



LINUX 7&8

WITH ANSIBLE AUTOMATION REFRENCE CUM LAB MANUAL

By Musabuddin Syed
Redhat certified

VirtualPath Techno Solutions

E-Mail: info@virtualpathtech.com

Phone: +91 799 309 6092

website: www.virtualpathtech.com

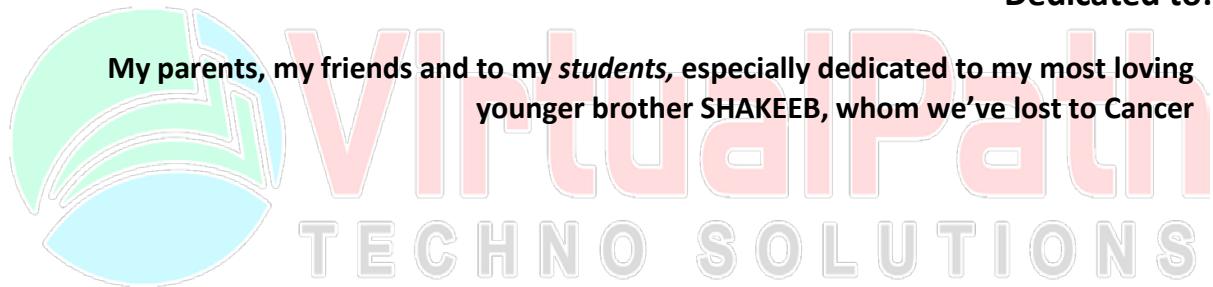
website: www.musab.in



LINUX 6&7 COMPARITIVE STUDY GUIDE CUM LAB MANUAL:

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied

Dedicated to:



Our Address:

VirtualPath Techno Solutions:

Phone/WhatsApp :+91 799 309 609292

Email: info@virtualpathtech.com

website: www.virtualpathtech.com

website: www.musab.in

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of Red Hat or other companies

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, is trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

XFS® is a trademark of Silicōn Graphics International Corp. or its subsidiaries in the United States and/or other countries

FOREWORD

About the Author:

Musabuddin Syed is a highly acclaimed trainer, author and solutions provider. He regularly trains students in in-house, online and corporate at VirtualPath Techno Solutions. He has an experience of more than 10 years in industry and Training, where he has delivered more than 400 batches successfully in various technologies.

I'm Highly Indebted To:

Almighty God

My Family

KernelSphere Technologies

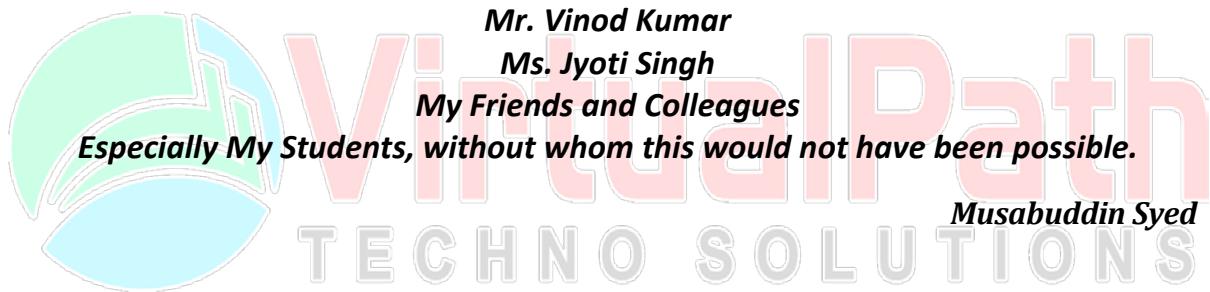
Mr. Vinod Kumar

Ms. Jyoti Singh

My Friends and Colleagues

Especially My Students, without whom this would not have been possible.

Musabuddin Syed



Words to the Students

Though we have taken utmost efforts to present you this book error free, but still it may contain some errors or mistakes. Students are encouraged to bring, if there are any mistakes or errors in this document to our notice. So that it may be rectified in the next edition of this document.

This document provides good information on every topic and lab practices. This could become more effective if equally good practice is done. I urge the readers/students to do rigorous practice to polish your skill sets.

You can reach us on the following email address

info@virtualpathtech.com

musab@virtualpathtech.com

musabsyd@gmail.com

Go through our website and blog

www.virtualpathtech.com

www.musab.in

OTHER COURSES AT VIRTUALPATH

LINUX CLUSTERS

DEVOPS

AWS

OPENSTACK

IBM &EMC SAN

NETAPP &
NETAPP CLUSTER

VMWARE

ORACLE DBA, RAC
& GOLDEN GATE

IBM AIX, POWER
HA, LPAR VIO

SHELL SCRIPTING

And Many More...

Table of Contents

1. Introduction to Linux	07-13
2. Basic Commands.....	14-32
3. RHEL 8 Basic Installation	33-45
4. Managing File Systems and Partitions	46-61
5. Swap Spaces Management	62-65
6. Logical Volume Management (LVM)	66-79
7. RHEL 8 LVM based Installation	80-82
8. User and Group Administration	83-93
9. Controlling Access to the Files	94-101
10. Enhanced User Security with SUDO	102-109
11. Network Configuration and Troubleshooting.....	110-122
12. NIC Teaming	123-128
13. Managing SELinux (Basics)	129-138
14. Booting Procedure of RHEL7/8 and Troubleshooting	139-147
15. Manage Installed Services	148-151
16. Introduction to Firewalld	152-155
17. Introduction to Cokpit in RHEL8	156-158
18. Backup and Restore (tar&gzip)	159-161
19. Job Automation with Cronjobs	162-167
20. Administrating Remote System	168-176
21. Software Management	177-197
22. Managing Processes	198-212
23. NFS (Network File System) Server	213-221
24. Samba Server	222-229
25. DNS (Domain Name System) Server	230-239
26. Mail Server	240-244
27. Web Server (Apache)	245-259

BONUS SERVERS & TOPICS FOR SELF-LEARNING

28. Squid Proxy Server	261-267
29. DHCP Server	268-273

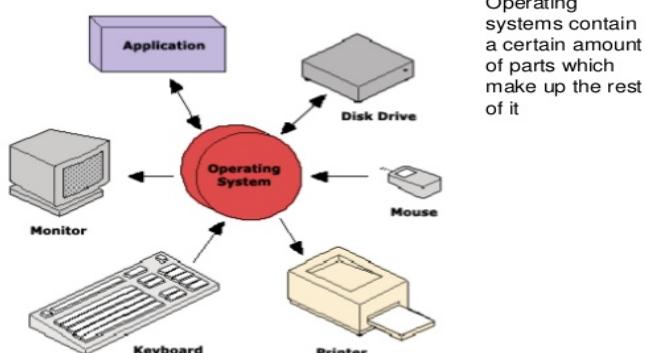
ANSIBLE AUTOMATION..... 274



This page has been left blank intentionally

INTRODUCTION TO UNIX & LINUX

What is an operating system?



Operating systems contain a certain amount of parts which make up the rest of it

What is Operating System?

Operating system is an interface between user and the computer hardware. The hardware of the computer cannot understand the human readable language as it works on binaries i.e. 0's and 1's. Also it is very tough for humans to understand the binary language, in such case we need an interface which can translate human language to hardware and vice-versa for effective communication.

Types of Operating System:

- Single User - Single Tasking Operating System
- Single User - Multitasking Operating System
- Multi User - Multitasking Operating System

Single User - Single Tasking Operating System

In this type of operating system only one user can log into system and can perform only one task at a time.

E.g.: MS-DOS

Single User - Multi tasking operating System

This type of O/S supports only one user to log into the system but a user can perform multiple tasks at a time, browsing internet while playing songs etc.

E.g.: Windows -98, XP, vista, 7,8,10 etc.

Multi User - Multi Tasking Operating System

This type of O/S provides multiple users to log into the system and also each user can perform various tasks at a time. In a broader term multiple users can logged in to system and share the resources of the system at the same time.

E.g.: UNIX, LINUX etc.

HISTORY OF UNIX

In the beginning, there was AT&T.

Bell Labs' Ken Thompson developed UNIX in 1969 so he could play games on a scavenged DEC PDP-7. With the help of Dennis Ritchie, the inventor of the "C" programming language, Ken rewrote UNIX entirely in "C" so that it could be used on different computers. In 1974, the OS was licensed to universities for educational purposes. Over the years, hundreds of people added and improved upon the system, and it spread into the commercial world. Dozens of different UNIX "flavors" appeared, each with unique qualities, yet still having enough similarities to the original AT&T version. All of the "flavors" were based on either AT&T's System V or Berkeley System Distribution (BSD) UNIX, or a hybrid of both.

During the late 1980's there were several of commercial implementations of UNIX:

- Apple Computer's A/UX
- AT&T's System V Release 3
- Digital Equipment Corporation's Ultrix and OSF/1 (renamed to DEC UNIX)
- Hewlett Packard's HP-UX
- IBM's AIX
- Lynx's Real-Time UNIX
- NeXT's NeXTStep
- Santa Cruz Operation's SCO UNIX
- Silicon Graphics' IRIX
- SUN Microsystems' SUN OS and Solaris
- and dozens more.

The Open Standards Foundation is a UNIX industry organization designed to keep the various UNIX flavors working together. They created operating systems guidelines called POSIX to encourage inter-operability of applications from one flavor of UNIX to another. Portability of applications to different gave UNIX a distinct advantage over its mainframe competition.

Then came the GUIs. Apple's Macintosh operating system and Microsoft's Windows operating environment simplified computing tasks, and made computers more appealing to a larger number of users. UNIX wizards enjoyed the power of the command line interface, but acknowledged the difficult learning curve for new users. The Athena Project at MIT developed the X Windows Graphical User Interface for UNIX computers. Also known as the X11 environment, corporations developed their own "flavors" of the UNIX GUIs based on X11. Eventually, a GUI standard called Motif was generally accepted by the corporations and academia.

During the late 1990's Microsoft's Windows NT operating system started encroaching into traditional UNIX businesses such as banking and high-end graphics. Although not as reliable as UNIX, NT became popular because of the lower learning curve and its similarities to Windows 95 and 98. Many traditional

UNIX companies, such as DEC and Silicon Graphics abandoned their OS for NT. Others, such as SUN, focused their efforts on niche markets, such as the Internet.

Linus Torvalds had a dream. He wanted to create the coolest operating system in the world that was free for anyone to use and modify. Based on an obscure UNIX flavor called MINIX, Linus took the source code and created his own flavor, called Linux. Using the power of the Internet, he distributed copies of his OS all over the world, and fellow programmers improved upon his work. In 1999, with a dozen versions of the OS and many GUIs to choose from, Linux is causing a UNIX revival. Knowing that people are used to the Windows tools, Linux developers are making applications that combine the best of Windows with the best of UNIX.

UNIX Principles

- **Everything is a file:-** UNIX system have many powerful utilities designed to create and manipulate files. The UNIX security model is based around the security of files. By treating everything as a file, you can secure access to hardware in the same way as you secure access to a document.
- **Configuration data stored in text:** - Storing configuration in text allows an administrator to move a configuration from one machine to another easily, provide the ability to roll back a system configuration to a particular date and time.
- **Small, Single-Purpose Programs:** - UNIX provides many utilities.
- **Avoid captive user interfaces:-**
- **Ability to chain programs together to perform complex tasks:-** A core design feature of UNIX is that output of one program can be the input for another. This gives the user the flexibility to combine many small programs together to perform a larger, more complex task.

GNU Project/ FSF

- GNU project started in 1984
 - a) Goal: Create 'free' UNIX clone
 - b) By 1990, nearly all required user space application created.
Example:-gcc, emacs, etc.
- Free Software Foundation
 - a) Non-Profit organization that manages the GNU project.

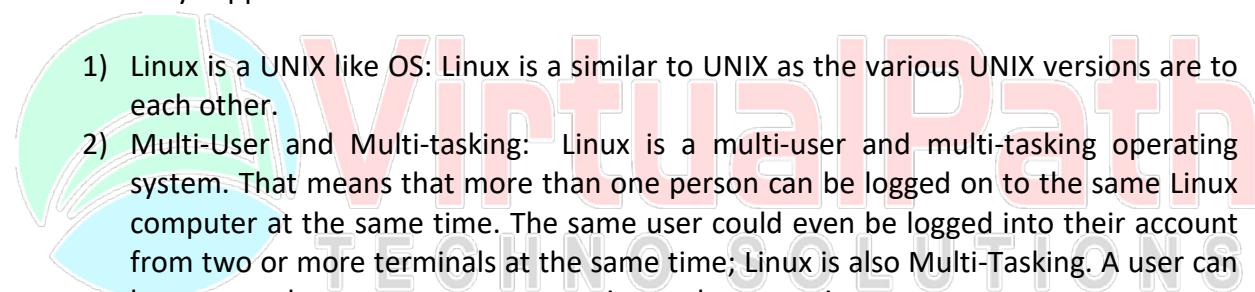
GPL – GNU (General Public License)

- primary license for open source software
- encourages free software
- All enhancements and changes to GPL software must also be GPL
- Often called 'copy left' (All rights reversed)

Linux Origins

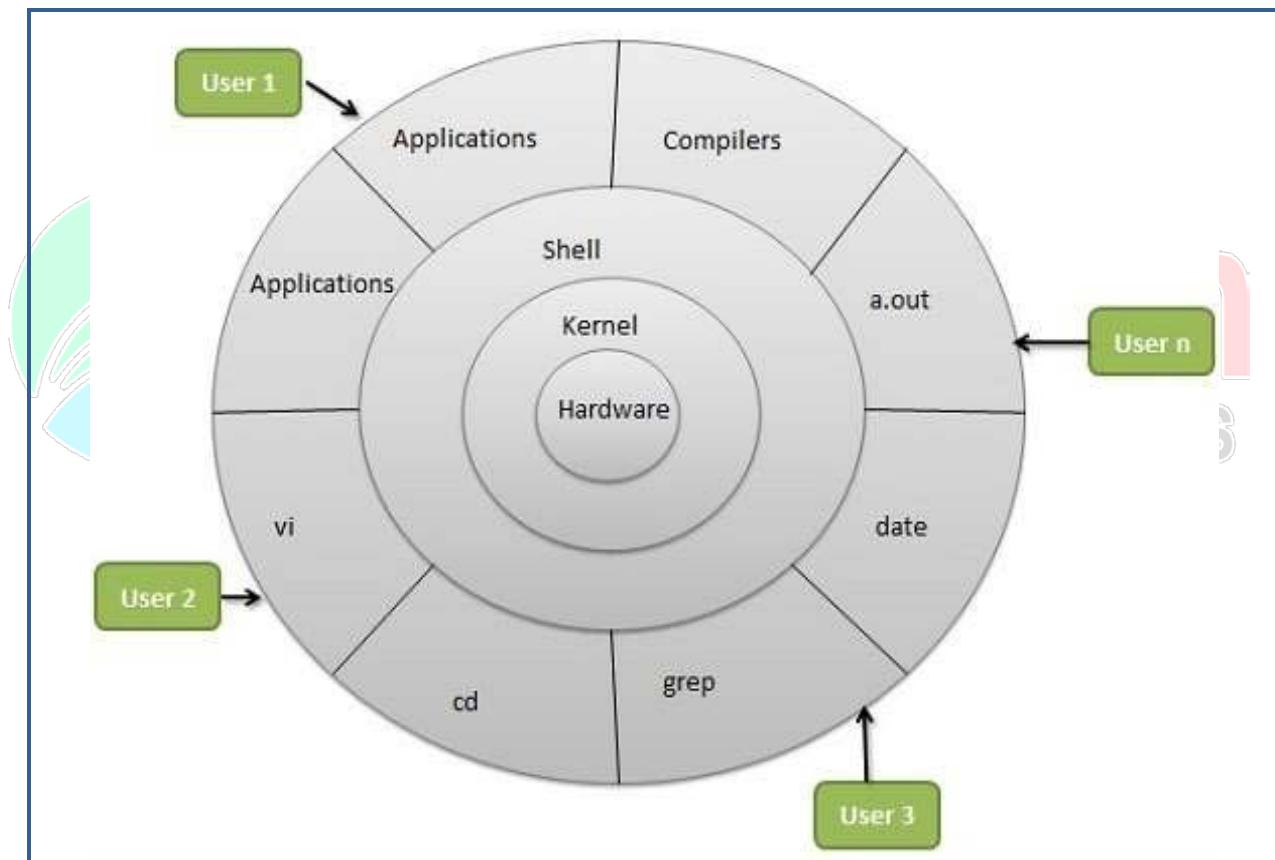
- **LINUS TORVALDS**
 - a) Finnish college student in 1991
 - b) Created Linux Kernel
- When Linux Kernel combined with GNU applications, complete free UNIX like OS was developed.

Why Linux?

- Fresh implementation of UNIX APIs
 - Open source development model
 - Supports wide variety of hardware
 - Supports many networking protocols and Configurations
 - Fully supported
- 
- 1) Linux is a UNIX like OS: Linux is a similar to UNIX as the various UNIX versions are to each other.
 - 2) Multi-User and Multi-tasking: Linux is a multi-user and multi-tasking operating system. That means that more than one person can be logged on to the same Linux computer at the same time. The same user could even be logged into their account from two or more terminals at the same time; Linux is also Multi-Tasking. A user can have more than one program executing at the same time.
 - 3) Wide hardware support: Red Hat Linux support most pieces modern x86 compatible PC hardware
 - 4) Fully Supported: Red Hat Linux is a fully supported distribution Red Hat Inc. provides many support programs for the smallest to the largest companies.

ARCHITECTURE OF UNIX

The architecture of UNIX can be divided into three levels of functionality, as shown in Figure. The lowest level is the *kernel*, which schedules tasks, manages resources, and controls security. The next level is the *shell*, which acts as the user interface, interpreting user commands and starting applications. The highest level is *utilities*, which provides utility functions. In other words it is the USER level, as user is the one who operates those utilities.



FILESYSTEM HIERARCHY

Linux uses single rooted, inverted tree like file system hierarchy

/

This is top level directory
It is parent directory for all other directories
It is called as ROOT directory
It is represented by forward slash (/)
C:\ of windows

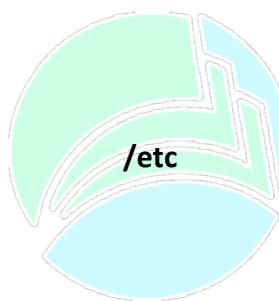
/root

it is home directory for root user (super user)
It provides working environment for root user
C:\Documents and Settings\Administrator

/home

it is home directory for other users
It provide working environment for other users (other than root)
c:\Documents and Settings\username

/boot



it contains bootable files for Linux
Like vmlinuz (kernel).... ntoskrnl
Initrd (INITial Ram Disk)and
GRUB (GRand Unified Boot loader).... boot.ini, ntldr

it contains all configuration files
Like /etc/passwd..... User info
/etc/resolv.conf... Preferred DNS
/etc/dhcpd.conf.... DHCP server
C:\windows\system32\dirvers\

/usr

by default soft wares are installed in /usr directory
(UNIX Sharable Resources)
c:\program files

/opt

It is optional directory for /usr
It contains third party softwares
c:\program files

/bin

it contains commands used by all users
(Binary files)

/sbin

it contains commands used by only Super User (root)
(Super user's binary files)

/dev

it contains device files
Like /dev/hda ... for hard disk
/dev/cd rom ... for cd rom
Similar to device manager of windows

/proc

it contain process files

Its contents are not permanent, they keep changing

It is also called as Virtual Directory

Its file contain useful information used by OS

Like /proc/meminfo... information of RAM/SWAP

/proc/cpuinfo... information of CPU

/var

it contains variable data like mails, log files

/tmp

contains the temporary files for small period of time

/mnt

it is default mount point for any partition

It is empty by default

/media

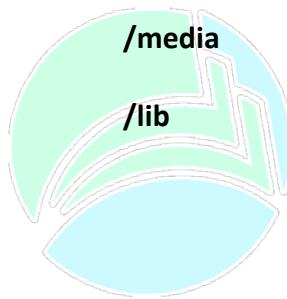
it contains all of removable media like CD-ROM, pen drive

/lib

it contains library files which are used by OS

It is similar to dll files of windows

Library files in Linux are SO (shared object) files



VirtualPath
TECHNO SOLUTIONS

UNIX BASIC COMMANDS

Creating, Removing, Copying, Moving files & Directories

Creating a file in Linux

Using cat command:

- cat (Concatenate) command is used to create a file and to display and modify the contents of a file.
- **To create a file**

```
# cat > filename (say myfile)
```

Hello World

Ctrl+d (To save the file)

```
[root@musab1 ~]# cat > myfile
HELLO WORLD
[root@musab1 ~]#
```

To display the content of the file

```
# cat filename (say myfile)
```

```
[root@musab1 ~]# cat myfile
HELLO WORLD
[root@musab1 ~]#
```

To append the data in the already existing file

```
# cat >> <filename>
```

```
# cat >> myfile
```

Ctrl+d (to exit back)

```
[root@musab1 ~]# cat >> myfile
WELCOME TO LINUX
[root@musab1 ~]#
```

Creating multiple files at same time using touch command

```
#touch <filename> <filename> <filename>
```

```
#touch file1 file2 file3
```

Note: to check the files use # ls command

```
[root@musab1 ~]# touch file1 file2 file3
[root@musab1 ~]# ls
anaconda-ks.cfg  Documents  file1  file3
Desktop          Downloads  file2  install.log
```

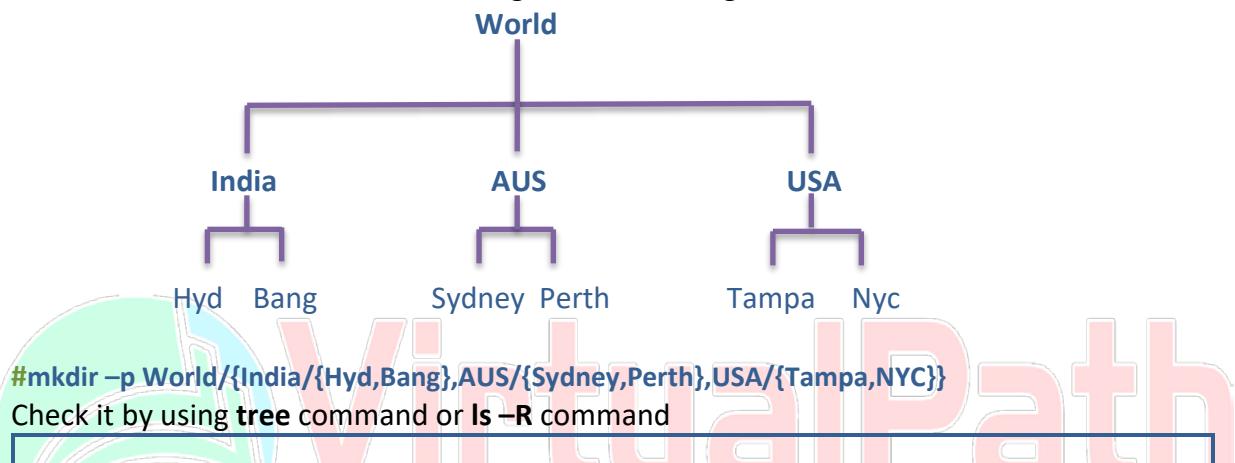
Creating a Directory

```
#mkdir <dir name>
#mkdir mydir
```

```
[root@musab1 ~]# mkdir mydir
[root@musab1 ~]# ls
anaconda-ks.cfg  Documents  file1  file3      install.log.syslog  mydir
Desktop          Downloads  file2  install.log  Music           myfile
```

Making multiple directories inside a directory

Let us make some directories according to the following architecture in one command.



```
[root@musab1 ~]# mkdir -p World/{India/{Hyd,Bang},AUS/{Sydney,Perth},USA/{Tampa, NYC}}
[root@musab1 ~]# tree World
World
├── AUS
│   ├── Perth
│   └── Sydney
├── India
│   ├── Bang
│   └── Hyd
└── USA
    ├── NYC
    └── Tampa

9 directories, 0 files
[root@musab1 ~]#
```

Copying files into directory

```
#cp <source filename> <destination directory in which to paste the file>
#cp file1 mydir
```

```
[root@musab1 ~]# cp file1 mydir
[root@musab1 ~]# cd mydir
[root@musab1 mydir]# ls
file1
```

Copying directories from one location to other

```
# cp -rvfp <dir name> <destination name>
#cp -rvfp mydir2 mydir
```

```
[root@musab1 ~]# cp -rvfp mydir2 mydir
`mydir2' -> `mydir/mydir2'
`mydir2/file5' -> `mydir/mydir2/file5'
`mydir2/file2' -> `mydir/mydir2/file2'
`mydir2/file1' -> `mydir/mydir2/file1'
`mydir2/file4' -> `mydir/mydir2/file4'
`mydir2/file3' -> `mydir/mydir2/file3'
[root@musab1 ~]# cd mydir
[root@musab1 mydir]# ls
file1  mydir2
```

Moving files from one location to other (cut and Paste)

```
#mv <filename> <Destination directory>
#mv myfile mydir
```

```
[root@musab1 ~]# ls
anaconda-ks.cfg  Documents  file1  file3      install.log.syslog  mydir  myfile
Desktop          Downloads   file2  install.log  Music           mydir2 -p
[root@musab1 ~]# mv myfile mydir
[root@musab1 ~]# cd mydir
[root@musab1 mydir]# ls
file1  mydir2  myfile
```

Moving a Directory from one location to other

```
#mv <dir name> <destination dir name>
#mv mydir mydir2
```

```
[root@musab1 ~]# ls
anaconda-ks.cfg  Documents  file1  file3      install.log.syslog  mydir  -p
Desktop          Downloads   file2  install.log  Music           mydir2  Pictures
[root@musab1 ~]# mv mydir mydir2
[root@musab1 ~]# cd mydir2
[root@musab1 mydir2]# ls
file1  file2  file3  file4  file5  mydir
[root@musab1 mydir2]#
```

Renaming a File

```
#mv <old name> <new name>
#mv file1 newfile
```

```
[root@musab1 mydir]# ls
file1  mydir2  myfile
[root@musab1 mydir]# cat myfile
HELLO WORLD
WELCOME TO LINUX
[root@musab1 mydir]# mv myfile new
[root@musab1 mydir]# ls
file1  mydir2  new
[root@musab1 mydir]# cat new
HELLO WORLD
WELCOME TO LINUX
[root@musab1 mydir]#
```

Renaming a Directory

- The procedure and command for renaming the directory is exactly same as renaming a file.

```
#mv old name new name
#mv mydir newdir
```

```
[root@musab1 mydir2]# ls
file1 file2 file3 file4 file5 mydir
[root@musab1 mydir2]# ls mydir
file1 mydir2 new
[root@musab1 mydir2]# mv mydir newdir
[root@musab1 mydir2]# ls
file1 file2 file3 file4 file5 newdir
[root@musab1 mydir2]# ls newdir
file1 mydir2 new
```

Removing a File

#rm filename or #rm -f filename (without prompting)

```
[root@musab1 mydir2]# ls
file1 file2 file3 file4 file5 newdir
[root@musab1 mydir2]# rm file1
rm: remove regular empty file `file1'? y
Without prompting: 
[root@musab1 mydir2]# rm -f file1
[root@musab1 mydir2]# ls
file2 file3 file4 file5 newdir
[root@musab1 mydir2]#
```

Removing an Empty directory

#rmdir dirname

```
[root@musab1 ~]# ls
anaconda-ks.cfg Documents file2 install.log Music newdir -p
Desktop Downloads file3 install.log.syslog mydir2 newfile Pictures
[root@musab1 ~]# ls newdir
[root@musab1 ~]# rmdir newdir
[root@musab1 ~]# ls
anaconda-ks.cfg Documents file2 install.log Music newfile Pictures
Desktop Downloads file3 install.log.syslog mydir2 -p Public
[root@musab1 ~]#
```

Removing a directory with files or directories inside

A dir which is having some contents inside it cannot be removed by **rmdir** command. There are two ways to delete the directory with contents.

- i. Remove the contents inside the directory and then run **rmdir** command
- ii. Run **#rm -rf dirname** (where r stands for recursive and f stands for forcefully).

```
[root@musab1 ~]# ls
anaconda-ks.cfg  Documents  file2  install.log      Music
Desktop          Downloads  file3  install.log.syslog mydir2
[root@musab1 ~]# ls mydir2
file2  file3  file4  file5  newdir
[root@musab1 ~]# rm -rf mydir2
[root@musab1 ~]# ls
anaconda-ks.cfg  Documents  file2  install.log      Music
Desktop          Downloads  file3  install.log.syslog newfile
[root@musab1 ~]# 
```



It has 3 modes:

- 1 Command Mode
- 2 Insert mode (edit mode)
- 3 extended command mode

Note: When you open the vim editor, it will be in the command mode by default.

In the command mode the cursor's can be used as
h/l/k/j to move cursor left/right/up/down

Insert Mode:

i	To begin insert mode at the cursor position
I	To insert at the beginning of line
a	To append to the next word's letter
A	To Append at the end of the line
o	To insert a new line below the cursor position
O	To insert a new line above the cursor position

Command Mode:

gg	To go to the beginning of the file
G	To go to end of the file
w	To move the cursor forward, word by word
b	To move the cursor backward, word by word
nw	To move the cursor forward to n words (5W)
nb	To move the cursor backward to n words (5B)
u	To undo last change (word)
U	To undo the previous changes (entire line)
Ctrl+R	To redo the changes
yy	To copy a line
nyy	To copy n lines (5yy or 4yy)
p	To paste line below the cursor position
P	To paste line above the cursor position
dw	To delete the word letter by letter (like Backspace)
x	To delete the world letter by letter (like DEL Key)
dd	To delete entire line
ndd	To delete n no. of lines from cursor position(5dd)
/	To search a word in the file

Extended Mode: (Colon Mode)

Extended Mode is used for save and quit or save without quit using "Esc" Key with ":"

Esc+:w	To Save the changes
Esc+:q	To quit (Without saving)
Esc+:wq	To save and quit
Esc+:w!	To save forcefully
Esc+wq!	To save and quit forcefully
Esc+:x	To save and quit
Esc+:X	To give password to the file and remove password
Esc+:20(n)	To go to line no 20 or n
Esc+: se nu	To set the line numbers to the file
Esc+:se nonu	To Remove the set line numbers

To open multiple files in vim editor

#vim -o file1 file2

To switch between files use Ctrl +w

Listing files and directories:

- #ls list the file names
- #ls -l long listing of the file
- #ls -l filename to see the permissions of a particular file
- #ls -al shows the files in ascending order of modification.
- #ls p* All the files start with p.

#ls ?ample	Files with any first character and has ample
#ls -l*	Directory listing only
#ls -l directory name	to see the permissions of a particular directory
#ls [ae]*	First character of the filename must be a or e.
# ls [!ae]*	! Symbol complements the condition that follows. The characters must not be a or e.
#ls [a-m][c-z][4-9]	list all the files in specific range

Types of Files:

Symbol	Type of File
-	Normal file
d	Directory
l	Link file (shortcut)
b	Block file (Harddisk, Floppy disk)
c	Character file (Keyboard, Mouse)

Symbolic Link

There are two types of Links:-

	Soft Link	Hard link
1	Size of link file is equal to no. of characters in the name of original file	Size of both file is same
2	Can be created across the Partition	Can't be created across the partition
3	inode no. of source and link file is different	inode no. of both file is same
4	if original file is deleted, link is broken and data is lost	If original file is deleted then also link will contain data
5	SHORTCUT FILE	BACKUP FILE

Creating a soft link:

ln -s <source file> <destination>

```
[root@musabi ~]# ls
anaconda-ks.cfg  Documents  file2  install.log      Music      -p
Desktop          Downloads  file3  install.log.syslog newfile   Pictures
[root@musabi ~]# ln -s newfile softlink
[root@musabi ~]# ls -li newfile softlink
1445159 -rw-r--r--. 1 root root 0 Feb 13 13:55 newfile
1439034 lwxrwxrwx. 1 root root 7 Feb 13 15:16 softlink -> newfile
[root@musabi ~]#
```

Creating a Hard link:

#ln <source file> <Destination>

```
[root@musab1 ~]# ls
anaconda-ks.cfg  Documents  file2  install.log      Music
Desktop          Downloads   file3  install.log.syslog newfile
[root@musab1 ~]# ln newfile hardlink
[root@musab1 ~]# ls
anaconda-ks.cfg  Documents  file2  hardlink      install.log.syslog
Desktop          Downloads   file3  install.log    Music
[root@musab1 ~]# ls -li newfile hardlink
1445159 -rw-r--r--. 2 root root 0 Feb 13 13:55 hardlink
1445159 -rw-r--r--. 2 root root 0 Feb 13 13:55 newfile
[root@musab1 ~]#
```

Regular Expressions, Pipelines & I/O Redirections

Grep:

Grep stands for **Global Regular Expression Print**. It is used to pick out the required expression from the file and print the output. If grep is combined with another command it can be used to pick out the selected word, phrase from the output of first command and print it.

Examples of Grep:

Let us pick the information about **root** from the file **/etc/passwd** (/etc/passwd contains information about all the users present in the system)

#grep root /etc/passwd

```
[root@ linux ~]# grep root /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
[root@ linux ~]#
```

To avoid case sensitivity of the word (i.e. the word may be uppercase or lowercase) use **-i**

#grep -i linux test (lets grep the word **linux** whether upper or lower case in the file **test**)

```
[root@musab1 ~]# grep -i linux test
LINUX is freedom
linux is freedom
Linux is freedom
[root@musab1 ~]#
```

To display a word and 2 lines after the word:

#grep -nA2 wheel /etc/group

```
[root@ linux ~]# grep -nA2 wheel /etc/group
11:wheel:x:10:root
12-mail:x:12:mail,postfix
13-uucp:x:14:uucp
[root@ linux ~]#
```

To display a word and 2 lines after the word:

```
#grep -nB2 wheel /etc/group
```

```
[root@ linux ~]# grep -nB2 wheel /etc/group
9-mem:x:8:
10-kmem:x:9:
11:wheel:x:10:root
```

To display the things except the given word

```
#grep -v world test
```

```
[root@musab1 ~]# cat test
linux is freedom
Hello world
Welcome to my world
[root@musab1 ~]# grep -v world test
linux is freedom
```

To display the searched word in color

```
#grep --color root /etc/passwd
```

Combining grep with other commands

cat myfile | grep -i linux (pipe | is used to combine to commands)

#ls -l | grep -i myfile

ifconfig |grep -i eth0

Like this we can combine grep with many commands which we will see in later chapters

Filter Commands:

- Filter commands are used to filter the output so that the required things can easily be picked up. The commands which are used to filter the output are

**#less
#more
#head
#tail
#sort
#cut
#sed**

- less:**

The **less** command is used to see the output line wise or page wise.

Ex: less /etc/passwd

```
root:x:0:0:root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
```

Note: -press **Enter** key to scroll down line by line (or)

Use **d** to go to next page

Use **b** to go to previous page

Use **/** to search for a word in the file

Use **v** to go vi mode where you can edit the file and once you save it you will back to less command

more:

more is exactly same like **less**

Ex: #more /etc/passwd

Note: -press **Enter** key to scroll down line by line (or)

Use **d** to go to next page

Use **/** to search for a word in the file

Use **v** to go vi mode where you can edit the file and once you save it you will back to more command

head:

It is used to display the top **10 lines** of the file.

Ex:# head /etc/passwd

```
[root@ linux ~]# head /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
```

To display the custom lines

#head -n /etc/passwd (where n can be any number)

```
[root@ linux ~]# head -5 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

tail:

It is used to display the **last 10** lines of the file

#tail /etc/passwd

```
[root@ linux ~]# tail /etc/passwd
apache:x:48:48:Apache:/var/www:/sbin/nologin
nslcd:x:65:55:LDAP Client User:::/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
pulse:x:496:494:PulseAudio System Daemon:/var/run/pulse:/sbin/nologin
gdm:x:42:42::/var/lib/gdm:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72:72:::/sbin/nologin
visitor:x:500:500:visitor:/home/visitor:/bin/bash
```

To display the custom lines

#tail -n /etc/passwd (where n can be any number)

```
[root@ linux ~]# tail -5 /etc/passwd
user:x:500:500: user:/home/ user:/bin/bash
amit:x:501:501::/home/amit:/bin/bash
vivek:x:502:502::/home/vivek:/bin/bash
musab:x:503:503::/home/musab:/bin/bash
rahul:x:504:504::/home/rahul:/bin/bash
```

Sort:

It is used to sort the output in numeric or alphabetic order

#sort filename

```
[root@musab1 ~]# cat test
Linux is freedom
Linux is freedom
Welcome to my world
Welcome to my world
Hello world
Hello world
[root@musab1 ~]# sort test
Hello world
Hello world
Linux is freedom
Linux is freedom
Welcome to my world
Welcome to my world
[root@musab1 ~]#
```

To sort the file according to numbers

#sort -d test or #sort -h test

```
[root@musab1 ~]# cat test
6. Linux is freedom
3. Linux is freedom
1. Welcome to my world
2. Welcome to my world
4. Hello world
7. Hello world
[root@musab1 ~]# sort -d test
1. Welcome to my world
2. Welcome to my world
3. Linux is freedom
4. Hello world
6. Linux is freedom
7. Hello world
```

To remove the duplicate entries from the output

#sort -u test

```
[root@musab1 ~]# cat test
Linux is freedom
Linux is freedom
Welcome to my world
Welcome to my world
Hello world
Hello world
[root@musab1 ~]# sort -u test
Hello world
Linux is freedom
Welcome to my world
```



cut command:

The cut command is used to pick the given expression (in columns) and display the output.

cut -d -f filename (where d stands for delimiter ex. :, " " etc and f stands for field)

```
[root@ linux ~]# cut -d: -f1 /etc/passwd
root
bin
daemon
adm
lp
sync
shutdown
halt
mail
uucp
```

To delimit spaces and print the field

#cut -d " " -f1 filename

To delimit commas and print the field

#cut -d, -f1 filename

```
[root@linux ~]# cat hello
hello,how,are,you
[root@linux ~]# cut -d, -f1 hello
hello
```

sed command:

sed stands for **stream editor**, which is used to search a word in the file and replace it with the word required to be in the output

Note: it will only modify the output, but there will be no change in the original file.

#sed 's/searchfor/replacewith/g' filename

```
[root@musab1 ~]# cat test
Linux is freedom
Linux is freedom
Welcome to my world
Welcome to my world
Hello world
Hello world
[root@musab1 ~]# sed 's/Linux/LINUX/g' test
LINUX is freedom
LINUX is freedom
Welcome to my world
Welcome to my world
Hello world
Hello world
```

I/O Redirection:

Redirection is a process where we can copy the output of any command(s), file(s) into a new file. There are two ways of redirecting the output into a file.

Using **>** or **>> filename** after the command, and

Using **tee** command

Let's see the > and >> option first

Syn: command > new file

Note: if the given name of the file is not available a new file will be created automatically. If the file already exists then it will overwrite contents of that file.

```
[root@musab1 ~]# cat test
Linux is freedom
[root@musab1 ~]# sed 's/Linux/LINUX/g' test > test2
[root@musab1 ~]# cat test2
LINUX is freedom
```

Appending another output in same the same file

```
[root@musab1 ~]# cat file2
Welcome to MyWorld
[root@musab1 ~]# cat file2 >>test2
[root@musab1 ~]# cat test2
LINUX is freedom
Welcome to MyWorld
```

Likewise there are many options where we can use redirections

Ex:

Copying contents of two files in a new file

```
#cat file1 file2 > file3
```

Using tee command:

The above options of redirections will not display any output, but directly save the output in a file. Using tee command will not only redirect the output to new file but it will also display the output.

Syn: cat <filename> | tee <new file name>

Note: if the given name of the file (newfile) is not available a new file will be created automatically. If the file already exists then it will overwrite contents of the file.

```
#cat file2 |tee file3
```

```
[root@musab1 ~]# cat file2 |tee file3
Welcome to MyWorld
[root@musab1 ~]# cat file3
Welcome to MyWorld
```

Appending data in the same file using tee command

Syn: cat filename |tee -a filename2

```
#cat test | tee -a file2
```

```
[root@musab1 ~]# cat test |tee -a file2
Linux is freedom
[root@musab1 ~]# cat file2
Welcome to MyWorld
Linux is freedom
```

Find command:

find command is used to find the files or directory's path, it is exactly like the find option in windows where you can search for a file.

Syntax: find / (under root) –option filename

Options that can be used with find command:

Option	Usage
-name	For searching a file with its name
-inum	For searching a file with particular inode number
-type	For searching a particular type of file
-user	For files whose owner is a particular user
-group	For files belonging to particular group

Finding a File with name

#find / -name newfile

```
[root@musab1 ~]# find / -name newfile
/root/newfile
```

Finding a file with its inode number

#find / -inum 1445159

```
[root@musab1 ~]# find / -inum 1445159
find: `/proc/28735/task/28735/fd/5': No such file or directory
find: `/proc/28735/task/28735/fdinfo/5': No such file or directory
find: `/proc/28735/fd/5': No such file or directory
find: `/proc/28735/fdinfo/5': No such file or directory
/root/hardlink
/root/newfile
[root@musab1 ~]#
```

Finding the files, whose owner is a user called "ktuser"

#find / -user myuser

```
[root@musab1 ~]# find / -user myuser
find: `/proc/28763/task/28763/fd/5': No such file or directory
find: `/proc/28763/task/28763/fdinfo/5': No such file or directory
find: `/proc/28763/fd/5': No such file or directory
find: `/proc/28763/fdinfo/5': No such file or directory
/var/spool/mail/myuser
/home/myuser
/home/myuser/.bash_profile
/home/myuser/.gnome2
/home/myuser/.bash_logout
/home/myuser/testfile
```

Finding the files whose group is “myuser”

#find / -group myuser

```
[root@musab1 ~]# find / -group myuser
find: `/proc/28780/task/28780/fd/5': No such file or directory
find: `/proc/28780/task/28780/fdinfo/5': No such file or directory
find: `/proc/28780/fd/5': No such file or directory
find: `/proc/28780/fdinfo/5': No such file or directory
/home/myuser
/home/myuser/.bash_profile
/home/myuser/.gnome2
/home/myuser/.bash_logout
/home/myuser/testfile
```

File Permissions:

Permissions are applied on three levels:

- Owner or User level
- Group level
- Others level

Access modes are of three types:

- r read only
- w write/edit/delete/append
- x execute/run a command

Access modes are different on file and directory:

Permissions	Files	Directory
r	Open the file	'ls'/list the contents of directory
w	Write, edit, append, delete file	Add/Del/Rename contents of directory
x	To run a command/shell script	To enter into directory using 'cd'

```
[root@musab1 ~]# ls -l myfile
-rw-r--r--. 2 root root 0 Feb 13 13:55 myfile
[root@musab1 ~]# ls -ld mydir
drwxr-xr-x. 2 root root 4096 Feb 13 16:43 mydir
```

Filetype+permission, links, owner, group name of owner, size in bytes, date of modification, file name

Permission can be set on any file/dir by two methods:

1 Symbolic method (ugo)

2 Absolute methods (numbers)

1 Symbolic method (ugo):

- Symbolic mode: General form of symbolic mode is:
chmod [who] [+/-/=] [permissions] file
 who → To whom the permissions to be assigned
 User/owner (u); group (g); others (o)

Example:

Assigning different permissions to the file (user=rwx, group=rw and others=r)

#chmod u=rwx,g=rw,o=r myfile (where myfile is the name of the file)

```
[root@musab1 ~]# chmod u=rwx,g=rw,o=r myfile
[root@musab1 ~]# ls -l myfile
-rwxrw-r--. 2 root root 0 Feb 13 13:55 myfile
[root@musab1 ~]#
```

Assigning full permission to the file i.e. rwx to all

#chmod ugo=rwx <file name>

```
[root@musab1 ~]# chmod ugo=rwx myfile
[root@musab1 ~]# ls -l myfile
-rwxrwxrwx. 2 root root 0 Feb 13 13:55 myfile
```

Likewise you can add or remove permissions from any file for anyone (user group or other)

- #chmod u+x myfile (Adding execute permission to user only)
- #chmod go-wx myfile (Removing write and execute permissions from group and other)
- #chmod go+wx myfile (Adding write and execute permissions from group and other)
- #chmod go=r myfile (Giving only read permission to group and other)

2 Absolute Method (numbers)

In Absolute method we use numbers instead of using symbols i.e.

- Read=4
- Write=2
- Execute=1

Assigning different permissions to the file (user=rwx, group=rw and others=r)

#chmod 764 myfile (where 7 means rwx i.e. 4+2+1, rw=6 i.e. 4+2 and 1 indicates x)

```
[root@musab1 ~]# ls -l myfile
-rwxrwxrwx. 2 root root 0 Feb 13 13:55 myfile
[root@musab1 ~]# chmod 764 myfile
[root@musab1 ~]# ls -l myfile
-rwxrw-r--. 2 root root 0 Feb 13 13:55 myfile
```

Assigning full permission to the file i.e. rwx to all

#chmod 777 myfile

```
[root@musab1 ~]# ls -l myfile
-rwxrw-r--. 2 root root 0 Feb 13 13:55 myfile
[root@musab1 ~]# chmod 777 myfile
[root@musab1 ~]# ls -l myfile
-rwxrwxrwx. 2 root root 0 Feb 13 13:55 myfile
```

Likewise you can give different permissions according to your requirement

Removing all permissions from others

#chmod 770 myfile (where 0 indicates no permissions)

Note: All the above permissions and procedure is same for files and directories.

Umask:

When we create any file using touch, cat or vi commands they get created with default file permissions as stored in umask (**User file creation mask**).umask is a 4 digit octal number which tells Unix which of the three permissions are to be denied rather than granted. Umask will decide that what should be the default permissions for a file and directory when it is created.

The default umask value is 0022

#umask

```
[root@musab1 ~]# umask
0022
```

Calculation of default permissions for file and directory, basing upon the umask value

Note: For a file by default it cannot have the execute permission, so the maximum full permission for a file at the time of creation can be **666** (i.e. 777 -111 = 666), whereas a directory can have full permissions i.e. **777**

- The full permission for the file 666
- Minus the umask value - 022
- The default permission for file is 644 (rw-,r--,r--)

```
[root@musab1 ~]# umask
0022
[root@musab1 ~]# touch test
[root@musab1 ~]# ls -l test
-rw-r--r--. 1 root root 0 Feb 13 16:55 test
```

- The full permission for the directory 777
- Minus the umask value - 022
- The default permission for file is 755 (rwx, r-x, r-x)

```
[root@musab1 ~]# umask
0022
[root@musab1 ~]# mkdir testdir
[root@musab1 ~]# ls -ld testdir
drwxr-xr-x. 2 root root 4096 Feb 13 16:56 testdir
```

Modifying the umask value:

#umask 002

The Modified default Permission for a file will be **666-002=664** i.e. **rw,rw,r**, and for the directory it will be **777-002=775** i.e. **rwx,rwx,r-x**.

```
[root@musab1 ~]# umask  
0022  
[root@musab1 ~]# umask 002  
[root@musab1 ~]# umask  
0002
```

Note: Create a file and a directory and check for the default permissions.

Visit www.musab.in for all basics video tutorial by Musab Syed

These were the few things amongst the basics; keep working to furnish your basics. After All, *"if the foundation is good then only the building can stand still"*



RHEL 8 BASIC INSTALLATION

Minimum and Recommended Requirements to install RHEL7/8

MINIMUM	RECOMMENDED
Intel/AMD Dual core processor	Intel/AMD Quad Core Processor
4GB RAM	16GB RAM
25GB HARD DISK SPACE	50GB HARD DISK SPACE

Note: RHEL7/8 comes only in 64 bit version

Minimum Partition creation and sizes for basic installation

Partitions	Sizes
/ {root}	20-30 GB APPROX
/boot	1 GB
SWAP	Twice of RAM approx

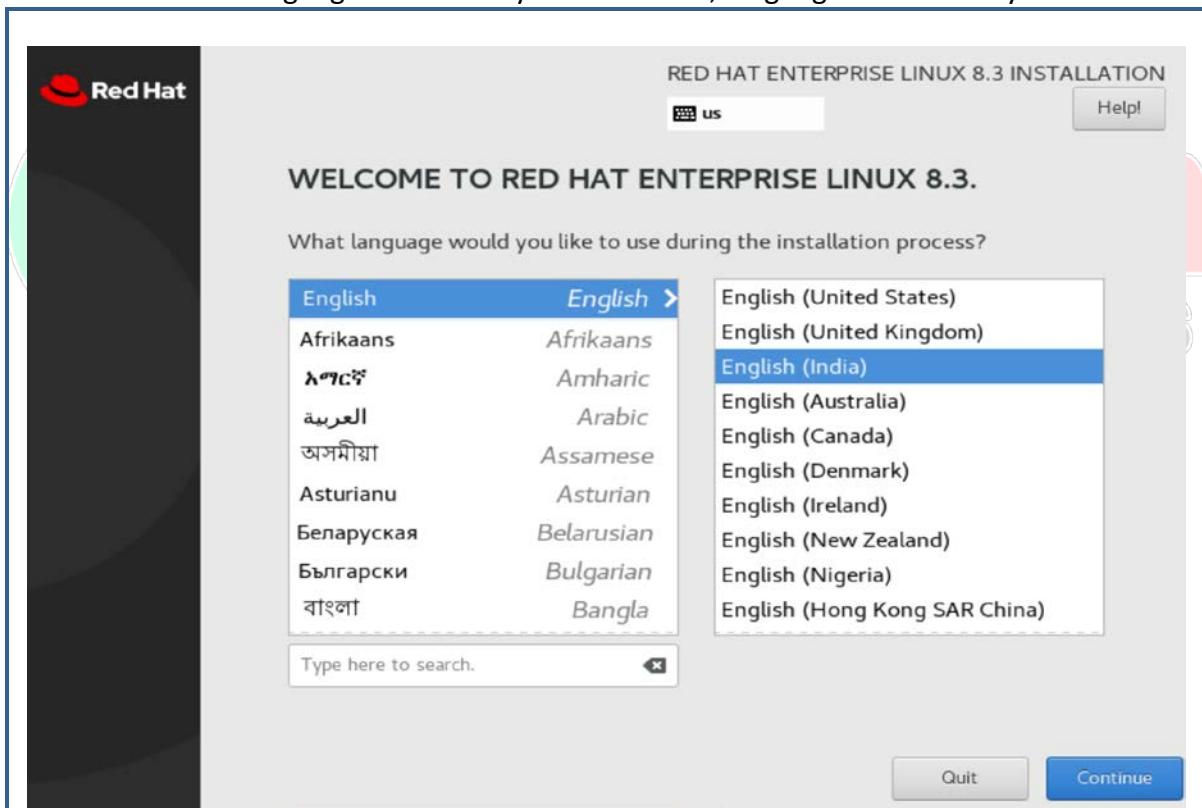
Installing RHEL7/8 with above specification

- Enter into BIOS setting and make CD/DVD Drive as first boot device
- Make sure that VT {Virtual Technology} is enabled for RHEL7-64 bit systems
- Insert the RHEL 7 CD/DVD into CD/DVD drive and boot the system
- If booted from CD/DVD Rom the following screen will be displayed

- Move the cursor to **install Red Hat Enterprise linux 8.x**, hit Enter



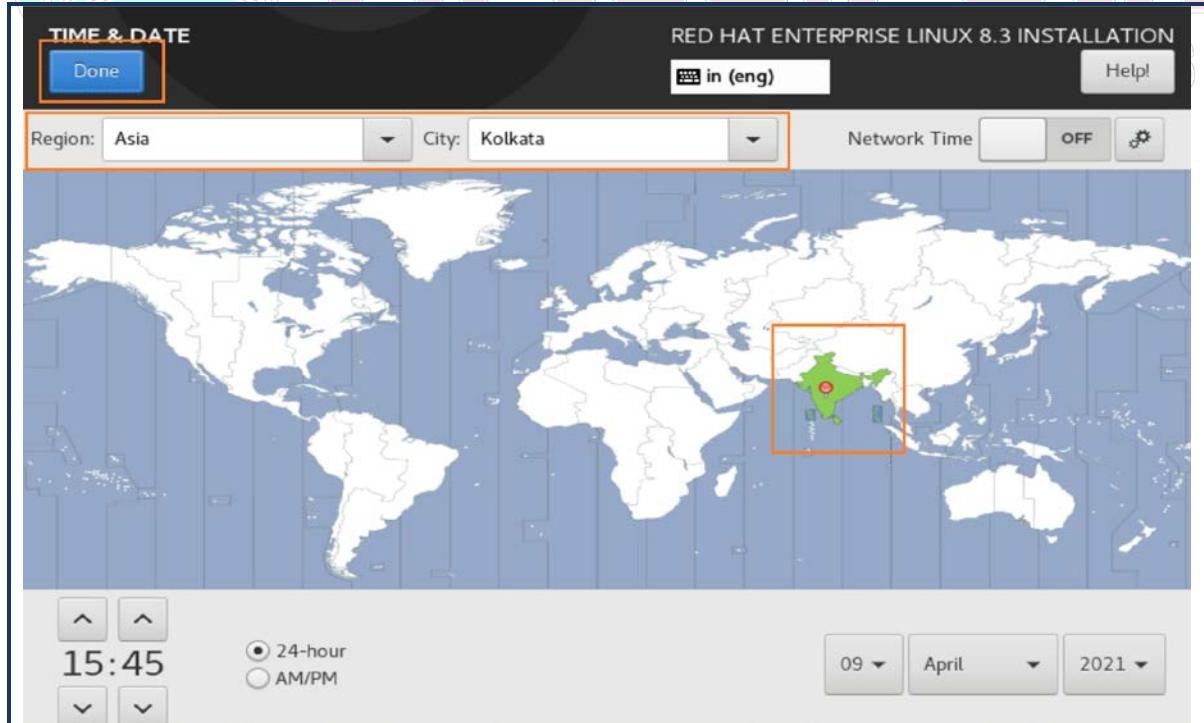
- Select the Language and Country for time zone, language and currency



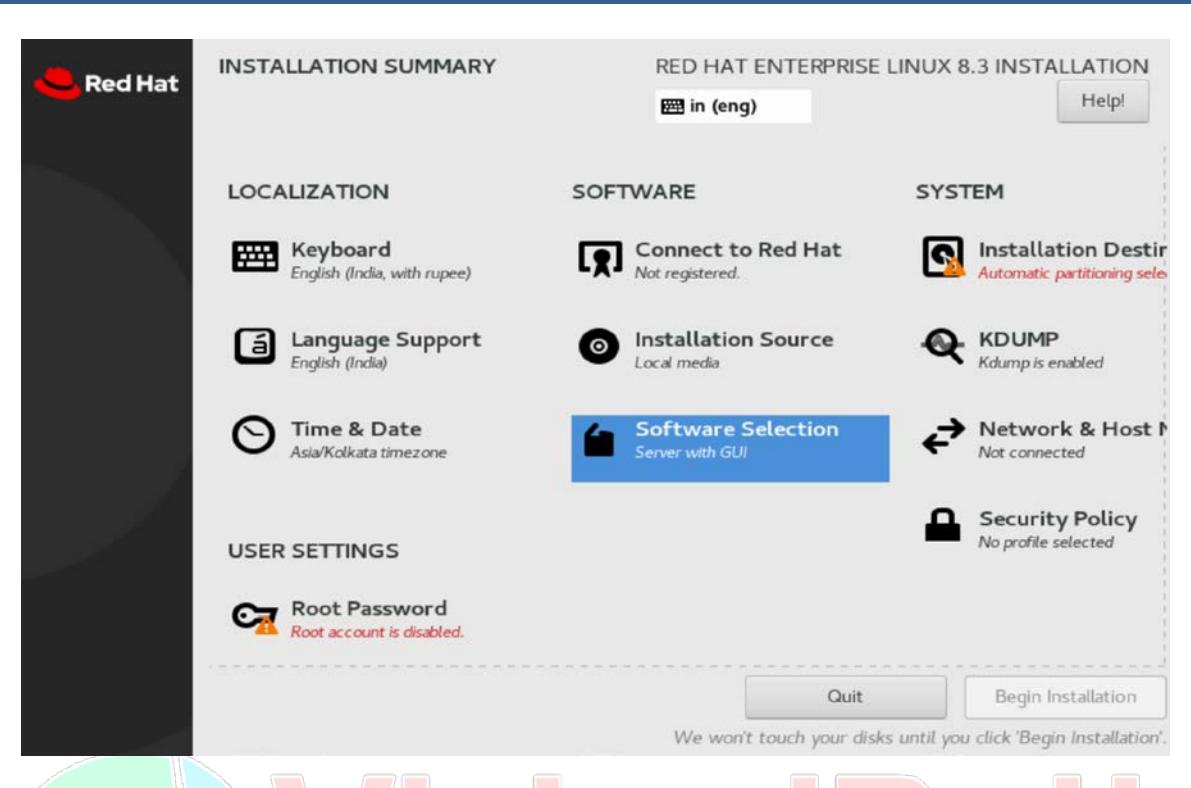
- Select date and time to make time zone selection



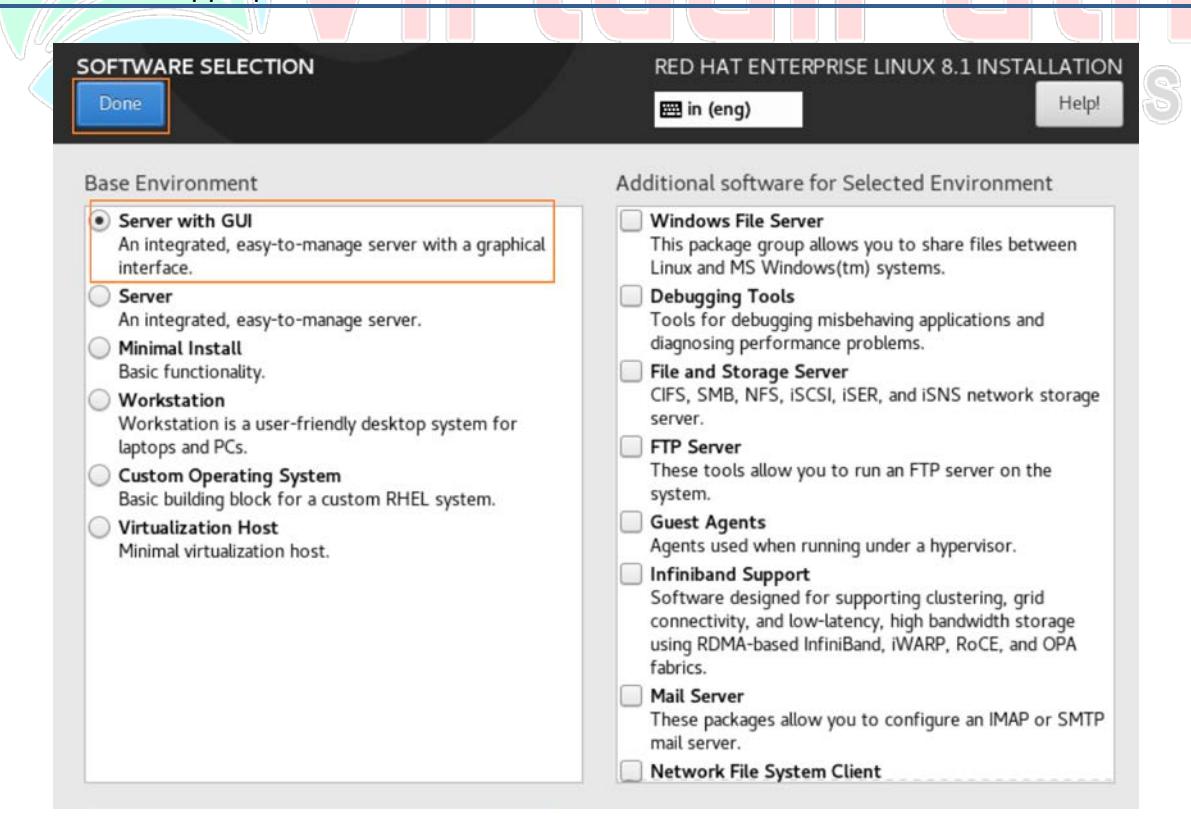
- Make the selection as per your country and click on **done**



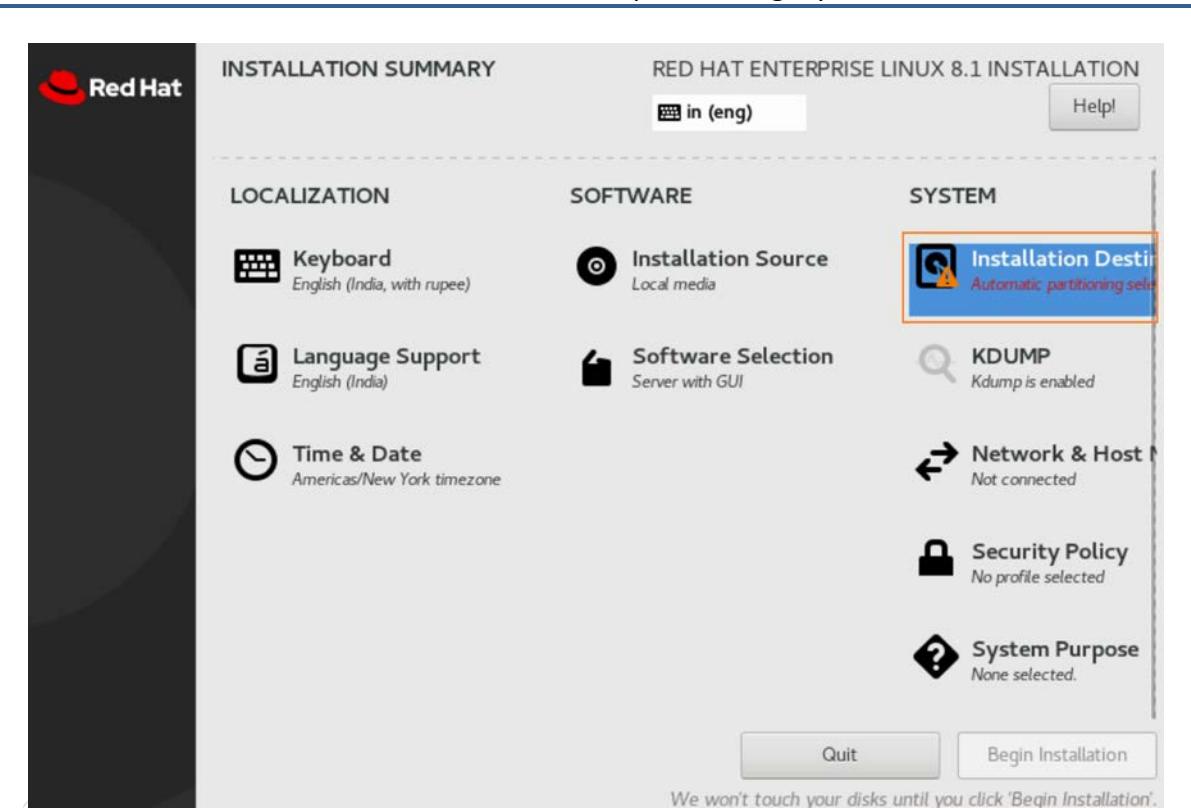
- Select **Software Selection** for selecting software to install during installation



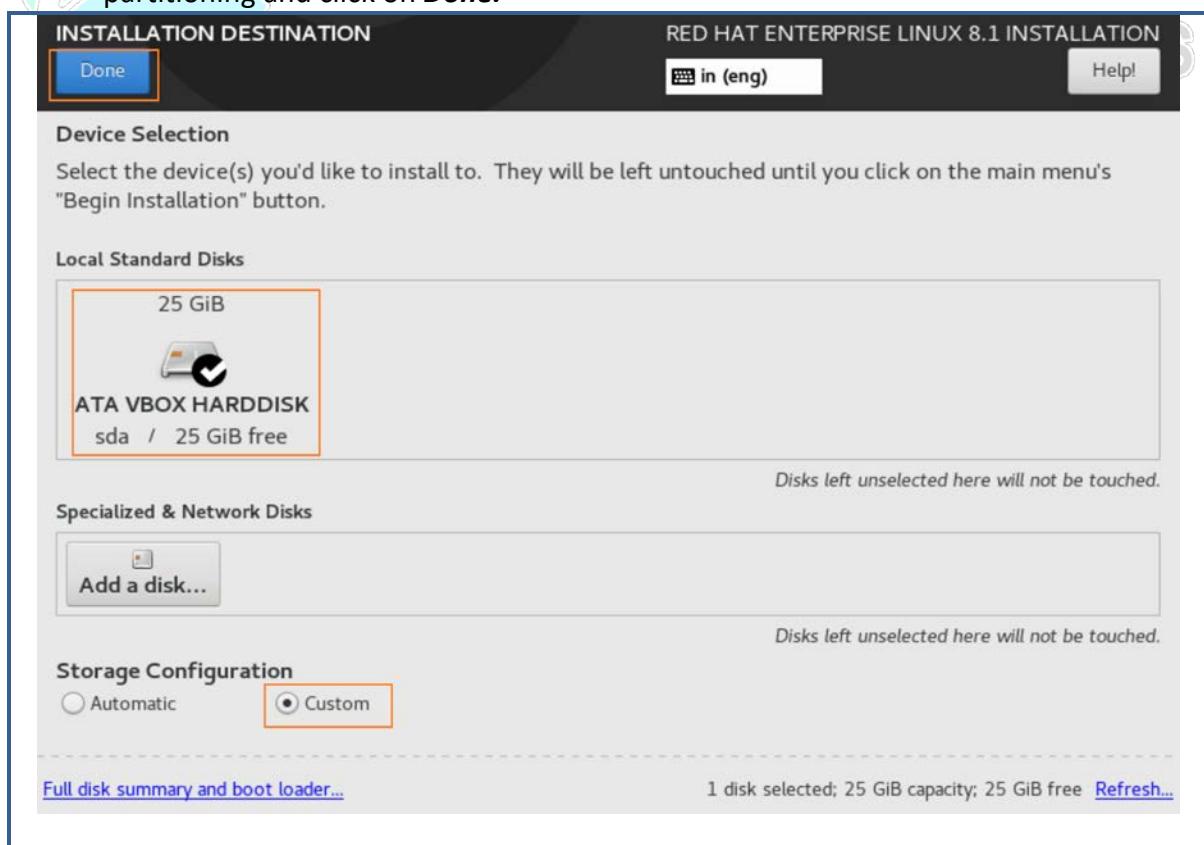
- Select appropriate environment for installation and click on **done**



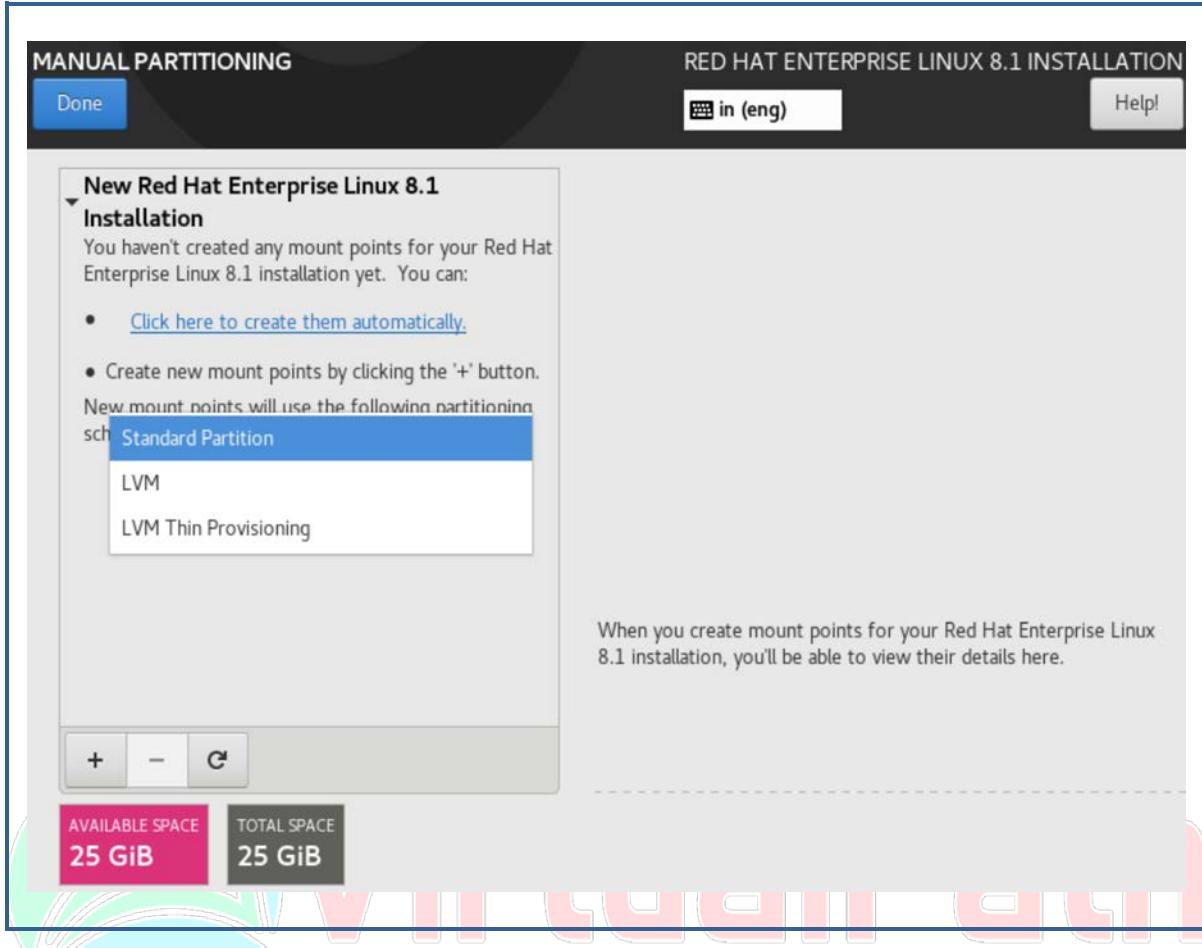
- Click on **INSTALLATION DESTINATION** for partitioning layout



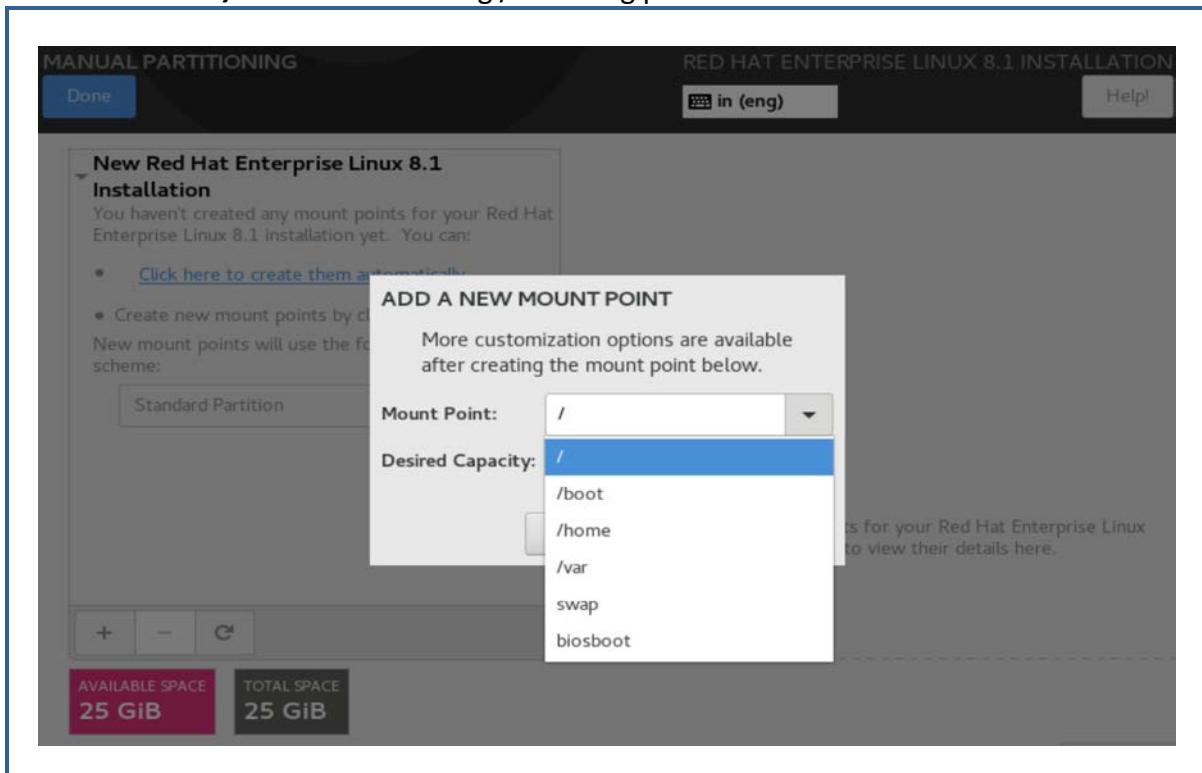
- Select the Disk which you want to use for installation, Select **Custom**, for manual partitioning and click on **Done**.



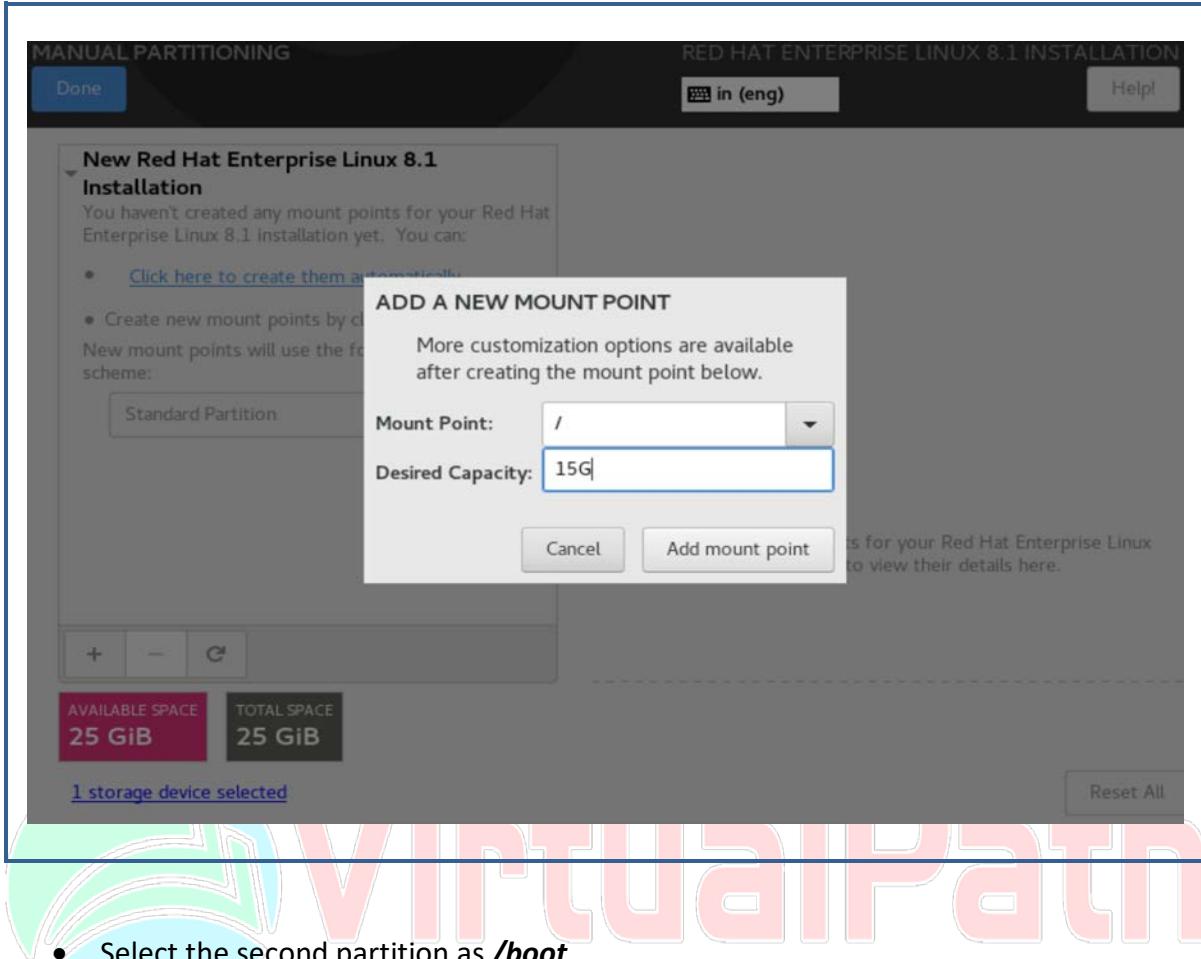
- Select the style of partitioning type preferable



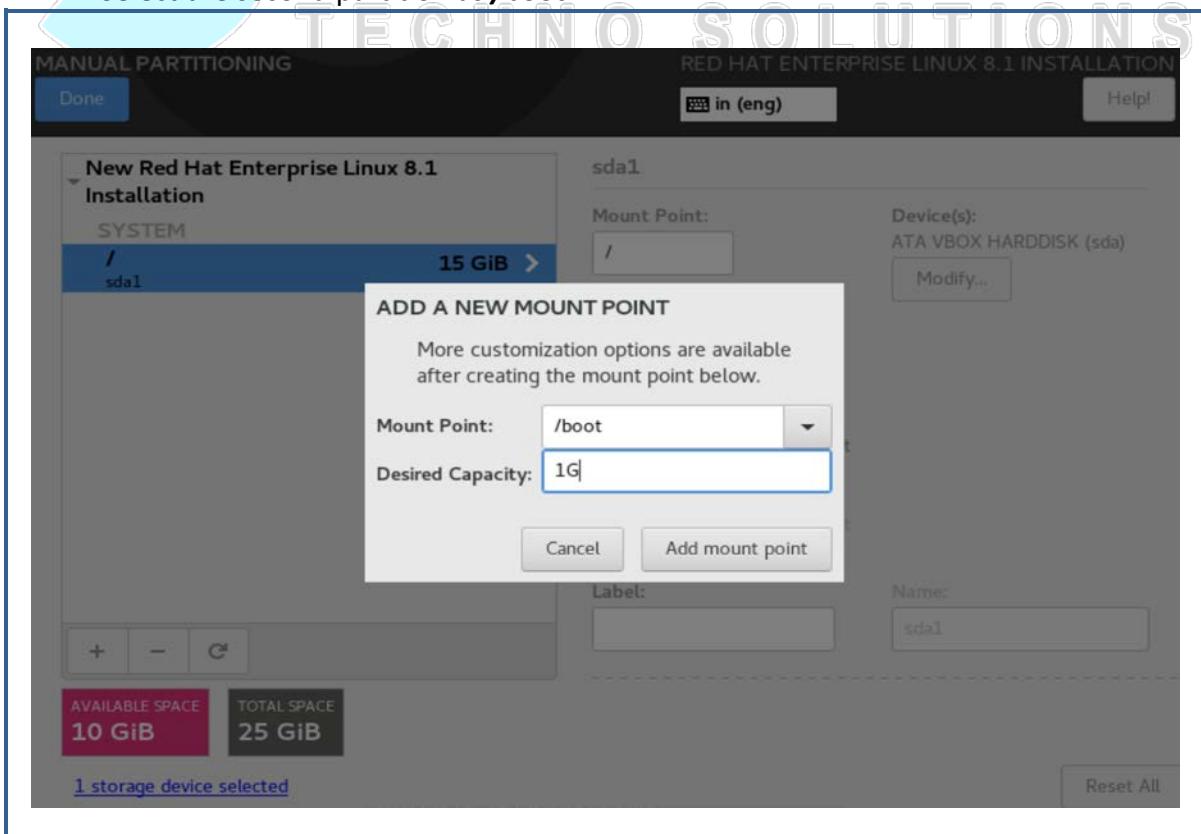
- Click on +/- button for adding /removing partitions



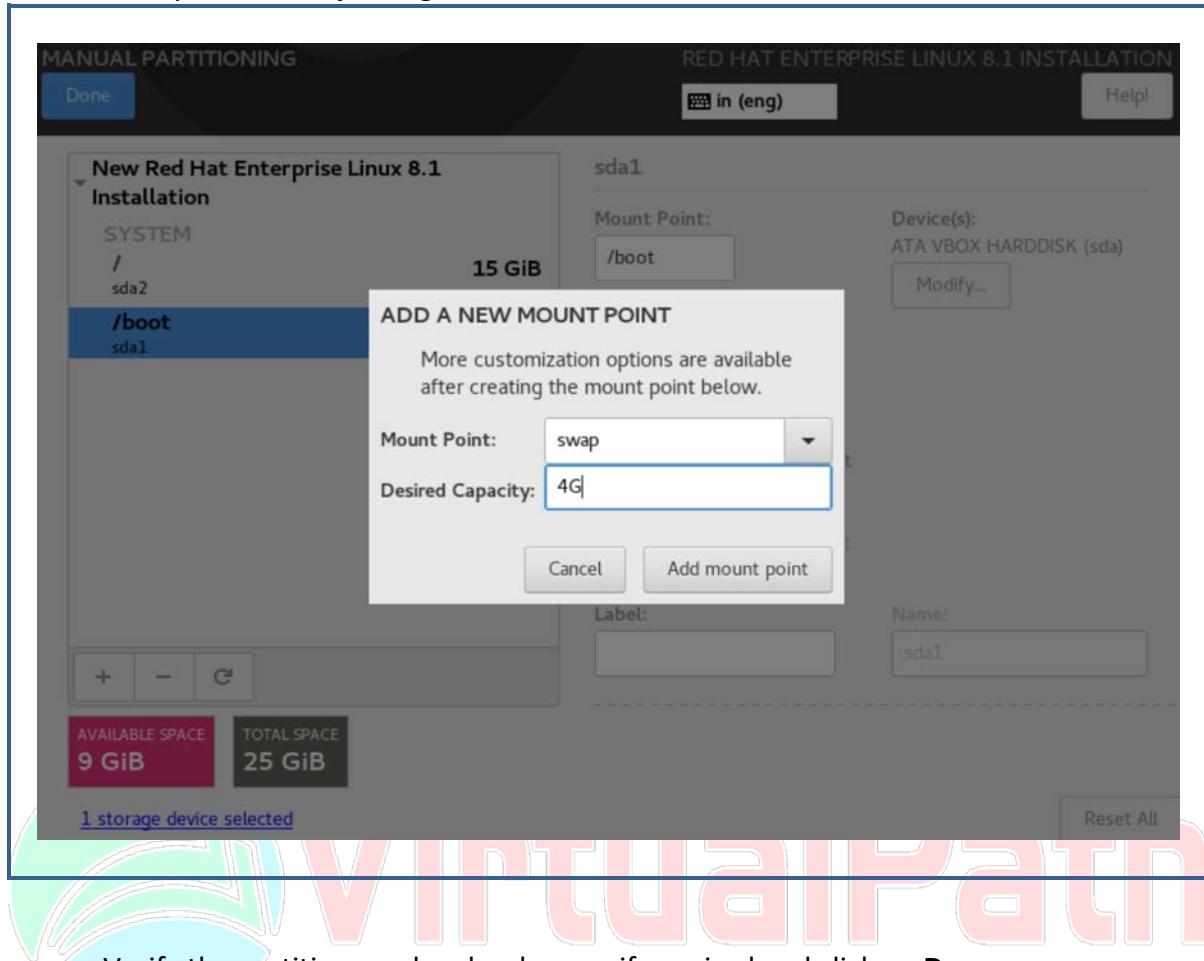
- Select **/** for root partition and assign required size in KB, MB, GB



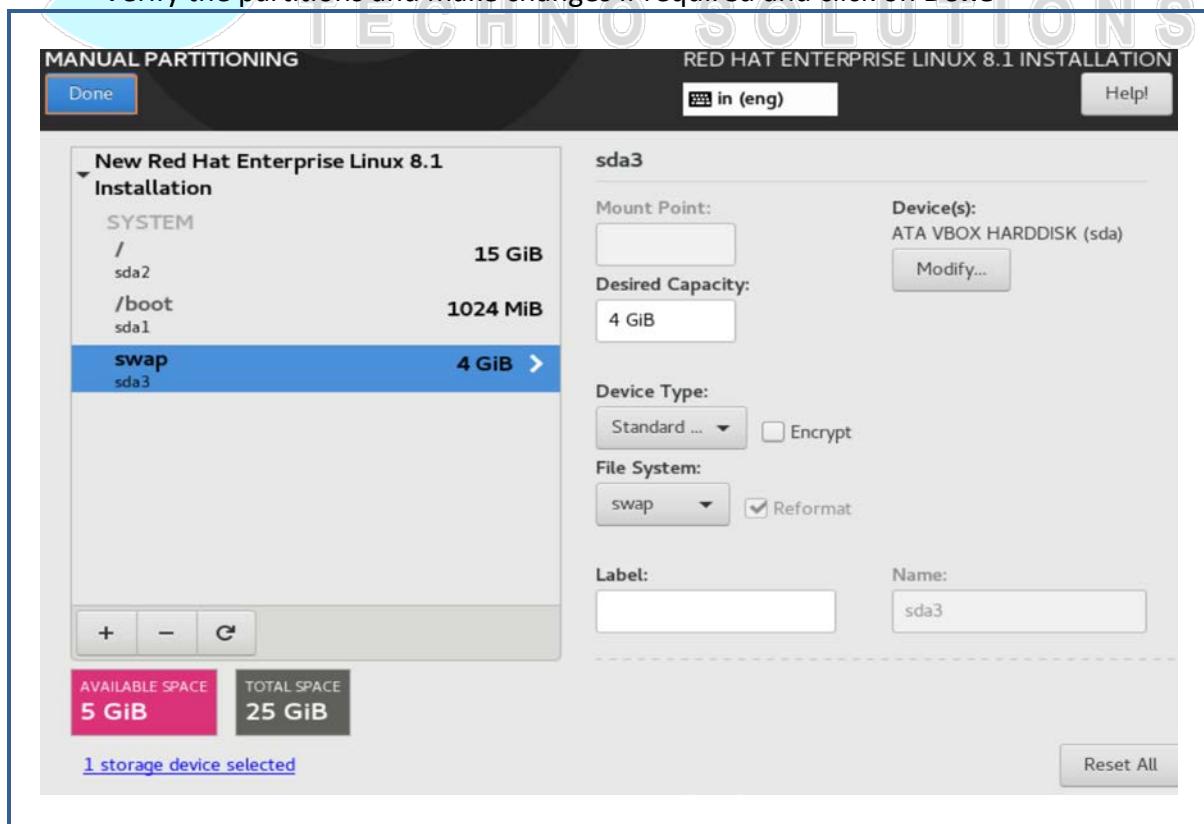
- Select the second partition as **/boot**



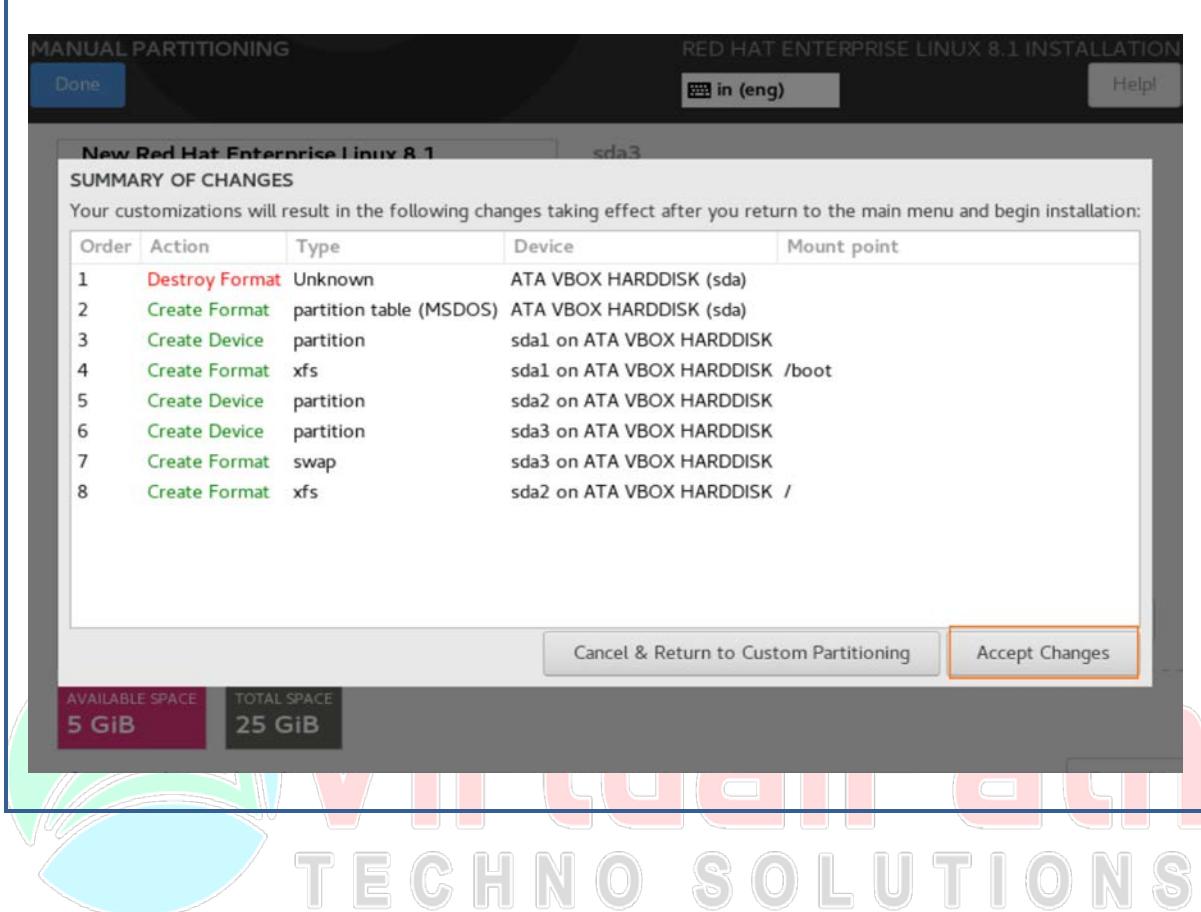
- Finally, select **swap** and give the size



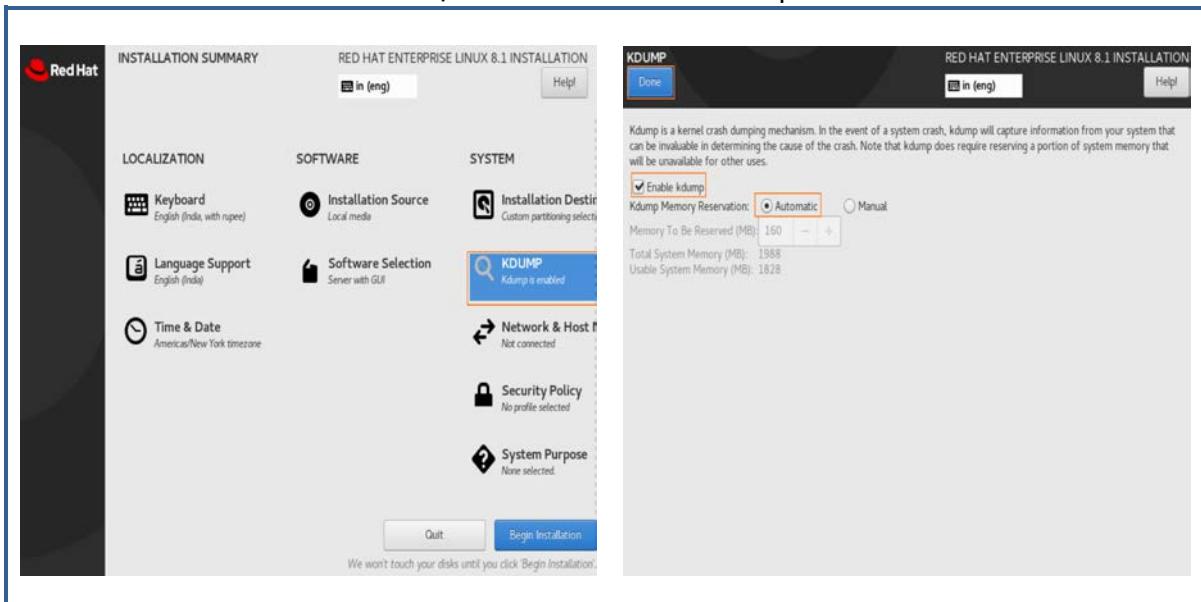
- Verify the partitions and make changes if required and click on **Done**



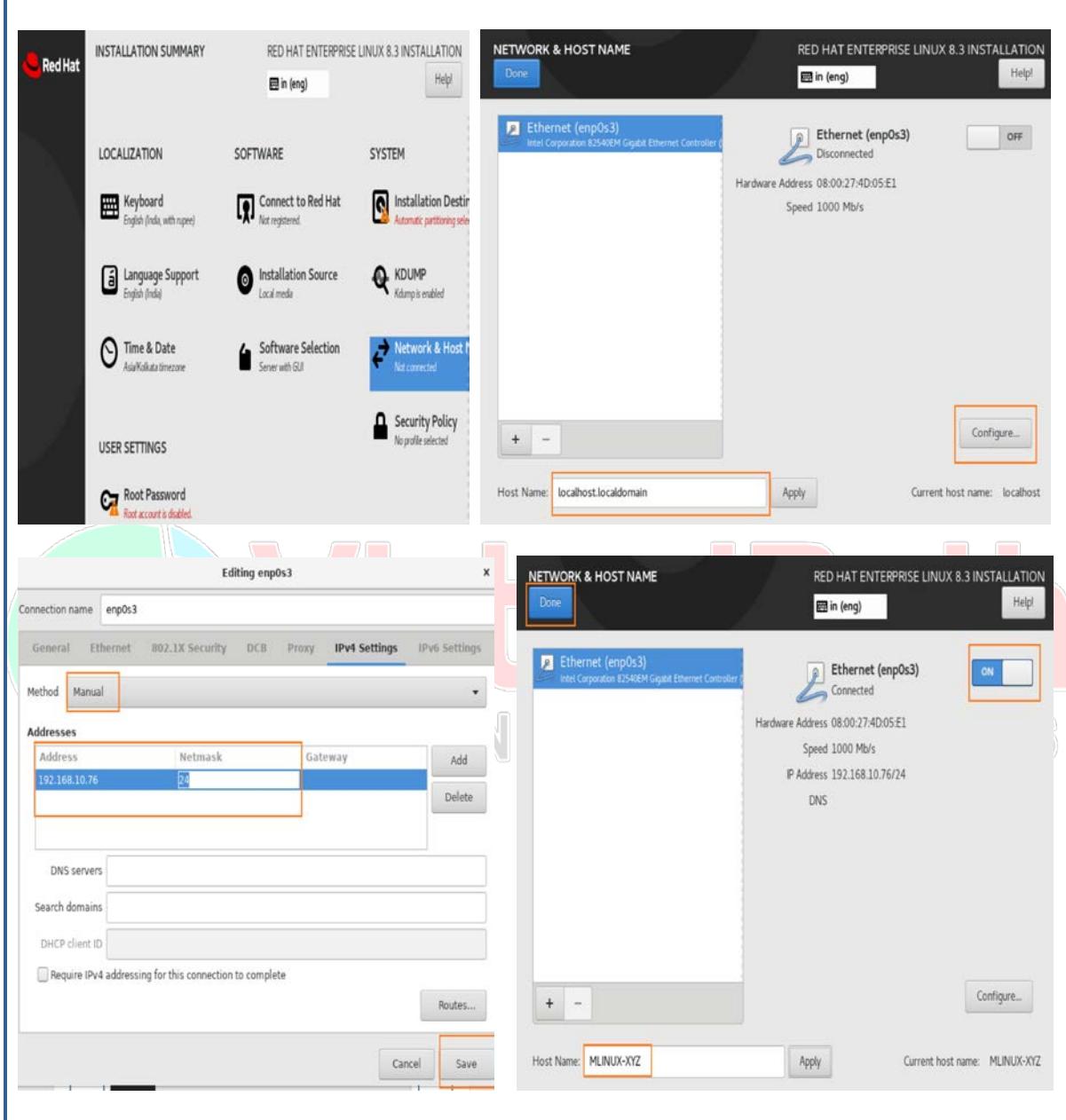
- Accept the changes for continue



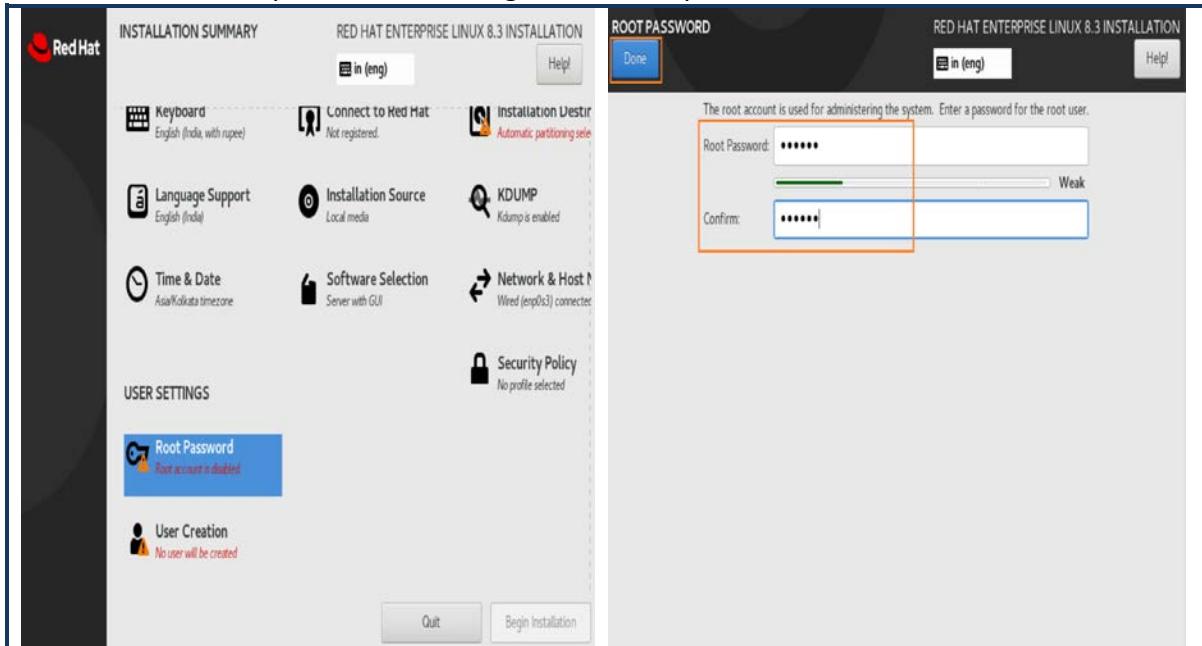
- Click on **KDUMP** to enable/disable kernel crash dump mechanism.



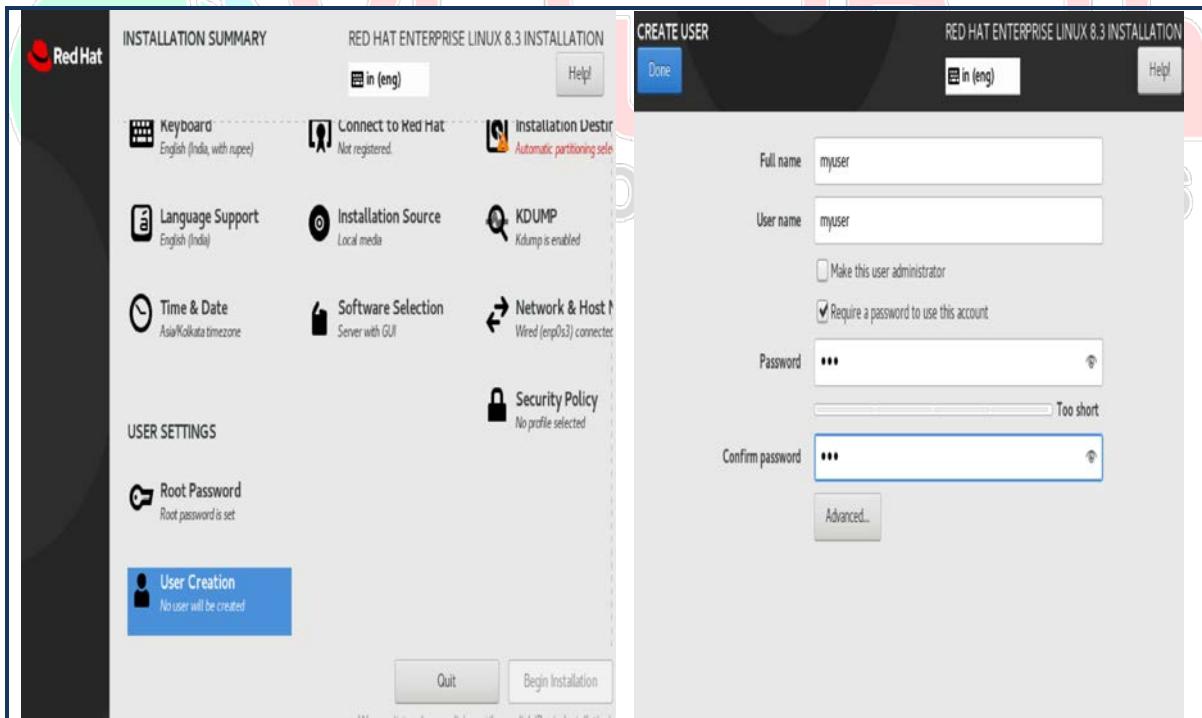
- Click on **Network&Hostname** to set IP address, which can be skipped and set later also.



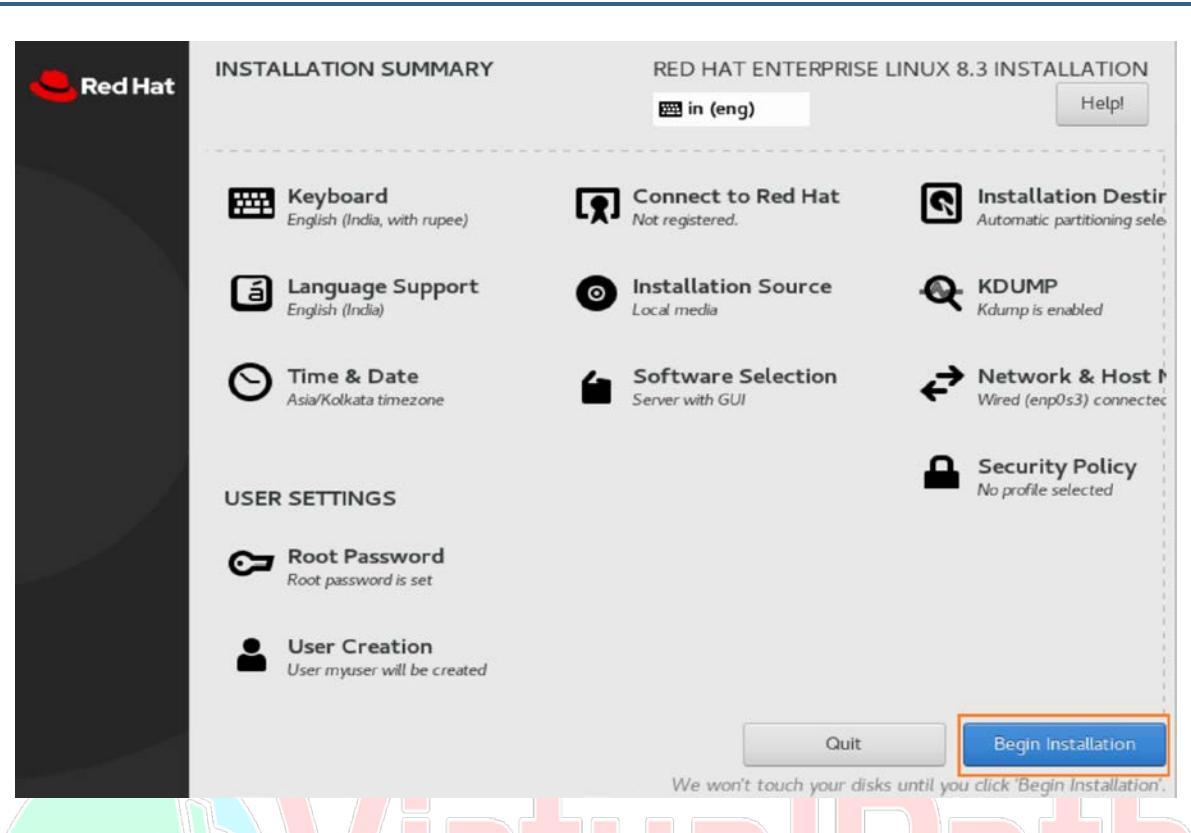
- Click on root password to assing admin level password for root user



- Click on User creation to create an additional normal user



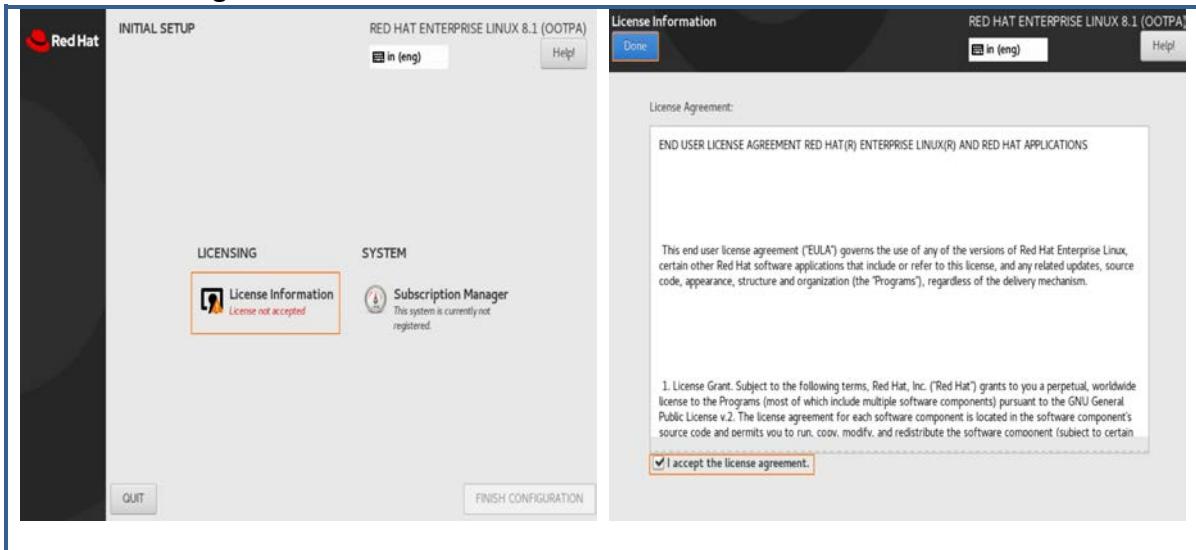
- Finally, Click on ***Begin Installation*** to start the installation



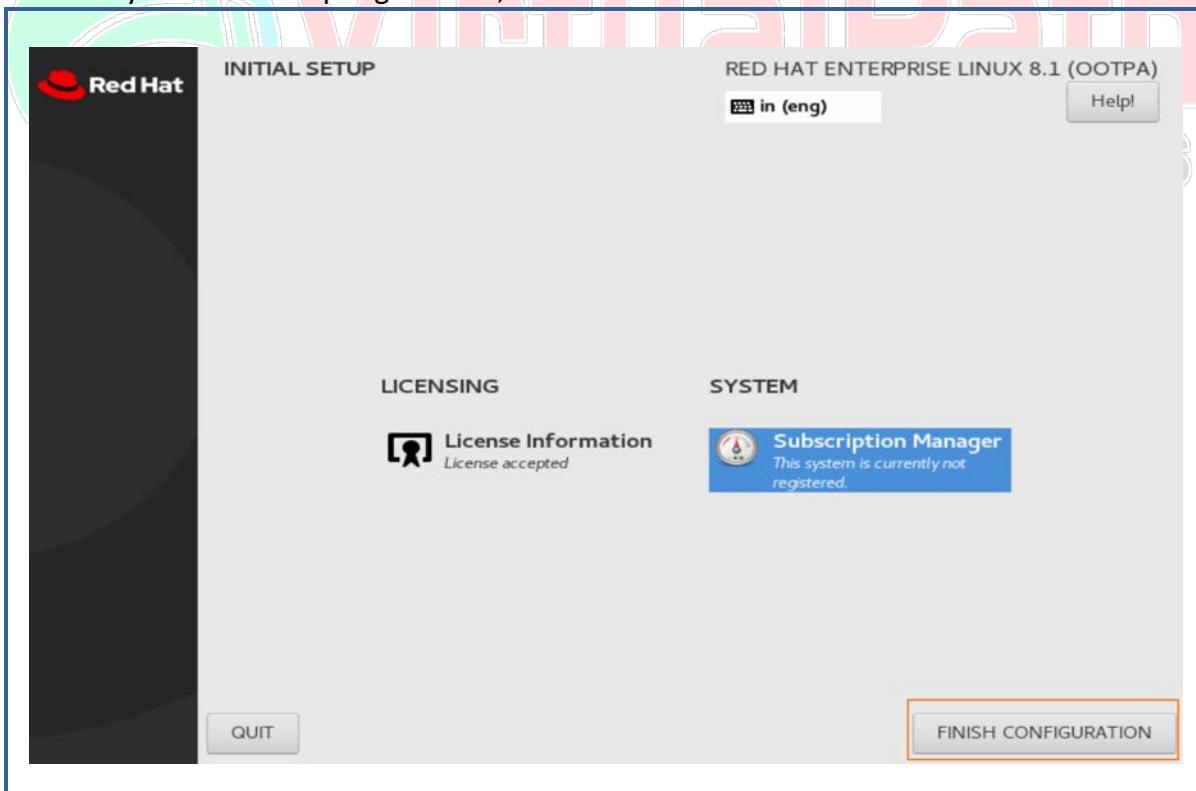
- Once the installation is completed, reboot the machine while clicking on ***Reboot***



- After booting first time, click on **LICENSE INFORMATION** to see and accept the license agreement.



- Read and Accept the EUL Agreement and click on done
- If you have subscription, register your machine with **Redhat Network** for updates and support.
- If you want to skip registration, click on **FINISH CONFIGURATION**



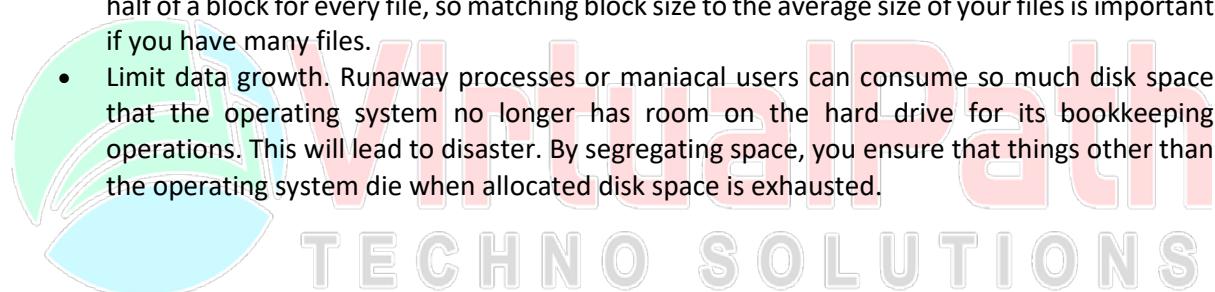
MANAGING PARTITIONS & FILE SYSTEMS

What is a partition?

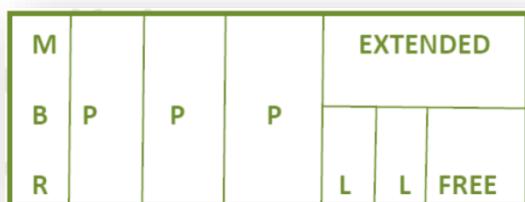
Partitioning is a means to divide a single hard drive into many logical drives. A partition is a contiguous set of blocks on a drive that are treated as an independent disk. A partition table is an index that relates sections of the hard drive to partitions.

Why have multiple partitions?

- Encapsulate your data. Since file system corruption is local to a partition, you stand to lose only some of your data if an accident occurs.
- Increase disk space efficiency. You can format partitions with varying block sizes, depending on your usage. If your data is in a large number of small files (less than 1k) and your partition uses 4k sized blocks, you are wasting 3k for every file. In general, you waste on average one half of a block for every file, so matching block size to the average size of your files is important if you have many files.
- Limit data growth. Runaway processes or maniacal users can consume so much disk space that the operating system no longer has room on the hard drive for its bookkeeping operations. This will lead to disaster. By segregating space, you ensure that things other than the operating system die when allocated disk space is exhausted.



Disk Partitioning Criteria:



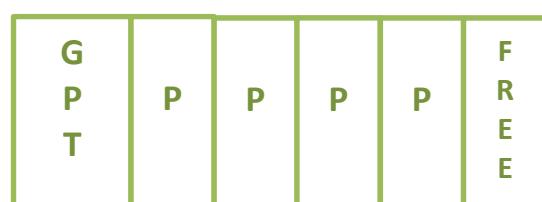
MBR = MASTER BOOT RECORD

P= PRIMARY PARTITION

EXTENDED= EXTENDED PARTITION

L= LOGICAL PARTITION

FREE= FREE SPACE



GPT = GUID PARTITION TABLE

P= PARTITION

FREE= FREE SPACE

The Structure of Disk Partition

- On the disk where O/S is installed, will have the first partition as **MBR/GPT**.
- **MBR** is a Master Boot Record, which contains two important utilities, **IPL** (Initial Program Loader) and **PTI** (Partition Table information). Which supports up to 2TB of disk size
- **GPT** is **GUID Partition Table**, which also contains **IPL** as well as **PTI**, supports up to **2 Zettabyte (1024 EB (Exabyte)=1 ZB (zettabyte))**.
- **IPL** is responsible for booting the operating system, because it contains the **boot loader**.
- In earlier versions of Linux i.e. up to **RHEL 4**, the default boot loader was **LILO** (Linux Loader). But, since **RHEL5** onwards it has been changed to **GRub** (Grand Unified Boot loader), which is far more superior to **LILO**. In **RHEL 7 GRub2** has been introduced.
- The **PTI** (Partition Table information) is the information about the number of partitions on the disk, sizes of the partition and types of partitions.

THE CRITERIA OF DISK PARTITIONING IN MBR:

- Every disk can have only **4 Primary Partitions** (3 Primary + 1 Extended).
- **Primary Partition** is a partition which usually holds the **operating system**.
- **Extended Partition** is a special type of primary partition which can be subdivided into multiple logical partitions. As there can be only 3 primary partitions per disk, and if the user is required to make further partitions then all the space remaining on the disk should be allocated to extended partition, which can be used to create the logical partitions later. There can be only **one extended partition** per disk.
- With the help of extended partition, you can create up to 60 partitions, but with *scsi* disk, you can create max up to 16 max no. of partitions
- **Logical partitions** are the partitions which are created under extended partition, all the space in the extended partition can be used to create any number of logical partitions.

THE CRITERIA OF DISK PARTITIONING IN GPT:

- It supports up to 128 partitions per disk

Disk Identification:

Different type of disks will be having different initials in Linux

- **IDE** drive will be shown as **/dev/hd** (ex: **hda, hdb, hdc**)
- **SCSI, SATA etc.**, drive will be shown as **/dev/sd** (ex: **sda, sdb, sdc**)

FILE SYSTEM:

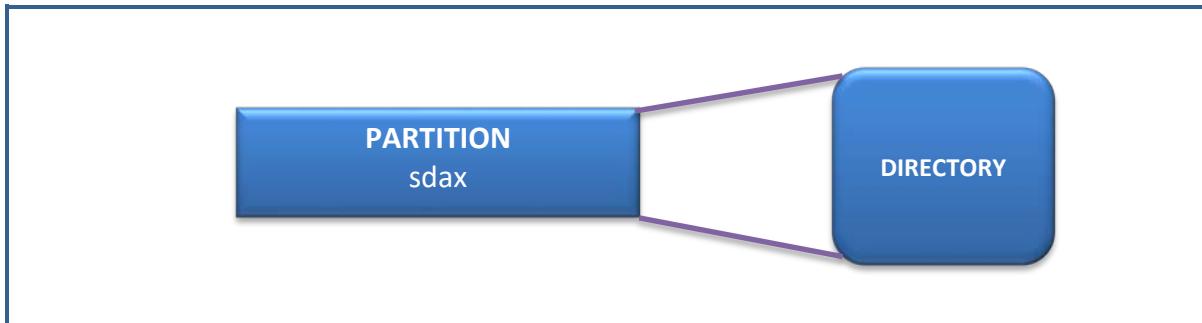
- It is method of storing the data in an organized fashion on the disk. Every partition on the disk except **MBR** and **Extended partition** should be assigned with some file system in order to make them store the data. File system is applied on the partition by formatting it with a particular type of file system.

Types of file systems supported in RHEL 6,7 & 8:

- The file systems supported in Linux are **ext2, ext3 ext4, vfat xfs** in RHEL 6, 7&8 etc.
- Ext/xfs** file system is the widely used file system in Linux, whereas vfat is the file system to maintain a common storage between **Linux and windows** (in case of multiple o/s)

S.NO	EXT2	EXT3	EXT4	XFS
1.	Stands for Second Extended File System	Stands for Third Extended File System	Stands for Fourth Extended File System	Xtents File system/ X File System
2.	It was introduced in 1993	It was introduced in 2001	It was introduced in 2008.	It was introduced in 1993.
3.	Does not have journaling feature.	Supports Journaling Feature.	Supports Journaling Feature.	Supports Journaling Feature.
4.	Maximum File size can be from 16 GB to 2 TB	Maximum File Size can be from 16 GB to 2 TB	Maximum File Size can be from 16 GB to 2 TB	Maximum File Size can be from 16TB to 8 EB
5.	Maximum ext2 file system size can be from 2 TB to 32 TB	Maximum ext3 file system size can be from 2 TB to 32 TB	Maximum ext4 file system size is 1 EB (Exabyte) . 1 EB = 1024 PB (Petabyte) . 1 PB = 1024 TB (Terabyte) .	Maximum xfs file system size is 16 EB (Exabyte) . 1 EB = 1024 PB (Petabyte) . 1 PB = 1024 TB (Terabyte) .
6.	Cannot convert ext file system to ext2.	You can convert an ext2 file system to ext3 file system directly (without backup/restore).	All previous ext file systems can easily be converted into ext4 file system. You can also mount an existing ext3 f/s as ext4 f/s (without having to upgrade it).	N/A

MOUNTING:-



- It is a method of attaching a directory to the file system in order to access the partition and its file system is known as mounting.
- The mount point is the directory (usually an empty one) in the currently accessible file system to which an additional file system is mounted.
- The /mnt directory exists by default on all Unix-like systems. Its, or usually its subdirectories (such as /mnt/floppy and /mnt/usb), are intended specifically for use as mount points for removable media such as CDROMs, USB key drives and floppy disks.

Files which is related to mounting in Linux:

- **/etc/mtab** is a file which stores the information of all the currently mounted file systems; it is dynamic and keeps changing.
- **/etc/fstab** is the file which keeps information about the permanent mount point. If you want to make your mount point permanent, so that it will be mounted even after reboot, then you need to make an appropriate entry in this file.

LAB WORK:-

To view the existing partitions (RHEL6)

#fdisk -l or parted -l

```
[root@ linux1 ~]# fdisk -l

Disk /dev/sda: 42.9 GB, 42949672960 bytes
255 heads, 63 sectors/track, 5221 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000405e4

      Device Boot      Start        End      Blocks   Id  System
/dev/sda1  *           1          26       204800   83  Linux
Partition 1 does not end on cylinder boundary.
/dev/sda2            26         2576     20480000   83  Linux
/dev/sda3            2576        2837     2102378+   82  Linux swap / Solaris

[root@ linux1 ~]# parted -l
Model: ATA QEMU HARDDISK (scsi)
Disk /dev/sda: 42.9GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start    End     Size   Type      File system    Flags
 1      1049kB  211MB   210MB  primary   ext4          boot
 2      211MB   21.2GB  21.0GB  primary   ext4
 3      21.2GB  23.3GB  2153MB primary   linux-swap(v1)
```

Note: Observe in the above picture that the device name is **/dev/sda**.

To view the existing partitions (RHEL7/8)

#fdisk -l or parted -l

```
[root@ rh7 ~]# fdisk -l
Disk /dev/sda: 85.9 GB, 85899345920 bytes, 167772160 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: 0x00082084

Device Boot Start End Blocks Id System
/dev/sda1 * 2048 1026047 512000 83 Linux
/dev/sda2 1026048 52226047 25600000 83 Linux
/dev/sda3 52226048 56420351 2097152 82 Linux swap / Solaris

[root@ rh7 ~]# parted -l
Model: ATA QEMU HARDDISK (scsi)
Disk /dev/sda: 85.9GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number Start End Size Type File system Flags
1 1049kB 525MB 524MB primary xfs boot
2 525MB 26.7GB 26.2GB primary xfs
3 26.7GB 28.9GB 2147MB primary linux-swap(v1)
```

Partition Administration using fdisk

To enter into disk utility, the syntax is

#fdisk <disk name>

#fdisk /dev/sda

```
[root@ktcl5 Desktop]# fdisk /dev/sda
```

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to switch off the mode (command 'c') and change display units to sectors (command 'u').

Command (m for help):

Command action

- a toggle a bootable flag
- b edit bsd disklabel
- c toggle the dos compatibility flag
- d delete a partition
- l list known partition types
- m print this menu
- n add a new partition
- o create a new empty DOS partition table
- p print the partition table
- q quit without saving changes
- s create a new empty Sun disklabel
- t change a partition's system id
- u change display/entry units
- v verify the partition table
- w write table to disk and exit
- x extra functionality (experts only)

Command (m for help):

- Use m to list out various options that can be used in fdisk.

Creating a new partition

#fdisk /dev/sda

- Use **p** to list out the partition information first and
- Use **n** to create a new partition.

```
[root@ tcl5 Desktop]# fdisk /dev/sda

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
switch off the mode (command 'c') and change display units to
sectors (command 'u').

Command (m for help): p

Disk /dev/sda: 32.2 GB, 32212254720 bytes
64 heads, 32 sectors/track, 30720 cylinders
Units = cylinders of 2048 * 512 = 1048576 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00090a50

      Device Boot      Start        End      Blocks   Id  System
/dev/sda1  *           2         201     204800   83  Linux
Partition 1 does not end on cylinder boundary.
/dev/sda2            202        8201    8192000   83  Linux
Partition 2 does not end on cylinder boundary.
/dev/sda3            8202       12201    4096000   83  Linux
Partition 3 does not end on cylinder boundary.
/dev/sda4            12202      30720    18963456    5  Extended
Partition 4 does not end on cylinder boundary.
/dev/sda5            12204      15203    3072000   83  Linux
/dev/sda6            15205      17204    2048000   82  Linux swap / Solaris

Command (m for help): q
```

Now use n to create a new partition and verify it again with p.

```
Command (m for help): n
First cylinder (12202-30720, default 12202): 17205
Last cylinder, +cyinders or +size{K,M,G} (17205-30720, default 30720): +500M

Command (m for help): p

Disk /dev/sda: 32.2 GB, 32212254720 bytes
64 heads, 32 sectors/track, 30720 cylinders
Units = cylinders of 2048 * 512 = 1048576 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00090a50

      Device Boot      Start        End      Blocks   Id  System
/dev/sda1  *           2         201     204800   83  Linux
Partition 1 does not end on cylinder boundary.
/dev/sda2            202        8201    8192000   83  Linux
Partition 2 does not end on cylinder boundary.
/dev/sda3            8202       12201    4096000   83  Linux
Partition 3 does not end on cylinder boundary.
/dev/sda4            12202      30720    18963456    5  Extended
Partition 4 does not end on cylinder boundary.
/dev/sda5            12204      15203    3072000   83  Linux
/dev/sda6            15205      17204    2048000   82  Linux swap / Solaris
/dev/sda7            17205      17705    513008   83  Linux
```

Deleting a partition

Let's delete the partition we've created above i.e. /dev/sda7

- Use **d** to delete a partition and specify the device name, in our case it is **7**.

```
Command (m for help): d
Partition number (1-7): 7
```

Note: Never delete the system partitions i.e. **1-3**

Saving the partition changes

Every time you make a partition or delete a partition, the changes made has to be saved using **w**, otherwise the creation and deletion will not be considered to be happen. For practice purpose you can make any no. of partition and delete it and just quit using **q** so that it will not be saved.

```
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.
[root@ cl5 Desktop]#
```

Updating the partition table without restarting the system

After creating or deleting a partition the changes will be effected in the partition table only after the restart of the system. But there is a way to avoid this circumstance. We can use **partprobe** or **partx** command to update the partition information without restarting the system. **Note: in RHEL8 it is not needed to manually update, as it gets update by default.**

```
#partprobe /dev/sda
Or
#partx -a /dev/sda (while adding), #partx -d /dev/sda (while deleting)
Or
#kpartx /dev/sda
```

Note: In **RHEL6** **partprobe** is not functioning properly, so it is recommended to use **partx**, whereas in **RHEL7** it is functioning perfectly.

- Read the following file to check status of partition table
- **#cat /proc/partitions**

RHEL6

```
[root@musabl ~]# partx -a /dev/sda
BLKPG: Device or resource busy
error adding partition 1
BLKPG: Device or resource busy
error adding partition 2
BLKPG: Device or resource busy
error adding partition 3
[root@musabl ~]# cat /proc/partitions
major minor #blocks name
8 0 41943040 sda
8 1 204800 sda1
8 2 25600000 sda2
8 3 2097152 sda3
8 4 31 sda4
8 5 515977 sda5
```

RHEL7

```
[root@ rh7 ~]#
[root@ rh7 ~]# partprobe /dev/sda
[root@ rh7 ~]# cat /proc/partitions
major minor #blocks name
11 0 3655680 sr0
8 0 83886080 sda
8 1 512000 sda1
8 2 25600000 sda2
8 3 2097152 sda3
8 4 1 sda4
8 5 512000 sda5
```

"Now then we have learnt creating a partition. Let's see how to format a partition with a particular file system"

Formatting a partition with ext4 filesystem in rhel6, 7 or 8

After creating a partition we need to assign some file system to it so that we can start storing the data into it. To format a partition the following syntax is used.

```
# mkfs.<file system type> <partition name>
#mkfs.ext4 /dev/sda7 (where sda7 is our newly created partition)
```

```
[root@mlinux6 ~]# mkfs.ext4 /dev/sda5
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
129024 inodes, 515976 blocks
25798 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67633152
63 block groups
8192 blocks per group, 8192 fragments per group
2048 inodes per group
Superblock backups stored on blocks:
    8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409
```

- Likewise you can format the different partitions with different file systems like
- #mkfs.ext3 /dev/sdax
- #mkfs.vfat /dev/sdax

Formatting a partition with xfs in rhel7

```
#mkfs.xfs /dev/sda5
```

```
[root@ rh7 ~]# mkfs.xfs /dev/sda5
meta-data=/dev/sda5              isize=256      agcount=4, agsize=32000 blks
                                =           sectsz=512   attr=2, projid32bit=1
                                =
                                =
data     =                         bsize=4096   blocks=128000, imaxpct=25
                                =
                                =
naming  =version 2               bsize=4096   ascii-ci=0 ftype=0
log     =internal log            bsize=4096   blocks=853, version=2
                                =
realtime =none                  sectsz=512   sunit=0 blks, lazy-count=1
                                extsz=4096   blocks=0, rtextents=0
```

Note: Even after formatting the partition, we cannot add the data into the partition. In order to add the data in the partition it is required to be mounted.

Mounting a partition

Mounting is a procedure where we attach a directory to the file system. There are two types of mounting which will be used in Linux or any UNIX.

- **Temporary Mounting**
- **Permanent Mounting**

Temporary Mounting

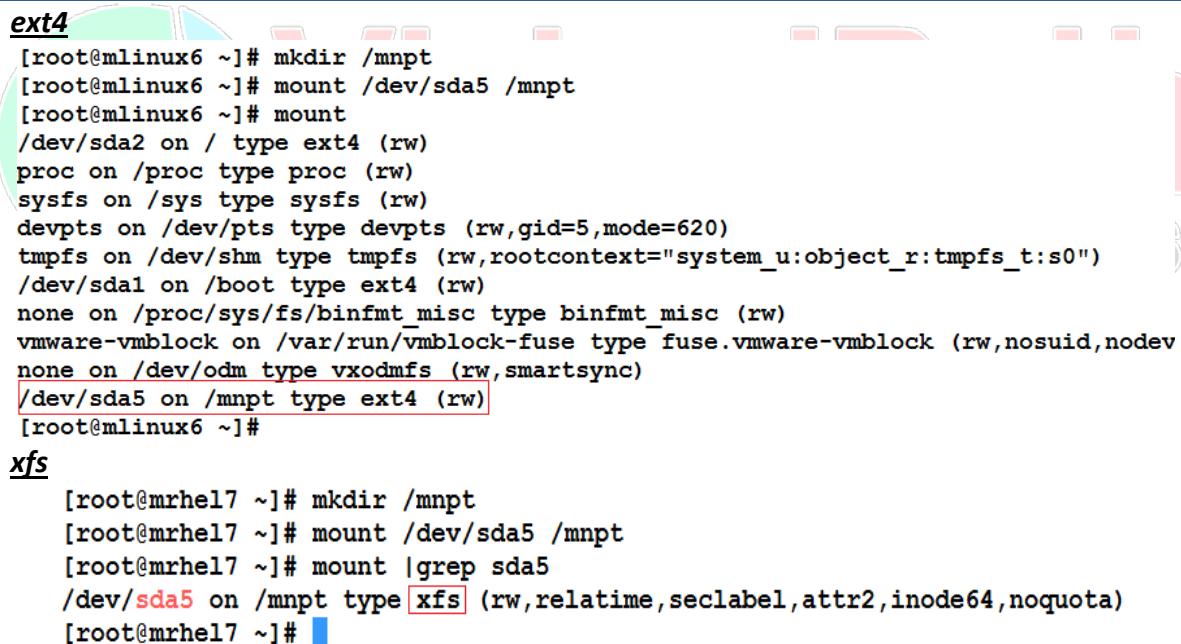
In a temporary mount point we will create a directory and mount it, but this mount point will last only till the system is up, once it is rebooted the mounting will be lost.

Syntax:

```
#mount <device name> <directory name (mount point)>
#mount /dev/sda5 /mnpt
[root@mrhel7 ~]# mkdir /mnpt
[root@mrhel7 ~]# mount /dev/sda5 /mnpt
```

To View all the mounted partitions

#mount



```
ext4
[root@mlinux6 ~]# mkdir /mnpt
[root@mlinux6 ~]# mount /dev/sda5 /mnpt
[root@mlinux6 ~]# mount
/dev/sda2 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw,rootcontext="system_u:object_r:tmpfs_t:s0")
/dev/sda1 on /boot type ext4 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
vmware-vmblock on /var/run/vmblock-fuse type fuse.vmware-vmblock (rw,nosuid,nodev
none on /dev/odm type vxodmfs (rw,SMARTsync)
/dev/sda5 on /mnpt type ext4 (rw)
[root@mlinux6 ~]#
xfs
[root@mrhel7 ~]# mkdir /mnpt
[root@mrhel7 ~]# mount /dev/sda5 /mnpt
[root@mrhel7 ~]# mount |grep sda5
/dev/sda5 on /mnpt type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
[root@mrhel7 ~]#
```

- Now we have successfully mounted the partition we can access it and can store the data
- To add the data access the mount point
- #cd /mnpt. Now, add the data and exit the directory

Unmounting a partition

#umount <mount point directory>

#umount /mnpt

Verify it with **mount** command.

Permanent Mounting

Permanent mounting procedure is exactly same like temp mounting, but here we will update the **/etc/fstab** file with the mounting details, so that it will be mounted even after the system is reboot.

Steps To make a permanent mount point:

- Make a directory or use an existing directory
- Add entry in **/etc/fstab** file
- Use **mount -a** command to check it is mounting. (**mount -a** will mount all the entry placed in **/etc/fstab**)

Here we will be using our existing **/kernel** directory as mount point which is created previously.

#vim **/etc/fstab**

#	# /etc/fstab	# Created by anaconda on Wed Sep 9 05:29:56 2015	#	# 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	#
	UUID=f25262b9-f44f-42fb-8cba-663b1e77cd83	/		xfs	defaults	1 1
	UUID=bce2a1cf-2138-4570-8180-e13c710fbfb6	/boot		xfs	defaults	1 2
	UUID=d43b526a-93d0-4523-9499-b22e50c90329	swap		swap	defaults	0 0
	/dev/sda5	/mnpt		ext4	defaults	0 0
	/dev/sda6	/mnpt2		xfs	defaults	0 0



Note: For xfs, you can put xfs in place ext4

#mount -a

```
[root@mlinux7 ~]# mount -a
[root@mlinux7 ~]# mount |grep sda*
/dev/sda2 on / type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
nfsd on /proc/fs/nfsd type nfsd (rw,relatime)
/dev/sda1 on /boot type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
gvfsd-fuse on /run/user/0/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,rel
/dev/sda5 on /mnpt type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/sda6 on /mnpt2 type ext4 (rw,relatime,seclabel,data=ordered)
[root@mlinux7 ~]#
[root@mlinux7 ~]#
```

You can now access the directory and add, delete or modify the contents and can also unmount the file system at any point

Mounting a partition permanently with its block id (UUID)

- To check the uuid of a partition use **blkid /dev/sda7** command.
- Copy the uuid
- Make an entry in **/etc/fstab** using UUID
- Verify it with **mount -a** option

```
[root@ cl5 ~]# blkid /dev/sda7
/dev/sda7: LABEL= UUID="f489d0b1-ffca-4e21-917a-7b82c0edd255" TYPE="ext4"
[root@ cl5 ~]#
```

#vim /etc/fstab

tmpfs	/dev/shm	tmpfs	defaults	0 0
devpts	/dev/pts	devpts	gid=5,mode=620	0 0
sysfs	/sys	sysfs	defaults	0 0
proc	/proc	proc	defaults	0 0
UUID=f489d0b1-ffca-4e21-917a-7b82c0edd255		/mnpt2	ext4 defaults	0 0

Now mount it with **mount -a** command and verify it with **mount** command

Sometimes a directory reflects error while unmouting, the possible causes for it are

- You are in the same directory and trying to unmount it. Check with **pwd** command
- Some users are present in the directory and using the contents in it.
- Check with **fuser -cu /mnpt or /dev/sda5 and lsof /mnpt or /dev/sda6**

```
[root@mlinux7 ~]# fuser -cu /mnpt
/mnpt: 44673c(myuser) 44710c(myuser)
[root@mlinux7 ~]# lsof /mnpt
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
bash 44673 myuser cwd DIR 8,5 6 128 /mnpt
vim 44710 myuser cwd DIR 8,5 6 128 /mnpt
```

- Kill the open connections using **fuser -ck /mnpt**, you could also try **umount -l** (lazy unmount)
- Now you can use **umount** command to unmount the file system.

To view the usage information of mounted partition:

To view the usage information of mounted partition use the command **df -h**

#df -h,

```
[root@mlinux7 ~]# df -h
Filesystem Size Used Avail Use% Mounted on
/dev/sda2 20G 2.9G 18G 15% /
devtmpfs 899M 0 899M 0% /dev
tmpfs 913M 140K 913M 1% /dev/shm
tmpfs 913M 9.1M 904M 1% /run
tmpfs 913M 0 913M 0% /sys/fs/cgroup
/dev/sda1 497M 154M 344M 31% /boot
tmpfs 183M 32K 183M 1% /run/user/0
/dev/sda5 497M 26M 472M 6% /mnpt
/dev/sda6 190M 1.6M 175M 1% /mnpt2
tmpfs 183M 0 183M 0% /run/user/1000
[root@mlinux7 ~]#
```

To view the size of a file or directory

To view the size of the file or directory uses the command **du -h file or directory name**.

To see more details about partition, FS and uuid of the partition

Also see **#lsblk -l , #lsblk -s, #lsblk -Fs**

PARTITIONING WITH PARTED COMMAND

In order to partitions in Linux we generally use ***fdisk*** command. The concept of ***fdisk*** is to work within the utility. What if one wants to create partitions directly with command line or script?

The answer is ***parted*** command, which allow us to create, remove or modify partitions either within utility or directly on command line.

Let's learn the way to work with parted utility and (or) command line.

To list the disk and partitions

#**parted -l**

```
[root@mlinux21 ~]# parted -l
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 26.8GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system     Flags
 1      1049kB  1075MB  1074MB  primary   xfs             boot
 2      1075MB  17.2GB  16.1GB  primary   xfs
 3      17.2GB  21.5GB  4295MB  primary   linux-swap(v1)
 4      21.5GB  26.8GB  5368MB  extended
 5      21.5GB  22.5GB  1023MB  logical   xfs             lba
```

```
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:
```

Creating the partition with parted utility

#**parted <disk path>**

#**parted /dev/sda**

```
[root@mlinux21 ~]# parted /dev/sda
GNU Parted 3.2
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

Getting help in parted utility

```
(parted) help
align-check TYPE N
help [COMMAND]
mklabel,mktable LABEL-TYPE
mkpart PART-TYPE [FS-TYPE] START END
name NUMBER NAME
print [devices|free|list,all|NUMBER]
    partitions, or a particular partition
quit
rescue START END
resizepart NUMBER END
rm NUMBER
select DEVICE
disk_set FLAG STATE
disk_toggle [FLAG]
set NUMBER FLAG STATE
toggle [NUMBER [FLAG]]
unit UNIT
version
```

align-check TYPE N	check parti
help [COMMAND]	print genera
mklabel,mktable LABEL-TYPE	create a new
mkpart PART-TYPE [FS-TYPE] START END	make a parti
name NUMBER NAME	name partiti
print [devices free list,all NUMBER]	display the
partitions, or a particular partition	
quit	exit program
rescue START END	rescue a los
resizepart NUMBER END	resize parti
rm NUMBER	delete parti
select DEVICE	choose the d
disk_set FLAG STATE	change the F
disk_toggle [FLAG]	toggle the s
set NUMBER FLAG STATE	change the F
toggle [NUMBER [FLAG]]	toggle the s
unit UNIT	set the defa
version	display the

Printing the partitions

```
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 26.8GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1       1049kB  1075MB  1074MB  primary    xfs          boot
 2       1075MB  17.2GB   16.1GB   primary    xfs
 3       17.2GB   21.5GB   4295MB  primary    linux-swap(v1)
 4       21.5GB   26.8GB   5368MB  extended
 5       21.5GB   22.5GB   1023MB  logical    xfs          lba
```

Adding a new partition

```
4       21.5GB  26.8GB  5368MB  extended
5       21.5GB  22.5GB  1023MB  logical    xfs          lba

(parted) mkpart
Partition type? [logical]? Primary/Extended/logical
File system type? [ext2]? Any desired Filesystem
Start? 22.5GB  Size to start with
End? 23GB     size where to end
```

Note: In the above example, we created 500MB partition.

Print and confirm the partition

```
#print (or) p
```

```
(parted) p
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 26.8GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1      1049kB  1075MB  1074MB  primary   xfs          boot
 2      1075MB  17.2GB  16.1GB  primary   xfs
 3      17.2GB  21.5GB  4295MB  primary   linux-swap(v1)
 4      21.5GB  26.8GB  5368MB  extended
 5      21.5GB  22.5GB  1023MB  logical   xfs
 6      22.5GB  23.0GB  499MB   logical   ext2        lba
```

Removing the partition

```
(parted) rm
Partition number? 6
(parted) p
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 26.8GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1      1049kB  1075MB  1074MB  primary   xfs          boot
 2      1075MB  17.2GB  16.1GB  primary   xfs
 3      17.2GB  21.5GB  4295MB  primary   linux-swap(v1)
 4      21.5GB  26.8GB  5368MB  extended
 5      21.5GB  22.5GB  1023MB  logical   xfs        lba
```

Creating a new partition table MBR/GPT on a new disk

```
[root@mlinux21 ~]# parted /dev/sdb
GNU Parted 3.2
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mklabel
New disk label type? gpt (msdos/gpt)
Warning: The existing disk label on /dev/sdb will be destroyed
to continue?
Yes/No? Yes
(parted) p
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:
```

Note: msdos for MBR & gpt for GPT partition table

Creating 500MB partition directly with parted command

```
#parted <disk path> mkpart <partition number> <start val> <end value>
#parted /dev/sdb mkpart 1 1 500MB
```

```
[root@mlinux21 ~]# parted -l |grep -A5 /dev/sdb
Warning: Unable to open /dev/sr0 read-write (Read-only
has been opened read-only.
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:
```

Number	Start	End	Size	File system	Name	Flags
--------	-------	-----	------	-------------	------	-------

```
[root@mlinux21 ~]# parted /dev/sdb mkpart 1 1 500MB
Information: You may need to update /etc/fstab.
```

```
[root@mlinux21 ~]# parted -l |grep -A7 /dev/sdb
Warning: Unable to open /dev/sr0 read-write (Read-only
has been opened read-only.
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:
```

Number	Start	End	Size	File system	Name	Flags
1	1049kB	500MB	499MB	ext4		1

Removing/deleting a partition using parted command

```
#parted <disk path> rm <partition number>
#parted /dev/sdb rm 1
```

```
[root@mlinux21 ~]# parted /dev/sdb rm 1
Information: You may need to update /etc/fstab.
```

```
[root@mlinux21 ~]# parted -l |grep -A7 /dev/sdb
Warning: Unable to open /dev/sr0 read-write (Read-only
has been opened read-only.
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:
```

Number	Start	End	Size	File system	Name	Flags
--------	-------	-----	------	-------------	------	-------

Creating a Primary/Extended/Logical partition in MBR/msdos table disk

```
#parted <disk path> mkpart <type of partition (primary/extended/logical)
```

```
<Start val> <end val>
```

```
#parted /dev/sda mkpart logical 22.5GB 23GB
```

```
[root@mlinux21 ~]# parted -l
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 26.8GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

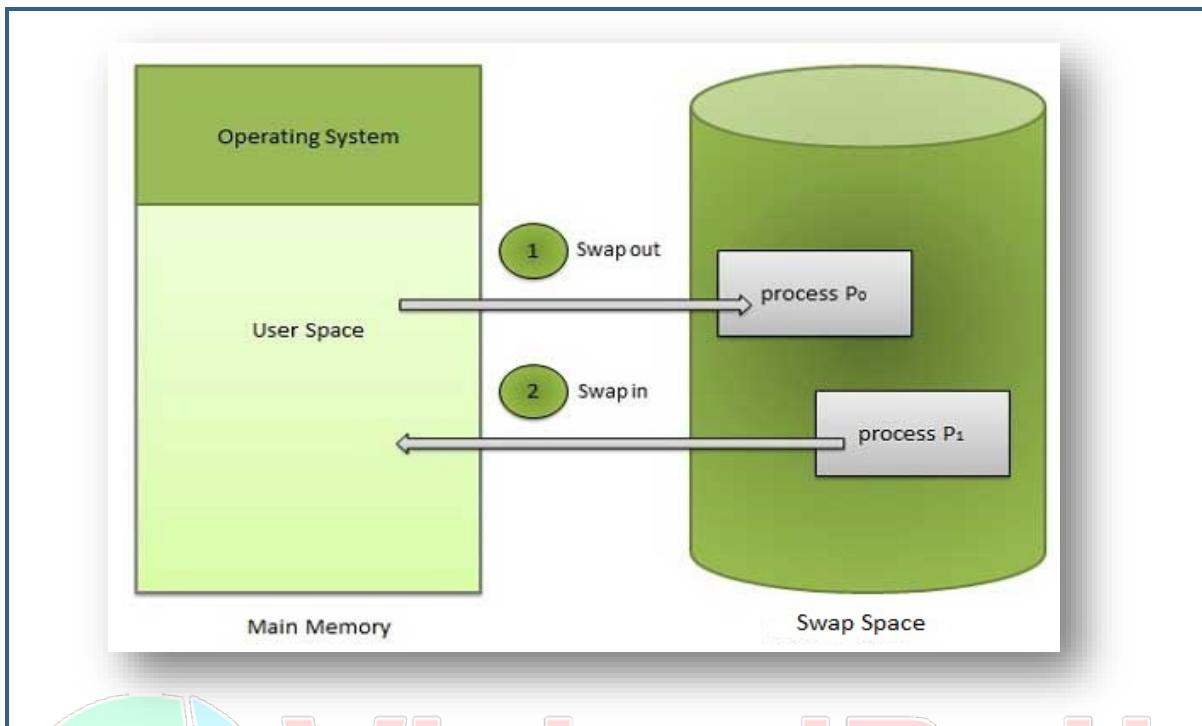
Number  Start   End     Size    Type      File system
 1       1049kB  1075MB  1074MB  primary   xfs
 2       1075MB  17.2GB  16.1GB  primary   xfs
 3       17.2GB  21.5GB  4295MB  primary   linux-swap(v1)
 4       21.5GB  26.8GB  5368MB  extended 
 5       21.5GB  22.5GB  1023MB  logical   xfs
```

```
[root@mlinux21 ~]# parted /dev/sda mkpart logical 22.5GB 23GB
Information: You may need to update /etc/fstab.
```

```
[root@mlinux21 ~]# parted -l
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 26.8GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system
 1       1049kB  1075MB  1074MB  primary   xfs
 2       1075MB  17.2GB  16.1GB  primary   xfs
 3       17.2GB  21.5GB  4295MB  primary   linux-swap(v1)
 4       21.5GB  26.8GB  5368MB  extended 
 5       21.5GB  22.5GB  1023MB  logical   xfs
 6       22.5GB  23.0GB  499MB   logical
```

SWAP SPACES MANAGEMENT



Swap space in Linux is used when the amount of physical memory (RAM) is full. If the system needs more memory resources and the RAM is full, inactive pages in memory are moved to the swap space. While swap space can help machines with a small amount of RAM, it should not be considered a replacement for more RAM. Swap space is located on hard drives, which have a slower access time than physical memory.

Recommended System Swap Space

Amount of RAM in the System	Recommended Amount of Swap Space
4GB of RAM or less	a minimum of 2GB of swap space
4GB to 16GB of RAM	a minimum of 4GB of swap space
16GB to 64GB of RAM	a minimum of 8GB of swap space
64GB to 256GB of RAM	a minimum of 16GB of swap space
256GB to 512GB of RAM	a minimum of 32GB of swap space
512GB TO 1TB of RAM	a minimum of 64 GB of swap space

The Basic Rule for the Size of SWAP:

Apart from the above recommendation a basic rule is applied to create the swap partitions

- if the size of the RAM is **less than or equal to 2GB**, then size of **SWAP=2 X RAM SIZE**
- If the size of the RAM is **more than 2GB**, then size of **SWAP= 2GB + size of the RAM**

Swap space is compulsory to be created at the time of installation. But, additional swap spaces can be created and deleted at any point of time, when it is required. Sometimes we need to increase the swap space, so we create additional swap spaces which will be added to the existing swap space to increase the size.

Commands to be used in maintaining Swap spaces

- To see the memory size and the swap space size
#free -m, #free -h
- To see the swap usage use
#swapon -s
- To format the partition with swap file system use
#mkswap <partition name>
- To activate the swap space use
#swapon <partition name>
- To deactivate the swap space use
#swapoff <partition name>

Creating a Swap partition

- Create a normal partition using fdisk and change hex code to make it swap partition.
- The hex code for SWAP is **82**. (To change the use **t** in fdisk and list all the hex code use **I**)
- Update the partition table using **partprobe** command

```
[root@ linux /]# fdisk /dev/sda

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
switch off the mode (command 'c') and change display units to
sectors (command 'u').

Command (m for help): n
First cylinder (4426-6527, default 4426):
Using default value 4426
Last cylinder, +cylinders or +size{K,M,G} (4426-6527, default 6527): +500M

Command (m for help): t
Partition number (1-6): 6
Hex code (type L to list codes): 82
Changed system type of partition 6 to 82 (Linux swap / Solaris)

Command (m for help): p

Disk /dev/sda: 53.7 GB, 53687091200 bytes
255 heads, 63 sectors/track, 6527 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0006d4e7

      Device Boot   Start     End   Blocks   Id  System
/dev/sda1            1    3825  30720000   8e  Linux LVM
/dev/sda2    *     3825    3851    204800   83  Linux
/dev/sda3     3851    4361   4096000   82  Linux swap / Solaris
/dev/sda4     4361    6527  17406303+   05  Extended
/dev/sda5     4361    4425     521957   83  Linux
/dev/sda6     4426    4490     522081   82  Linux swap / Solaris
```

Format the partition with swap file system

```
#mkswap /dev/sda6
```

```
[root@ linux ]# mkswap /dev/sda6
Setting up swap space version 1, size = 522076 KiB
no label, UUID=d3d25afa-71d6-4339-a88a-8640f2680a74
[root@ linux ]#
```

Turn on the newly created swap space and verify it.

- To turn on the swap space the syntax is

```
#swapon /dev/sda6
```

```
[root@ linux ]# swapon /dev/sda6
[root@ linux ]# swapon -s
Filename                                Type      Size   Used   Priority
/dev/sda3                               partition 4095992 0     -1
/dev/sda6                               partition 522072  0     -2
[root@ linux ]# free -m
              total     used     free   shared   buffers   cached
Mem:       2007      741     1266       0         4      272
-/+ buffers/cache:        464     1543
Swap:      4509      0     4509
```

Making the Newly Created SWAP Partition to be activated after reboot

- In order to make the swap partition mount automatic after reboot, we need to make an entry in **/etc/fstab** file.

```
#vim /etc/fstab
```

```
# /etc/fstab
# Created by anaconda on Wed Nov 10 06:40:21 2010
#
# 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/vg_l... linux-rootlv /          ext4    defaults    1 1
UUID=ce33cb92-21b8-49c0-95fc-17f40437d44b /boot   ext4    defaults    1 2
/dev/mapper/vg_l... linux-homelv /home        ext4    defaults    1 2
/dev/mapper/vg_l... linux-usrlv /usr         ext4    defaults    1 2
/dev/mapper/vg_l... linux-varlv /var         ext4    defaults    1 2
UUID=60dcea45-f68b-473d-b953-0fbcc5b63d5fc swap   swap    defaults    0 0
tmpfs           /dev/shm        tmpfs   defaults    0 0
devpts          /dev/pts        devpts  gid=5,mode=620 0 0
sysfs           /sys           sysfs   defaults    0 0
proc             /proc          proc    defaults    0 0
/dev/mapper/tpart /k            ext4    defaults    0 0
/dev/sda6        swap          swap    defaults    0 0
```

Note: In place of device name (sda6) blkid can also be used

- To Test auto activation of swap after putting entries in fstab use the following command
#swapon -a (*It works the same way how mount -a would work*)

Removing the SWAP Partition

- Deactivate the swap partition
#swapoff <device name>
- Remove the entry from **/etc/fstab**.
- Delete the partition through **fdisk**

Using File as a Swap Space:

At times it is seen that disks may run out of space and a partition may not be created to add a new swap space. In this situation a file can also be used as a swap space.

Steps to Create File as a swap space.

Step1: Create a file with required size like 1G for example, using dd command

```
#dd if=/dev/urandom of=/swap-file bs=1M count=1024
```

```
[root@mlinux7 ~]# dd if=/dev/urandom of=/swap-file bs=1M count=1024
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB) copied, 9.04193 s, 119 MB/s
[root@mlinux7 ~]#
[root@mlinux7 ~]# du -h /swap-file
1.0G    /swap-file
```

Step2: Change the permission of the file to 600

```
[root@mlinux7 ~]# ls -l /swap-file
-rw-r--r--. 1 root root 1073741824 Jan 24 20:26 /swap-file
[root@mlinux7 ~]# chmod 600 /swap-file
[root@mlinux7 ~]# ls -l /swap-file
-rw-----. 1 root root 1073741824 Jan 24 20:26 /swap-file
```

Step3: Format the file with swap file system

```
#mkswap -f /swap-file
```

```
[root@mlinux7 ~]# mkswap -f /swap-file
mkswap: /swap-file: warning: wiping old swap signature.
Setting up swapspace version 1, size = 1048572 KiB
no label, UUID=1a56a0cf-7f74-40ef-805f-72cb7788abb3
```

Step4: Activate the swap file

```
#swapon /swap-file
```

```
[root@mlinux7 ~]# swapon /swap-file
[root@mlinux7 ~]# swapon -s
Filename                                     Type      Size   Used   Priority
/dev/sda3                                    partition 4194300 1664    -2
/swap-file                                    file     1048572 0       -3
```

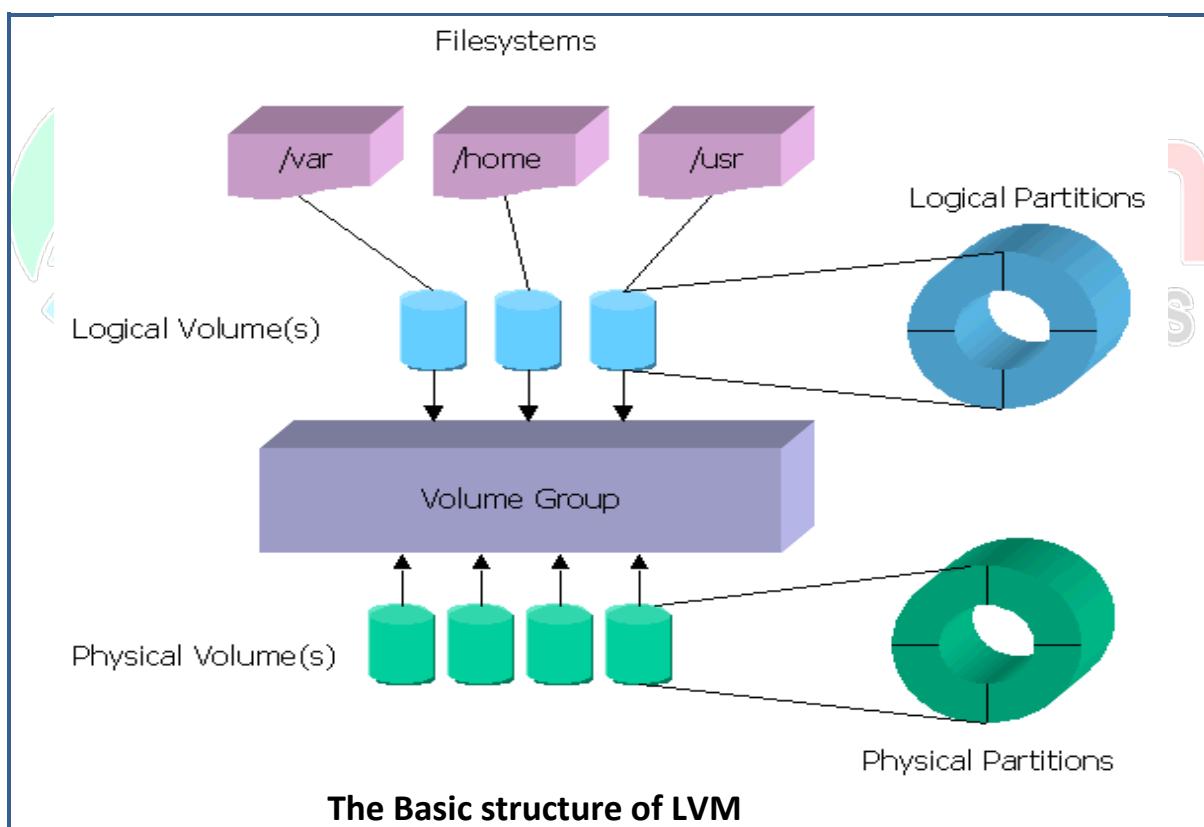
Step5: Make it permanent by adding entry in /etc/fstab

```
[root@mlinux7 ~]# vim /etc/fstab
[root@mlinux7 ~]# tail -3 /etc/fstab
/dev/sda5          /test        xfs        defaults      0 0
/dev/myvg/mylv    /my         xfs        defaults      0 0
UUID="1a56a0cf-7f74-40ef-805f-72cb7788abb3" swap      swap    defaults      0 0
```

Logical Volume Management

The Linux Logical Volume Manager (LVM) is a mechanism to virtualize the disks. It can create "virtual" disk partitions out of one or more physical hard drives, allowing you to grow, shrink, or move those partitions from drive to drive as your needs change. It also allows you to create larger partitions than you could achieve with a single drive. Traditional uses of LVM have included databases and company file servers, but even home users may want large partitions for music or video collections, or for storing online backups. LVM can also be convenient ways to gain redundancy without sacrificing flexibility.

A typical example for the need of LVM can be, assuming that we are having a disk of size 2GB and we start adding the data in the form of a single file, eventually it grows to the size of 2GB. In this case the possibility is, you go for another disk which is larger than 2GB, let's say 4GB. But what if the file again grows more than 4GB? How far you will be migrating file from one disk to another so on and so forth? It requires a down time as well which is not possible in real time, so to avoid these circumstances we implement LVM and store data in LV's whose size can be easily increased whenever required without a downtime.



Above picture shows the structure of LVM. LVM consists of **Physical Volumes**, **Volume Group**, **Logical Volumes** and finally **file systems**. The Physical partitions are known as **Physical Extents (PE)**, and the logical partitions are known as **logical Extents (LE)**

Components of LVM in Linux:

- **Physical Volumes (PV)**
- **Physical Extent (PE)**
- **Volume Group (VG)**
- **Logical Volume (LV)**
- **Logical Extent (LE)**

Physical Volume (PV)

It is the standard partition that you add to the LVM. Normally, a physical volume is a standard primary or logical partition with the hex code **8e**.

Physical Extent (PE)

It is a chunk of disk space. Every PV is divided into a number of equal sized PEs.

Volume Group (VG)

It is composed of a group of PV's and LV's. It is the organizational group for LVM.

Logical Volume (LV)

It is composed of a group of LEs. You can format and mount any file system on an LV. The size of these LV's can easily be increased or decreased as per the requirement.

Logical Extent (LE)

It is also a chunk of disk space. Every LE is mapped to a specific PE.

LVM Command	Function
pvs	Displays all the physical volumes
vgs	Displays all volume groups in the system
lvs	Displays all the logical volumes in the system
pvdisplay	Displays detailed information on physical volumes
vgdisplay	Displays detailed information on volume groups
lvdisplay	Displays detailed information on logical volumes
pvcreate	Create a new physical volume
vgcreate	Create a new volume group.
lvcreate	Creates a new logical volume
vgextend	Add a new physical disk to a volume group.
vgreduce	Reduces a volume group by removing a PV from it.
lvextend	Extends the size of a logical volume
lvreduce	Reduces the size a logical volume
lvresize	Resizes a logical volume, i.e., increase as well as decrease the size
pvmove	Moves the contents of a PV from one PV to another
lvremove	Removes /Deletes a logical volume
vgremove	Removes /Deletes a volume group
pvremove	Removes/Deletes a PV

LAB WORK:-

Creating a Physical Volume (PV)

- Create a partition using fdisk, and change the hex code of it to **8e**.
- Save and exit the fdisk and update the partition table using **partx -a** command

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1		1	3825	30720000	8e	Linux LVM
/dev/sda2	*	3825	3851	204800	83	Linux
/dev/sda3		3851	4361	4096000	82	Linux swap / Solaris
/dev/sda4		4361	6527	17406303+	5	Extended
/dev/sda5		4361	4425	521957	83	Linux
/dev/sda6		4426	4490	522081	82	Linux swap / Solaris
/dev/sda7		4491	4555	522081	83	Linux

```
Command (m for help): [t]
Partition number (1-7): 7
Hex code (type L to list codes): [8e]
Changed system type of partition 7 to 8e (Linux LVM)
```

- Create a PV on newly created partition i.e. **/dev/sda7**.
- Verify it by **pvs** or **pvdisplay** command
- **Syn:** #**pvcreate <partition name>**
#**pvcreate /dev/sda7**

```
[root@ linux Desktop]# pvcreate /dev/sda7
Physical volume "/dev/sda7" successfully created
```

```
[root@ linux Desktop]# pvs
PV          VG      Fmt Attr PSize   PFree
/dev/sda1  vg    linux lvm2 a-  29.29g  1.95g
/dev/sda7           lvm2 a-  509.84m 509.84m
```

```
[root@ linux Desktop]# pvdisplay
"/dev/sda7" is a new physical volume of "509.84 MiB"
--- NEW Physical volume ---
PV Name          /dev/sda7
VG Name
PV Size         509.84 MiB
Allocatable     NO
PE Size          0
Total PE         0
Free PE          0
Allocated PE     0
PV UUID          RzuHEg-ks6y-cvem-C5F4-tfk8-veco-mJqs46
```

- The above command will list all the PVs in the system, if you want to see the details only for a particular PV, then use
#pvdisplay <partition name> i.e. **#pvdisplay /dev/sda7**

Creating a Volume Group (VG)

- After creating a PV, the next step is to create a **Volume Group** or **VG**
- To create a VG the syntax is
#vgcreate <name for the VG> <PV name>
#vgcreate myvg /dev/sda7

```
[root@mlinux7 ~]# vgcreate myvg /dev/sda7
Volume group "myvg" successfully created
```

- Verify it by using the following command
#vgs or **#vgdisplay <vgname>**

```
[root@mlinux7 ~]# vgs
  VG #PV #LV #SN Attr   VSize   VFree
  myvg   1   0   0 wz--n- 508.00m 508.00m
[root@mlinux7 ~]# vgdisplay
--- Volume group ---
VG Name           myvg
System ID
Format           lvm2
Metadata Areas   1
Metadata Sequence No  1
VG Access        read/write
VG Status        resizable
MAX LV
Cur LV
Open LV
Max PV
Cur PV
Act PV
VG Size          508.00 MiB
PE Size          4.00 MiB
Total PE         127
Alloc PE / Size  0 / 0
Free PE / Size   127 / 508.00 MiB
VG UUID          qfPegi-5PXa-xZF0-9Aqc-I3IO-YaXL-w3RXpj
```

- To check all the **VGs** detail you can also use the command
#vgdisplay
- It will list out all the VGs in the system in detail.



Logical Volume Creation

- Once we are ready with a **Volume Group** then it's the time to create a **Logical Volume LV**
- The syntax for creating an **LV** is
- #lvcreate -L <size of LV> -n <name for LV> <VG name>**
- #lvcreate -L 300M -n mylv myvg** (To create a LV of 300MB)

```
[root@mlinux7 ~]# lvcreate -L 300M -n mylv myvg
Logical volume "mylv" created.
[root@mlinux7 ~]#
[root@mlinux7 ~]#
```

- Verify the **LV** by using the following commands
- #lvs** or **#lvdisplay** to display all the **LVs** available in the system
- #lvdisplay <VG name>** to display the **LVs** of a particular **Volume Group**
- #lvdisplay myvg**

```
[root@mlinux7 ~]# lvs
  LV   VG   Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  mylv myvg -wi-a---- 300.00m
[root@mlinux7 ~]# lvdisplay myvg
--- Logical volume ---
LV Path          /dev/myvg/mylv
LV Name          mylv
VG Name          myvg
LV UUID          MsjU7i-DRzu-UrGo-ESAO-dowh-2mJz-tz3Mgu
LV Write Access  read/write
LV Creation host, time mlinux7.mb.com, 2017-02-14 16:07:54 +0530
LV Status        available
# open           0
LV Size          300.00 MiB
Current LE       75
Segments         1
Allocation       inherit
Read ahead sectors auto
- currently set to 8192
Block device     253:0
```

- Note:** The output for only **lvdisplay** command is very lengthy to show, it is recommended that you run the command on the system and check it out. The syntax is given above.

Adding File system to the LV and Mounting it.

- As per now we have our VG created so is our LV. In order make it accessible we need to format it with a file system like ext4 or xfs
 - The syntax for formatting an LV is exactly like formatting a normal partition, Instead of **/dev/partition name** we use the path of **LV** that will be something like **/dev/vg/lv**
 - **#mkfs.ext4 /dev/myvg/mylv** or **#mkfs.xfs /dev/myvg/mylv**

```
[root@mlinux7 ~]# mkfs.ext4 /dev/myvg/mylv
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
76912 inodes, 307200 blocks
15360 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=33947648
38 block groups
8192 blocks per group, 8192 fragments per group
2024 inodes per group
Superblock backups stored on blocks:
           8193, 24577, 40961, 57345, 73729, 204801, 221185
```

```
[root@mlinux7 ~]# mkfs.xfs -f /dev/myvg/mylv
meta-data=/dev/myvg/mylv      isize=256  agcount=4, agsize=19200 blks
                               =          sectsz=512  attr=2, projid32bit=1
                               =
                               =          crc=0    finobt=0
data     =          bsize=4096  blocks=76800, imaxpct=25
         =
         =          sunit=0   swidth=0 blks
naming   =version 2          bsize=4096  ascii-ci=0 ftype=0
log      =internal log       bsize=4096  blocks=853, version=2
         =
realtime =none               extsz=4096  blocks=0, rtxtents=0
```

Mounting:

- Mounting an LV is exactly same like a normal partition, again the path for mounting will be **/dev/vg/lv**
 - Create a directory over which the LV should be mounted.
 - **#mount </dev/vgname/lvname> /directory name**
 - **#mount /dev/myvg/mylv /mydir**
 - Verify the mounting with **mount** command
 - Make it a permanent mount by making an entry in **/etc/fstab**

```
[root@mlinux7 ~]# mkdir /mydir  
[root@mlinux7 ~]# mount /dev/mvvq/mylv /mydir
```

```
#vim /etc/fstab
#
# /etc/fstab
# Created by anaconda on Tue Feb 14 12:16:02 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
#
UUID=f896b19f-142b-4796-919a-2f32e61e0cd1 / xfs defaults 0 0
UUID=42298fe6-71d8-4c41-a3dc-8a9dcffe4e40 /boot xfs defaults 0 0
UUID=5c384537-24fb-42c5-a763-0f02d6f33ae6 swap swap defaults 0 0
/dev/mvvg/mylvw /mydir xfs defaults 0 0
```

- Now you can access it and add the data as usual.

Extending a Volume Group

- Extending a volume group is actually adding a new PV to the volume group.
- To extend a volume group we need to create a new partition using fdisk. Don't forget to change its **hex code** to **8e** and update the partition table using **partprobe** command
- Create a PV on the newly created partition using **pvcreate** command
- Add the partition to the **VG** using **vgextend** command, the syntax for it is
- #**vgextend <VG name> <PV name>**
- #vgextend myvg /dev/sda8**
- Verify it **pvs** command

```
[root@mlinux7 ~]# pvcreate /dev/sda8
Physical volume "/dev/sda8" successfully created
[root@mlinux7 ~]# pvs
PV          VG   Fmt Attr PSize  PFree
/dev/sda7  myvg lvm2 a--  508.00m 208.00m
/dev/sda8            lvm2 ---  500.00m 500.00m
[root@mlinux7 ~]#
[root@mlinux7 ~]# vgextend myvg /dev/sda8
Volume group "myvg" successfully extended
[root@mlinux7 ~]# pvs
PV          VG   Fmt Attr PSize  PFree
/dev/sda7  myvg lvm2 a--  508.00m 208.00m
/dev/sda8  myvg lvm2 a--  496.00m 496.00m
[root@mlinux7 ~]#
```

Increasing the size of a logical volume

- Sometimes the file system size may be full, so we need to increase the size of the LV to continue adding the data in it.
- The size of LV can be increased online, no downtime is required.
- Check the current size of the LV by using **#df -h** command.
- Increase the size of the LV by using **lvextend** or **lvresize** command, the syntax for it is
- #**lvextend -L <+addition size> </dev/vg/lv name>** (syntax for **lvresize** is also same)
- #lvextend -L +200M /dev/myvg/mylv**
- Update the file system by using **resize2fs** command in case of ext filesystem and use **xfs_growfs** in case of xfs filesystem
- #resize2fs /dev/vg/lv name, #xfs_growfs /dev/vg/lv**
- #resize2fs /dev/myvg/mylv , #xfs_growfs /dev/myvg/mylv**
- Verify the change by using **df -h** command

```
[root@mlinux7 ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda2       20G  2.9G  18G  15% /
devtmpfs        899M    0  899M  0% /dev
tmpfs          913M   84K  913M  1% /dev/shm
tmpfs          913M  9.0M  904M  1% /run
tmpfs          913M    0  913M  0% /sys/fs/cgroup
/dev/mapper/myvg-my lv 297M   16M  282M  6% /mydir
/dev/sda1       497M  154M  344M 31% /boot
```

- Increasing the size of the LV and updating the file system**

<pre>[root@mlinux7 ~]# lvextend -L +200M /dev/myvg/mylv Size of logical volume myvg/mylv changed from 300.00 MiB (75 extents) to 500.00 MiB (125 extents). Logical volume mylv successfully resized. [root@mlinux7 ~]# xfs_growfs /dev/myvg/mylv meta-data=/dev/mapper/myvg-myvg isize=256 = sectsz=512 = crc=0 data = bsize=4096 = sunit=0 naming =version 2 bsize=4096 log =internal bsize=4096 = sectsz=512 realtime =none extsz=4096 data blocks changed from 76800 to 128000 For xfs Filesystem</pre>	<pre>[root@mlinux6 ~]# resize2fs /dev/myvg/mylv resize2fs 1.41.12 (17-May-2010) Filesystem at /dev/myvg/mylv is mounted on /mydir; old_desc_blocks = 2, new_desc_blocks = 2 Performing an on-line resize of /dev/myvg/mylv to The filesystem on /dev/myvg/mylv is now 512000 blo For ext Filesystems</pre>
---	--

- Verify it by df -h**

<pre>[root@mlinux7 ~]# df -h Filesystem Size Used Avail Use% Mounted on /dev/sda2 20G 2.9G 18G 15% / devtmpfs 899M 0 899M 0% /dev tmpfs 913M 84K 913M 1% /dev/shm tmpfs 913M 9.1M 904M 1% /run tmpfs 913M 0 913M 0% /sys/fs/cgroup /dev/mapper/myvg-myvg 497M 16M 482M 4% /mydir /dev/sda1 497M 154M 344M 31% /boot tmpfs 183M 16K 183M 1% /run/user/42 tmpfs 183M 0 183M 0% /run/user/0</pre>	
---	--

Reducing the size of the LV

- Reducing the size of an LV is a bit complicated task, there are few things which you need to keep in mind before reducing the size of an LV.
 - LV size cannot be reduced online, it requires a down time i.e. unmounting the file system.
 - Check the consistency of the File system.
 - Update the file system about the size. I.e. what its size will be after reduction.
 - Finally reduce the size. **Huh....! Lots of things to do!!!!**
- If any of the above things are missed then it will be a mess, you may corrupt the file system and LV.

Let's start the steps carefully

- Check the size of the lv using **df -h** command
- Unmount the LV using **umount** command
- Check the consistency of file system by using **e2fsck** command
- #e2fsck -f /dev/myvg/mylv.**
- Update the file system by using **resize2fs** command
- #resize2fs /dev/myvg/mylv 300M** (where **300M** is the approximate total size of LV after reduction)

- Now reduce the size by using # **lvreduce -L -200M /dev/myvg/mylv** command
- We know the size of LV is around 500MB, from previous picture in case of extending the size of LV.
- Or else you can run **df -h** and verify it again.
- Unmount the LV by using **umount** command

```
[root@mlinux6 ~]# umount /mydir
```

- Check the consistency of the file system.

```
[root@mlinux6 ~]# e2fsck -f /dev/myvg/mylv
e2fsck 1.41.12 (17-May-2010)
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
/dev/myvg/mylv: 11/127512 files (0.0% non-contiguous),
```

- Update the file system about the size after reduction

```
[root@mlinux6 ~]# resize2fs /dev/myvg/mylv 300M
resize2fs 1.41.12 (17-May-2010)
Resizing the filesystem on /dev/myvg/mylv to 307200 (1k) blocks.
The filesystem on /dev/myvg/mylv is now 307200 blocks long.
```

- Finally reduce the size of the LV using **lvreduce** command. It will prompt you about the change type **y** to continue with reduction.

```
[root@mlinux6 ~]# lvreduce -L -200M /dev/myvg/mylv
WARNING: Reducing active logical volume to 300.00 MiB
THIS MAY DESTROY YOUR DATA (filesystem etc.)
Do you really want to reduce mylv? [y/n]: y
Size of logical volume myvg/mylv changed from 500.00 MiB (125 extents) to 300.00 MiB (75 extents).
Logical volume mylv successfully resized
```

- Mount the LV and run the command **df-h**, to verify the change in the size of LV
- #mount -a** (if an entry is passed in **/etc/fstab** use this command), else do manual mounting as shown below
- #df -h**

```
[root@mlinux6 ~]# mount /dev/myvg/mylv /mydir
[root@mlinux6 ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda2        24G   6.8G   16G  30% /
tmpfs           491M    72K  491M   1% /dev/shm
/dev/sda1       190M   28M  153M  16% /boot
/dev/mapper/myvg-mylv
                  283M  2.1M  266M   1% /mydir
```

Note: xfs filesystem does not support file system reduction

Moving or Migrating the data from one PV to another.

- There might be a situation where the **PV** might be failing and it is required to be replaced, in such case, we need to migrate or move the data from such **PV** to the other and isolate the **PV**.
- **The Steps to migrate the PV are**
 - Access the mount point of failing **PV** and check the data in it,
 - Verify the size of the **PV** by **pvs** command or **pvdisplay** command.
 - Unmount the file system on that **PV (optional)**
 - Add new **PV**, which should be of the same size or higher than that of the replacing **PV** to the volume group.
 - Migrate the **PVs** contents to the new **PV** using following command
 - **#pvmmove <Old PV> <New PV>**
 - Mount back the **LV**, access the mount point and verify the data in it.
 - Remove the faulty **PV** from Volume Group.

Okay! So let's do the practical following above steps.

- Access the mount point of the failing PV and check the data in it,

```
[root@mlinux6 ~]# cd /mydir
[root@mlinux6 mydir]# ls
lost+found
[root@mlinux6 mydir]# touch mydir{1..10}
[root@mlinux6 mydir]# ls
lost+found  mydir1  mydir10  mydir2  mydir3  mydir4  mydir5
[root@mlinux6 mydir]#
```

- Verify the size of the PV by **pvs** command or **pvdisplay** command.

```
[root@mlinux6 ~]# pvs
  PV          VG  Fmt Attr PSize   PFree
  /dev/sda6    myvg lvm2 a--  516.00m 216.00m
  /dev/sda7    myvg lvm2 a--  516.00m 516.00m
[root@mlinux6 ~]# pvdisplay
--- Physical volume ---
PV Name           /dev/sda6
VG Name           myvg
PV Size           517.69 MiB / not usable 1.69 MiB
Allocatable       yes
PE Size           4.00 MiB
Total PE          129
Free PE           54
Allocated PE      75
PV UUID           46Gvd5-z04y-tneS-S4v1-1ySh-TG8J-PiqnXy
```

- **Unmount the file system on that PV (this is optional as migration can be done online as well)**
- **#umount /mydir**

- Add new PV which should be of the same size or higher than that of the replacing PV to the volume group.
- In our case the size of the failing PV is around **500MB**, so we need to add a PV whose size is at least **500MB** or more
- I have created another partition from fdisk i.e. **/dev/sda7** with the size around **500MB**

```
[root@mlinux6 ~]# pvs
  PV          VG  Fmt  Attr PSize   PFree
  /dev/sda6   myvg lvm2 a--  516.00m 216.00m
  /dev/sda7           lvm2 ---  517.69m 517.69m
[root@mlinux6 ~]# vgextend myvg /dev/sda7
  Volume group "myvg" successfully extended
[root@mlinux6 ~]# pvs
  PV          VG  Fmt  Attr PSize   PFree
  /dev/sda6   myvg lvm2 a--  516.00m 216.00m
  /dev/sda7   myvg lvm2 a--  516.00m 516.00m
[root@mlinux6 ~]#
```

- Migrate the PV's contents to the new PV using following command
- #pvmove <Old PV> <New PV>
- #pvmove /dev/sda6 /dev/sda7

```
[root@mlinux6 ~]# pvmove /dev/sda6 /dev/sda7
  /dev/sda6: Moved: 4.0%
  /dev/sda6: Moved: 100.0%
```

- Mount back the LV, access the mount point and verify the data in it.

```
[root@mlinux6 ~]# mount -a
[root@mlinux6 ~]# cd /mydir
[root@mlinux6 mydir]# ls
lost+found mydir1 mydir10 mydir2 mydir3 mydir4 mydir5 mydir6 mydir7 mydir8 mydir9
[root@mlinux6 mydir]#
```

- Remove the free PV from Volume Group.
- As the data is moved safely, now let's remove the faulty PV from the volume group.
- The syntax to remove a PV from a VG is
- #vgreduce <vg name> <PV name>
- #vgreduce myvg /dev/sda6

```
[root@mlinux6 ~]# vgreduce myvg /dev/sda6
  Removed "/dev/sda6" from volume group "myvg"
[root@mlinux6 ~]# pvs
  PV          VG  Fmt  Attr PSize   PFree
  /dev/sda7           lvm2 ---  517.69m 517.69m
  /dev/sda6   myvg lvm2 a--  516.00m 216.00m
[root@mlinux6 ~]#
```

Deleting/Removing the LV:

- To Delete/Remove an LV, first unmount the file system.
- Remove the entry from **/etc/fstab**.
- Use the command **lvremove** i.e.
- **#lvremove <LV name>**
- **#lvremove /dev/myvg/mylv** (it will prompt to you to continue, press **y** to continue)
- Verify it by using **lvdisplay** command

```
[root@mlinux6 ~]# umount /mydir
[root@mlinux6 ~]# lvremove /dev/myvg/mylv
Do you really want to remove active logical volume mylv? [y/n]: y
Logical volume "mylv" successfully removed
[root@mlinux6 ~]# lvdisplay myvg
[root@mlinux6 ~]#
```

- As we was having only one LV and that is now deleted, that's why it is not showing any LVs after executing **lvdisplay** command.

Deleting a Volume Group

- To delete the volume group, make sure that if there is any LV in it, it should not be mounted. Because while removing a vg it will also remove LV's inside it. In our case we have no LV in our volume group, so we will not be concerned about it.
- To delete a VG, use the following command.
- **#vgremove <vgname>**
- **#vgremove myvg**

```
[root@mlinux6 ~]# vgremove myvg
Volume group "myvg" successfully removed
[root@mlinux6 ~]# vgs
No volume groups found
```

Deleting a Physical Volume

- Deleting a PV is very simple. The only thing we should check that the PV we are going to delete should not belong to any volume group. We can only delete a PV which is free.
- The syntax to delete a PV is
- **#pvremove <PV name>**
- **#pvremove /dev/sda6**
- **#pvremove /dev/sda7 OR**
- **#pvremove /dev/sda{6,7}** (To remove multiple PVs in one command)

```
[root@mlinux6 ~]# pvremove /dev/sda{6,7}
Labels on physical volume "/dev/sda6" successfully wiped
Labels on physical volume "/dev/sda7" successfully wiped
[root@mlinux6 ~]# pvs
```

- If you want you can verify it by using **pvs** or **pvdisplay** commands

Building anything requires lots of concentration, hard work, and patience, but to destroy it, it is just a matter of seconds. Isn't it....!

Creating a VG by customized PE size

- To create a VG with specifying an PE size,
- First create a partition and also create a pv

```
[root@ktlinux ~]# fdisk -l

Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000003d37

      Device Boot      Start        End    Blocks   Id  System
/dev/sda1            1       1785    14336000   8e  Linux LVM
/dev/sda2     *       1785       1811     204800   83  Linux
/dev/sda3       1811       2072    2097152   82  Linux swap / Solaris
/dev/sda4       2072       2610    4325849    5  Extended
/dev/sda5       2072       2136    518412+   8e  Linux LVM

[root@ linux ~]# pvcreate /dev/sda5
Physical volume "/dev/sda5" successfully created
```

- To create a vg with custom PE size use
`#vgcreate <name for the vg> -s <size of PE(1-128)> <pv names>`
`#vgcreate myvg -s 16 /dev/sda5`

```
[root@mlinux7 ~]# vgcreate myvg -s 16 /dev/sda5
Volume group "myvg" successfully created
```

- Verify the PE size using `vgdisplay` command

```
[root@mlinux7 ~]# vgdisplay myvg
--- Volume group ---
VG Name           myvg
System ID
Format           lvm2
Metadata Areas   1
Metadata Sequence No  1
VG Access        read/write
VG Status        resizable
MAX LV
Cur LV
Open LV
Max PV
Cur PV
Act PV
VG Size          496.00 MiB
PE Size          16.00 MiB
Total PE         31
Alloc PE / Size  0 / 0
Free PE / Size   31 / 496.00 MiB
VG UUID          YyxEjM-JaVu-cy0X-2WTb-d2Ur-BRLb-PZsjSs
```

Creating LV by specifying no. of LE instead of giving size in MB or GB.

- To create an LV using LE, the things to keep in mind are
- **Size of LE = Size of PE**
- **For example if the size of PE is 16, then the size of LE will also be 16.**

Steps to create an LV based on LE

- The syntax to create an LV with no. of LE is

```
#lvcreate -l <no. of LE> -n <name for the LV> <volume group name>
#lvcreate -l 25 -n mylv myvg
```

```
[root@mlinux7 ~]# lvcreate -l 25 -n mylv myvg
Logical volume "mylv" created.
```

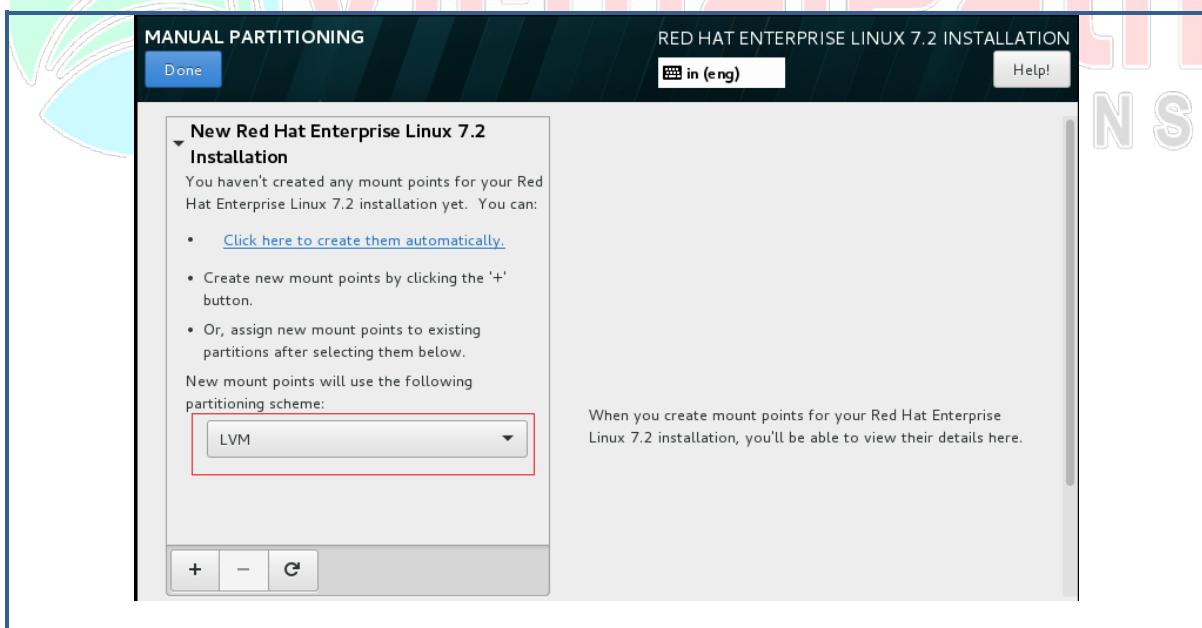
- Now check the size of the LV “mylv” using lvdisplay command

```
#lvdisplay myvg
```

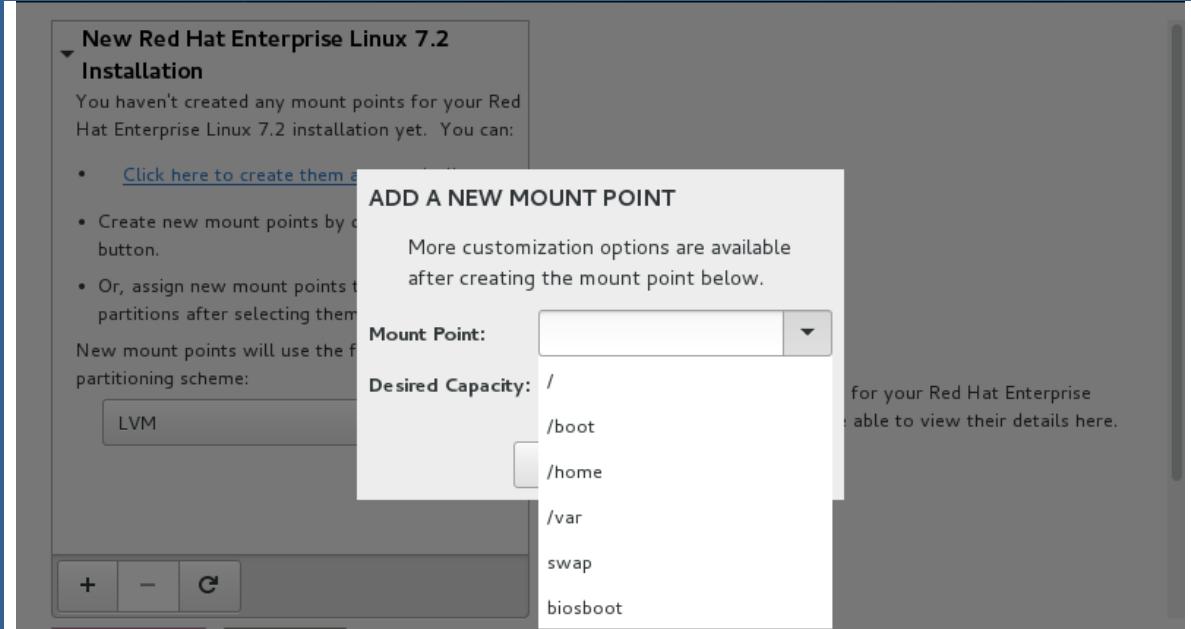
```
[root@mlinux7 ~]# lvdisplay myvg
--- Logical volume ---
LV Path          /dev/myvg/mylv
LV Name          mylv
VG Name          myvg
LV UUID          PYo57X-55qe-yRMu-dIso-rXX4-Mn0F-VZzYjA
LV Write Access  read/write
LV Creation host, time mlinux7.mb.com, 2017-02-15 14:45:38 +0530
LV Status        available
# open           0
LV Size          400.00 MiB
Current LE       25
Segments         1
Allocation       inherit
Read ahead sectors auto
- currently set to 8192
Block device     253:0
```

INSTALLING RHEL7/8 USING LVM PARTITIONING

- The only difference in a normal installation and **LVM** installation is that instead of creating normal partition we will create a **VG** and then **LVs** for all partitions, except **/boot** and **swap**.
- The advantage of installing Linux using **LVM** is that, if any of system partition is running out of space and required more space, in case of normal partitioning it is not possible to increase the size of a partition once it is created. But, using LVM the space can be dynamically increased whenever it is required.
- Even if there is no space remaining in the disk some space can be borrowed from other **LVMs** and can easily be assigned to required system partition to fulfill its need.
- LVM** provides a greater scalability to the administrator and avoid uncertain down time to the server.
- LVM** ensures the possibility of increasing and decreasing the sizes whenever required and prevents unnecessary loss of time.
- Start the installation normally as done previously, but only at the time of partitioning follow the steps below.



- Select the **LVM** scheme and click on **+** to proceed.



New Red Hat Enterprise Linux 7.2 Installation

You haven't created any mount points for your Red Hat Enterprise Linux 7.2 installation yet. You can:

- Click here to create them all at once.
- Create new mount points by clicking the Add button.
- Or, assign new mount points to existing partitions after selecting them.

New mount points will use the following partitioning scheme:

LVM

+ - G

ADD A NEW MOUNT POINT

More customization options are available after creating the mount point below.

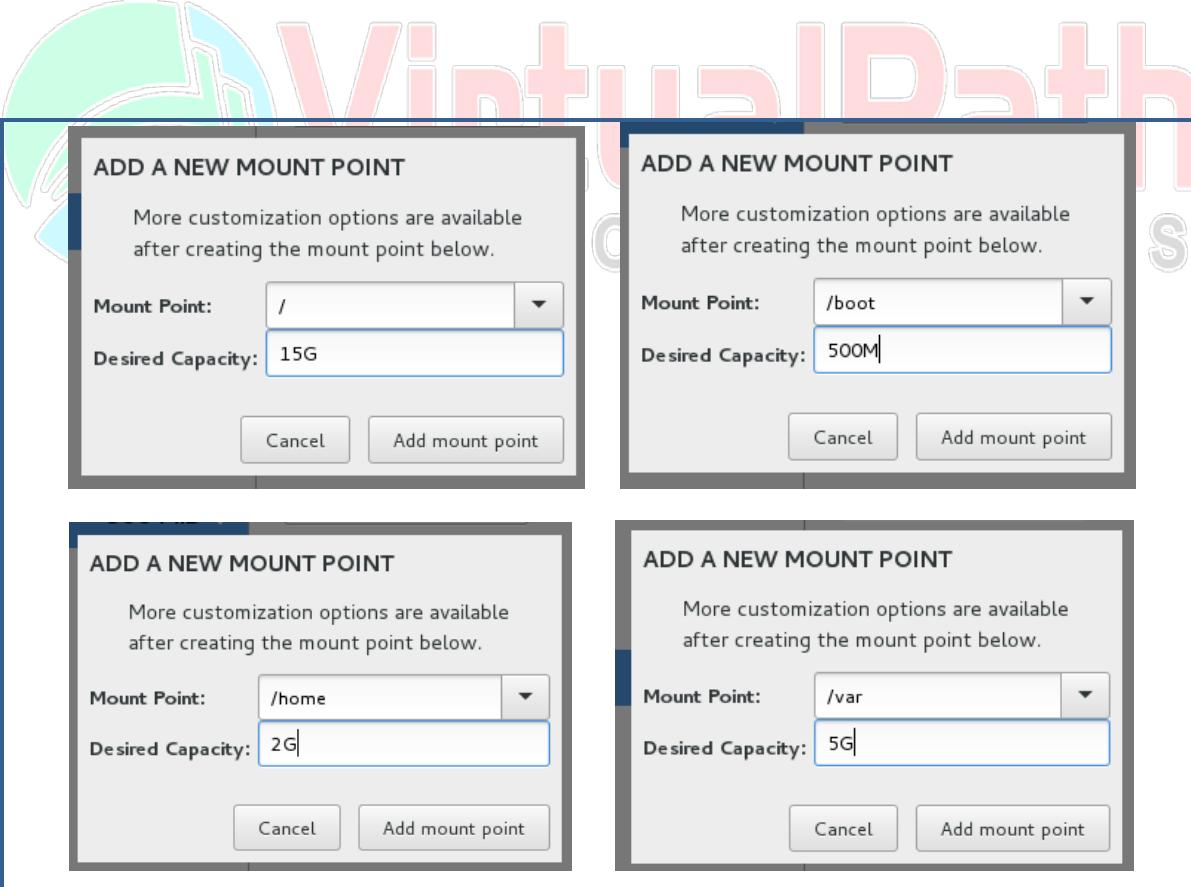
Mount Point:

Desired Capacity:

/
/boot
/home
/var
swap
biosboot

for your Red Hat Enterprise Linux 7.2 installation. You will be able to view their details here.

- Select an appropriate mount point and assign the size. Repeat it till all important mount points are created



ADD A NEW MOUNT POINT

More customization options are available after creating the mount point below.

Mount Point:

Desired Capacity:

ADD A NEW MOUNT POINT

More customization options are available after creating the mount point below.

Mount Point:

Desired Capacity:

ADD A NEW MOUNT POINT

More customization options are available after creating the mount point below.

Mount Point:

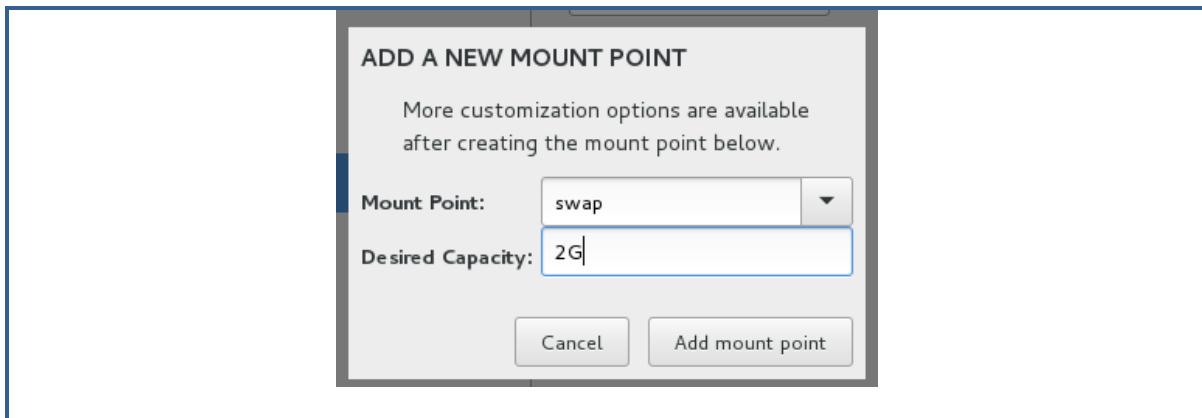
Desired Capacity:

ADD A NEW MOUNT POINT

More customization options are available after creating the mount point below.

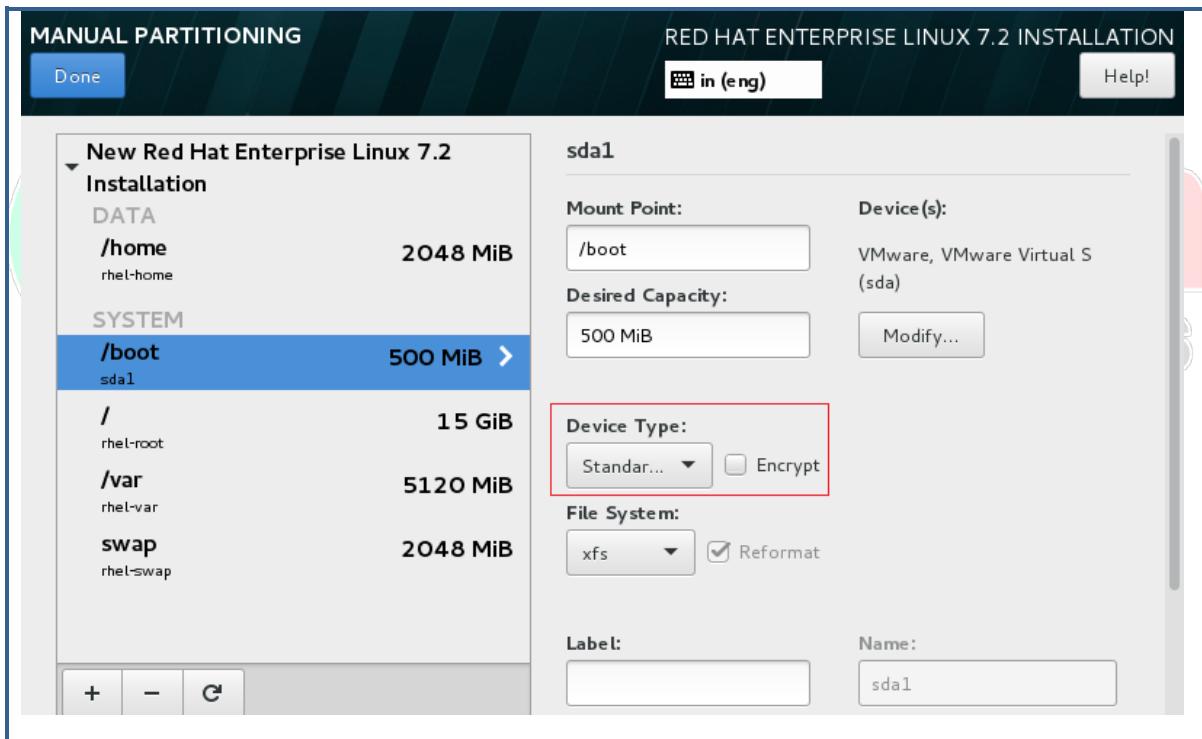
Mount Point:

Desired Capacity:



- Give appropriate size and create /, /boot/, /home/, /var and swap

Note: All the sizes listed above are based on the availability of the space. It is no-where a recommended or minimum sizes. The sizes can be based on your requirements. But / should get more size as it will be having "/usr", "/opt" & "/tmp" as well, which in rhel6 used to be created separately.



- Verify the sizes and click on "**Done**" to continue with the installation. Complete the installation as usual as we have done previously at the beginning of the course.
- Note: /boot is automatically created as "**Standard Partition**" even though the scheme is selected as "**LVM**". Isn't it awesome?

Practice the LVM Concept well; as it is the most important part in Linux and in any UNIX operating system as well.

That sums up the LVM concept in Linux

USER AND GROUP ADMINISTRATION

PART- I USER ADMINISTRATION

In Linux/Unix user is one who uses the system. There can be at least one or more than one users in Linux at a time. Users on a system are identified by a username and a userid. The username is something that users would normally refer to, but as far as the operating system is concerned this is referred to using the user id (or uid). The username is typically a user friendly string, such as your name, whereas the user id is a number. The words username and userid are often (incorrectly) used interchangeably. The user id numbers should be unique (one number per user). If you had two usernames with the same user id, effectively their permissions would be the same and the files that they create would appear to have been created by the same user. This should not be allowed and the **useradd** command will not allow usernames to share the same userid.

Some Important Points related to Users:

- Users and groups are used to control access to files and resources
- Users login to the system by supplying their username and password
- Every file on the system is owned by a user and associated with a group
- Every process has an owner and group affiliation, and can only access the resources its owner or group can access.
- Every user of the system is assigned a unique user ID number (the UID)
- Users name and UID are stored in **/etc/passwd**
- User's password is stored in **/etc/shadow** in encrypted form.
- Users are assigned a **home directory** and a program that is run when they login (**Usually a shell**)
- Users cannot read, write or execute each other's files without permission.

Types of users In Linux and their attributes:

TYPE	EXAMPLE	USER ID (UID)	GROUP ID (GID)	HOME DIRECTORY	SHELL
Super User	root	0	0	/root	/bin/bash
System User	ftp, ssh, apache nobody	1 to 999	1 to 999	/var/ftp, /var/www/html /var/named, etc.	/sbin/nologin
Normal User	Visitor, myuser,etc	1000 to 60000	1000 to 60000	/home/user name	/bin/bash

In Linux there are three types of users.

1. Super user or root user

Super user or the root user is the most powerful user. He is the administrator user.

2. System user

System users are the users created by the software or applications. For example if we install Apache it will create a user apache. These kinds of users are known as system users.

3. Normal user

Normal users are the users created by root user. They are normal users like Rahul, Musab etc. Only the root user has the permission to create or remove a user.

Whenever a user is created in Linux things created by default:-

- A home directory is created(/home/username)
- A mail box is created(/var/spool/mail/username)
- unique UID & GID are given to user

Linux uses UPG (User Private Group) scheme

- It means that whenever a user is created it has its own private group
- For Example if a user is created with the name **musab**, then a primary group for that user will be **musab** only

There are two important files a user administrator should be aware of

1. "/etc/passwd"
2. "/etc/shadow"

Each of the above mentioned files have specific formats

1. /etc/passwd

```
[root@ linux ~]# head /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
```

The above fields are

- **root** =name
- **x**= link to password file i.e. /etc/shadow
- **0 or 1**= UID (user id)
- **0 or 1**=GID (group id)
- **root or bin** = comment (brief information about the user)
- **/root or /bin** = home directory of the user
- **/bin/bash or /sbin/nologin** = shell

2. /etc/shadow

```
root:$1fdgsdfsdksdkffefje:14757:0:99999:7:::
```

The fields are as follows,

1. **root** = User name
2. **:\$1fdgsdfsdksdkffefje** = Encrypted password
3. **14757** = Days since that password was last changed.
4. **0** = Days after which password must be changed.
5. **99999** = Days before password is to expire that user is warned.
6. **7** = Days after the password is expires that the user is disabled.
7. A reserved field.

LAB WORK:-

Creating a user

- The syntax for creating a user in Linux is
- # **useradd <username> <options>** or #**useradd <options> <username>**
- **options** are
- -u user id
- -G Secondary group id
- -g primary group id
- -d home directory
- -c comment
- -s shell

Let's create a user with default attributes.

- When no option is used with **useradd** command the options like **UID**, **GID**, **home dir** and **shell** will be assigned default.
- #**useradd <username>**
- #**useradd myuser**

```
[root@mlinux7 ~]# useradd myuser
[root@mlinux7 ~]# tail /etc/passwd
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs/nobody
pulse:x:171:171:PulseAudio System Daemon:/var/run/pulse
gdm:x:42:42::/var/lib/gdm:/sbin/nologin
gnome-initial-setup:x:989:984::/run/gnome-inhibit
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd
tcpdump:x:72:72::/sbin/nologin
myuser:x:1000:1000::/home/myuser:/bin/bash
```

Observe that the uid, gid, home dir, and shell is assigned automatically.

Let's create a user with customized attributes

- Create a user with following attributes
- Name = myuser2
- uid=1050
- comment =MGR
- shell = /bin/csh (c shell)
- #useradd -u 1050 -c MGR -s /bin/csh myuser2 (rhel7)

```
[root@mlinux7 ~]# useradd -u 1050 -c MGR -s /bin/csh myuser2
[root@mlinux7 ~]# tail /etc/passwd
pulse:x:171:171:PulseAudio System Daemon:/var/run/pulse:/sbin/nologin
gdm:x:42:42::/var/lib/gdm:/sbin/nologin
gnome-initial-setup:x:989:984::/run/gnome-initial-setup/:/sbin/nologin
avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72:72::/sbin/nologin
myuser:x:1000:1000::/home/myuser:/bin/bash
myuser2:x:1050:1050:MGR:/home/myuser2:/bin/csh
```

Assigning password to the user:

- As a root user we can assign any password to any user
- The syntax for assigning a password is
- #**passwd** to assign password to current user (the one with which you have logged in, if it is root then root's password will be changed)
- #**passwd <user name>** to assign a password to a specific user, only root can assign password to other user.

```
[root@mlinux7 ~]# passwd myuser
Changing password for user myuser.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
```

Note: If the “passwd” command is executed without any username, it would consider the Name of active or currently logged in user. Say, if you have logged in with root, it will try to change the root password

Modifying the user's attribute

- After creating a user if we need to modify the attributes of user like changing uid, changing secondary group id or adding a comment, locking or unlocking the user account, can be done by following command
- Syntax. # usermod <options> <username>
- options are:
- all the options which are used with useradd command can be used and also the following,
- l to change login name
- L to LOCK account
- U to UNLOCK account
- ex. # usermod -l newname oldname (changing the name of the user)
- ex. # usermod -L username to lock the user account; #passwd -l username
- ex. # usermod -U username to unlock the user account; #passwd -u username
- Note: - when an account is locked it will show! (Exclamation mark) in /etc/shadow file.
- The account lock/unlock details can also be checked by "#passwd -S" command

Locking and unlocking a user account:

- To lock a user a/c use the following
- #usermod -L < user name>; #passwd -l <user name>
- #usermod -L myusr; #passwd -l myuser
- Verify it in /etc/shadow file, it shows exclamation mark before user account or try using #passwd -S <user name>

```
[root@mlinux7 ~]# usermod -L myuser
[root@mlinux7 ~]# passwd -l myuser
myuser !LK 2017-02-16 0 99999 7 -1 (Password locked.)
```

OR

```
[root@mlinux7 ~]# passwd -l myuser
Locking password for user myuser.
passwd: Success
[root@mlinux7 ~]# passwd -S myuser
myuser !LK 2017-02-16 0 99999 7 -1 (Password locked.)
[root@mlinux7 ~]#
```

"/etc/shadow" file status

```
[root@mlinux7 ~]# tail /etc/shadow |grep myuser
myuser:!$6$70F3QNKR$dfCus1NqGy.eThCxfyYyicasCKGX6
:::
```

Unlocking a user a/c:

- Unlock the above a/c
- **#usermod -U < user name >; #passwd -u <user name>**
- **#usermod -U myuser ; #passwd -u myuser**
- Verify it in **/etc/shadow** file, it shows exclamation mark before user account or try using **#passwd -S username**

```
[root@mlinux7 ~]# usermod -U myuser
[root@mlinux7 ~]# passwd -S myuser
myuser PS 2017-02-16 0 99999 7 -1 (Password set, SHA512 crypt.)
```

OR

```
[root@mlinux7 ~]# passwd -u myuser
Unlocking password for user myuser.
passwd: Success
[root@mlinux7 ~]# passwd -S myuser
myuser PS 2017-02-16 0 99999 7 -1 (Password set, SHA512 crypt.)
```

“/etc/shadow” file status

```
[root@mlinux7 ~]# tail /etc/shadow |grep myuser
myuser:$6$70F3QNKR$dfCuslNqGy.eThCx fyYyicaSCkGX6
:::
```

- Observe in both pictures that once the account is unlocked the exclamation is gone.

The password parameters

- For any user we can set the parameters for the password, like **min and max password age, password expiration warnings and a/c expiration date** etc.
- To view the advanced parameters of the user, use **#chage -l < user name>**
- **#chage -l myuser**

[root@mlinux7 ~]# chage -l myuser	
Last password change	: Feb 16, 2017
Password expires	: never
Password inactive	: never
Account expires	: never
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

- **Last password change:** When the password was changed last time.
- **Password expires:** Password expiry date
- **Password inactive:** After password expiry grace period before the account gets locked.
- **Account expires:** Date on which the account expires.
- **Minimum number of days b/w password change:** Once the password is changed, it cannot be changed until a min period of specified date. [0] means never.
- **Max number of days b/w password change:** After changing the password how long it will be valid for.
- **Number of days of warning before password expires:** Start of warnings to change the password, no. of days before the password expires.

Changing the password parameters:

- Changing of the password parameters can be done by two ways.
1. #chage <user name >
 2. #chage <option> <value> <username>
- Let's see the first method and then the other.
 - To set the password parameters of a user "myuser" to
 - Min password age : 1 days
 - Max password age: 7 days
 - Password expiration warnings: 2 days before password expires
 - Password inactive [-1]: 1 one day later the account will be locked, after password expiry.
 - A/C expiration date: 2017-03-31 (March 31st 2017)
 - #chage myuser

```
[root@mlinux7 ~]# chage myuser
Changing the aging information for myuser
Enter the new value, or press ENTER for the default

      Minimum Password Age [0]: 1
      Maximum Password Age [99999]: 7
      Last Password Change (YYYY-MM-DD) [2017-02-16]:
      Password Expiration Warning [7]: 2
      Password Inactive [-1]: 1
      Account Expiration Date (YYYY-MM-DD) [-1]: 2017-03-31

[root@mlinux7 ~]#
[root@mlinux7 ~]# chage -l myuser
Last password change : Feb 16, 2017
Password expires       : Feb 23, 2017
Password inactive      : Feb 24, 2017
Account expires        : Mar 31, 2017
Minimum number of days between password change : 1
Maximum number of days between password change : 7
Number of days of warning before password expires : 2
```

- The second method is for, if you want to change a particular field of password aging policy
- #chage <option> <value> <username>
- The options which can be used are as follows
 - -m for Min password age
 - -M for Max password age
 - -d for last time the password is changed. (*Note: if given d 0, it will force the user to change password at next login*)
 - -W Password expiration warnings
 - -I Password inactive [-1 means inactive].
 - -E A/C expiration date

- Let's see how to change only the account expiration date

```
[root@mlinux7 ~]# chage -E 2017-04-30 myuser
[root@mlinux7 ~]# chage -l myuser
Last password change : Feb 16, 2017
Password expires      : Feb 23, 2017
Password inactive     : Feb 24, 2017
Account expires       : Apr 30, 2017
Minimum number of days between password change : 1
Maximum number of days between password change : 7
Number of days of warning before password expires : 2
```

Likewise you can use any option listed above and change any particular field in password aging parameters.

Forcing a user to change the password at next login:

Sometimes it is required to force the users to change their password at next login. This can be done using following syntax

- #chage -d 0 <username>, (where 0 = zero days since last password change)
- #chage -d 0 myuser

```
[root@mlinux7 ~]# chage -d 0 myuser
[root@mlinux7 ~]# chage -l myuser
Last password change : password must be changed
Password expires      : password must be changed
Password inactive     : password must be changed
Account expires        : Apr 30, 2017
Minimum number of days between password change : 1
Maximum number of days between password change : 7
Number of days of warning before password expires : 2
```

At next login

login as: myuser
myuser@192.168.100.20's password:
You are required to change your password immediately (root enforced)
Last failed login: Tue Feb 14 14:46:17 IST 2017 from 192.168.100.3 on ssh:notty
There was 1 failed login attempt since the last successful login.
WARNING: Your password has expired.
You must change your password now and login again!
Changing password for user myuser.
Changing password for myuser.

Deleting a User:

- To delete a user the syntax used is
- #userdel <username> it will only delete the user but home directory will be there. To delete the user with its home directory and mailbox, use the following command.
- #userdel -r < user name >
- #userdel -r myuser2

```
[root@mlinux7 ~]# userdel -r myuser2
[root@mlinux7 ~]# cd /home
[root@mlinux7 home]# ls
myuser
[root@mlinux7 home]# cd /var/spool/mail/
[root@mlinux7 mail]# ls
myuser  rpc
```

We're now done with user administration, let's see what's in part-II

PART-II GROUP ADMINISTRATION

GROUPS

- Users are assigned to groups with unique group ID numbers (the GID)
- The group name and GID are stored in **/etc/group**
- Each user is given their own private group
- They can also be added to their groups to gain additional access
- All users in a group can share files that belong to the group

Each user is a member of at least one group, called a primary group. In addition, a user can be a member of an unlimited number of secondary groups. Group membership can be used to control the files that a user can read and edit. For example, if two users are working on the same project you might put them in the same group so they can edit a particular file that other users cannot access.

- A user's primary group is defined in the **/etc/passwd** file and Secondary groups are defined in the **/etc/group** file.
- The primary group is important because files created by this user will inherit that group affiliation.

Creating a Group with default options :

- To create a group the syntax is
- **#groupadd <name for the group>**
- **#groupadd mygroup**

```
[root@mlinux7 ~]# groupadd mygrp
[root@mlinux7 ~]# tail /etc/group
postdrop:x:90:
postfix:x:89:
ntp:x:38:
sshd:x:74:
tcpdump:x:72:
stapusr:x:156:
stapsys:x:157:
stapdev:x:158:
myuser:x:1000:
mygrp:x:1001:
```

Creating a group with user specified group id (GID)

- **#groupadd <option> <name for the group>**
- **#groupadd -g 1050 mygrp2**
- Verify it in /etc/group

```
[root@mlinux7 ~]# groupadd -g 1050 mygrp2
[root@mlinux7 ~]# tail /etc/group
postfix:x:89:
ntp:x:38:
sshd:x:74:
tcpdump:x:72:
stapusr:x:156:
stapsys:x:157:
stapdev:x:158:
myuser:x:1000:
mygrp:x:1001:
mygrp2:x:1050:
```

Modifying the properties of the group

- To modify the group properties the syntax is
- **#groupmod <option> <arguments> <group name>**
- The options are
- -g to change the group id
- -o to override the previous assigned id, if it matches with the new one.
- -n to change the group name

Changing the GID of the group

- **#groupmod -g 1100 mygrp**
- Verify it in /etc/group

```
[root@mlinux7 ~]# groupmod -g 1100 mygrp
[root@mlinux7 ~]# tail /etc/group
postfix:x:89:
ntp:x:38:
sshd:x:74:
tcpdump:x:72:
stapusr:x:156:
stapsys:x:157:
stapdev:x:158:
myuser:x:1000:
mygrp:x:1100:
mygrp2:x:1050:
```

Changing the name of the group

- The syntax for changing the group name is
- **#groupmod -n <new name > < existing name >**
- **#groupmod -n mygrp mygroup**

```
[root@mlinux7 ~]# groupmod -n mygroup mygrp
[root@mlinux7 ~]# grep mygroup /etc/group
mygroup:x:1100:
```

Adding and Removing Members to a Group

- Adding single or multiple users to the group with various attributes
- #gpasswd < option> <arguments> <group name>
- Options:
 - -M For Adding Multiple users to a group
 - -a for Adding a single user to a group
 - -A for Adding a group Administrator
 - -d removing a user from a group
- #gpasswd -M <user>,<user>,<user> <group>
- #gpasswd -M u1,u2,u3 mygroup

```
[root@mlinux7 ~]# gpasswd -M u1,u2,u3 mygroup
[root@mlinux7 ~]# tail /etc/group
tcpdump:x:72:
stapusr:x:156:
stapsys:x:157:
stapdev:x:158:
myuser:x:1000:
mygrp2:x:1050:
mygroup:x:1100:u1,u2,u3
```

Adding a single user using gpasswd

- #gpasswd -a u4 mygroup (verify it in /etc/group)

```
[root@mlinux7 ~]# useradd u4
[root@mlinux7 ~]# gpasswd -a u4 mygroup
Adding user u4 to group mygroup
[root@mlinux7 ~]# grep mygroup /etc/group
mygroup:x:1100:u1,u2,u3,u4
[root@mlinux7 ~]#
```

Making a user as an administrator of the group

- #gpasswd -A u1 mygroup (verify it in /etc/gshadow)

```
[root@mlinux7 ~]# grep mygroup /etc/gshadow
mygroup:!:u1,u2,u3,u4
[root@mlinux7 ~]# gpasswd -A u1 mygroup
[root@mlinux7 ~]# grep mygroup /etc/gshadow
mygroup:!:u1:u1,u2,u3,u4
[root@mlinux7 ~]#
```

Removing a user from the group

- #gpasswd -d u2 mygroup

```
[root@mlinux7 ~]# grep mygroup /etc/group
mygroup:x:1100:u1,u2,u3,u4
[root@mlinux7 ~]# gpasswd -d u2 mygroup
Removing user u2 from group mygroup
[root@mlinux7 ~]# grep mygroup /etc/group
mygroup:x:1100:u1,u3,u4
```

Removing a group

- #groupdel <group name>; #groupdel mygroup

CONTROLLING ACCESS TO FILES

In this chapter we will be dealing with two things.

1. Special Permissions or Advanced Permission
2. Access Control List (ACL)

Let's first begin with Special Permissions

1. Special Permissions or Advanced Permission

- There are three special permissions that can be assigned to a file or directory apart from basic file permissions(rwx), they are
- **SUID – SET USER ID**
- **SGID – SET GROUP ID**
- **STICKY BIT**

Permission	Symbolic Form	Numeric Form	Syntax
SETUID	s or S	4	#chmod u+s or #chmod 4766
SETGID	s or S	2	#chmod g+s or #chmod 2766
STICKY BIT	t or T	1	#chmod o+t or #chmod 1766

Note: Where **s**= **setuid + execute** permission and **S**= **setuid only**. Same is for **SGID** and also for sticky bit.

SUID – SET USER ID

Change user ID on execution. If SETUID bit is set, when the file will be executed by a user, the process will have the same rights as the owner of the file being executed. Many of the system commands are the best example for SUID, basically the owner of the commands will be **root**, but still a normal user can execute it.

Example

- By default passwd command is having uid, so all users can run that command but if uid is removed and a normal user wants to user execute it, then they would not be able to use it to update /etc/shadow with new passwd.

```
[root@client ~]# which passwd
/usr/bin/passwd
[root@client ~]# ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 27832 Jan 29 2014 /usr/bin/passwd
```

Note: observe that in the permissions “**-rwsr-xr-x**” it contains an “**s**”, which means SUID is placed.

- Let's remove uid on passwd command and logged in as normal user and check the results

```
[root@client ~]# chmod u-s /usr/bin/passwd
[root@client ~]# ls -l /usr/bin/passwd
-rwxr-xr-x. 1 root root 27832 Jan 29 2014 /usr/bin/passwd
[root@client ~]# su - myuser
Last login: Sun Jan 13 08:16:01 EST 2019 on pts/0
[myuser@client ~]$ passwd
Changing password for user myuser.
Changing password for myuser.
(current) UNIX password:
New password:
Retype new password:
passwd: Authentication token manipulation error
```

SGID – SET GROUP ID

Set group ID, used on executable files to allow the file to be run as if logged into the group (like SUID but uses file group permissions)

SGID can also be used on a directory so that every file created in that directory will have the directory group owner rather than the group owner of the user creating the file.

Example

- When a directory is created and its group is set to some group. Now if SGID is applied to it, and the group member creates files and directory inside it, then it will get the same group rather than getting user's primary group
- Let's see it practically.

```
[root@mlinux7 ~]# mkdir mydir
[root@mlinux7 ~]# chgrp mygroup mydir
[root@mlinux7 ~]# ls -ld mydir
drwxr-xr-x. 2 root mygroup 6 Feb 16 16:25 mydir
[root@mlinux7 ~]# chmod 777 mydir
[root@mlinux7 ~]# ls -ld mydir
drwxrwxrwx. 2 root mygroup 6 Feb 16 16:25 mydir
```

- Login as other user, access the directory, create some files and check the group it is getting. (It will be getting the logged in user's group)

```
[u1@mlinux7 ~]$ whoami
u1
[u1@mlinux7 ~]$ cd /mydir
[u1@mlinux7 mydir]$ touch f1
[u1@mlinux7 mydir]$ ll
total 0
-rw-rw-r--. 1 u1 u1 0 Feb 16 16:38 f1
[u1@mlinux7 mydir]$
```

- Now, as a root, assign SGID on the directory
- #chmod g+s /mydir

```
[root@mlinux7 ~]# chmod g+s /mydir
[root@mlinux7 ~]# ls -ld /mydir
drwxrwsrwx. 2 root mygroup 15 Feb 16 16:38 /mydir
```

- Try creating other files with other user(s) in the same directory, instead of getting it's own group it would now be inheriting directory's group

```
[u1@mlinux7 mydir]$ whoami
u1
[u1@mlinux7 mydir]$ touch f2 f3
[u1@mlinux7 mydir]$ ll
total 0
-rw-rw-r--. 1 u1 u1 0 Feb 16 16:38 f1
-rw-rw-r--. 1 u1 mygroup 0 Feb 16 16:51 f2
-rw-rw-r--. 1 u1 mygroup 0 Feb 16 16:51 f3
```

STICKY BIT

If sticky bit is applied on a file or directory, then only root and owner of that file or directory can delete it. Even if others are having full permissions they cannot delete or edit the contents of the directory

- Let see it practically.
- Apply sticky bit to the directory

```
[root@mlinux7 /]# ls -ld /mydir
drwxrwsrwx. 2 root mygroup 33 Feb 16 16:51 /mydir
[root@mlinux7 /]# chmod o+t /mydir
[root@mlinux7 /]# ls -ld /mydir
drwxrwsrwt. 2 root mygroup 33 Feb 16 16:51 /mydir
```

- Access the directory with other user and try deleting the files

```
[u2@mlinux7 ~]$ 
[u2@mlinux7 ~]$ whoami
u2
[u2@mlinux7 ~]$ cd /mydir
[u2@mlinux7 mydir]$ ll
total 0
-rw-rw-r--. 1 u1 u1      0 Feb 16 16:38 f1
-rw-rw-r--. 1 u1 mygroup 0 Feb 16 16:51 f2
-rw-rw-r--. 1 u1 mygroup 0 Feb 16 16:51 f3
[u2@mlinux7 mydir]$ rm -f f1
rm: cannot remove 'f1': Operation not permitted
[u2@mlinux7 mydir]$ rm -f f2
rm: cannot remove 'f2': Operation not permitted
[u2@mlinux7 mydir]$ rm -f f3
rm: cannot remove 'f3': Operation not permitted
```

Note: If the owner or root tries to modify or delete the contents it would be allowed

2. Access Control List (ACL)

- Define more fine-grained discretionary access rights for files and directories.
- Often, you want to share files among certain groups and specific users. It is a good practice to designate a directory for that purpose. You want to allow those groups and users to read, and write files in that directory, as well as create new files into the directory. Such special permissions can be given using ACL.

To check the acl permission syntax is:

- `#getfacl <option> <dir/file name>`
- **Options:**
- `-d` Displays the default ACL
- `-R` Recurses into subdirectories

`#getfacl /mydir`

```
[root@mlinux6 ~]# ls -ld /mydir
drwxr-xr-x. 3 root root 4096 Feb 16 21:38 /mydir
[root@mlinux6 ~]# getfacl /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
group::r-x
other::r-x
```

- Now let's assign full permission to the directory and then apply acl on it, so that we can analyze how acl will work.

```
[root@mlinux6 ~]# chmod 777 /mydir
[root@mlinux6 ~]# ls -ld /mydir
drwxrwxrwx. 3 root root 4096 Feb 16 21:38 /mydir
[root@mlinux6 ~]#
```

- Okay, now we are ready to apply acl, but first lets understand the command and option in details.
- The syntax to apply acl is
- `#setfacl <option> < argument > < file or directory name >`
- **The options are,**
- `-m` Modifies an ACL
- `-x` Removes an user/group from ACL
- `-R` Recurses into subdirectories
- `-b` completely banishing/removing the ACL from a file/directory

The possible arguments are

- `u: user`
- `g: group`

To assign **read and execute** permission to a particular user the syntax could be

- **#setfacl -m u: <username>: <permissions> <file or dir name>**
- **#setfacl -m u1:rx /mydir**
- Verify it by using **getfacl** command

```
[root@mlinux6 ~]# setfacl -m u1:rx /mydir
[root@mlinux6 ~]# getfacl /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
user:u1:r-x
group::rwx
mask::rwx
other::rwx
```

- Now login as u1 user and try to create a file inside /mydir, as we have not assigned write permission to u1 user, though it is having full permissions, still it will not allow u1 user to create a file inside it.

```
[root@mlinux6 ~]# su - u1
[u1@mlinux6 ~]$ ls -ld /mydir
drwxrwxrwx+ 3 root root 4096 Feb 16 21:38 /mydir
[u1@mlinux6 ~]$ cd /mydir
[u1@mlinux6 mydir]$ touch f1
touch: cannot touch `f1': Permission denied
[u1@mlinux6 mydir]$
```

Observe that when you check for the permissions it is showing a + sign after normal permission, that indicate that ACL is applied on this directory.

To assign read write and execute permission to a particular group

- **#setfacl -m g:<group name>:<permissions> <file or directory name>**
- **#setfacl -m g:g1:rwx /mydir**

```
[root@mlinux6 ~]# getfacl /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
user:u1:r-x
group::rwx
group:g1:rwx
mask::rwx
other::rwx
```

Now you know how to apply acl on any file or directory, let me just give one more examples which you can broaden your understandings.

Assigning read and execute permission for a user and a group at same time.

- #setfacl -m u:u1:rx,g:g1:rx /mydir

```
[root@mlinux6 ~]# setfacl -m u:u1:rx,g:g1:rx /mydir
[root@mlinux6 ~]# getfacl /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
user:u1:r-x
group::rwx
group:g1:r-x
mask::rwx
other::rwx
```

Likewise you can explore applying acl to any user, group, or others in many ways.

Removing acl for a particular user

- #setfacl -x u:<username> <dir name>
- #setfacl -x u1 /mydir; #setfacl -x u:u1 /mydir

```
[root@mlinux6 ~]# setfacl -x u1 /mydir
[root@mlinux6 ~]# getfacl /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
group::rwx
group:g1:r-x
mask::rwx
other::rwx
```

Removing acl for a particular group

- #setfacl -x g:<group name> <directory name>
- #setfacl -x g:g1 /mydir

```
[root@mlinux6 ~]# setfacl -x g:g1 /mydir
[root@mlinux6 ~]# getfacl /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
group::rwx
mask::rwx
other::rwx
```

Removing all ACL permissions from a file or directory

- `#setfacl -b <dir name>`
- `#setfacl -b /ktdir`

As we have removed acl for a group and a user, let's apply back some acl on **ktdir** and remove it using above command

```
[root@mlinux6 ~]# setfacl -m u:u1:rwx,u:u2:rw,u:u3:r,g:g1:rwx,g:g2:rwx,g:g3:--- /mydir
[root@mlinux6 ~]# getfacl /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
user:u1:rwx
user:u2:rw-
user:u3:r--
group::rwx
group:g1:rwx
group:g2:r-x
group:g3:---
mask::rwx
other::rwx

[root@mlinux6 ~]# setfacl -b /mydir
[root@mlinux6 ~]# getfacl /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
group::rwx
other::rwx
```

To Apply /Remove ACL recursively on a directory and its contents

- `#setfacl -Rm u:u1:rwx,g:g1:rw /mydir`
- `#setfacl -Rb /mydir` (To recursively banishing ACL from directory and its contents)

```
[root@mlinux6 /]# setfacl -Rm u:u1:rwx,g:g1:rw /mydir
[root@mlinux6 /]# ls -l /mydir
total 28
-rw-rwrxr--+ 1 root root      0 Feb 16 22:22 f1
-rw-rwrxr--+ 1 root root      0 Feb 16 22:22 f2
-rw-rwrxr--+ 1 root root      0 Feb 16 22:22 f3
drwxrwxrwx---+ 2 root root 16384 Feb 16 21:38 lost+found
```

- To remove a user or group from acl
- #setfacl -Rx g:g1 /mydir (use #setfacl -Rx u1 /mydir for a user)

```
[root@mlinux6 ~]# setfacl -Rx u1 /mydir
[root@mlinux6 ~]# getfacl -R /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
group::rwx
group:g1:rwx-
mask::rwx
other::rwx

# file: mydir/lost+found
# owner: root
# group: root
user::rwx
group::---
group:g1:rwx-
mask::rwx-
other::---
```

To remove ACL completely from a directory recursively

- #setfacl -Rb /mydir

```
[root@mlinux6 ~]# setfacl -Rb /mydir
[root@mlinux6 ~]# ls -l /mydir
total 16
-rw-r--r--. 1 root root      0 Feb 16 22:22 f1
-rw-r--r--. 1 root root      0 Feb 16 22:22 f2
-rw-r--r--. 1 root root      0 Feb 16 22:22 f3
drwx-----. 2 root root 16384 Feb 16 21:38 lost+found
[root@mlinux6 ~]# getfacl -R /mydir
getfacl: Removing leading '/' from absolute path names
# file: mydir
# owner: root
# group: root
user::rwx
group::rwx
other::rwx
```

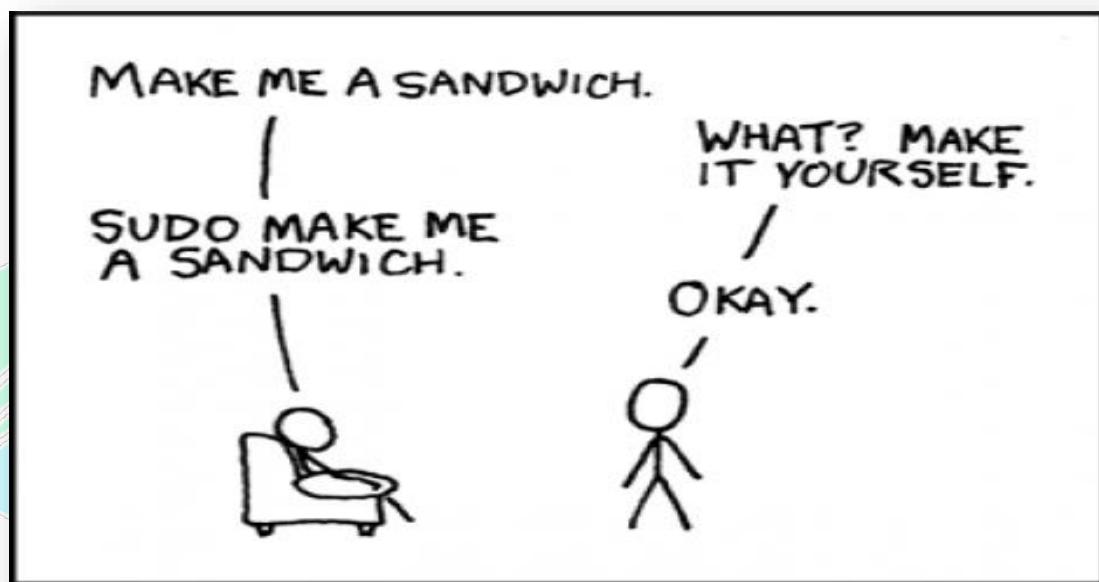


There are still much more experiments can be done, go ahead and read man pages for more details

ENHANCED USER SECURITY WITH SUDO

SUDO

- Sudo stands for either "substitute user do" or "super user do" (depending upon how you want to look at it). What sudo does is incredibly important and crucial to many Linux distributions. Effectively, sudo allows a user to run a program as another user (most often the root user). There are many that think sudo is the best way to achieve "best practice security" on Linux
- Users can login using their username and password and can issue administrative commands placing sudo in front of the commands, e.g. sudo rpm -Uvh *.rpm , to run the command which installs and updates programs in Linux (rpm).



- The file **/etc/sudoers** file has the rules that users have to follow when using sudo command. That means that whatever commands access is provided to any user in **/etc/sudoers** file, that user can only run those commands.
- Do not edit the **/etc/sudoers** directly; instead use "**visudo**" command to edit the sudoers file. There are two reasons for that- it prevents two users from editing the file at the same time, and it also provides limited syntax checking. Even if you are the only root user, you need the syntax checking, so use "visudo".

Advantages of using SUDO

Two of the best advantages about using sudo are:

- **Limited user privileges**

As we have studied above that we can restrict users to use certain commands as a privileged user as per the role of the user.

E.g.: Networking commands for Network user and Admin commands for Admin users etc.

- **Logs of the actions done by users**

All commands executed by sudo users will be stored in **/var/log/secure** file, but still if you want you can make your own log file by passing an entry in **/etc/sudoers** file at the bottom as “**Defaults logfile=/var/log/sudo.log**” or whatever name you want, to save the logs of what commands is executed by which sudo user.

The /etc/sudoers file

- As we learnt above that it is the configuration file for sudo users, which is used to assign specific commands to the specific users or groups.
- Always use **visudo** command to edit this file. it prevents two users from editing the file at the same time, and it also provides limited syntax checking .
- When you run **visudo** command the output will be as follows

```
## Next comes the main part: which users can run what software on
## which machines (the sudoers file can be shared between multiple
## systems).
## Syntax:
##
##       user      MACHINE=COMMANDS
##
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)        ALL
```

- As you can see there is basically one line
- **root ALL=(ALL) ALL**
- This lines means that the user root can execute from ALL terminals, acting as ALL (any) users, and run ALL (any) command.
- So the first part is the **user**, the second is the **terminal** from where the user can use sudo, the third is **as which user he may act**, and the last one, is which **commands** he may run.
- The advantage of **visudo** command , while editing if there are any syntax error it will be reflected as follows

```
[root@ cl5 ~]#
[root@ cl5 ~]# visudo
>>> /etc/sudoers: [syntax error near line 93] <<<
visudo: Warning: unused User_Alias   ADMIN
What now? ■
```

LAB WORK:-

Allow a user "myuser" all privileges like root

- To assign root privileges to user add a line by using sudoers file as shown below.
#visudo (save the sudoers file as we save a vim file using "**wq!**")

```
##      user      MACHINE=COMMANDS
##
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root  ALL=(ALL)      ALL
myuser  ALL=(ALL)      ALL
```

- Now logged in as myuser and run admin commands like **fdisk -l** etc
- First try to run fdisk command normally and see what happens.

```
[root@ cl5 ~]# su - myuser
[myuser@ cl5 ~]$ fdisk -l
[myuser@ cl5 ~]$ fdisk /dev/sda

Unable to open /dev/sda
[myuser@ cl5 ~]$
```

It will not allow a normal user to run privileged user's command

- Now run the same command using **sudo** before command
#sudo fdisk -l (or) **#sudo fdisk /dev/sda**

```
[myuser@ cl5 ~]$ sudo fdisk -l
[sudo] password for myuser:

Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0002e9e6

Device Boot      Start        End      Blocks   Id
/dev/sda1            1       1785    14336000   8e
/dev/sda2      *        1785       1811      204800   83
/dev/sda3            1811       1941     1048576   82
```

Note: Only for the first time of the session it will prompt for user's password to continue, but for rest of the process it will continue normally as shown below

```
[myuser@ cl5 ~]$ sudo fdisk /dev/sda
```

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to switch off the mode (command 'c') and change display units to sectors (command 'u').

Command (m for help):

Allow a group called mygroup, all root privileges.

- Let's first check the members of mygroup and then apply root privileges.

```
#tail /etc/gshadow
```

```
[root@ cl5 ~]# tail /etc/gshadow
fuse:::
stapdev:::
stapusr:::
gdm:::
student::::
myuser:::
mygroup:::musab,rahul,sai
musab:::
rahul:::
sai::::
[root@ cl5 ~]#
```

- Okay as we know the users in mygroup, let's assign it root privileges.

```
#visudo and look for the below line.
```

```
## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING,
## Allows people in group wheel to run all commands
# %wheel      ALL=(ALL)      ALL
%mygroup      ALL=(ALL)      ALL
```

- Now, login as one of the user of mygroup try root commands

```
[root@ cl5 ~]# su - musab
[musab@ cl5 ~]$ sudo parted -l
```

We trust you have received the usual lecture from the local System Administrator. It usually boils down to these three things:

- #1) Respect the privacy of others.
- #2) Think before you type.
- #3) With great power comes great responsibility.

```
[sudo] password for musab:
Model: VMware, VMware Virtual S (scsi)
Disk /dev/sda: 21.5GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type      File system    Flags
 1      1049kB  14.7GB  14.7GB  primary               lvm
 2      14.7GB   14.9GB  210MB   primary    ext4          boot
 3      14.9GB   16.0GB  1074MB  primary   linux-swap(v1)
```

Allow a user "myuser2" to run all commands without prompting for his password any time.

- To allow run all commands, the syntax we have already seen, but allow him run command's without prompting password a small change is to be made,

```
## Allows people in group wheel to run all commands
# %wheel      ALL=(ALL)      ALL
# %mygroup    ALL=(ALL)      ALL
## Same thing without a password
# %wheel      ALL=(ALL)      NOPASSWD: ALL
myuser2     ALL=(ALL)      NOPASSWD: ALL
```

- Now login as that user and check whether password is prompted or not

```
[root@ cl5 ~]# su - myuser2
[myuser2@ cl5 ~]$ sudo parted -l
Model: VMware, VMware Virtual S (scsi)
Disk /dev/sda: 21.5GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type      File system    Flags
 1      1049kB  14.7GB  14.7GB  primary   ext4          lvm
 2      14.7GB  14.9GB  210MB   primary   ext4          boot
 3      14.9GB  16.0GB  1074MB  primary   linux-swap(v1)
```

Note: - The same can be done for groups also, try it!

Restrict a user "myuser" to run only two root commands.

- This task is very simple; just modify the previous permissions assign to myuser.
- Let's give myuser to run only #fdisk and #parted command access.
- First check the complete path of those command by using following command

#which fdisk

#which parted

```
[root@ cl5 ~]# which fdisk
/sbin/fdisk
[root@ cl5 ~]# which parted
/sbin/parted
[root@ cl5 ~]#
```

- Let's assign both above paths in sudoers file

#visudo

```
## Syntax:
##
##      user      MACHINE=COMMANDS
##
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)      ALL
myuser  ALL=(ALL)      /sbin/fdisk, /sbin/parted
```

- Login as myuser and try assigned commands and other commands as well

```
[root@ cl5 ~]# su - myuser
[myuser@ cl5 ~]$ sudo fdisk -l
[sudo] password for myuser:

Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes

[myuser@ cl5 ~]$ sudo parted -l
Model: VMware, VMware Virtual S (scsi)
Disk /dev/sda: 21.5GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
[myuser@ cl5 ~]$ sudo useradd maarij
Sorry, user myuser is not allowed to execute '/usr/sbin/useradd maarij' as root
on cl5.my.com.
[myuser@ktcl5 ~]$
```

Note: - Try the same for groups also. It is exactly same

Allow a group “mygroup” to run only network related commands as sudo user

- To allow a group run only network commands, first uncomment the following line

```
## Networking
# Cmnd_Alias NETWORKING = /sbin/route, /sbin/ifconfig, /bin/ping, /sbin/dhclient, /usr/bin/net,
sbin/iptables, /usr/bin/rfcomm, /usr/bin/wvdial, /sbin/iwconfig, /sbin/mii-tool

## Networking
Cmnd_Alias NETWORKING = /sbin/route, /sbin/ifconfig, /bin/ping, /sbin/dhclient, /usr/bin/net,
bin/iptables, /usr/bin/rfcomm, /usr/bin/wvdial, /sbin/iwconfig, /sbin/mii-tool
```

Observe that we have just remove ‘#’ before the line to make the line readable. And also observe that it contains all networking commands.

- Just replace “**ALL**” with “**NETWORKING**” from the last field of mygroup line.

```
## Allows people in group wheel to run all commands
# %wheel        ALL=(ALL)        ALL
%mygroup        ALL=(ALL)        NETWORKING
```

NOTE: - **NETWORKING** is the name of the command alias, which was uncommented line.

- Now login as one of the member of mygroup and try some commands assigned it.

```
[root@ cl5 ~]# su - rahul
[rahul@ cl5 ~]$ sudo fdisk -l
[sudo] password for rahul:
Sorry, user rahul is not allowed to execute '/sbin/fdisk -l' as root on cl5.my.com.
[rahul@ cl5 ~]$ sudo route
[sudo] password for rahul:
Kernel IP routing table
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface
192.168.10.0     *               255.255.255.0   U         0      0        0 eth0
link-local       *               255.255.0.0     U         1002   0        0 eth0
```

Create a customize commands alias and assign it to mygroup with network command.

- Okay, first we need to create an alias say "CUSTOM" with some commands and assign it to mygroup in addition to NETWORK commands.
- Let's firs get the path of the command need to be in CUSTOM alias

```
[root@ cl5 ~]# which service
/sbin/service
[root@ cl5 ~]# which rpm
/bin/rpm
[root@ cl5 ~]# which yum
/usr/bin/yum
[root@ cl5 ~]#
```

- Okay, now let's create an alias for these commands and assign it to mygroup
- **#visudo**

```
## Networking
Cmnd_Alias NETWORKING = /sbin/route, /sbin/ifconfig, /bin/ping,
/bin/iptables, /usr/bin/rfcomm, /usr/bin/wvdial, /sbin/iwconfig,
## Custom
Cmnd_Alias CUSTOM = /sbin/service, /bin/rpm, /usr/bin/yum
```

- What are you waiting for! Assign it to ktgroup and save the file.

```
## Allows people in group wheel to run all commands
# %wheel      ALL=(ALL)        ALL
%mygroup      ALL=(ALL)        NETWORKING, CUSTOM
```

- Login as one of the users in mygroup and try newly added commands.

```
[root@ tcl5 ~]# su - rahul
[rahul@ cl5 ~]$ sudo rpm -q vsftpd
vsftpd-2.2.2-6.el6.i686
[rahul@ cl5 ~]$ sudo yum list installed |grep -i vsftpd
This system is not registered with RHN.
RHN support will be disabled.
vsftpd.i686                               2.2.2-6.el6
                                         enterpriseLinux-201009221732.i386/6.0
```

Allowing user all commands except few commands

- Most of the time it might require to block few commands from users, even though all commands have been granted. It is an important aspect as per the security view.
- Let's assume we want to block "**visudo**" and "**su**" commands
- Find out the path of those two command needs to be block

```
[root@mlinux7 ~]# which su
/usr/bin/su
[root@mlinux7 ~]# which visudo
/usr/sbin/visudo
[root@mlinux7 ~]#
```

- Now, allow all the commands to myuser and block the above two by using an “!” exclamation mark

```
## Allow root to run any commands anywhere
root    ALL=(ALL)          ALL
myuser  ALL=(ALL)          ALL, !/usr/bin/su, !/usr/sbin/visudo
```

- Login as myuser, try other commands and then go for blocked one and check whether they are allowed or not

```
[root@mlinux7 ~]# su - myuser
Last login: Fri Feb 24 23:05:16 IST 2017 on pts/0
[myuser@mlinux7 ~]$ sudo route
[sudo] password for myuser:
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
192.168.100.0   0.0.0.0        255.255.255.0  U      100    0        0 eno16777736
192.168.122.0   0.0.0.0        255.255.255.0  U      0       0        0 virbr0
[myuser@mlinux7 ~]$ sudo whoami
root
[myuser@mlinux7 ~]$ sudo who am i
root      pts/1        2017-02-24 23:41  (192.168.100.3)
```

Now let's try the blocked ones:

```
[myuser@mlinux7 ~]$ sudo su -
Sorry, user myuser is not allowed to execute '/bin/su -' as root on mlinux7.mb.com.
[myuser@mlinux7 ~]$ sudo visudo
Sorry, user myuser is not allowed to execute '/sbin/visudo' as root on mlinux7.mb.com.
[myuser@mlinux7 ~]$
```

Note: As a user to know how many commands are allowed via sudo, login as a user and run #sudo -l command

Note: Checkout sudoers file for more option and try it out on your own!!!!

Network Configuration & Troubleshooting

Networking:

It is a connection between two or more machines to communicate with each other.

The basic requirements for Networking are:

1. **NIC (Network Interface Controller or Card)**
2. **Media**
3. **Topology**
4. **Protocol**
5. **IP Addresses**

1. NIC (Network Interface Controller or Card)

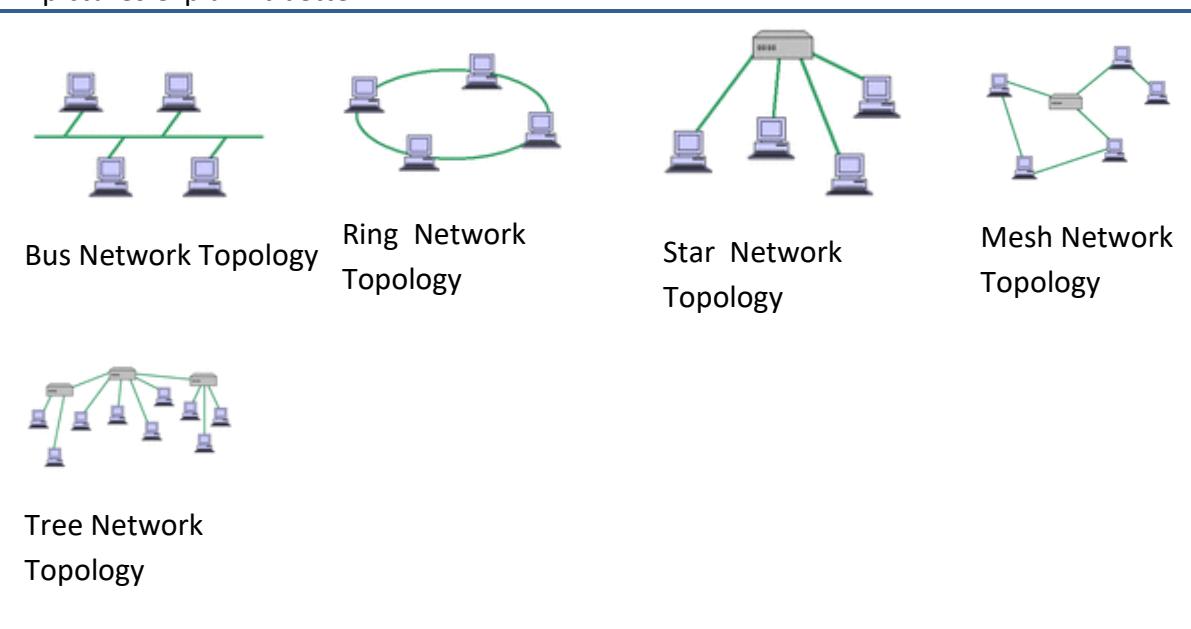
A **network interface controller** (also known as a **network interface card**, **network adapter**, **LAN adapter** and by similar terms) is a computer hardware component that connects a computer to a computer network. Each NIC will be having a unique MAC addresses (**Media Access Control address**) to avoid conflicts between same NIC adapters. In Linux these NIC adapter is represented by the word “**eth**” or “**ens**”. Example if there are two Ethernet adapters in the system then it will be denoted as eth0, eth1, ens1, ens2 etc.

2. Media

Media is the medium via which two different computer's NIC card will be connected. The best example for media is Cable. Example **RJ 45, CAT 5** etc.

3. Topology

Topology is the scheme or design in which the computers in the network will be connected to each other. Example for topology is Bus, Ring, star, mesh, tree topologies. The following pictures explain it better.



4. Protocol

A **network protocol** defines rules and conventions for communication between network devices. Protocols for computer networking all generally use packet switching techniques to send and receive messages in the form of *packets*.

Network protocols include mechanisms for devices to identify and make connections with each other, as well as formatting rules that specify how data is packaged into messages sent and received. Some protocols also support message acknowledgement and data compression designed for reliable and/or high-performance network communication. Hundreds of different computer network protocols have been developed each designed for specific purposes and environments.

Examples for Protocols are **TCP/IP (Transmission Control Protocol)**, **UDP (User Datagram Protocol)**, **HTTP**. The most widely and regularly used protocols for transferring data are TCP and UDP. Let's analyze some basic differences between **TCP/IP** and **UDP**.

TCP/IP	UDP
Transmission Control Protocol	User Datagram Protocol
It is connection Oriented	Connectionless
Reliable	Non-Reliable
TCP Acknowledgement will be sent/received	No Acknowledgement for UDP
Slow Communication	Faster Communication
Protocol Number for TCP is 6	Protocol Number for UDP is 17
HTTP, FTP, SMTP uses TCP	DNS, DHCP uses UDP

5. IP ADDRESS

An IP address can be thought of as being similar to a phone number. Just as every person who communicates with a telephone is using a phone with a unique phone number, every computer that is on the Internet has a unique IP address. Not only on internet but within an organization every computer is assigned an IP address so that they can communicate with each other. Basically IP addressing is very deep concept. To understand the concept of IP address we need to understand some important aspect of IP Address which is

- IP Address Classes
- Subnet mask
- Gateway

The above concepts in IP Addressing are very important to understand networking clearly.

- [IP Address Classes](#)

The IP addresses are further broken down into classes. These classes are A, B, C, D, E and their possible ranges can be seen in Figure below.

Class	Start	End	Default subnet mask	CIDR
Class A	0.0.0.0	127.255.255.255	255.0.0.0	/8
Class B	128.0.0.0	191.255.255.255	255.255.0.0	/16
Class C	192.0.0.0	223.255.255.255	255.255.255.0	/24
Class D (multicast)	224.0.0.0	239.255.255.255		
Class E (reserved)	240.0.0.0	255.255.255.255		

*CIDR - Classless Inter-Domain Routing

* 127.0.0.0 to 127.255.255.255 is reserved for loopback address

[Loopback](#)

A special IP number (127.0.0.1), that is designated for the software loopback interface of a machine. 127.0.0.0 Through 127.255.255.255 is also reserved for loopback and is used for internal testing on local machines.

[Multicast](#)

Multicasting allows a single message to be sent to a group of recipients. **Emailing**, **teleconferencing**, are examples of multicasting. It uses the network infrastructure and standards to send messages.

- [Subnet Mask](#)

A subnet mask allows users to identify which part of an IP address is reserved for the network and which part is available for host use. By looking at the IP address alone, especially now with classless inter-domain routing, users cannot tell which part of the address is which. Adding the subnet mask or netmask gives users all the information needed to calculate network and host portions of the address with ease. In summary, knowing the subnet mask can allow users to easily calculate whether IP addresses are on the same subnet or not.

A commonly used netmask is a 24-bit netmask as seen below.

Netmask:	255.	255.	255.	0
Binary:	11111111	11111111	11111111	00000000
Netmask length	8	16	24	--

- Gateway

A gateway is a network point that provides entrance into another network. On the Internet, a node or stopping point can be either a gateway node or a host (end-point) node. Both the computers of Internet users and the computers that serve pages to users are host nodes. The computers that control traffic within your company's network or at your local Internet service provider (ISP) are gateway nodes.

For example let's say our network is 192.168. something and we want to send a file to other computer on 10.10.network, so we need a gateway to communicate between two computers of different networks.

Some Important configuration files/directories of network configurations

#/etc/sysconfig/network-scripts is the directory which keeps the configuration of network devices connected to the system.

<pre>[root@ linux network-scripts]# ls ifcfg-eth0 ifdown-isdn ifup-aliases ifcfg-lo ifdown-post ifup-bnep ifdown ifdown-ppp ifup-eth ifdown-bnep ifdown-routes ifup-ipp ifdown-eth ifdown-sit ifup-ipv6 ifdown-ipp ifdown-tunnel ifup-isdn ifdown-ipv6 ifup ifup-ppip</pre>	<pre>[root@mlinux7 network-scripts]# ls ifcfg-eno16777736 ifdown-ipp ifcfg-lo ifdown-ipv6 ifdown ifdown-isdn ifdown-bnep ifdown-post ifdown-eth ifdown-ppp ifdown-ib ifdown-routes</pre>
RHEL6	RHEL7/8

#/etc/sysconfig/network (in rhel6) & #/etc/hostname (in rhel7) is the files which keeps the information about the hostname assigned to the system. If you want to change the hostname permanently, you need to change the hostname in this file.

RHEL6
<pre>[root@mlinux6 ~]# cat /etc/sysconfig/network NETWORKING=yes HOSTNAME=mlinux6.mb.com</pre>

RHEL7/8
<pre>[root@mlinux7 ~]# cat /etc/hostname mlinux7.mb.com</pre>

#/etc/hosts a file which is responsible for resolving hostname into IP locally, in other word it acts as local DNS if DNS server is not accessible. There is no change In this file in either versions

<pre>[root@mlinux7 ~]# cat /etc/hosts 127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4 ::1 localhost localhost.localdomain localhost6 localhost6.localdomain6 192.168.100.10 mlinux6.mb.com mlinux6</pre>

#/etc/resolv.conf is a file which keeps the address of DNS server to which the clients will be accessing to resolve IP to hostname and hostname to IP.

<pre>[root@ linux ~]# cat /etc/resolv.conf # Generated by NetworkManager search vpts.com nameserver 192.168.10.98</pre>

LAB WORK:

- To check the ip address assign to all the interfaces

#ifconfig

```
[root@mlinux7 ~]# ifconfig
eno16777736: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.100.20 netmask 255.255.255.0 broadcast 192.168.100.255
        inet6 fe80::20c:29ff:fea:9af0 prefixlen 64 scopeid 0x20<link>
            ether 00:0c:29:1a:9a:f0 txqueuelen 1000 (Ethernet)
                RX packets 382 bytes 35593 (34.7 KiB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 246 bytes 42590 (41.5 KiB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 0 (Local Loopback)
            RX packets 532 bytes 43824 (42.7 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 532 bytes 43824 (42.7 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

- To check the ip of a particular interface

#ifconfig < adapter name >

#ifconfig eth0; #ifconfig ensxx

```
[root@mlinux7 ~]# ifconfig eno16777736
eno16777736: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.100.20 netmask 255.255.255.0 broadcast 192.168.100.255
        inet6 fe80::20c:29ff:fea:9af0 prefixlen 64 scopeid 0x20<link>
            ether 00:0c:29:1a:9a:f0 txqueuelen 1000 (Ethernet)
                RX packets 420 bytes 38917 (38.0 KiB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 267 bytes 46296 (45.2 KiB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

- To check the hostname of the system.

#hostname

```
[root@mlinux6 ~]# hostname
mlinux6.mb.com
```

- To check ip of the host

#hostname -i

```
[root@ linux ~]# hostname -i
192.168.10.98 127.0.0.1
```

- To check whether DNS is resolving or not

```
#host < ip address >
```

```
#host 192.168.10.95
```

```
[root@ linux ~]# host 192.168.10.98
98.10.168.192.in-addr.arpa domain name pointer linux.mb.com.
```

```
#host <hostname>
```

```
#host mylinux.kt.com
```

```
[root@ linux ~]# host linux.mb.com.
linux.mb.com has address 192.168.10.98
```

- Same with “nslookup” command

```
#nslookup < ip address >
```

```
#nslookup < hostname >
```

```
[root@ linux ~]# nslookup 192.168.10.98
Server:          192.168.10.98
Address:         192.168.10.98#53

98.10.168.192.in-addr.arpa      name = mb.com.
98.10.168.192.in-addr.arpa      name = linux.mb.com.
```

```
[root@ linux ~]# nslookup linux.mb.com
Server:          192.168.10.98
Address:         192.168.10.98#53

Name:    linux.mb.com
Address: 192.168.10.98
```



- The most common command used to check DNS function is “dig”

```
#dig <hostname>
```

```
[root@ linux ~]# dig linux.mb.com

; <>> DiG 9.7.0-P2-RedHat-9.7.0-5.P2.el6 <>>    linux.mb.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11898
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
; linux.mb.com.                      IN      A

;; ANSWER SECTION:
linux(mb).com.        10800   IN      A      192.168.10.98

;; AUTHORITY SECTION:
linux(mb).com.        10800   IN      NS     linux(mb).com.
```

With ip address

```
#dig -x <ip address>
#dig -x 192.168.10.98
```

```
[root@ linux ~]# dig -x 192.168.10.98

; <>> DiG 9.7.0-P2-RedHat-9.7.0-5.P2.el6 <>> -x 192.168.10.98
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4532
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;98.10.168.192.in-addr.arpa. IN PTR

;; ANSWER SECTION:
98.10.168.192.in-addr.arpa. 10800 IN PTR linux.mb.com.
98.10.168.192.in-addr.arpa. 10800 IN PTR mb.com.

;; AUTHORITY SECTION:
10.168.192.in-addr.arpa. 10800 IN NS linux.mb.com.
```

- Checking network connectivity using ping command

```
#ping < ip address >
#ping 192.168.10.95
```

```
[root@ linux ~]# ping 192.168.10.95
PING 192.168.10.95 (192.168.10.95) 56(84) bytes of data.
64 bytes from 192.168.10.95: icmp_seq=1 ttl=64 time=0.115 ms
64 bytes from 192.168.10.95: icmp_seq=2 ttl=64 time=0.140 ms
64 bytes from 192.168.10.95: icmp_seq=3 ttl=64 time=0.099 ms
64 bytes from 192.168.10.95: icmp_seq=4 ttl=64 time=0.111 ms
64 bytes from 192.168.10.95: icmp_seq=5 ttl=64 time=0.110 ms
^C
--- 192.168.10.95 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4431ms
rtt min/avg/max/mdev = 0.099/0.115/0.140/0.013 ms
```

Note: use **ctrl + c** to stop pinging.

- To limit the pinging for specific number of counts

```
#ping -c <counts> <ip address>
#ping -c 2 192.168.10.95
```

```
[root@ linux ~]# ping -c 2 192.168.10.95
PING 192.168.10.95 (192.168.10.95) 56(84) bytes of data.
64 bytes from 192.168.10.95: icmp_seq=1 ttl=64 time=0.100 ms
64 bytes from 192.168.10.95: icmp_seq=2 ttl=64 time=0.106 ms

--- 192.168.10.95 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.100/0.103/0.106/0.003 ms
```

Changing the hostname

- Check the current hostname with **hostname** command
- The syntax for changing the hostname is
#hostname <new name>
#hostname mlinux6.mb.com

```
[root@mlinux6 ~]# hostname
mlinux2.mb.com
[root@mlinux6 ~]# hostname mlinux6.mb.com
[root@mlinux6 ~]# hostname
mlinux6.mb.com
```

Note: The above change is temporary and will be last only till you are logged in, if you want to change it permanently edit the **/etc/sysconfig/network (rhel6)** and **/etc/hostname (rhel7)** file and then logout and login to confirm the change.

#vim /etc/sysconfig/network; #vim /etc/hostname delete the previous hostname and add the new name.

```
mlinux6.mb.com
• Now logoff and logon and check the hostname
[root@mlinux6 ~]# hostname
mlinux6.mb.com
```

Note: Once you logout and login again the change will be permanent, observe the highlighted region above.

- **hostnamectl command**

In rhel7 a new command has been introduced to see the hostname/system details as well as changing hostname permanently.

```
[root@node1 ~]# hostnamectl
  Static hostname: node1.private
    Icon name: computer-vm
    Chassis: vm
      Machine ID: 36462571f9e14f54b7cf5065ee8cd22b
        Boot ID: ca0f6031684a47e59abe13ab49790b23
      Virtualization: vmware
Operating System: Red Hat Enterprise Linux Server 7.6 (Maipo)
      CPE OS Name: cpe:/o:redhat:enterprise_linux:7.6:GA:server
        Kernel: Linux 3.10.0-957.el7.x86_64
      Architecture: x86-64
```

- Changing hostname permanently with **hostnamectl** command

#hostnamectl set-hostname <new name>

```
[root@node1 ~]# hostnamectl set-hostname mlinux7.vpts.com
[root@node1 ~]# hostname
mlinux7.vpts.com
[root@node1 ~]# cat /etc/hostname
mlinux7.vpts.com
```

Assigning /Changing the IP Address

Steps for changing the IP Address

To change the IP Address use the following command line

To see the all the available connection

#nmcli con show

```
[root@mlinux2 ~]# nmcli con show
NAME      UUID                                  TYPE      DEVICE
ens3     e4a31311-1ac5-4614-83ee-03629af62ddc 802-3-ethernet ens3
```

To see the details of any connection

#nmcli con show <connection name>

```
[root@mlinux2 ~]# nmcli con show ens3
connection.id:                           ens3
connection.uuid:                          e4a31311-1ac5-4614-83ee-03629af62ddc
connection.interface-name:                ens3
connection.type:                          802-3-ethernet
connection.autoconnect:                  yes
connection.autoconnect-priority:        0
connection.timestamp:                   1466229442

ipv4.addresses:                          192.168.106.82/24
ipv4.gateway:                           192.168.106.1
```

To add a new connection

#nmcli con add con-name <name of connection> ifname <interface name> type <type of connection> autoconnect <yes/no> ip4/ip6 <ip add> gw4/gw6 <gateway>

```
[root@mlinux2 ~]# nmcli con add con-name mycon ifname ens3 type ethernet autoconnect no
ip4 192.168.106.182/24 gw4 192.168.106.1
```

```
[root@mlinux2 ~]# nmcli con show
NAME      UUID                                  TYPE      DEVICE
mycon    268e6fb2-662f-4482-af68-161a1a4ad775 802-3-ethernet  --
ens3     e4a31311-1ac5-4614-83ee-03629af62ddc 802-3-ethernet ens3
```

To activate a connection

#nmcli con up <con name>

```
[root@mlinux2 ~]# nmcli con show
NAME      UUID                                  TYPE      DEVICE
mycon    268e6fb2-662f-4482-af68-161a1a4ad775 802-3-ethernet  --
ens3     e4a31311-1ac5-4614-83ee-03629af62ddc 802-3-ethernet ens3
[root@mlinux2 ~]# nmcli con up mycon
```

```
Pinging 192.168.106.182 with 32 bytes of data:
Request timed out.
Reply from 192.168.106.182: bytes=32 time=9ms TTL=63
Reply from 192.168.106.182: bytes=32 time=10ms TTL=63
Reply from 192.168.106.182: bytes=32 time=11ms TTL=63
Reply from 192.168.106.182: bytes=32 time=1ms TTL=63
```

```
[root@mlinux2 ~]# nmcli con show
NAME      UUID                                  TYPE      DEVICE
mycon    268e6fb2-662f-4482-af68-161a1a4ad775 802-3-ethernet ens3
ens3     e4a31311-1ac5-4614-83ee-03629af62ddc 802-3-ethernet --
```

To modify an existing connection

```
#nmcli con mod <con-name> attributes <value>
```

```
[root@mlinux2 ~]# nmcli con mod mycon ipv4.addresses 192.168.106.188/24
[root@mlinux2 ~]# nmcli con show mycon |grep ipv4.addresses
ipv4.addresses: 192.168.106.188/24
```

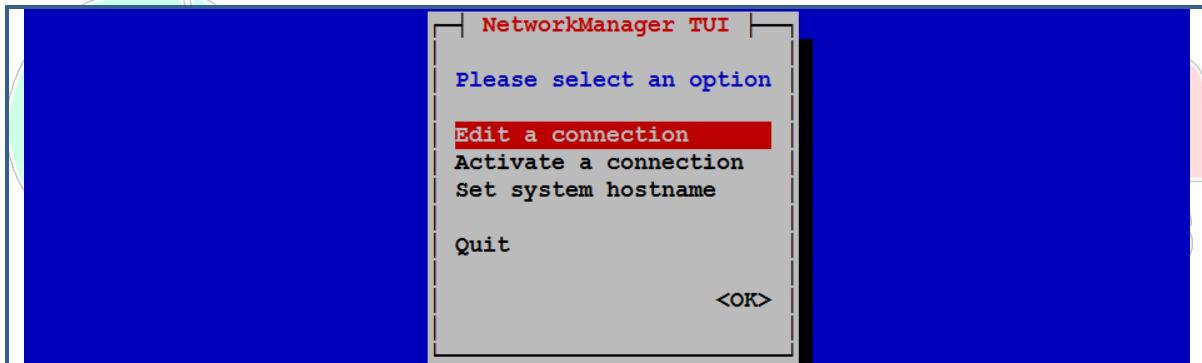
To delete a connection

```
#nmcli con del <con-name>
```

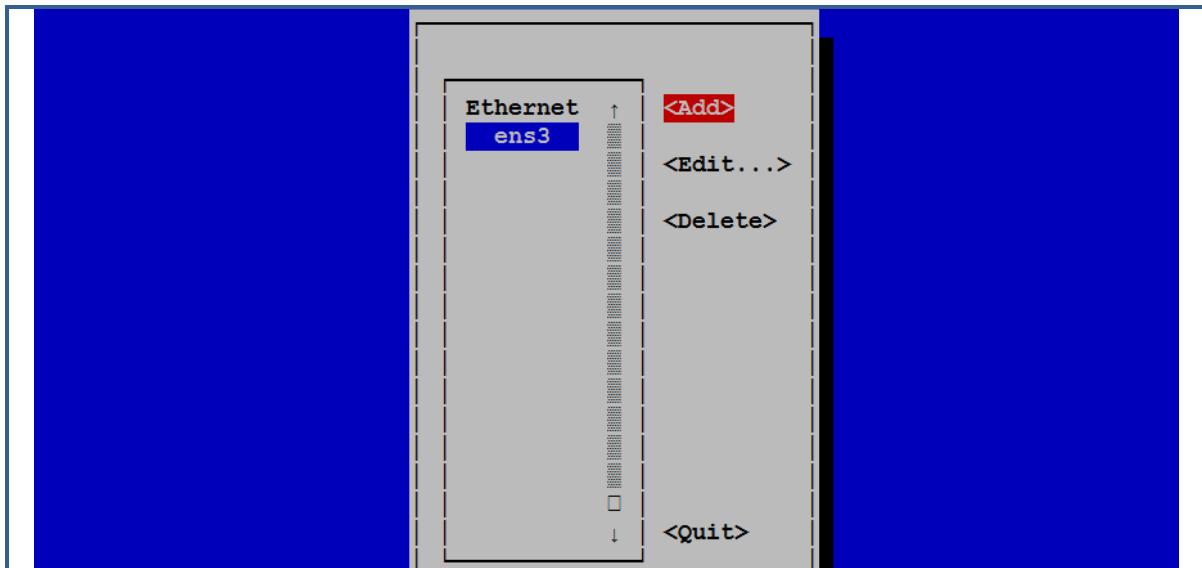
```
[root@mlinux2 ~]# nmcli con show
NAME      UUID                                  TYPE      DEVICE
mycon    268e6fb2-662f-4482-af68-161a1a4ad775  802-3-ethernet  --
ens3     e4a31311-1ac5-4614-83ee-03629af62ddc  802-3-ethernet  ens3
[root@mlinux2 ~]# nmcli con del mycon
[root@mlinux2 ~]# nmcli con show
NAME      UUID                                  TYPE      DEVICE
ens3     e4a31311-1ac5-4614-83ee-03629af62ddc  802-3-ethernet  ens3
[root@mlinux2 ~]#
```

Using nmtui for a text based tool

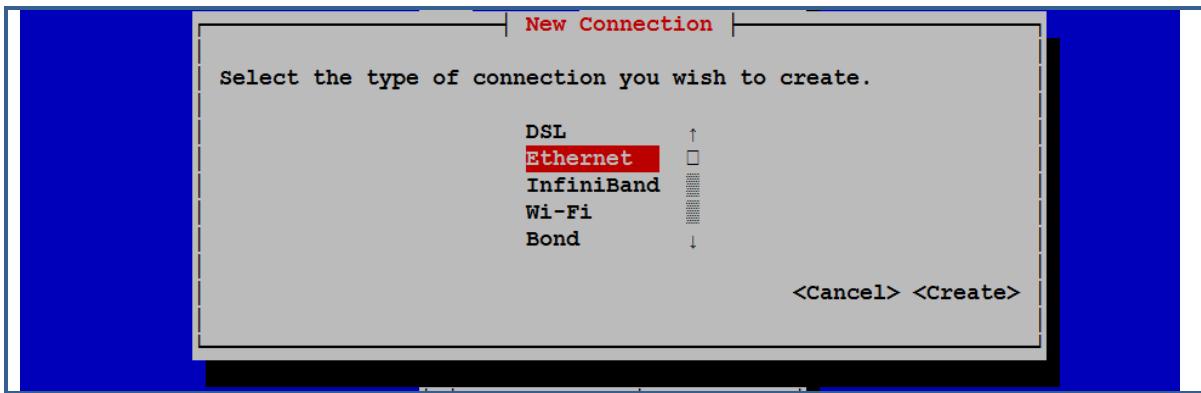
```
#nmtui
```



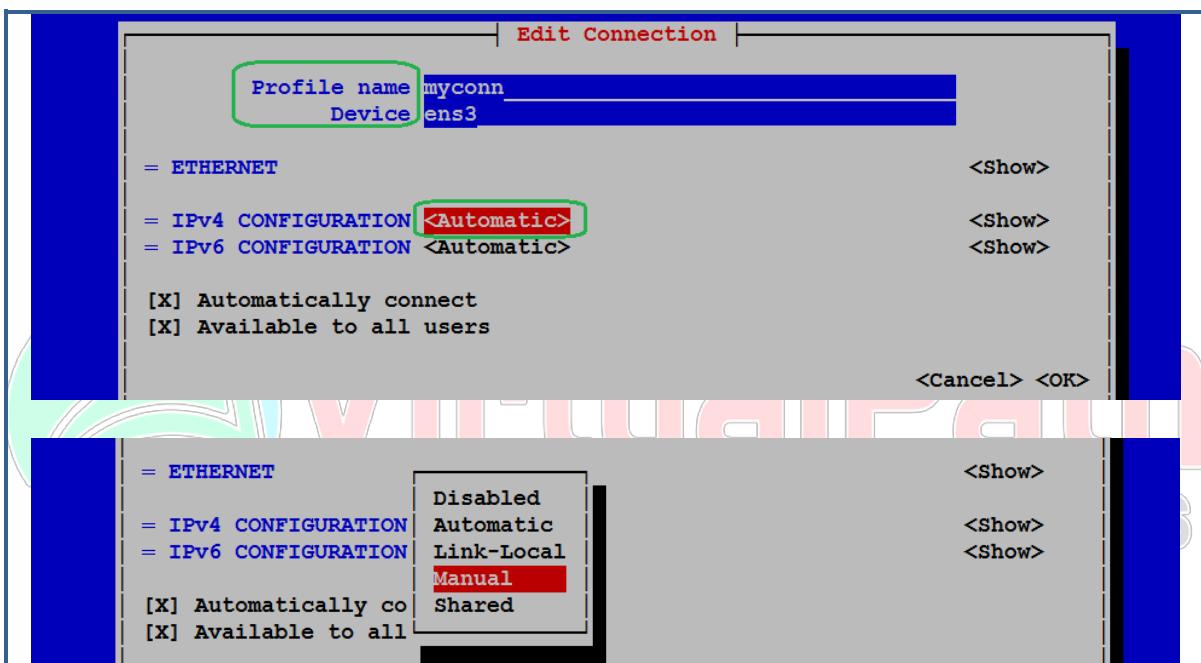
Move the cursor to *Edit a connection* and press Enter



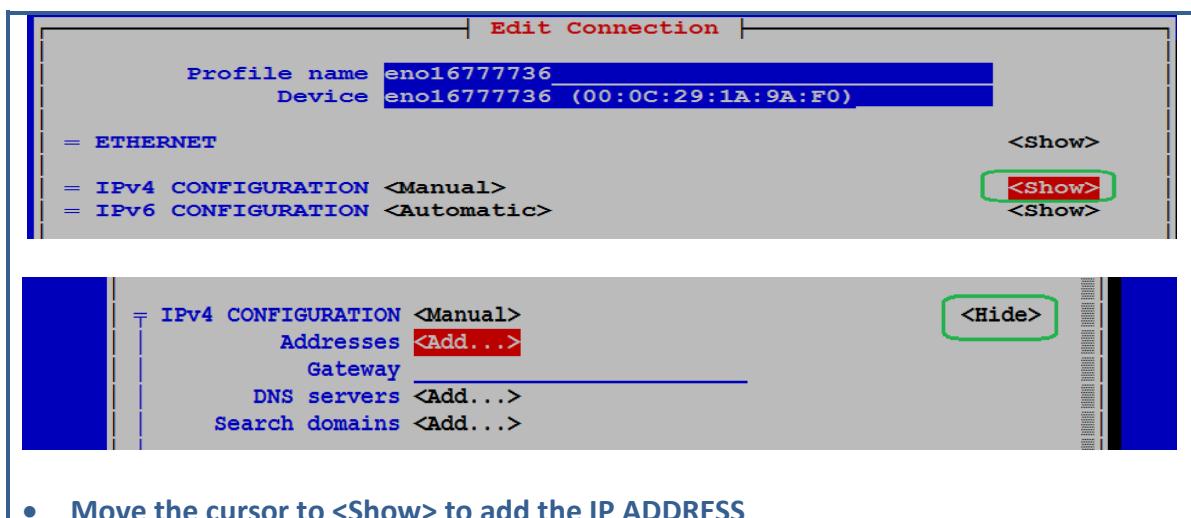
Move the cursor to '*Add*' to add a new connection and press Enter



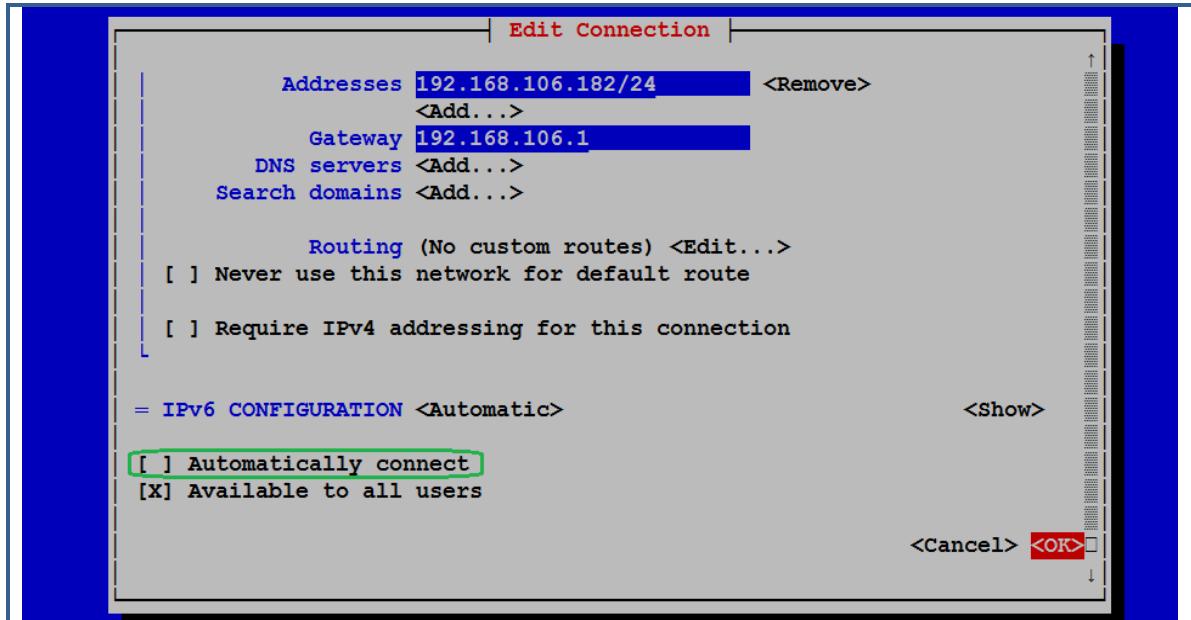
Now select the type of connection, ex., Ethernet and press Enter



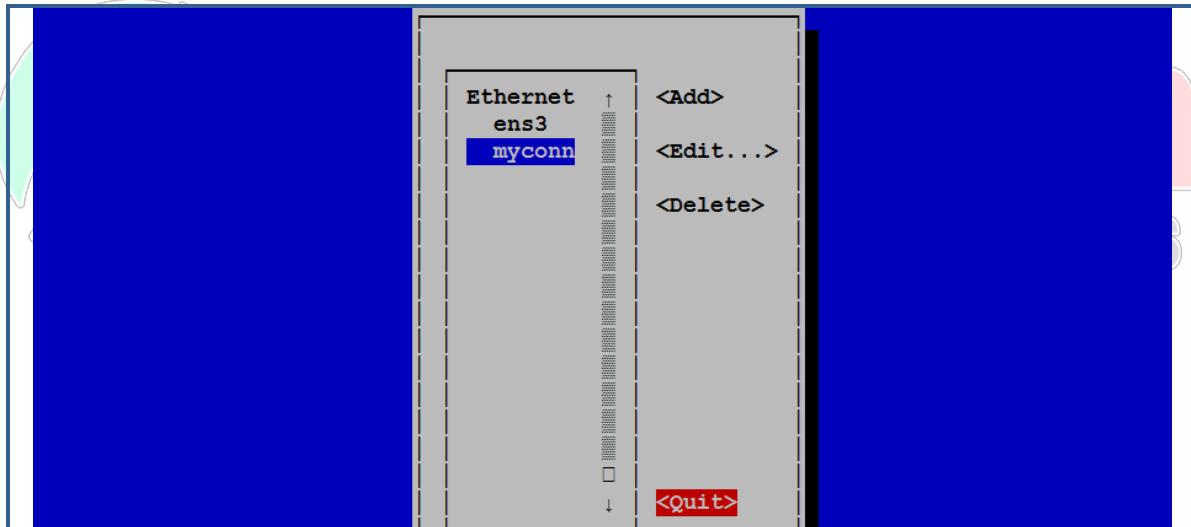
Assign the connection name, Device name and change the ipv4 from automatic to manual to continue



- Move the cursor to <Show> to add the IP ADDRESS



Move the cursor to “add” to assign the ip address move cursor to “ok” Enter.

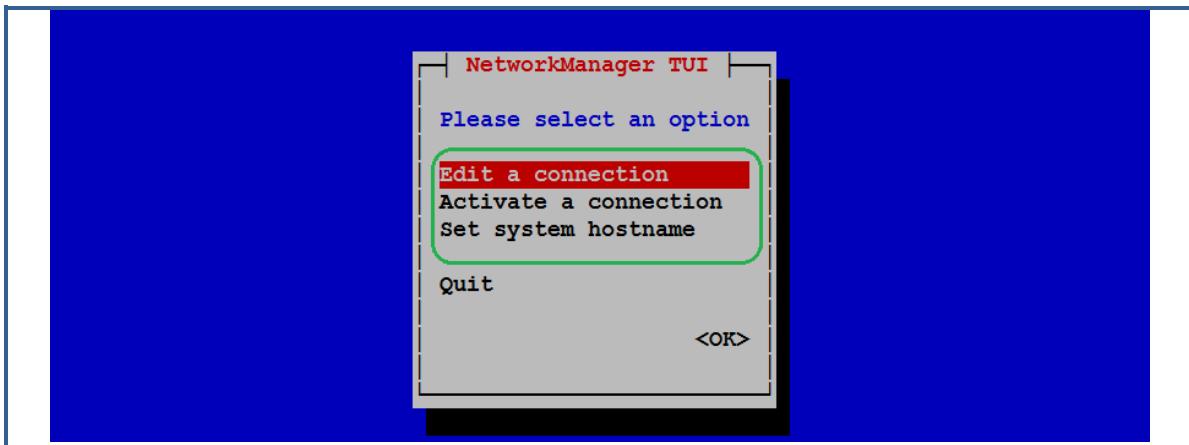


move the cursor to “Quit” button and press Enter to quit the utility

Check the connection by nmcli con show command

```
[root@mlinux2 ~]# nmcli con show
NAME      UUID                                  TYPE      DEVICE
myconn    62351fe9-9b6b-4596-b030-d4fe208c095f  802-3-ethernet --
ens3      e4a31311-1ac5-4614-83ee-03629af62ddc  802-3-ethernet ens3
```

In the same fashion, the hostname, connections can be created and modify



- To Know more about the NIC card/adapter use
#ethtool <adapter name>

```
[root@client ~]# ethtool ens192
Settings for ens192:
    Supported ports: [ TP ]
    Supported link modes:  1000baseT/Full
                           10000baseT/Full
    Supported pause frame use: No
    Supports auto-negotiation: No
    Supported FEC modes: Not reported
    Advertised link modes: Not reported
    Advertised pause frame use: No
    Advertised auto-negotiation: No
    Advertised FEC modes: Not reported
    Speed: 10000Mb/s
    Duplex: Full
    Port: Twisted Pair
    PHYAD: 0
    Transceiver: internal
    Auto-negotiation: off
    MDI-X: Unknown
    Supports Wake-on: uag
    Wake-on: d
    Link detected: yes
```



- To see the same in summarized way

mii-tool <adapter name>

```
[root@localhost ~]# mii-tool enp0s3
enp0s3: no autonegotiation, 1000baseT-FD flow-control, link ok
```

NIC TEAMING

NIC teaming is the concept of combining or bonding 2 or more network interfaces into one logical interface to provide high throughput and redundancy. This practice is popular especially with critical servers where high availability is expected at all times. In a server with 2 or more **NIC** cards, the concept of NIC teaming is critical in the event where one NIC card fails. With NIC teaming, the logical network interface will ensure that the remaining NIC will continue functioning and serving the purpose of the defective NIC. In this guide, we take you through the configuration of NIC teaming in CentOS 8 and RHEL 8.

Let's take a look at some of the concepts around



- **Teamd** – This is a daemon that allows you to configure a team network interface. Teamd is a part of the libteam project and leverages the libteam library for implementation of load balancing and round-robin logic.
- **Teamdctl** – This is a utility tool for querying an instance of teamd for configuration information and detailed statistics.
- **Runners** – These are distinct units of code in JSON format used for implementing different concepts of NIC teaming such as Round robin

The modes of the Runners:

- **Round-robin**: In this mode, data is transmitted across all ports in turn.
- **Load-balance**: Traffic is distributed across all NICs.
- **Active-backup**: Where one link or port is activated as the rest are reserved as a backup. This is used for failover to provide redundancy as we shall later see in this guide.

Steps to configure NIC teaming

Creating a NIC team adapter

```
$ nmcli connection add type team con-name team0 ifname team0 config
'{"runner": {"name": "activebackup"}}'
```

Check the details of network connection

nmcli connection show

```
[root@mlinux4 ~]# nmcli connection add type team con-name team0 ifname team0
config '{"runner": {"name": "activebackup"}}'
Connection 'team0' (1e0fc38a-261b-4be0-bf6d-3592ad543196) successfully added.
[root@mlinux4 ~]# nmcli con show
NAME      UUID                                  TYPE      DEVICE
team0    1e0fc38a-261b-4be0-bf6d-3592ad543196  team      team0
enp0s3   d7c9e8db-444c-4a3a-8b20-2720838fed85  ethernet  enp0s3
virbr0   ab4153b2-3de1-4bb4-9276-42c80dfa39a1  bridge    virbr0
virbr0   aabbda7-0a54-4bd4-95ce-7f68388e1b15  bridge    --
```

Modify the team connection properties as per requirement

```
$ nmcli con mod team0 ipv4.addresses 192.168.10.100/24
```

```
$ nmcli con mod team0 ipv4.gateway 192.168.10.1
```

```
$ nmcli con mod team0 ipv4.dns 8.8.8.8 → (optional)
```

```
$ nmcli con mod team0 ipv4.method manual
```

```
$ nmcli con mod team0 connection.autoconnect yes
```

```
[root@mlinux4 ~]# nmcli con mod team0 ipv4.addresses 192.168.10.100/24
[root@mlinux4 ~]# nmcli con mod team0 ipv4.gateway 192.168.10.1
[root@mlinux4 ~]# nmcli con mod team0 ipv4.method manual
[root@mlinux4 ~]# nmcli con mod team0 connection.autoconnect yes
```

Adding the slaves to the team

```
$ nmcli con add type team-slave con-name team0-slave1 ifname enp0s8
master team0
```

```
$ nmcli con add type team-slave con-name team0-slave2 ifname enp0s9
master team0
```

```
[root@mlinux4 ~]# nmcli dev status
DEVICE      TYPE      STATE           CONNECTION
enp0s3      ethernet  connected       enp0s3
team0       team      connected       team0
virbr0     bridge    connected       virbr0
enp0s8      ethernet  disconnected   --
enp0s9      ethernet  disconnected   --
lo          loopback unmanaged      --
virbr0-nic  tun      unmanaged      --
[root@mlinux4 ~]# nmcli con add type team-slave con-name team0-slave1 ifname
enp0s8 master team0
Connection 'team0-slave1' (309788a8-9c74-46e6-b6fc-2b686de93a37) successfully
added.
[root@mlinux4 ~]# nmcli con add type team-slave con-name team0-slave2 ifname
enp0s9 master team0
Connection 'team0-slave2' (818eb0b8-0d5a-4789-9540-bffa9bd3fb48) successfully
added.
```

To View the team interface and slaves, run the command:

\$ nmcli connection show

```
[root@mlinux4 ~]# nmcli con show
NAME           UUID                                  TYPE      DEVICE
enp0s3         d7c9e8db-444c-4a3a-8b20-2720838fed85  ethernet  enp0s3
team0          1e0fc38a-261b-4be0-bf6d-3592ad543196  team      team0
virbr0         ab4153b2-3de1-4bb4-9276-42c80dfa39a1  bridge    virbr0
team0-slave1   309788a8-9c74-46e6-b6fc-2b686de93a37  ethernet  enp0s8
team0-slave2   818eb0b8-0d5a-4789-9540-bffa9bd3fb48  ethernet  enp0s9
virbr0         aabbda7-0a54-4bd4-95ce-7f68388e1b15    bridge    --
```

Restart or just up the team0 connection

\$ nmcli connection down team0 && nmcli connection up team0

Or

\$nmcli con up (this is also fine to refresh a connection)

```
[root@mlinux4 ~]# nmcli connection down team0 && nmcli connection up team0
Connection 'team0' successfully deactivated (D-Bus active path: /org/freedesktop/NM/1/Connections/4)
Connection successfully activated (master waiting for slaves) (D-Bus active path: /org/freedesktop/NM/1/ActiveConnection/17)
```

Check the IP for team0

\$ip addr show team0 or ifconfig team0

```
[root@mlinux4 ~]# ifconfig team0
team0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.100  netmask 255.255.255.0  broadcast 192.168.10.255
              inet6 fe80::3876:b06c:9bfe:10f  prefixlen 64  scopeid 0x20<link>
                ether 08:00:27:17:25:42  txqueuelen 1000  (Ethernet)
                  RX packets 0  bytes 0 (0.0 B)
                  RX errors 0  dropped 0  overruns 0  frame 0
                  TX packets 25  bytes 3928 (3.8 KiB)
                  TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

To check the details of team

```
[root@mlinux4 ~]# teamdctl team0 state
setup:
  runner: activebackup
ports:
  enp0s8
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 0
  enp0s9
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 0
runner:
  active port: enp0s8
[root@mlinux4 ~]#
```

Testing the failover in active-backup team setup

To test the functionality of active-backup team setup fail the active adapter and check the network reachability

```
$ nmcli device disconnect enp0s8/ ifconfig down enp0s8
```

```
$ sudo teamdctl bond0 state
```

```
[root@mlinux4 ~]# nmcli dev disconnect enp0s8
Device 'enp0s8' successfully disconnected.
[root@mlinux4 ~]# teamdctl team0 state
setup:
  runner: activebackup
ports:
  enp0s9
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 0
runner:
  active port: enp0s9
[root@mlinux4 ~]#
```

Deleting the team adapter/connection

- Bring down team adapter/connection
- Delete slaves first and finally the team adapter/connection

```
[root@mlinux4 ~]# nmcli con down team0
Connection 'team0' successfully deactivated (D-Bus active path:
top/NetworkManager/ActiveConnection/17)

[root@mlinux4 ~]# nmcli con delete team0-slave1
Connection 'team0-slave1' (309788a8-9c74-46e6-b6fc-2b686de93a37)
deleted.
[root@mlinux4 ~]# nmcli con delete team0-slave2
Connection 'team0-slave2' (818eb0b8-0d5a-4789-9540-bffa9bd3fb48)
deleted.

[root@mlinux4 ~]# nmcli con delete team0
Connection 'team0' (1e0fc38a-261b-4be0-bf6d-3592ad543196)
d.

[root@mlinux4 ~]# nmcli con show
NAME      UUID                                     TYPE      DEVICE
enp0s3   d7c9e8db-444c-4a3a-8b20-2720838fed85  ethernet  enp0s3
virbr0   ab4153b2-3de1-4bb4-9276-42c80dfa39a1  bridge   virbr0
```

The same way you can create NIC-TEAM for load-balancer

- nmcli connection add type team con-name team0 ifname team0 config


```
'{"runner": {"name": "loadbalance"}}'
```

 or
- nmcli connection add type team con-name team0 ifname team0 config


```
'{"runner": {"name": "roundrobin"}}'
```

```
[root@mlinux4 ~]# nmcli connection add type team con-name team0 ifname team0
config '{"runner": {"name": "loadbalance"}}'
Connection 'team0' (6628f699-e295-43c8-adf1-eba6f877dc47) successfully added.
[root@mlinux4 ~]# nmcli con show
NAME      UUID                                  TYPE      DEVICE
team0     6628f699-e295-43c8-adf1-eba6f877dc47  team      team0
enp0s3   d7c9e8db-444c-4a3a-8b20-2720838fed85  ethernet  enp0s3
virbr0   ab4153b2-3de1-4bb4-9276-42c80dfa39a1  bridge    virbr0
```

Rest of the process is same as we did earlier by modifying team by adding IP and setting other parameters

```
[root@mlinux4 ~]# nmcli con mod team0 ipv4.addresses 192.168.10.100/24
[root@mlinux4 ~]# nmcli con mod team0 ipv4.method manual
[root@mlinux4 ~]# nmcli con mod team0 connection.autoconnect yes
```

Adding slaves to the team

```
[root@mlinux4 ~]# nmcli con add type team-slave con-name team0-slave1 ifname
enp0s8 master team0
Connection 'team0-slave1' (21df80cb-2a23-471a-8e37-6791733f9338) successfully
added.
[root@mlinux4 ~]#
[root@mlinux4 ~]# nmcli con add type team-slave con-name team0-slave2 ifname
enp0s9 master team0
Connection 'team0-slave2' (9de13455-1899-4efa-8f16-3da0c14dbfaa) successfully
added.
```

Restart the team0 connection to update the changes and check the IP and team details.

```
[root@mlinux4 ~]# nmcli connection down team0 && nmcli connection up team0
Connection 'team0' successfully deactivated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/21)
Connection successfully activated (master waiting for slaves) (D-Bus active p
ath: /org/freedesktop/NetworkManager/ActiveConnection/24)
[root@mlinux4 ~]# ifconfig team0
team0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.100  netmask 255.255.255.0  broadcast 192.168.10.255
              inet6 fe80::3903:9ff9:77f4:a812  prefixlen 64  scopeid 0x20<link>
                    ether 08:00:27:a1:13:77  txqueuelen 1000  (Ethernet)
                          RX packets 0  bytes 0 (0.0 B)
                          RX errors 0  dropped 0  overruns 0  frame 0
                          TX packets 22  bytes 3561 (3.4 KiB)
                          TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

Check the details of the team

```
[root@mlinux4 ~]# teamdctl team0 state
setup:
    runner: loadbalance
ports:
    enp0s8
        link watches:
            link summary: up
            instance[link_watch_0]:
                name: ethtool
                link: up
                down count: 0
    enp0s9
        link watches:
            link summary: up
            instance[link_watch_0]:
                name: ethtool
                link: up
                down count: 0
```

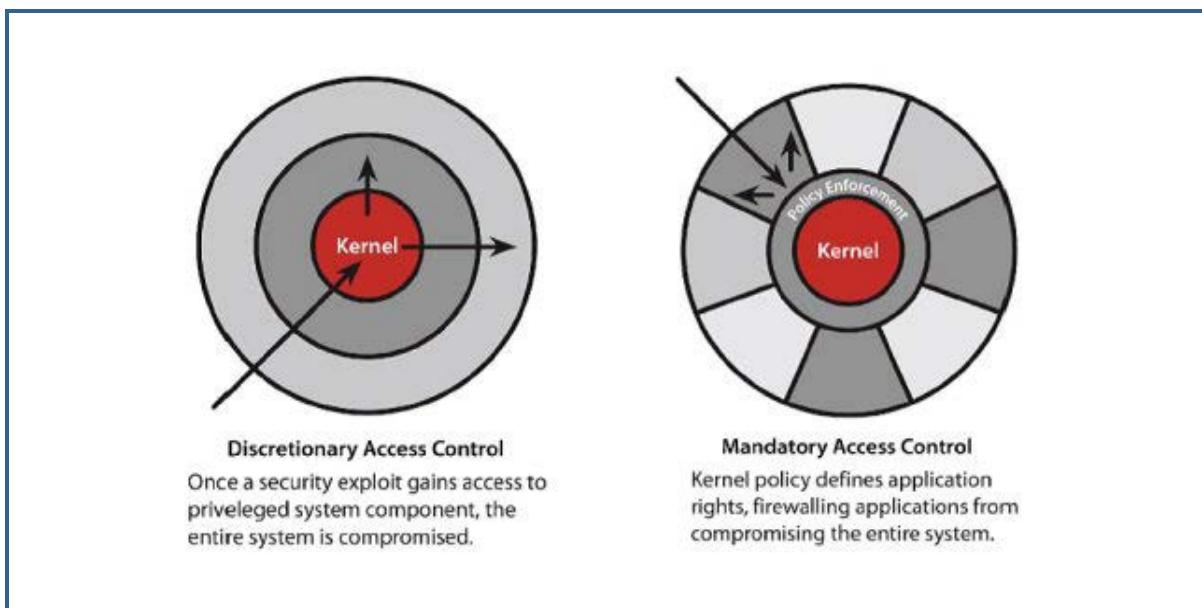
Testing the load-balance of the team

```
[root@mlinux4 ~]# ethtool team0 |grep -i speed
    Speed: 2000Mb/s
[root@mlinux4 ~]#
[root@mlinux4 ~]# nmcli dev disconnect enp0s8
Device 'enp0s8' successfully disconnected.
[root@mlinux4 ~]# teamdctl team0 state
setup:
    runner: loadbalance
ports:
    enp0s9
        link watches:
            link summary: up
            instance[link_watch_0]:
                name: ethtool
                link: up
                down count: 0
[root@mlinux4 ~]# ethtool team0 |grep -i speed
    Speed: 1000Mb/s
[root@mlinux4 ~]#
```

MANAGING SELINUX (BASICS OF SELINUX)

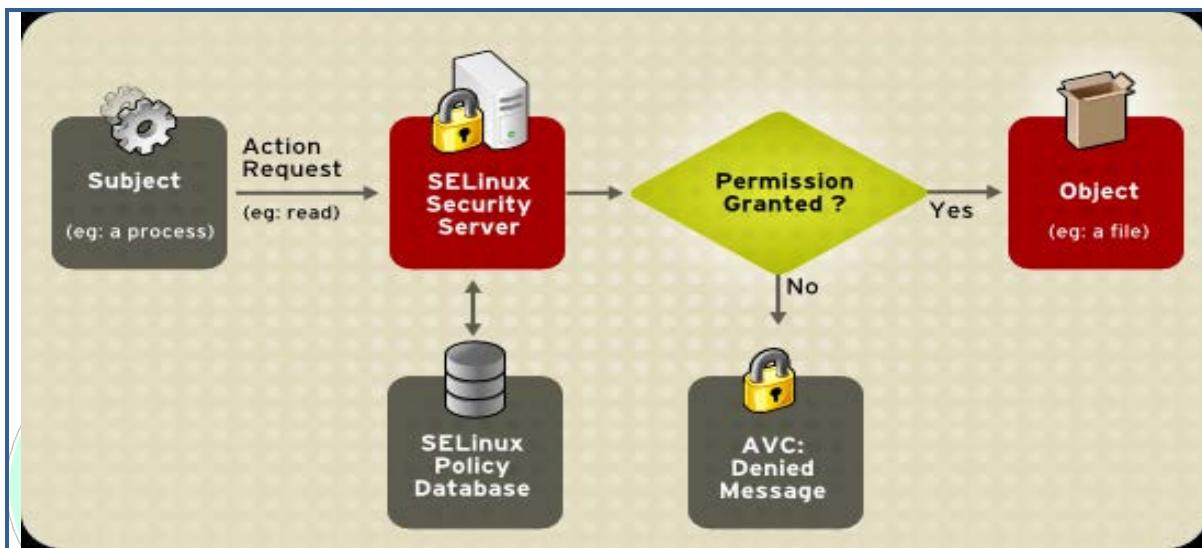
Basic SELinux Security Concepts

- SELinux is a security enhancement to Linux that allows users and administrators more control over which users and applications can access which resources, such as files. Standard Linux access controls, such as file modes (-rwxr-xr-x) are modifiable by the user and applications that the user runs, whereas SELinux access controls are determined by a policy loaded on the system and not changeable by careless users or misbehaving applications.
- SELinux also adds finer granularity to access controls. Instead of only being able to specify who can read, write or execute a file, for example, SELinux lets you specify who can unlink, append only, and move a file and so on. SELinux allows you to specify access to many resources other than files as well, such as network resources and inter-process communication (IPC).
- SELinux provides a flexible **Mandatory Access Control (MAC)** system built into the Linux kernel. Under standard Linux **Discretionary Access Control (DAC)**, an application or process running as a user (UID or SUID) has the user's permissions to objects such as files, sockets, and other processes. Running a MAC kernel protects the system from malicious or flawed applications that can damage or destroy the system. The following picture explains more detailed about both Access controls.



- The SELinux Decision Making Process**

When a subject, (for example, an application), attempts to access an object (for example, a file), the policy enforcement server in the kernel checks an *access vector cache* (AVC), where subject and object permissions are cached. If a decision cannot be made based on data in the AVC, the request continues to the security server, which looks up the *security context* of the application and the file in a matrix. Permission is then granted or denied, with an avc: denied message detailed in /var/log/messages if permission is denied. The security context of subjects and objects is applied from the installed policy, which also provides the information to populate the security server's matrix.



- Important SELinux configuration Files**

/etc/selinux/config is the main configuration file of SELinux.

/etc/sysconfig/selinux contains a symbolic link to the actual configuration file, /etc/selinux/config.

Note: If you want to turn on or off the SELinux security you need to make changes in the main configuration file i.e. /etc/selinux/config file. Well we'll see it later in this chapter.

```
[root@ linux ~]# cat /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Modes of SELinux

- There are three modes in which SELinux can be at a time, they are
- **Enforcing, Permissive and Disabled**
 - **Enforcing**
Enable and enforce the SELinux security policy on the system, denying access and logging actions
 - **Permissive**
Permissive mode is similar to Debugging Mode. In Permissive Mode, SELinux policies and rules are applied to subjects and objects, but actions (for example, Access Control denials) are not affected. The biggest advantage of Permissive Mode is that log files and error messages are generated based on the SELinux policy implemented.
 - **Disabled**
SELinux is turned off and no warn and log messages will be generated and stored.
- **Booleans**
Booleans are variables that can either be set as true or false. Booleans enhance the effect of SELinux policies by letting the system administrator fine tune a policy. A policy may protect a certain daemon or service by applying various access control rules. In real world scenarios, a system administrator would not like to implement all the access controls specified in the policy.

SELinux Policy

- The SELinux Policy is the set of rules that guide the SELinux security engine. It defines *types* for file objects and *domains* for processes. It uses roles to limit the domains that can be entered, and has user identities to specify the roles that can be attained. In essence, types and domains are equivalent, the difference being that types apply to objects while domains apply to processes.

SELinux Context

- Processes and files are labeled with a SELinux context that contains additional information, such as a SELinux user, role, type, and, optionally, a level.

LAB WORK:-

To check the SELinux Mode

#getenforce

```
[root@ linux ~]# getenforce
Enforcing
[root@ linux ~]#
```

#sestatus

[root@mlinux6 ~]# sestatus	
SELinux status:	enabled
SELinuxfs mount:	/selinux
Current mode:	enforcing
Mode from config file:	enforcing
Policy version:	24
Policy from config file:	targeted

RHEL6

[root@mlinux7 ~]# sestatus	
SELinux status:	enabled
SELinuxfs mount:	/sys/fs/selinux
SELinux root directory:	/etc/selinux
Loaded policy name:	targeted
Current mode:	enforcing
Mode from config file:	enforcing
Policy MLS status:	enabled
Policy deny_unknown status:	allowed
Max kernel policy version:	28

RHEL7/8

Note: Observe that there is a small change between 6, 7&8, which the mount point

Display the SELinux context of a file or directory.

- To display the context of a file the syntax is
#ls -Z <filename>

```
[root@ linux ~]# ls
anaconda-ks.cfg  Documents  install.log      ktfile  Pictures  Templates
Desktop          Downloads  install.log.syslog  Music   Public    Videos
[root@ktlinux ~]# ls -Z ktfile
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 ktfile
[root@ktlinux ~]#
```

- To display the context of a directory the syntax is

#ls -ldZ <directory name>

```
[root@ linux ~]# ls -ldZ Documents
drwxr-xr-x. root root unconfined_u:object_r:admin_home_t:s0 Documents
[root@ktlinux ~]#
```

Changing the SELinux Context of a file or directory

- To change the context of the file the steps are
 - Check the existing context of the file by
#ls -ldZ <filename>

```
[root@ linux ~]# ls -ldZ myfile
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 myfile
```

Observe that the type is **admin_home_t**, let's change it to **public_content_t**, so that it will be available for all users.

- To change the context of a file or directory the syntax is
#chcon -t <argument> <file/dir name>
#chcon -t public_content_t myfile

```
[root@ linux ~]# chcon -t public_content_t myfile
[root@ linux ~]# ls -ldZ myfile
-rw-r--r--. root root unconfined_u:object_r:public_content_t:s0 myfile
[root@ linux ~]#
```

- To change the context for a directory and its contents
- Check the context of both directory and its contents

```
[root@ linux ~]# ls -ldZ mydir
drwxr-xr-x. root root system_u:object_r:admin_home_t:s0 mydir
[root@ linux ~]# ls -lZmydir
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 file1
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 file2
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 file3
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 file4
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 file5
[root@ linux ~]#
```

- To change the context for a directory and its contents, the syntax is
#chcon -R -t <argument> <dir name>
#chcon -R -t public_content_t mydir

```
[root@ linux ~]# chcon -R -t public_content_t mydir
[root@ linux ~]# ls -ldZ mydir
drwxr-xr-x. root root system_u:object_r:public_content_t:s0 mydir
[root@ linux ~]# ls -lZmydir
-rw-r--r--. root root system_u:object_r:public_content_t:s0 file1
-rw-r--r--. root root system_u:object_r:public_content_t:s0 file2
-rw-r--r--. root root system_u:object_r:public_content_t:s0 file3
-rw-r--r--. root root system_u:object_r:public_content_t:s0 file4
-rw-r--r--. root root system_u:object_r:public_content_t:s0 file5
[root@ linux ~]#
```

Restoring back the modified SELinux context to its default value

- To restore the modified/changed SELinux context of a file to its default one, the syntax is
#restorecon -v <filename>
#restorecon -v myfile

```
[root@ linux ~]# ls -ldZmyfile
-rw-r--r--. root root unconfined_u:object_r:public_content_t:s0 myfile
[root@ linux ~]# restorecon -v myfile
restorecon reset /root/ktfile context unconfined_u:object_r:public_content_t:s0->system_u:object_r:admin_home_t:s0
[root@ linux ~]# ls -ldZmyfile
-rw-r--r--. root root system_u:object_r:admin_home_t:s0 myfile
[root@ linux ~]#
```

- To restore back the same of a directory with its contents, the syntax is

```
#restorecon -Rv <dir name >
```

```
#restorecon -Rv mydir
```

```
[root@mlinux6 ~]# ls -lZ mydir
-rw-r--r--. root root unconfined_u:object_r:public_content_t:s0 file1
-rw-r--r--. root root unconfined_u:object_r:public_content_t:s0 file2
-rw-r--r--. root root unconfined_u:object_r:public_content_t:s0 file3
-rw-r--r--. root root unconfined_u:object_r:public_content_t:s0 file4
-rw-r--r--. root root unconfined_u:object_r:public_content_t:s0 file5
[root@mlinux6 ~]# restorecon -R mydir
[root@mlinux6 ~]# ls -lZ mydir
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 file1
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 file2
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 file3
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 file4
-rw-r--r--. root root unconfined_u:object_r:admin_home_t:s0 file5
```

Note: For restoring the context of only the dir except its contents do not add "R" in the command.

Changing the default context for the file:

In this case we want to change the default context of the file, so that even the context is restored any time it should be changed to the context we want.

- First of all check the present context on the file

```
[root@mlinux3 ~]# ls -lZ /myfile
-rw-r--r--. 1 root root unconfined_u:object_r:etc_runtime_t:s0 0 Apr 19 22:50 /myfile
```

- Let's change the default context to something required for example **samba_share_t**

```
# semanage fcontext -a -t samba_share_t /myfile
```

```
[root@mlinux3 ~]# semanage fcontext -a -t samba_share_t /myfile
```

- The **-a** option adds a new record, and the **-t** option defines a type (**samba_share_t**).
Note: running this command does not directly change the type - myfile is still labeled with the **etc_runtime_t** type:
- The **semanage fcontext -a -t samba_share_t /etc/file1** command adds the following entry to **/etc/selinux/targeted/contexts/files/file_contexts.local**:

```
[root@mlinux3 ~]# cat /etc/selinux/targeted/contexts/files/file_contexts.local
# This file is auto-generated by libsemanage
# Do not edit directly.

myfile    system_u:object_r:samba_share_t:s0
 myfile    system_u:object_r:samba_share_t:s0
```

- Now restore the context and verify is it changing by default to **samba_share_t**

```
[root@mlinux3 ~]# restorecon -v /myfile
Relabeled /myfile from unconfined_u:object_r:etc_runtime_t:s0 to unconfined_u:object_r:samba_share_t:s0
[root@mlinux3 ~]# ls -lZ /myfile
-rw-r--r--. 1 root root unconfined_u:object_r:samba_share_t:s0 0 Apr 19 22:50 /myfile
[root@mlinux3 ~]#
```

Changing the default context for the directory and its content:

In this case we want to change the default context of the dir, so that even the context is restored any time it should be changed to the context we want.

- First of all check the present context on the file

```
[root@mlinux3 ~]# ls -ldZ /mydir
drwxr-xr-x. 2 root root unconfined_u:object_r:default_t:s0 45 Apr 19 23:14 /mydir
[root@mlinux3 ~]# ls -lZ /mydir
total 0
-rw-r--r--. 1 root root unconfined_u:object_r:default_t:s0 0 Apr 19 23:14 file1
-rw-r--r--. 1 root root unconfined_u:object_r:default_t:s0 0 Apr 19 23:14 file2
-rw-r--r--. 1 root root unconfined_u:object_r:default_t:s0 0 Apr 19 23:14 file3
[root@mlinux3 ~]#
```

- Let's change the default context to something required for example **httpd_sys_content_t**.

```
# semanage fcontext -a -t httpd_sys_content_t "/mydir(/.*)?"
```

```
[root@mlinux3 ~]# semanage fcontext -a -t httpd_sys_content_t "/mydir(/.*)?"
```

- The **-a** option adds a new record, and the **-t** option defines a type (**httpd_sys_content_t**). The **" /mydir(/.*)?"** regular expression causes the **semanage** command to apply changes to the **/mydir/** directory, as well as the files in it.
- Note: running this command does not directly change the type **- /mydir/** and files in it are still labeled with the **default_t** type:
- The **semanage fcontext -a -t httpd_sys_content_t "/mydir(/.*)?"** command adds the following entry to **/etc/selinux/targeted/contexts/files/file_contexts.local**:

```
[root@mlinux3 ~]# cat /etc/selinux/targeted/contexts/files/file_contexts.local
# This file is auto-generated by libsemanage
# Do not edit directly.

myfile    system_u:object_r:samba_share_t:s0
 myfile    system_u:object_r:samba_share_t:s0
 /mydir(/.*)?    system_u:object_r:httpd_sys_content_t:s0
[root@mlinux3 ~]#
```

- Now restore the context and verify is it changing by default to **httpd_sys_content**

```
[root@mlinux3 ~]# restorecon -R /mydir
[root@mlinux3 ~]# ls -ldZ /mydir
drwxr-xr-x. 2 root root unconfined_u:object_r:httpd_sys_content_t:s0 45 Apr 19 23:14 /mydir
[root@mlinux3 ~]# ls -lZ /mydir
total 0
-rw-r--r--. 1 root root unconfined_u:object_r:httpd_sys_content_t:s0 0 Apr 19 23:14 file1
-rw-r--r--. 1 root root unconfined_u:object_r:httpd_sys_content_t:s0 0 Apr 19 23:14 file2
-rw-r--r--. 1 root root unconfined_u:object_r:httpd_sys_content_t:s0 0 Apr 19 23:14 file3
[root@mlinux3 ~]#
```

Checking the Booleans and modifying it.

- To see the Booleans of a particular service, the syntax is

```
#getsebool -a |grep <service name >
#getsebool -a |grep ftp
```

Note1: if **grep** is not used it will list Booleans for all the services in the system and output will be very lengthy.

Note2: Booleans can only be checked and changed when **SELinux** is in enforcing or Permissive modes; if the SELinux is in disabled mode Booleans cannot be modified.

```
[root@ linux ~]# getenforce
Enforcing
[root@ linux ~]# getsebool -a |grep ftp
allow_ftpd_anon_write --> off
allow_ftpd_full_access --> off
allow_ftpd_use_cifs --> off
allow_ftpd_use_nfs --> off
ftp_home_dir --> off
ftpd_connect_db --> off
httpd_enable_ftp_server --> off
sftpd_anon_write --> off
sftpd_enable_homedirs --> off
sftpd_full_access --> off
sftpd_write_ssh_home --> off
tftp_anon_write --> off
[root@ linux ~]#
```

- To change any Boolean just copy the Boolean and give the option (the only possible option for a Boolean to enable and disable is **on/off**). The syntax for changing Boolean value is
#setsebool < Boolean > < option (on/off) >
#setsebool allow_ftpd_anon_write on; Verify the change with **getsebool** command.

```
[root@ tlinux ~]# setsebool allow_ftpd_anon_write on
[root@ tlinux ~]# getsebool -a |grep ftp
allow_ftpd_anon_write --> on
allow_ftpd_full_access --> off
allow_ftpd_use_cifs --> off
allow_ftpd_use_nfs --> off
ftp_home_dir --> off
ftpd_connect_db --> off
httpd_enable_ftp_server --> off
sftpd_anon_write --> off
sftpd_enable_homedirs --> off
sftpd_full_access --> off
sftpd_write_ssh_home --> off
tftp_anon_write --> off
[root@ tlinux ~]#
```

- Making boolean permanent**

#setsebool -P <Boolean> on/off to make change permanent

```
[root@node1 ~]# setsebool -P ftpd_anon_write on
[root@node1 ~]# getsebool -a |grep ftp
ftpd_anon_write --> on
ftpd_connect_all_unreserved --> off
ftpd_connect_db --> off
ftpd_full_access --> on
ftpd_use_cifs --> off
ftpd_use_fusefs --> off
```

Changing the Modes of SELinux

- To change the mode of SELinux the syntax is
#setenforce <option>
Options used are 0 or 1 (Where 0 means Permissive and 1 means Enforcing)
- To change the SELinux Mode to permissive
#setenforce 0
- Verify it by **getenforce** or **sestatus** command.

```
[root@ linux ~]# getenforce
Enforcing
[root@ linux ~]# setenforce 0
[root@ linux ~]# getenforce
Permissive
[root@ linux ~]# sestatus
SELinux status:          enabled
SELinuxfs mount:         /selinux
Current mode:            permissive
Mode from config file:  enforcing
Policy version:          24
Policy from config file: targeted
[root@ linux ~]# █
```

- To change the SELinux Mode back to Enforcing mode
#setenforce 1
- Verify the change

```
[root@ linux ~]# getenforce
Permissive
[root@ linux ~]# setenforce 1
[root@ linux ~]# getenforce
Enforcing
[root@ linux ~]# sestatus
SELinux status:          enabled
SELinuxfs mount:         /selinux
Current mode:            enforcing
Mode from config file:  enforcing
Policy version:          24
Policy from config file: targeted
[root@ linux ~]# █
```

Disabling and Enabling the SELinux Security

- To disable the SELinux protection or to change it to **disabled** Mode
- Edit the **/etc/selinux/config** file and change **SELINUX=disabled**
- Whenever changing the mode of **SELinux** from **Enforcing/Permissive** to **Disabled** or **Disabled** to **Permissive/Enforcing**, you need to restart the system so that the changes can take effect.
- First check the current status of **SELinux** and the configuration file.

```
[root@ linux ~]# getenforce
Enforcing
[root@ linux ~]# cat /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.

SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

- Now, edit the configuration file, restart the computer and check the status.

```
#vim /etc/selinux/config
#reboot (to reboot the system)
```

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.

SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

```
[root@ linux ~]# getenforce
Disabled
[root@ linux ~]# sestatus
SELinux status:                 disabled
[root@ linux ~]#
```

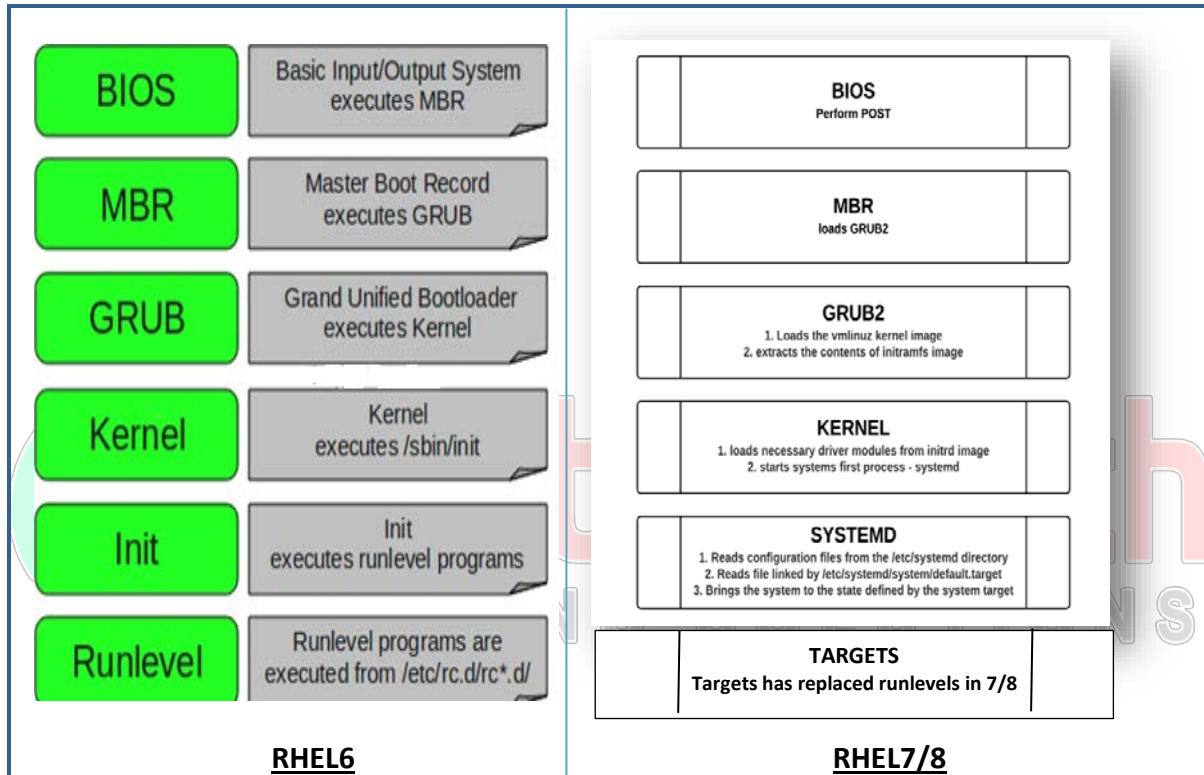
To Enable it back the procedure is exactly same as above, instead of **SELINUX=disabled** change it to **SELINUX=enforcing** or **permissive**. Don't forget to restart the system, unless the system is rebooted the changes will not take effect.

BOOTING PROCEDURE

Press the power button on your system, and after few moments you see the Linux login prompt.

Have you ever wondered what happens behind the scenes from the time you press the power button until the Linux login prompt appears?

The following are the 6 high level stages of a typical RHEL7/8 boot process.



1. BIOS/UEFI

- **BIOS** stands for Basic Input/Output System, **UEFI** stands for Unified Extensible Firmware Interface
- Performs some system integrity checks
- Searches, loads, and executes the boot loader program.
- It looks for boot loader in floppy, cd-rom, or hard drive. You can press a key (typically F12 or F2, but it depends on your system) during the BIOS startup to change the boot sequence.
- Once the boot loader program is detected and loaded into the memory, BIOS gives the control to it.
- So, in simple terms BIOS/UEFI loads and executes the MBR/GPT boot loader.

2. MBR/GPT

- MBR stands for Master Boot Record, GPT stands for GUID Partition Table
- It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda
- MBR is less than 512 bytes in size. This has three components 1) primary boot loader info in 1st 446 bytes 2) partition table info in next 64 bytes 3) mbr validation check in last 2 bytes.
- It contains information about GRUB/GRUB2 (or LILO in old systems).
- So, in simple terms MBR loads and executes the GRUB boot loader.

3. GRUB2

- The default bootloader used on RHEL 7&8 is GRUB 2. GRUB stands for GRand Unified Bootloader. GRUB 2 replaces the older GRUB bootloader also called as legacy GRUB.
- The GRUB 2 configuration file is located at **/boot/grub2/grub.cfg** and link at **/etc/grub2.cfg**(Do not edit this file directly).
- GRUB 2 menu-configuration settings are taken from **/etc/default/grub** when generating grub.cfg.

```
[root@localhost ~]# cat /etc/default/grub
GRUB_TIMEOUT=5
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto rhgb quiet"
GRUB_DISABLE_RECOVERY="true"
[root@localhost ~]#
```

- If changes are made to any of these parameters, you need to run **grub2-mkconfig** to re-generate the **/boot/grub2/grub.cfg** file
- **#grub2-mkconfig -o /boot/grub2/grub.cfg**

initramfs

- The job of the initial RAM file system is to preload the block device modules, such as for IDE, SCSI, or RAID, so that the root file system, on which those modules normally reside, can then be accessed and mounted.
- The initramfs is bound to the kernel and the kernel mounts this initramfs as part of a two-stage boot process.
- The dracut utility creates initramfs whenever a new kernel is installed.
- Use the lsinitrd command to view the contents of the image created by dracut

4. Kernel

- Mounts the root file system as specified in the “root=” in grub2.conf
- Kernel executes the systemd program

Since system is the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1. Do a ‘ps -ef | grep init’ and check the pid.

5. Systemd

1. systemd is the ancestor of all processes on a system.
2. systemd reads the file linked by /etc/systemd/system/default.target (for example, /usr/lib/systemd/system/multi-user.target) to determine the default system target (equivalent to run level). The system target file defines the services that systemd starts.
3. systemd brings the system to the state defined by the system target, performing system initialization tasks such as:
 1. Setting the host name
 2. Initializing the network
 3. Initializing SELinux based on its configuration
 4. Printing a welcome banner
 5. Initializing the system hardware based on kernel boot arguments
 6. Mounting the file systems, including virtual file systems such as the /proc file system
 7. Cleaning up directories in /var
 8. Starting swap

6. Runlevel/Targets

- Prior to RHEL 7, runlevels were used to identify a set of services that would start or stop when that runlevel was requested. Instead of runlevels, systemd uses the concept of *targets* to group together sets of services that are started or stopped. A target can also include other targets (for example, the multi-user target includes an nfs target).
- Depending on your default init level setting, the system will execute the programs from one of the following directories.

Traditional runlevel	New Target name	Symbolically linked to...
Runlevel 0	runlevel0.target	poweroff.target
Runlevel 1	runlevel1.target	rescue.target
Runlevel 2	runlevel2.target	multi-user.target
Runlevel 3	runlevel3.target	multi-user.target
Runlevel 4	runlevel4.target	multi-user.target
Runlevel 5	runlevel5.target	graphical.target
Runlevel 6	runlevel6.target	reboot.target

To know more about targets, pl read /etc/inittab file

LAB WORK:-

To check the default run level in linux

- To see the default run level in linux the command is
#who -r

```
[root@localhost ~]# who -r
  run-level 5 2016-08-02 09:06
[root@localhost ~]#
```

Changing the default run level/target

- To check the default run level

```
#systemctl get-default
```

```
[root@localhost ~]# systemctl get-default
graphical.target
```

- To change the default target/runlevel

```
#systemctl set-default multiuser.target
```

```
[root@localhost ~]# systemctl set-default multi-user.target
rm '/etc/systemd/system/default.target'
ln -s '/usr/lib/systemd/system/multi-user.target' '/etc/systemd/system/default.target'
[root@localhost ~]#
```

Now reboot the system and check in which runlevel it is.

```
#reboot
```

```
Red Hat Enterprise Linux Server 7.1 (Maipo)
Kernel 3.10.0-229.el7.x86_64 on an x86_64

localhost login: root
Password:
Last login: Wed Aug  3 13:55:42 on :1
[root@localhost ~]# who -r
    run-level 3  2016-08-03 14:40
[root@localhost ~]# _
```

- To start the graphical interface when you are in runlevel 3, use the following command

```
#startx
```

To see the details regarding the kernel installed

- To see the version of the kernel use

```
#uname -r
```

```
[root@localhost ~]# uname -r
3.10.0-229.el7.x86_64
[root@localhost ~]#
```

- To see the same thing with more details use

```
#uname -a
```

```
Linux localhost.localdomain 3.10.0-229.el7.x86_64 #1 SMP Thu Jan 29 18:37:38 EST 2015 x86_64 x86_64
x86_64 GNU/Linux
[root@localhost ~]#
```

Note: The same information can be seen in /boot/grub2/grub2.cfg

```
fi
linux16 /vmlinuz-3.10.0-229.el7.x86_64 root=UUID=d06af8d4-c931-4412-aa1-674309315701
nol�=auto rhgb quiet LANG=en_IN.UTF-8
initrd16 /initramfs-3.10.0-229.el7.x86_64.img
```

To change the default values in grub2 configuration file

- Changing the default timeout value in grub2.cfg
- Check the present timeout value with following command

```
[root@mlinux1 ~]# grep -i timeout /boot/grub2/grub.cfg
if [ x$feature_timeout_style = xy ] ; then
    set timeout_style=menu
    set timeout=5
# Fallback normal timeout code in case the timeout_style feature is
    set timeout=5
[root@mlinux1 ~]#
```

- Make the changes in /etc/default/grub file as per the requirement

```
[root@mlinux1 ~]# vim /etc/default/grub
GRUB_TIMEOUT=9
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 rhgb quiet"
GRUB_DISABLE_RECOVERY="true"
```

- Update the change in grub2 config file by using following command
- #grub2-mkconfig -o /boot/grub2/grub.cfg

```
[root@mlinux1 ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-3.10.0-327.el7.x86_64
Found initrd image: /boot/initramfs-3.10.0-327.el7.x86_64.img
Found linux image: /boot/vmlinuz-0-rescue-71773719b0e6437abb6a06811abed8c4
Found initrd image: /boot/initramfs-0-rescue-71773719b0e6437abb6a06811abed8c4
g
done
```

- Check back in grub2 config file whether the changes are applied

```
[root@mlinux1 ~]# grep -i timeout /boot/grub2/grub.cfg
if [ x$feature_timeout_style = xy ] ; then
    set timeout_style=menu
    set timeout=9
# Fallback normal timeout code in case the timeout_style feature is
    set timeout=9
[root@mlinux1 ~]#
```

To check the architecture of the O/S

- To check the architecture of the O/S the command is

```
#arch
#uname -m
```

```
[root@ktadm Desktop]# arch
x86_64
[root@ktadm Desktop]# uname -m
x86_64
[root@ktadm Desktop]#
```

To check the version of the O/S in the system

- To check the O/S version you have to navigate to the following file

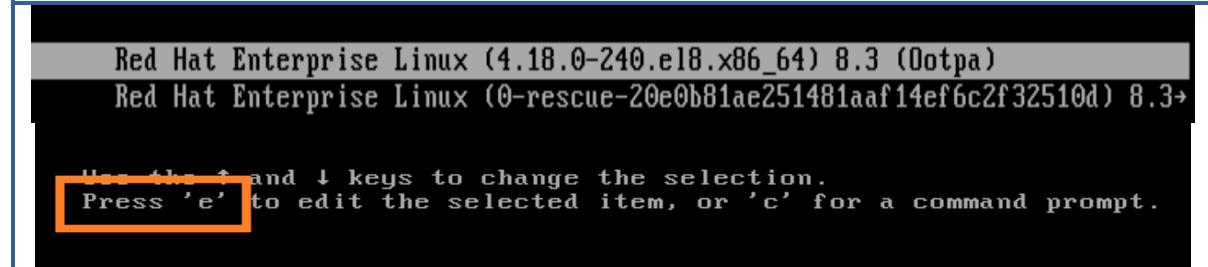
```
# cat /etc/redhat-release or # cat /etc/os-release
```

```
[root@localhost ~]# cat /etc/redhat-release
Red Hat Enterprise Linux release 8.1 (Ootpa)
```

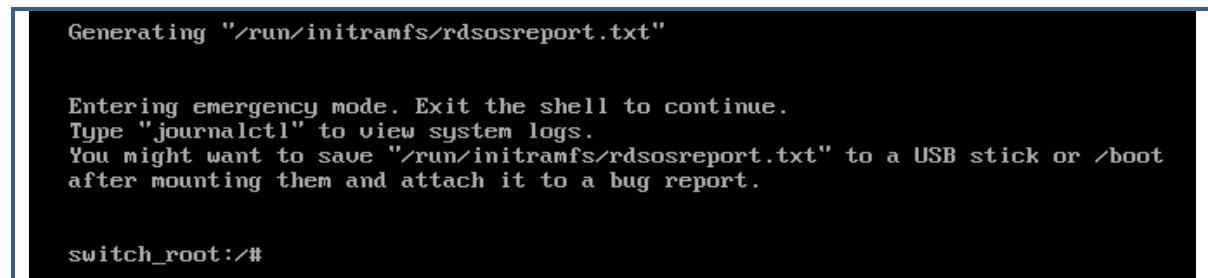
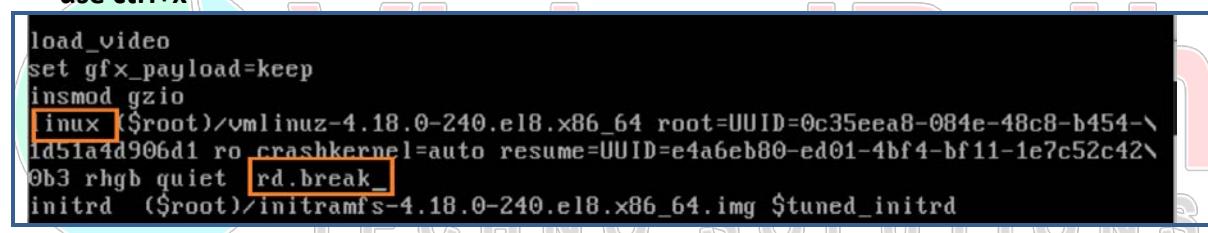
Recovering the lost password in RHEL 7/8

To recover the password the steps are

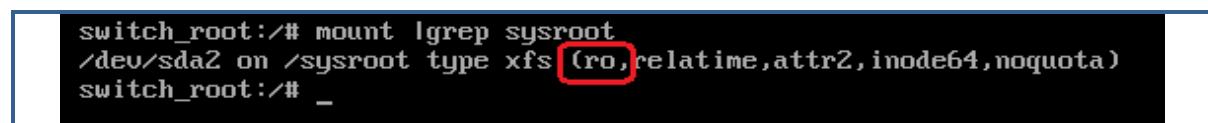
- Disturb the normal boot by pressing any key when RHEL 7 booting screen is displayed



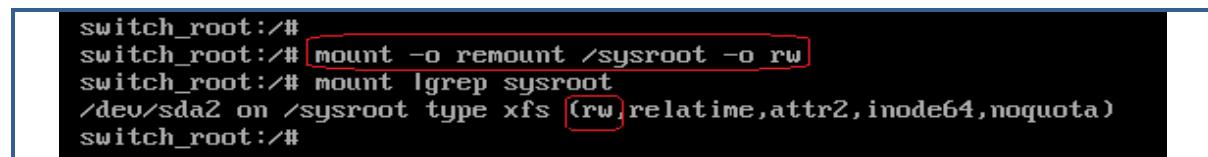
- Type “e” to edit the kernel argument
- Go to the end of kernel line “linux16” using ctrl+e, type rd.break. To continue booting use ctrl+x



- It will boot into emergency mode



- Check in which mode /sysroot is mounted, it would be in read-only mode



- Change the /sysroot to rw, by using above command

```
switch_root:/#  
switch_root:/# chroot /sysroot  
sh-4.2# ls  
bin boot dev etc home ktdir lib lib64 media mnt opt proc root run sbin srv sys tmp usr var  
sh-4.2#
```

- Now access the /sysroot using #chroot command

```
sh-4.2# passwd  
Changing password for user root.  
New password:  
BAD PASSWORD: The password fails the dictionary check - it is too simplistic/systematic  
Retype new password:  
passwd: all authentication tokens updated successfully.  
sh-4.2#
```

- To change the password use command #passwd and change the passwd

```
sh-4.2#  
sh-4.2# touch /.autorelabel  
sh-4.2# _
```

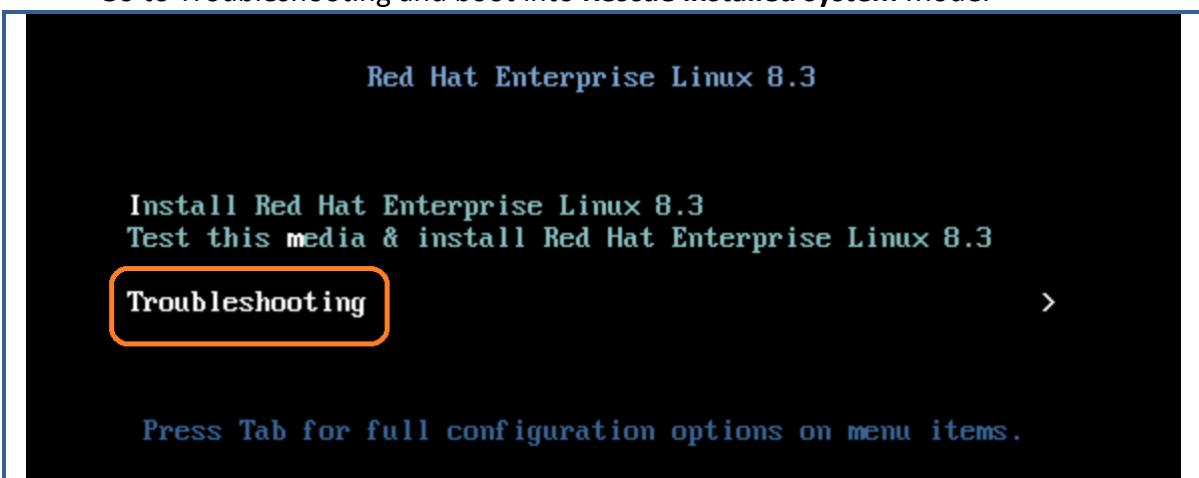
- To auto relabel selinux policies, create a blank hidden file “/.autorelabel”

```
sh-4.4# exit  
exit  
switch_root:/# exit  
logout
```

- Okay, Now we have successfully changed the password, type exit to leave the /sysroot and exit emergency mode. Once selinux policies relabeled system reboots, try the new password for root user after reboot.

Repairing the corrupted boot loader and recovering it

- There might be a situation where your boot loader i.e., **GRub** might got corrupted and you want to recover it or in other word repair it. Basically the repairing of **GRub** means installing a new grub on the existing one from **RHEL 7/8 installation media/DVD**.
- To recover the grub the steps are:
 - Boot the system with RHEL 7/8 DVD
 - Go to Troubleshooting and boot into **Rescue installed system Mode**.



Troubleshooting**Install Red Hat Enterprise Linux 8.3 in basic graphics mode****Rescue a Red Hat Enterprise Linux system****Run a memory test****Boot from local drive****Return to main menu****Press Tab for full configuration options on menu items.****Rescue**

The rescue environment will now attempt to find your Linux installation and mount it under the directory : /mnt/sysroot. You can then make any changes required to your system. Choose '1' to proceed with this step. You can choose to mount your file systems read-only instead of read-write by choosing '2'.

If for some reason this process does not work choose '3' to skip directly to a shell.

- 1) Continue
- 2) Read-only mount
- 3) Skip to shell
- 4) Quit (Reboot)

Please make a selection from the above:

- Move the cursor to "continue" tab, and hit enter

Rescue Shell

Your system has been mounted under .

If you would like to make the root of your system the root of the active system, run the command:

When finished, please exit from the shell and your system will reboot.

Please press ENTER to get a shell:

```
sh-4.4#  
sh-4.4#  
sh-4.4#
```

- Observe from above pic, that now your system has been mounted on /mnt/sysroot. It means where our system root is residing
- Press Enter to continue.

```
sh-4.4#  
sh-4.4# chroot /mnt/sysroot  
bash-4.4#
```

- Change the DVD root to system root in order to access OS with root credentials by using following command

#chroot /mnt/sysroot

```
bash-4.4# grub2-install /dev/sda
Installing for i386-pc platform.
Installation finished. No error reported.
bash-4.4#
```

- Install the grub on the /boot device i.e. /dev/sda by using following command
`#grub2-install <device name>`
`#grub2-install /dev/sda`
- If it shows no error reported, that means we have successfully recovered the grub.

```
bash-4.4# exit
exit
sh-4.4# poweroff
```

- Type “exit” to exit from system root
- Again type “exit” or “reboot” to reboot the system, or “#poweroff” to shutdown the system



All these steps are very critical and highly useful, please practice the stuff nicely.

MANAGING INSTALLED SERVICES

- Services are programs (called daemons) that once started run continuously in the background and are ready for input or monitor changes in your computer and respond to them. For example the Apache server has a daemon called **httpd** (the d is for daemon) that listens on port 80 on your computer and when it receives a request for a page it sends the appropriate data back to the client machine.
- Many services are required to run all the time however many can be safely turned off for both security reasons as running unnecessary services opens more doors into your computer, but also for performance reasons. It may not make much difference but your computer should boot slightly faster with less services it has to start on boot.
- One of the techniques in every Linux administrator's toolbox to improve security of a box is to turn off unneeded services.

INTRODUCTION TO *systemd* and *systemctl* command in RHEL7/8

Systemd is a system and service manager for Linux operating systems. It is designed to be backwards compatible with SysV init scripts, and provides a number of features such as parallel startup of system services at boot time, on-demand activation of daemons, support for system state snapshots, or dependency-based service control logic. In Red Hat Enterprise Linux 7, systemd replaces Upstart as the default init system.

Systemd is a replacement to the older traditional "System V init" system . systemd stands for system daemon. systemd was designed to allow for better handling of dependencies and have the ability to handle more work in parallel at system startup. systemd supports snapshotting of your system and the restoring of your systems state, keeps track of processes stored in what is known as a "cgroup" as opposed to the conventional "PID" method. systemd is now shipping by default with many popular Linux distributions such as Fedora, Mandriva, Mageia, Arch Linux, CentOS 7, RHEL 7.0 (Red Hat Enterprise Linux) and Oracle Linux 7.0. systemd refers to runlevels as targets.

Runlevel	Systemd Description
0	poweroff.target
1	rescue.target
2	multi-user.target
3	multi-user.target
4	multi-user.target
5	graphical.target
6	reboot.target

Service vs systemctl commands

service	systemctl	Description
service name start	systemctl start name.service	Starts a service.
service name stop	systemctl stop name.service	Stops a service.
service name restart	systemctl restart name.service	Restarts a service.
service name reload	systemctl reload name.service	Reloads configuration.
service name status	systemctl status name.service	Check the status of the service
service --status-all	systemctl list-units --type service --all	Displays the status of all services.
	systemctl is-active name.service	Checks if a service is running.

Chkconfig vs systemctl commands

chkconfig	systemctl	Description
chkconfig --list name	systemctl status name	Checks if a service is enabled
chkconfig service on	systemctl enable service	Enables a service
chkconfig service off	systemctl disable service	Disables a service
	Systemctl is-enable service	Checks if a service is enabled

LAB WORK:-

To check the status of ftp service “vsftpd”

- To check the status of the above service

```
#systemctl status vsftpd
```

```
[root@mlinux71 ~]# systemctl status vsftpd
vsftpd.service - Vsftpd ftp daemon
   Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; enabled)
   Active: active (running) since Fri 2016-09-30 07:03:05 IST; 5h 6min ago
     Process: 1312 ExecStart=/usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf (code=exited, status=0/SUCCESS)
    Main PID: 1351 (vsftpd)
      CGroup: /system.slice/vsftpd.service
              └─1351 /usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf

Sep 30 07:03:05 mlinux71.kt.com systemd[1]: Starting Vsftpd ftp daemon...
Sep 30 07:03:05 mlinux71.kt.com systemd[1]: Started Vsftpd ftp daemon.
[root@mlinux71 ~]#
```

To stop the ftp services

- To start the ftp service, the command is

```
#systemctl stop vsftpd
```

```
[root@mlinux71 ~]# systemctl stop vsftpd
[root@mlinux71 ~]# systemctl status vsftpd
vsftpd.service - Vsftpd ftp daemon
   Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; enabled)
   Active: inactive (dead) since Fri 2016-09-30 12:14:55 IST; 2s ago
     Process: 1312 ExecStart=/usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf (code=exited, status=0/SUCCESS)
    Main PID: 1351 (code=killed, signal=TERM)

Sep 30 07:03:05 mlinux71.kt.com systemd[1]: Starting Vsftpd ftp daemon...
Sep 30 07:03:05 mlinux71.kt.com systemd[1]: Started Vsftpd ftp daemon.
Sep 30 12:14:55 mlinux71.kt.com systemd[1]: Stopping Vsftpd ftp daemon...
Sep 30 12:14:55 mlinux71.kt.com systemd[1]: Stopped Vsftpd ftp daemon.
[root@mlinux71 ~]#
```

Reload the ftp services, may be required after doing some change in config file.

- To reload the service, the command is

```
#service crond reload
```

```
[root@mlinux71 ~]# systemctl reload crond
[root@mlinux71 ~]#
```

To restart the ftp or any service required when reload does not work

- To restart the ftp services, the command will be

```
#systemctl restart vsftpd
```

```
[root@mlinux71 ~]# systemctl restart vsftpd.service
[root@mlinux71 ~]#
```

Check the status of the service availability.

- To check the status of the service availability, use

```
#systemctl status vsftpd
```

```
[root@mlinux71 ~]# systemctl status vsftpd.service
vsftpd.service - Vsftpd ftp daemon
   Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; disabled)
   Active: active (running) since Fri 2016-09-30 12:25:53 IST; 3min 38s ago
     Main PID: 6986 (vsftpd)
       CGroup: /system.slice/vsftpd.service
               └─6986 /usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf

Sep 30 12:25:53 mlinux71.kt.com systemd[1]: Starting Vsftpd ftp daemon...
Sep 30 12:25:53 mlinux71.kt.com systemd[1]: Started Vsftpd ftp daemon.
[root@mlinux71 ~]#
```

Make the service enable at boot for vsftpd .

- To make the service enable at boot vsftpd service,

```
#systemctl enable vsftpd.service
```

```
[root@mlinux71 ~]# systemctl enable vsftpd.service
ln -s '/usr/lib/systemd/system/vsftpd.service' '/etc/systemd,
[root@mlinux71 ~]# systemctl is-enabled vsftpd.service
enabled
[root@mlinux71 ~]#
```

Make the service availability disabled for vsftpd

- To make the service availability disabled the command is
#systemctl disable vsftpd.service

```
[root@mlinux71 ~]# systemctl disable vsftpd.service
rm '/etc/systemd/system/multi-user.target.wants/vsftpd.service'
[root@mlinux71 ~]# systemctl is-enabled vsftpd.service
disabled
[root@mlinux71 ~]#
```

To start and enable the service at the same time

- To make the service availability disabled the command is
#systemctl enable vsftpd --now

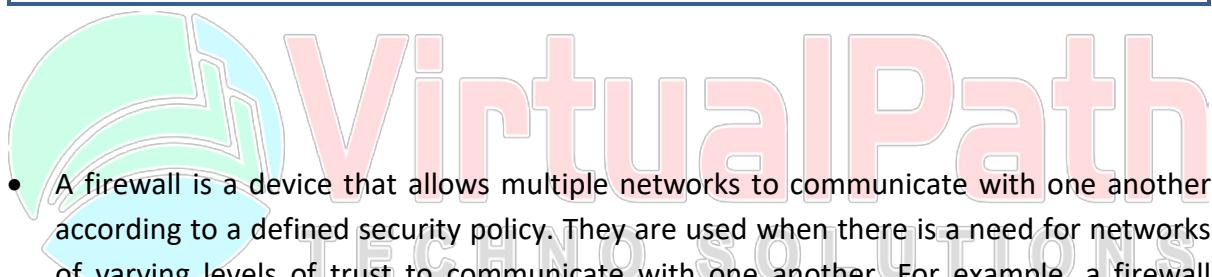
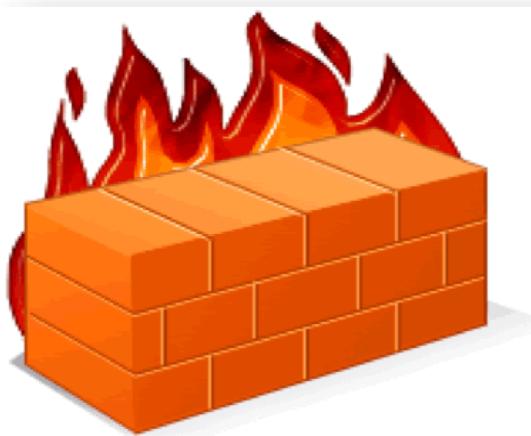
```
[root@mlinux1 ~]# systemctl enable vsftpd --now
Created symlink from /etc/systemd/system/multi-user.target.wants/vsftpd.service
service.
[root@mlinux1 ~]# systemctl status vsftpd
● vsftpd.service - Vsftpd ftp daemon
  Loaded: loaded (/usr/lib/systemd/system/vsftpd.service); enabled; vendor prese
  Active: active (running) since Sat 2020-03-28 11:35:58 IST; 27s ago
    Process: 2385 ExecStart=/usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf (code=exited,
   Main PID: 2386 (vsftpd)
     Tasks: 1
```

To stop and disable the service at the same time

- To make the service availability disabled the command is
#systemctl disable vsftpd --now

```
[root@mlinux1 ~]# systemctl status vsftpd
● vsftpd.service - Vsftpd ftp daemon
  Loaded: loaded (/usr/lib/systemd/system/vsftpd.service); disabled;
  Active: inactive (dead)
```

INTRODUCTION TO FIREWALL



- A firewall is a device that allows multiple networks to communicate with one another according to a defined security policy. They are used when there is a need for networks of varying levels of trust to communicate with one another. For example, a firewall typically exists between a corporate network and a public network like the Internet. It can also be used inside a private network to limit access to different parts of the network. Wherever there are different levels of trust among the different parts of a network, a firewall can and should be used.
- A firewall is a device that allows multiple networks to communicate with one another according to a defined security policy. They are used when there is a need for networks of varying levels of trust to communicate with one another. For example, a firewall typically exists between a corporate network and a public network like the Internet. It can also be used inside a private network to limit access to different parts of the network. Wherever there are different levels of trust among the different parts of a network, a firewall can and should be used.

LAB WORK:

To check the status of firewalld status

#firewall-cmd --state

```
[root@mlinux71 ~]# firewall-cmd --state  
running  
[root@mlinux71 ~]#
```

To check the list of trusted services and ports in firewalld

#firewall-cmd --list-all

```
[root@mlinux71 ~]# firewall-cmd --list-all  
public (default, active)  
  interfaces: bond0 ens3 ens8  
  sources:  
  services: dhcpcv6-client ssh  
  ports:  
  masquerade: no  
  forward-ports:  
  icmp-blocks:  
  rich rules:
```

To add a service into trusted list permanently

#firewall-cmd --add-service=<service name> --permanent (ex ftp service)

```
[root@mlinux71 ~]# firewall-cmd --add-service=ftp --permanent  
success  
[root@mlinux71 ~]#
```

The service will not be updated until reloading of firewall

To reload firewalld service

#firewall-cmd --reload

```
[root@mlinux71 ~]#  
[root@mlinux71 ~]# firewall-cmd --reload  
success  
[root@mlinux71 ~]# firewall-cmd --list-all  
public (default, active)  
  interfaces: bond0 ens3 ens8  
  sources:  
  services: dhcpcv6-client ftp ssh  
  ports:  
  masquerade: no  
  forward-ports:  
  icmp-blocks:  
  rich rules:
```

To remove the service from firewall trusted list permanently

```
#firewall-cmd --remove-service=ftp --permanent
#firewall-cmd --reload
```

```
[root@mlinux71 ~]# firewall-cmd --remove-service=ftp --permanent
success
[root@mlinux71 ~]# firewall-cmd --reload
success
[root@mlinux71 ~]# firewall-cmd --list-all
public (default, active)
  interfaces: bond0 ens3 ens8
  sources:
  services: dhcpv6-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

To add a port into firewall trusted lists permanently

```
#firewall-cmd --add-port=21/tcp --permanent
#firewall-cmd --reload
```

```
[root@mlinux71 ~]# firewall-cmd --add-port=21/tcp --permanent
success
[root@mlinux71 ~]# firewall-cmd --reload
success
[root@mlinux71 ~]# firewall-cmd --list-all
public (default, active)
  interfaces: bond0 ens3 ens8
  sources:
  services: dhcpv6-client ssh
  ports: 21/tcp
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```



To remove a port from firewall trusted lists permanently

```
#firewall-cmd --remove-port=21/tcp --permanent
#firewall-cmd --reload
```

```
[root@mlinux71 ~]# firewall-cmd --remove-port=21/tcp --permanent
success
[root@mlinux71 ~]# firewall-cmd --reload
success
[root@mlinux71 ~]# firewall-cmd --list-all
public (default, active)
  interfaces: bond0 ens3 ens8
  sources:
  services: dhcpv6-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

To start the firewall permanently

```
#systemctl start firewalld.service
#systemctl enable firewalld.service
```

```
[root@mlinux71 ~]# systemctl is-active firewalld.service
unknown
[root@mlinux71 ~]# systemctl start firewalld.service
[root@mlinux71 ~]# systemctl is-active firewalld.service
active
[root@mlinux71 ~]# systemctl enable firewalld.service
ln -s '/usr/lib/systemd/system/firewalld.service' '/etc/systemd/
fedoraproject.FirewallD1.service'
ln -s '/usr/lib/systemd/system/firewalld.service' '/etc/systemd/
get.wants/firewalld.service'
[root@mlinux71 ~]# systemctl is-enabled firewalld.service
enabled
[root@mlinux71 ~]#
```

To stop the firewall permanently

```
#systemctl stop firewalld.service
#systemctl disable firewalld.service
```

```
[root@mlinux71 ~]# systemctl is-active firewalld.service
active
[root@mlinux71 ~]# systemctl stop firewalld.service
[root@mlinux71 ~]# systemctl is-active firewalld.service
inactive
[root@mlinux71 ~]#
[root@mlinux71 ~]# systemctl is-enabled firewalld.service
enabled
[root@mlinux71 ~]# systemctl disable firewalld.service
rm '/etc/systemd/system/basic.target.wants/firewalld.service'
rm '/etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service'
[root@mlinux71 ~]# systemctl is-enabled firewalld.service
disabled
[root@mlinux71 ~]#
```

To add or remove multiple services/ports in firewall use following command

```
#firewall-cmd --add-service=1st service --add-service=2nd service –permanent
#firewall-cmd --reload
```

```
#firewall-cmd --add-port=portid/protocol --add-port=portid/protocol –permanent
#firewall-cmd --reload
```

INTRODUCTION TO COCKPIT IN RHEL8

- The Cockpit is a free and open source web-based server management tool. By default, Cockpit comes preinstalled on an RHEL 8 server. But, it is not activated. A sysadmin must enable it. One can see the server in a web browser and perform system tasks with a GUI/mouse. It is easy to start containers, administer storage or users, configure networks, and inspect log files on RHEL 8. The Cockpit web interface is user-friendly for new to Linux users and seasoned sysadmins too.
- Cockpit is a useful Web based GUI tool through which sysadmins can monitor and manage their Linux servers, it can also be used to manage networking and storage on servers, containers, virtual machines and inspections of system and application's logs.

Activating cockpit services:

As mentioned above cockpit tool comes pre-installed in RHEL8 and hence we don't have to install any package for it. The only thing we need to do is to activate the services related to cockpit and allow it through firewall.

It can be observed that while we login into any rhel8 machine, by default it gives a message on the screen related to cockpit, as shown below.

```
login as: root
root@192.168.10.30's password:
Activate the web console with: systemctl enable --now cockpit.socket

This system is not registered to Red Hat Insights. See https://cloud.redhat.com/
To register this system, run: insights-client --register
```

In order to use cockpit start the service of cockpit and check whether in firewall the services is allowed or not.

#systemctl enable --now cockpit.socket

```
[root@mlinux3 ~]# systemctl enable cockpit.socket --now
Created symlink /etc/systemd/system/sockets.target.wants/cockpit.socket → /usr/lib/systemd/s
[root@mlinux3 ~]# systemctl status cockpit.socket
● cockpit.socket - Cockpit Web Service Socket
   Loaded: loaded (/usr/lib/systemd/system/cockpit.socket; enabled; vendor preset: disabled)
   Active: active (listening) since Sun 2020-04-19 21:32:52 IST; 6s ago
     Docs: man:cockpit-ws(8)
     Listen: [::]:9090 (Stream)
   Process: 3221 ExecStartPost=/bin/ln -snf active.motd /run/cockpit/motd (code=exited, statu
   Process: 3214 ExecStartPost=/usr/share/cockpit/motd/update-motd localhost (code=exited, s
     Tasks: 0 (limit: 11501)
    Memory: 312.0K
      CGroup: /system.slice/cockpit.socket

Apr 19 21:32:52 mlinux3.vpts.com systemd[1]: Starting Cockpit Web Service Socket.
Apr 19 21:32:52 mlinux3.vpts.com systemd[1]: Listening on Cockpit Web Service Socket.
```

Check whether **cockpit** is allowed in firewall or not. If not, allow cockpit in firewall
#firewall-cmd --list-all

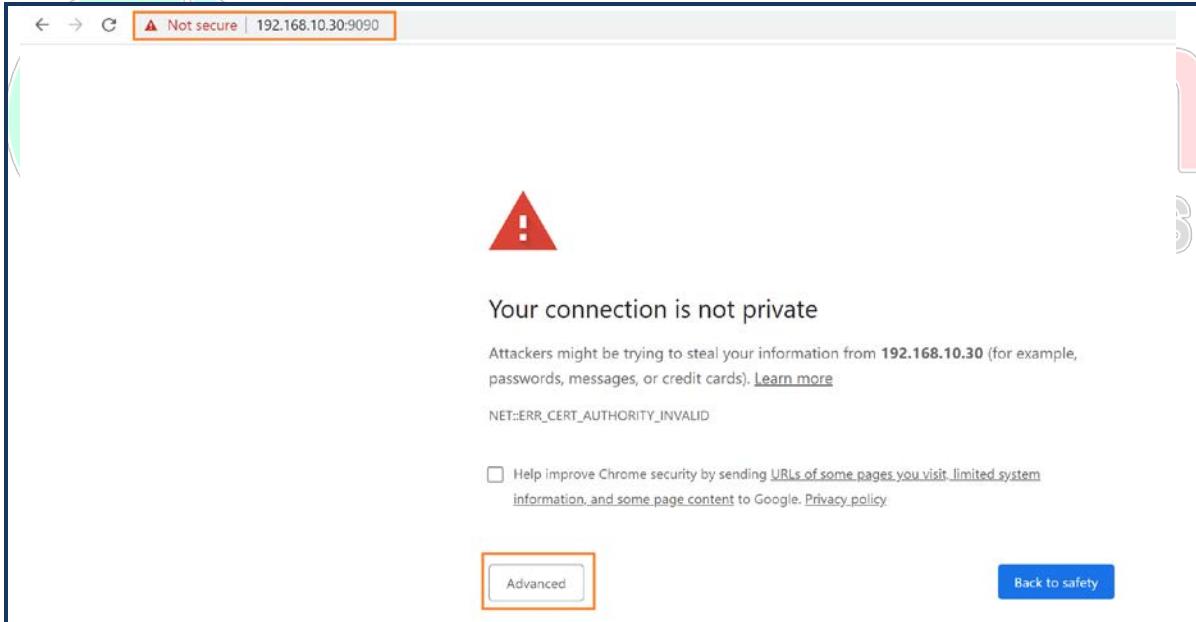
```
[root@mlinux3 ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3
  sources:
  services: cockpit dhcpcv6-client dns ftp http mounted
  ports: 8080/tcp
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

By default **cockpit** is pre-allowed in firewall. If it is not pre-allowed, then it can be added later by using following commands

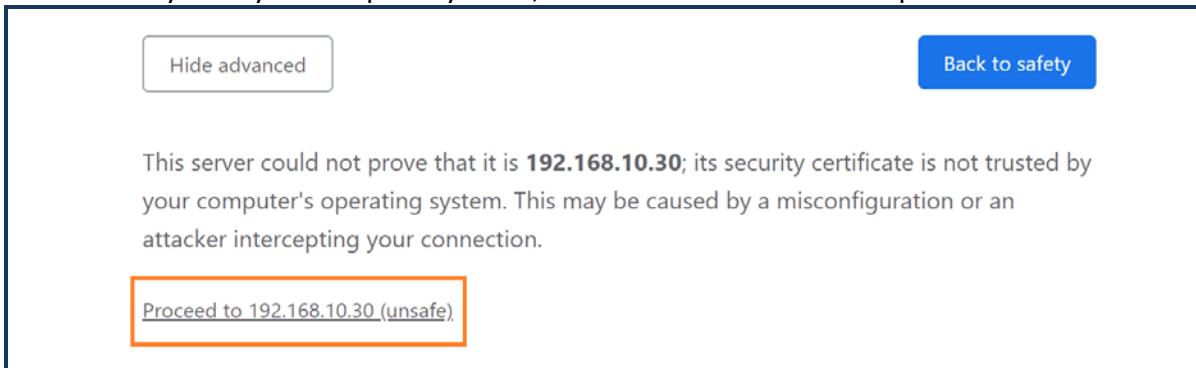
#firewall-cmd --add-service=cockpit –permanent
#firewall-cmd –reload

Now then the cockpit services are active and allowed in firewall, we can now access it through a browser using the following URL

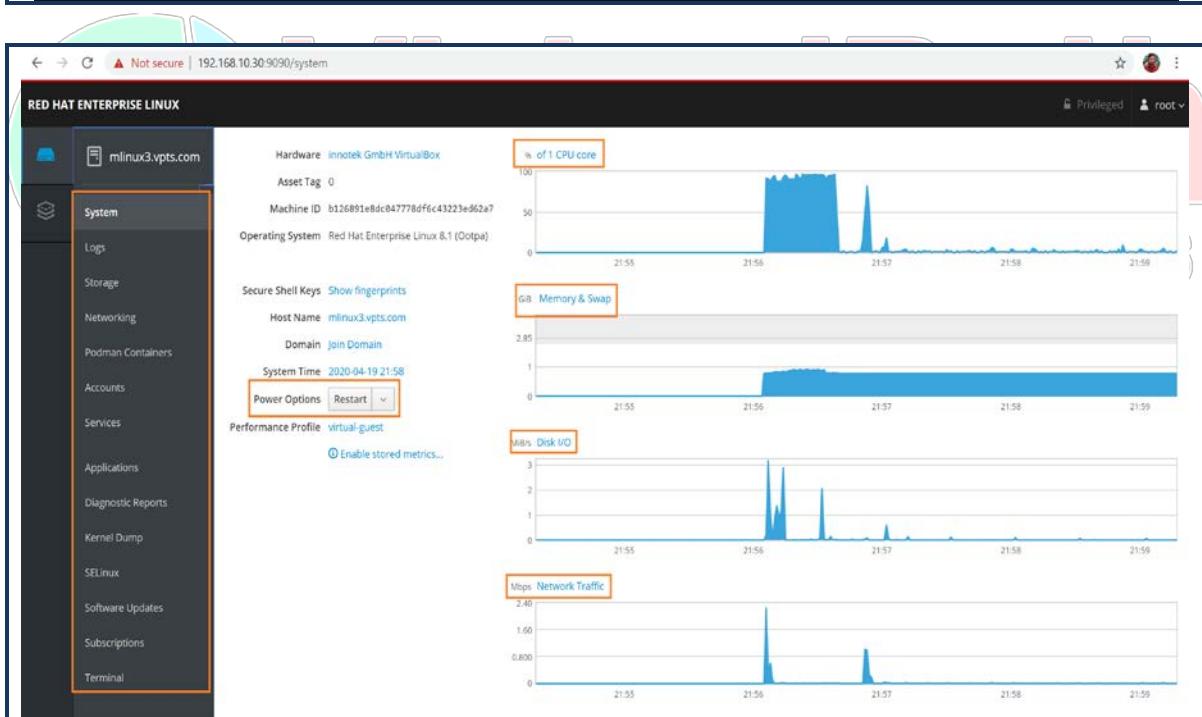
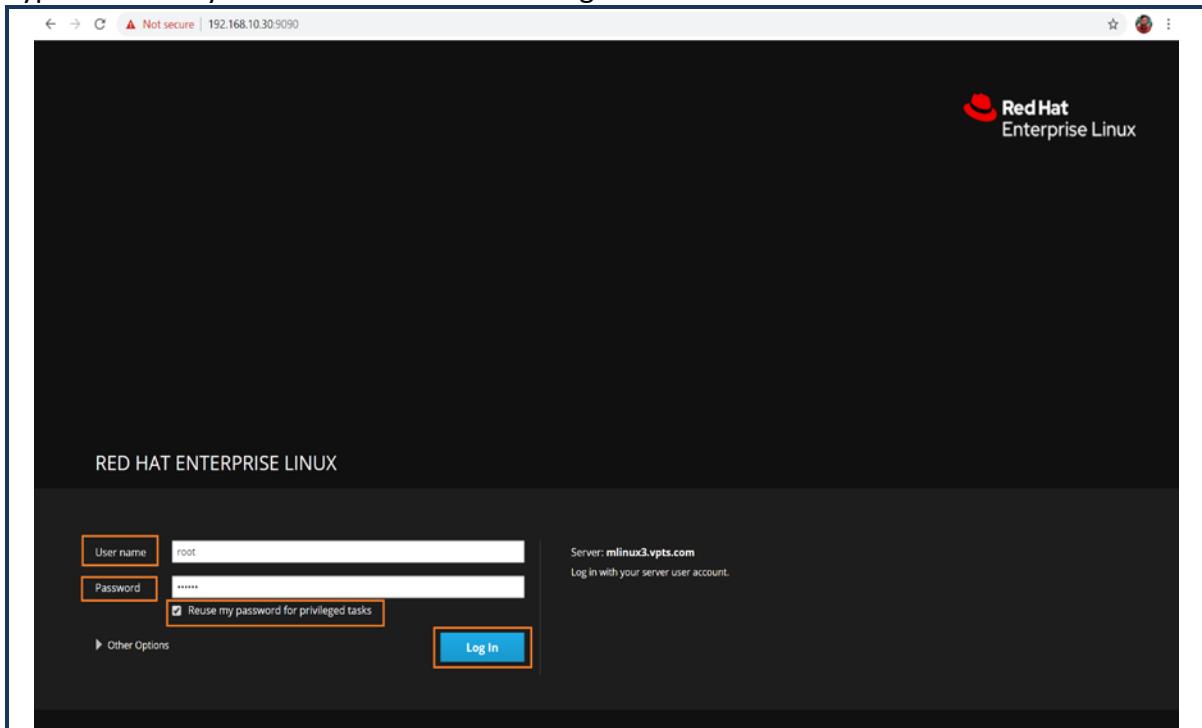
<https://<server IP>:9090>



Note: Initially it may reflect privacy error, click on **Advanced tab** and proceed to continue



Type root or any other user credentials to login with

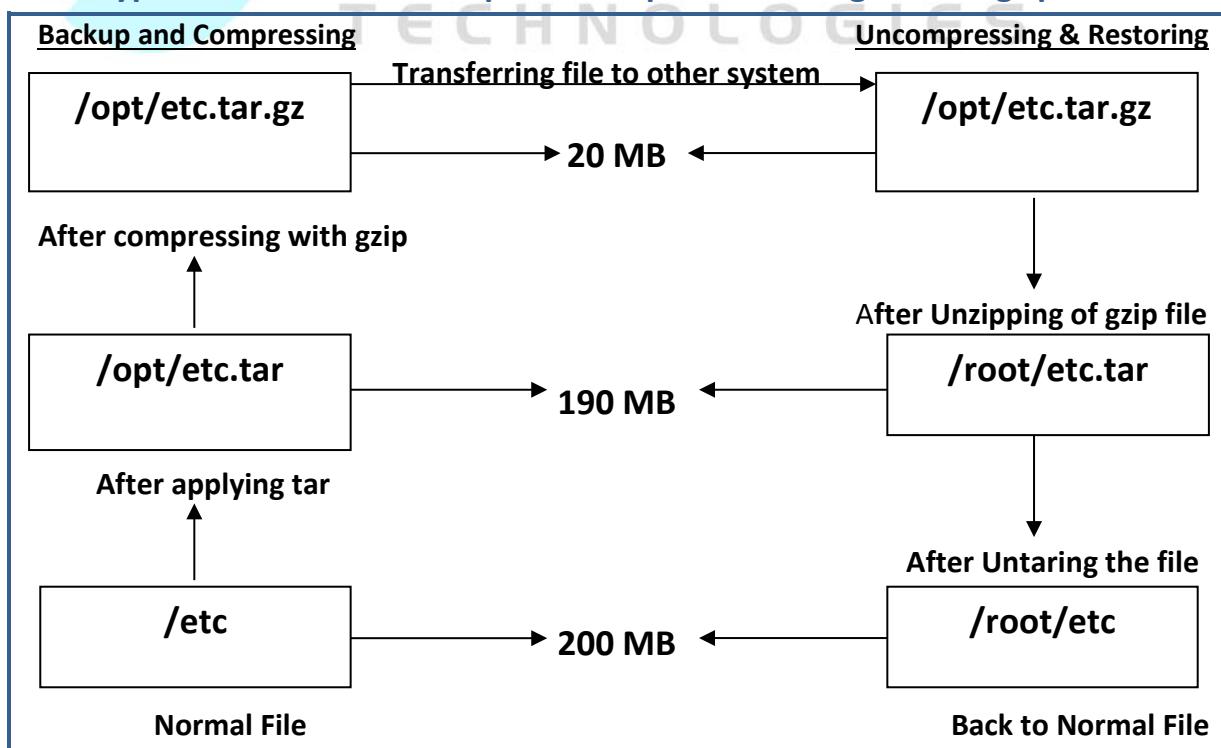


As you can see there are many options which can be explored to do various activities. Go ahead and explore tabs for more activities, in-fact the many important activities can be controlled or performed using cockpit.

BACKUP AND RESTORE

- In information technology, a **backup** or the process of **backing up** is making copies of data which may be used to *restore* the original after a data loss event.
- Backups have two distinct purposes
- The primary purpose is to recover data after its loss, be it by data deletion or corruption. Data loss is a very common experience of computer users. 67% of Internet users have suffered serious data loss.
- The secondary purpose of backups is to recover data from an earlier time, according to a user-defined data retention policy, typically configured within a backup application for how long copies of data are required.
- Backup is the most important job of a system administrator, as a system admin it is your duty to take backup of the data every day.
- Many companies have gone out of the market because of poor backup planning.
- The easiest way to back up your files is just copying. But if you have too many files to backup, copying and restoring may take too long time and it is not convenient. If there is a tool that can put many files into one file, the world will be better. Fortunately, 'tar' is used to create archive files. It can pack files or directories into a 'tar' file. It is like WinZip in Windows, without much compression.
- The **gzip** program compresses a single file. One important thing to remember about **gzip** is that, unlike **tar**, it replaces your original file with a compressed version. (The amount of compression varies with the type of data, but a typical text file will be reduced by 70 to 80 percent.)

A Typical scenario of backup and compression using tar and gzip



LAB WORK:-

To backup the file using tar

- To backup the file using tar the syntax is

```
#tar -cvf <destination and name to be > < source file>
#tar -cvf /opt/etc.tar /etc
```

```
[root@ linux ~]# tar -cvf /opt/etc.tar /etc
/etc/rc.d/rc0.d/K95firstboot
/etc/rc.d/rc0.d/K89iscsid
/etc/rc.d/rc0.d/K92iptables
/etc/rc.d/rc0.d/K50snmpd
/etc/rc.d/rc0.d/K03rhnsd
/etc/rc.d/rc0.d/K15httpd
/etc/rc.d/rc0.d/K80sssd
/etc/rc.d/rc0.d/K99microcode_ctl
/etc/rc.d/rc0.d/K83rpcgssd
/etc/rc.d/rc0.d/K84NetworkManager
/etc/rc.d/rc0.d/K50vsftpd
/etc/rc.d/rc0.d/K74nscd
/etc/rc.d/rc0.d/K83bluetooth
/etc/rc.d/rc0.d/K01smartd
/etc/rc.d/rc0.d/K02oddjobd
```

- Check the size of tar file by using du -h <file name > command

```
#du -h /opt/etc.tar
```

```
[root@ linux ~]# du -h /opt/etc.tar
29M    /opt/etc.tar
[root@ linux ~]#
```

Apply gzip on tar file and check the size.

- To apply gzip on a tar file, the syntax is

```
#gzip <file name>
#gzip /opt/etc.tar
```

```
[root@ linux ~]# gzip /opt/etc.tar
[root@ linux ~]#
```

- Now check the size of the file

```
[root@ linux ~]# cd /opt/
[root@ linux opt]# ls
etc.tar.gz  home  lost+found
[root@ linux opt]# du etc.tar.gz
7544    etc.tar.gz
[root@ linux opt]#
```

Transfer the file to other system and remove gzip and tar from it and check the size on every step.

- Let's transfer the file to other computer using scp

```
#scp /opt/etc.tar.gz 192.168.10.95:/root/
```

```
[root@ linux ~]# scp /opt/etc.tar.gz 192.168.10.95:/root/
etc.tar.gz                                         100% 7544KB   7.4MB/s  00:01
[root@ linux ~]#
```

- Login to the remote system, remove gzip it and check the size.
- To gunzip a file the syntax is

```
#gunzip <file name>
```

```
#gunzip etc.tar.gz
```

```
[root@ cl5 ~]# ls
anaconda-ks.cfg  Documents  etc.tar.gz  install.log.syslog  ktfile
Desktop          Downloads  install.log  ktdir                  Music
[root@ cl5 ~]# du -h etc.tar.gz
7.4M  etc.tar.gz
[root@ cl5 ~]# gunzip etc.tar.gz
[root@ cl5 ~]# ls
anaconda-ks.cfg  Documents  etc.tar      install.log.syslog  ktfile
Desktop          Downloads  install.log  ktdir                  Music
[root@ cl5 ~]# du -h etc.tar
29M  etc.tar
[root@ cl5 ~]#
```

Untar the file and check for the size of the file/directory

- To untar a file the syntax is

```
#tar -xvf <file name>
```

```
#tar -xvf etc.tar
```

```
[root@ cl5 ~]# tar -xvf etc.tar
|etc/sgml/xml-docbook-4.1.2-1.0-51.el6.cat
|etc/sgml/sgml-docbook-4.3-1.0-51.el6.cat
|etc/sgml/sgml.conf
|etc/sgml/xml-docbook.cat
|etc/sgml/sgml-docbook-4.1-1.0-51.el6.cat
[root@ cl5 ~]# ls
anaconda-ks.cfg  Downloads  install.log
Desktop          etc      install.log.syslog
Documents        etc.tar    ktdir
[root@ cl5 ~]# du -h etc
8.0K  etc/avahi/etc
8.0K  etc/avahi/services
32K   etc/avahi
4.0K   etc/openldap/cacerts
12K   etc/openldap
```

- To un-tar without unzipping, use #tar -zxvf (file name)
- To tar and zip together use #tar -zcvf destination and source.
- To view the content of tar file # tar -tvf filename.

JOB AUTOMATION

Automation with cron

- In any operating system, it is possible to create jobs that you want to reoccur. This process, known as **job scheduling**, is usually done based on user-defined jobs. For Red Hat or any other Linux, this process is handled by the cron service or a daemon called **crond**, which can be used to schedule tasks (also called *jobs*). By default, Red Hat comes with a set of predefined jobs that occur on the system (hourly, daily, weekly, monthly, and with arbitrary periodicity). As an administrator, however, you can define your own jobs and allow your users to create them as well.
- The importance of the job scheduling is that the critical tasks like taking backups, which the clients usually wants to be taken in nights, can easily be performed without the intervention of the administrator by scheduling a cron job. If the cron job is scheduled carefully than the backup will be taken at any given time of the client and there will be no need for the administrator to remain back at nights to take the backup.

Important Files related to cron and at

- **/etc/crontab** is the file which contains cron job format
- **/var/spool/cron/UserName** : contains job scheduled by users
- **/etc/cron.deny** is the file used to restrict the users from using cron jobs.
- **/etc/cron.allow** is used to allow only users whose names are mentioned in this file to use cron jobs. (this file does not exist by default)
- **/var/log/cron** is the log file for cron jobs

Crontab format

- To assign a job in the Crontab file the format used is the following



Options	Description
*	Is treated as a wild card. Meaning any possible value.
*/5	Is treated as every 5 minutes, hours, days, or months. Replacing the 5 with another numerical value will change this option.
2,4,6	Treated as an OR, so if placed in the hours, this could mean at 2, 4, or 6 o'clock.
9-17	Treats for any value between 9 and 17. So if placed in day of month this would be days 9 through 17. Or if put in hours it would be between 9 and 5.

Crontab Commands

Command	Explanation
crontab -e	Edit your crontab file, or create one if it doesn't already exist.
crontab -l	Display your crontab file.
crontab -r	Remove your crontab file.
crontab -u	If combined with -e, edit a particular user's Crontab file and if combined with -l, display a particular user's crontab file. If combined with -r, deletes a particular user's Crontab file

LAB WORK:-

To check the assigned cron jobs of currently logged in user

- To check the cron jobs the command is
#crontab -l

```
[root@ linux ~]# crontab -l
no crontab for root
[root@ linux ~]#
```

To check the cron jobs of a particular user

- To check a user's cron jobs, the syntax is
#crontab -l -u <user name>
#crontab -l -u myuser

```
[root@ linux ~]# crontab -l -u myuser
no crontab for myuser
[root@ linux ~]# crontab -lu myuser
no crontab for myuser
[root@ linux ~]#
```

Setting a job to display the current date for every minute on present console

- To set the above job the steps are
- Check the console on which you are working by following command
#tty

```
[root@ linux ~]# tty
/dev/pts/1
[root@ linux ~]#
```

Note: /dev/pts/1 is the console address

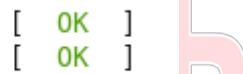
- Schedule the task as shown below
#crontab -e and enter the field as shown below and save it as in **VI editor**

```
*/1 * * * * date > /dev/pts/1
~[root@ linux ~]# crontab -e
crontab: installing new crontab
```

Note: where * means every possible value.

- Restart the cron services
#service crond restart (RHEL6) or #systemctl restart crond (RHEL7)

```
[root@ linux ~]# service crond restart
Stopping crond:
Starting crond:
[root@ linux ~]#
```



- Wait for a minute and check whether time is displaying or not. Every min time will be displayed as below.

```
[root@ linux ~]# Thu Oct 13 15:24:01 IST 2011
Thu Oct 13 15:25:01 IST 2011
Thu Oct 13 15:26:01 IST 2011
```

Schedule a cron job to create a directory “mydir” under “/root” on “Sunday 22 October at 1:30 AM”

- To schedule above job edit the crontab file as shown below and restart the service
#crontab -e

```
30 1 22 10 0 mkdir /root/mydir
~
:wq!#
[root@ linux ~]# crontab -e
crontab: installing new crontab
[root@ linux ~]#
```

Note: you can use 0 or 7 for Sunday.

Check whether it got created or not on scheduled day, if it created you can see the directory otherwise an error mail will be generated to your mail.

Schedule a job to run the backup script “bkpscript.sh” on every “Saturday 12:30 PM”

- In order to schedule above job the steps are.
- Check the location of script and also check whether it is having execute permission or not. If not then add the execute permissions to all user on it.

```
[root@ linux ~]# ls
anaconda-ks.cfg Desktop Downloads install.log.syslog
bkpscript.sh Documents install.log ktdir
[root@ linux ~]# pwd
/root
[root@ linux ~]# ls -l bkpscript.sh
-rw-r--r--. 1 root root 0 Oct 13 15:47 bkpscript.sh
[root@ linux ~]# chmod a+x bkpscript.sh
[root@ linux ~]# ls -l bkpscript.sh
-rwxr-xr-x. 1 root root 0 Oct 13 15:47 bkpscript.sh
[root@ linux ~]#
```

- Apply the job in **crontab**

```
#crontab -l
```

```
30 1 22 10 0 mkdir /root/mydir
30 12 * * 7 /root/bkpscript.sh
~  
:wq!
```

Schedule a job so that a user “myuser” should get a mail regarding meeting on 24th, 29th and 31st October at 2:25 PM.

- To set above task edit the crontab in following passion, and restart the service

```
#crontab -e -u <user name>
```

```
#crontab -e -u myuser
```

```
25 14 27,29,31 10 * echo "Meeting at 3:00 PM Today"
~  
:wq!
```

Schedule a job so that a user “myuser” should get the mail from 15th to 20th and 25th to 30st November as a reminder of some session at 2:25 PM

- This task is very much similar to the previous one but there is only a small change in format.

```
#crontab -e -u myuser
```

```
#crontab -e -u myuser
25 14 27,29,31 10 * echo "Meeting at 3:00 PM Today"
25 14 15-20,25-30 11 * echo "Class at study hall 3:00 PM Today"
~  
~  
:wq!
```

- There are still various method you can schedule the cron jobs, Do some **R&D** on it to find out more.

Restrict users “myuser” “john” “sam” from using cron jobs

- To restrict any user from using cron job facility, enter their names in **/etc/cron.deny** and save it

```
#vim /etc/cron.deny
```

```
myuser
john
sam
~
~
~
:wq!■
```

- Now login as one of those users and try to use crontab.

```
[root@ linux ~]# vim /etc/cron.deny
[root@ linux ~]# su - myuser
[ktuser@ linux ~]$ crontab -l
You (myuser) are not allowed to use this program (crontab)
See crontab(1) for more information
[myuser@ linux ~]$ crontab -e
You (myuser) are not allowed to use this program (crontab)
See crontab(1) for more information
[myuser@ linux ~]$ ■
```

- If you want to allow them to use cron job facilities again, remove their names from **/etc/cron.deny** file.

Allow only two users “musab” and “rahul” to use cron jobs out of all the users in the system

- Assuming that we have 100 users in our system, putting all 98 names in **/etc/cron.deny** file is a time consuming process. Instead of that, we can create one more file **/etc/cron.allow**, in which we can assign names of those users who are allowed to use cron jobs.
- Remove the **/etc/cron.deny** file and create **/etc/cron.allow**, still if both files are existing **cron.allow** file will be having precedence over **cron.deny** file. Just to avoid confusion it is good to remove **cron.deny** file

Note: **/etc/cron.deny** file exists by default, but we need to create **/cron.allow** file. If your name is not there in **cron.allow** file then you will not be allowed to use cron jobs, and as mentioned above, if both files are existing **cron.allow** file will be having precedence over **cron.deny** file. If neither **cron.deny** nor **cron.allow** files exists then only **root** can use cron jobs.

- Now, let's put those two users "musab" and "rahul" name in **/etc/cron.allow** file and check the results.

```
#vim /etc/cron.allow
```

```
musab
rahul
~
~
:wq!■
[root@ linux /]# vim /etc/cron.allow
[root@ linux /]# rm -f /etc/cron.deny
[root@ linux /]# su - vivek
[vivek@ linux ~]$ crontab -l
You (vivek) are not allowed to use this program (crontab)
See crontab(1) for more information
[vivek@ linux ~]$ exit
logout
[root@ linux /]# su - musab
[musab@ linux ~]$ crontab -l
no crontab for musab
[musab@ linux ~]$ exit
logout
[root@ linux /]# su - rahul
[rahul@ linux ~]$ crontab -l
no crontab for rahul
[rahul@ linux ~]$ exit
logout
[root@ linux /]# su -myuser
[myuser@ linux ~]$ crontab -l
You (myuser) are not allowed to use this program (crontab)
```



Note: To see man pages on cron job use **#man 4 crontabs** command

All the above are few examples to use cron, do some constant R&D's to know more about it.

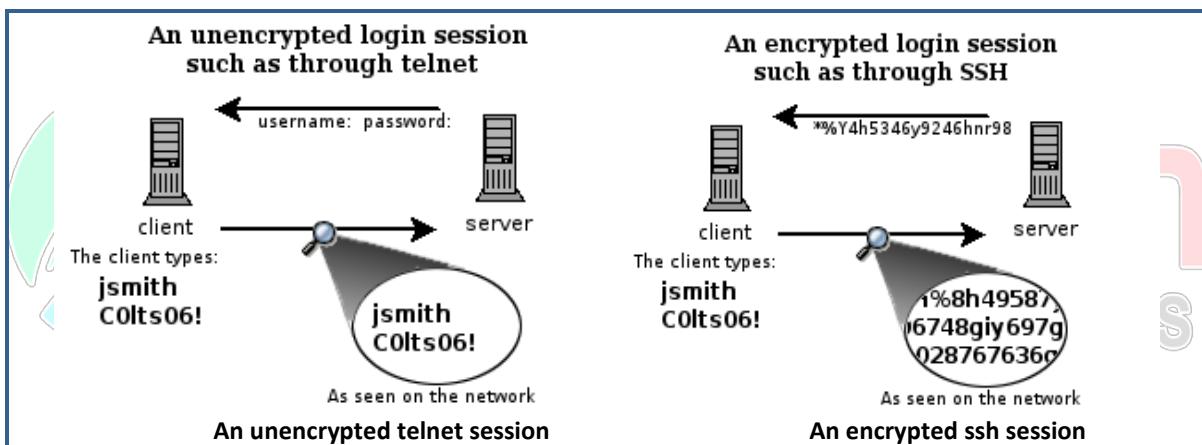
ADMINISTRATING REMOTE SYSTEM

- Remote shell Access using SSH

What Is SSH?

There are a couple of ways that you can access a shell (command line) remotely on most Linux/Unix systems. One of the older ways is to use the telnet program, which is available on most network capable operating systems. Accessing a shell account through the telnet method though poses a danger in that everything that you send or receive over that telnet session is visible in plain text on your local network, and the local network of the machine you are connecting to. So anyone who can "sniff" the connection in-between can see your username, password, email that you read, and command that you run. For these reasons you need a more sophisticated program than telnet to connect to a remote host.

SSH, which is an acronym for Secure SHell, was designed and created to provide the best security when accessing another computer remotely. Not only does it encrypt the session, it also provides better authentication facilities.



These two diagrams above show how a telnet session can be viewed by anyone on the network by using a sniffing program like Ethereal (now called Wireshark) or tcpdump. It is really rather trivial to do this and so anyone on the network can steal your passwords and other information. The first diagram shows user jsmith logging in to a remote server through a telnet connection. He types his username jsmith and password Colts06! which are viewable by anyone who is using the same networks that he is using.

The second diagram shows how the data in an encrypted connection like SSH is encrypted on the network and so cannot be read by anyone who doesn't have the session-negotiated keys, which is just a fancy way of saying the data is scrambled. The server still can read the information, but only after negotiating the encrypted session with the client.

- SSH configuration file is **/etc/ssh/sshd_config**
- SSH demon or service is **sshd**

LAB WORK:-

Accessing the remote machine using ssh

- To access the remote machine using ssh, the syntax is

#ssh <ip address/ host name of remote machine>

Note: hostname can only be used when the hostname is saved in **/etc/hosts** file or, if DNS is configured.

#ssh 192.168.10.98

```
[root@ linux .ssh]# ssh 192.168.10.95
The authenticity of host '192.168.10.95 (192.168.10.95)' can't be established.
RSA key fingerprint is 35:f4:34:b8:29:00:02:87:14:47:56:f5:bb:6b:4c:68.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.10.95' (RSA) to the list of known hosts.
```

The first time around it will ask you if you wish to add the remote host to a list of known_hosts, go ahead and say **yes**.

- Enter the password of the remote system correctly, once logged in check hostname and ip address to confirm login.

```
root@192.168.10.95's password:
Last login: Sun Sep 4 02:42:54 2011 from 192.168.1.10
[root@ cl5 ~]# hostname
cl5.mb.com
[root@ cl5 ~]# ifconfig eth0
eth0      Link encap:Ethernet HWaddr 00:0C:29:97:79:78
          inet addr:192.168.10.95 Bcast:192.168.10.255 Mask:255.255.255.0
```

- To leave the session, just type exit or logout command and you will be back to your own machine through which you are logged in.

```
[root@ cl5 ~]# hostname
cl5.mb.com
[root@ cl5 ~]# exit
logout
Connection to 192.168.10.95 closed.
[root@ linux .ssh]# hostname
linux.mb.com
[root@ linux .ssh]# ifconfig eth0
eth0      Link encap:Ethernet HWaddr 00:0C:29:A4:5E:C8
          inet addr:192.168.10.98 Bcast:192.168.10.255 Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fea4:5ec8/64 Scope:Link
```

Trust Relationship / Password less login using SSH keys

- As a system administrator, one person will be assigned to manage many systems, for example one person has to manage more than 10 systems at a time. In this situation admin has to transfer some files from one system to another 9 systems or vice versa, for every login on remote system it will prompt for password. Even for transferring files for every transfer we need to enter the password.
- Above situation will be very annoying for system admin to type password for every step. Therefore SSH provides a best way to escape password prompting every now and then.
- By generating SSH keys, a public key and a private key, an admin can copy the public key into other system and done, it will work as authorized access from the admin's system. Now whenever we are logging from admin's system to other system in which we have stored the public key of admin's system, it will not prompt us for password and we can login to that system as many time as we want without being prompt for the password.
- SSH keys are an implementation of public-key cryptography. They solve the problem of brute-force password attacks by making them computationally impractical.
- Public key cryptography uses a **public key** to **encrypt data** and a **private key** to **decrypt it**.



- To generate the SSH key pair, the syntax is
ssh-keygen

```
[root@ linux ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): █
```

It will prompt above to mention the file where these keys shoud be stored, to keep its default directory just press “Enter”. The default location will be **/root/.ssh/** directory

```
[root@ linux ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase): █
```

Now it will ask for passphrase, which will be used instead of password. The passphrase will only be asked every time you connect from other machine instead of its original password. Enter your desired passphrase twice, or leave it blank for no passphrase and press enter.

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
14:e4:49:b2:c4:b5:a3:b9:31:3e:f4:d6:11:5f:29:ee root@ktlinux.kt.com
The key's randomart image is:
+---[ RSA 2048]---+
|       .00=
|       ..= +
|       . * . . 0
|       + . + 0
|       * S . 0
|      o = . 0
|      + o . E
|      o
+-----+
```

Okay, now our keys are successfully generated, go to **/root/.ssh/** directory and check for the keys.

```
#cd /root/.ssh
```

```
[root@ linux ~]# cd /root/.ssh
[root@ linux .ssh]# ls
id_rsa id_rsa.pub known_hosts
[root@ linux .ssh]# cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAQEAw0sFIAZwmnB0KcBrRm71kze+RueIa0qx6HEPThdqdBpq0VG8IIBgKdLvrVZbm
PXlqmMnNDvJt7o8V9DSfh2vwqEk8SoCuSQz53PwGJWSfmFYepkVF+0qpe3hsv2vFzFJmAoMINZzobkiwNH6Up9cQFPqMdpmP
J5cTe24dQLQuasFUQwg/IF15PK8o7dk0CUf+86Pxdb9XS3qGIZ7n6ABsIhE0MTQ0F4uX/pnZNRWCzb7f8HFZ12x9Lsg20V0SU
bbSELZmDfT0mLCP0ErUZ7oK8oTELcXl/sQ3Yddr5b09Caeb410hKoNCUSiUOSIUmMrp3aSyaLIoSktJ89IK35DQ== root@kt
linux.mb.com
```

- The **id_rsa** is a private key and **id_rsa.pub** is the public key which will be used later to make password less login.

Copying the public key on Client system

- To copy the server's public key in client system, the command is
#ssh-copy-id -i <public key location> <clients IP address> (or user @ client IP)
#ssh-copy-id -i /root/.ssh/id_rsa.pub 192.168.10.95

```
[root@ linux ~]# ssh-copy-id -i /root/.ssh/id_rsa.pub 192.168.10.95
[root@192.168.10.95's password]
Now try logging into the machine, with "ssh '192.168.10.95'", and check in:
    .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
```

Enter the password of the client to proceed, check it on client side whether it is copied or not

Note: Public key can also be copied in a simple way by using
#ssh-copy-id <remote machine address>; ex: ssh-copy-id 192.168.10.95

Move to client machine and check whether the key is copied properly or not

- To check the key navigate to `/root/.ssh/` directory and check for `authorized_keys` file which will hold all the system which are authorized and will not be asked for password.

```
#cd /root/.ssh/
#cat authorized_keys
```

```
[root@ cl5 ~]# cd /root/.ssh/
[root@ cl5 .ssh]# ls
authorized_keys
[root@ cl5 .ssh]# cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAQEAvg0sFIAZwmnB0KcBrRm71kze+RueIa0qx6HEPThdqdBpq0VG8IIBgKdLvrVZbm
PX1qmMnNDvJt7o8V9DSfHl2vwqEk8SoCuSQz53PwGJWSfmFYepkVF+0qpe3hsv2vFzFjmAoMInZZobkiwNH6Up9cQFPqMdpmP
J5cTe24dQLQuasFUQwg/IF15PK8o7dk0CUf+86Pxdb9XS3qGIZ7n6ABsIhE0MTQOF4uX/pnZNRWCZb7f8HFZ12x9Lsg20V0SU
bbSELZmDfT0mLCPOErUZ7oK8oTELcXl/sQ3Yddr5b09Caeb410hKoNCUSIUOSIUmrp3aSyaLIo5kTJ89IK35DQ== [root@
linux.mb.com
[root@ cl5 .ssh]#
```

Try login to the client machine using SSH, check whether it is asking for password

- For logging into client machine the procedure is same as shown earlier.
- #ssh 192.168.10.95**

```
[root@mlinux1 ~]# ssh 192.168.10.95
Last login: Mon Sep 19 13:48:24 2016 from 192.168.10.95
[root@mlinux2 ~]# hostname
mlinux2.mb.com
[root@mlinux2 ~]#
```

Remote file transfer with SCP and RSYNC

SCP (SECURE COPY)

- scp stands for secure cp (copy), which means that you can copy files across an ssh connection that will be encrypted, and therefore secured. As scp will be using ssh protocol to transfer the data, hence it is termed as the safest method of transferring data from one location to another.

LAB WORK:

To copy a file using SCP to remote machine from source location

- We are having a file `myfile` in “`/`” directory, in the server `linux.mb.com` who’s IP is **192.168.10.98**, and we need to copy the same in **other** server’s i.e. `cl5.mb.com` with an IP **192.168.10.95**, `/root` directory.
- The syntax for SCP a file from source location.
`#scp <file name > <remote hosts IP >:<location to copy the file >`
`#scp /myfile 192.168.10.95:/root/`

```
[root@ linux ~]# hostname
linux.my.com
[root@ linux ~]# cat /myfile
Welcome to my Technologies
[root@ linux ~]# scp /ktfile 192.168.10.95:/root/
myfile
```

100% 31

- Now log in to destination system and check whether if the file is there.

```
[root@linux ~]# ssh 192.168.10.95
Last login: Fri Oct 14 15:29:23 2011 from 192.168.10.98
[root@cl5 ~]# cd /root/
[root@tcl5 ~]# ls
anaconda-ks.cfg  Documents  install.log      myfile  Pictures
Desktop          Downloads  install.log.syslog  Music   Public
[root@cl5 ~]# cat myfile
Welcome to my Technologies
```

To copy a file using SCP from a remote machine being in destination's location(reverse scp)

- Let's reverse the previous task, login to cl5 machine whose IP is 192.168.10.95, and transfer a file from linux machine whose IP is **192.168.10.98**
- Let's first remove the earlier copied file **myfile**, then copy it again from destination's location.
- The syntax for SCP a file from destination location.

#scp <source system's IP>:<location of file to be copied> <destination location to copy>

```
[root@cl5 ~]# rm myfile
rm: remove regular empty file `myfile'? y
[root@cl5 ~]# scp 192.168.10.98:/myfile /root/
myfile                                         100%   31
[root@cl5 ~]# cat myfile
Welcome to my Technologies
```

Note: Password will be asked for every transfer if public key is not saved on both locations, in our case we have already generated and copied the key, hence there is no password prompts.

To copy a directory using SCP to remote machine from source's location

- We are having a directory **mydir** in "/" directory, in the server **linux.mb.com** who's IP is **192.168.10.98**, and we need to copy the same in **other** server's i.e. **cl5(mb.com)** with an IP **192.168.10.95**, **/root** directory. Then,
- The syntax SCP a directory from source's location, the syntax is
#scp <option> <dir name > <remote hosts IP >:<location to copy the directory >
#scp -r /mydir 192.168.10.95:/root/

```
[root@linux ~]# tree /mydir
/ktdir
└── file1
    ├── file2
    ├── file3
    ├── file4
    └── file5

0 directories, 5 files
[root@linux ~]# scp -r /mydir 192.168.10.95:/root/
file1                                         100%   0
file4                                         100%   0
file5                                         100%   0
file2                                         100%   0
file3                                         100%   0
```

To copy a directory using SCP from a remote machine being in destination's location (reverse scp)

- Let's reverse the previous task, login to **cl5** machine who's IP is 192.168.10.95, and transfer a directory **mydir** from **linux** machine whose IP is **192.168.10.98**
- Let's first remove the earlier copied directory **mydir**, then copy it again being in destination's location.
- The syntax for SCP a file from destination location.

```
#scp <option> <source system's IP>:<location of file to be copied> <destination location to copy>
```

```
#scp -r 192.168.10.98:/mydir /root/
```

```
[root@ cl5 ~]# rm -rf /root/mydir/
[root@ cl5 ~]# scp -r 192.168.10.98:/mydir /root/
file1                               100%   0
file4                               100%   0
file5                               100%   0
file2                               100%   0
file3                               100%   0
[root@ cl5 ~]#
```

RSYNC (REMOTE SYNCHRONIZATION)

- rsync is a very good program for backing up/mirroring a directory tree of files from one machine to another machine, and for keeping the two machines "in sync." It's designed to speed up file transfer by copying the differences between two files rather than copying an entire file every time.
- For example, Assume that we are supposed to take the backup of a system and copy the same to another system. For first time we will copy entire directory, but every day if we copy entire directory it will kill lots of time. In such situation if rsync is used it will only copy the updated files/directories rather than copying all files/directories inside main directory, which saves lots of time and speedup the transfer
- If rsync is combined with ssh it makes a great utility to sync the data securely. If rsync is not used with ssh, the risk sniffing will always be there.

LAB WORK:-

Copy a directory using SCP, then update it and try rsync with SSH and check if the data is synced.

- As we have already copy a directory earlier using SCP from **linux** to **cl5** system, let's use it for rsync.
- Update the directory with some files in **linux** machine

```
[root@ linux ~]# cd /mydir
[root@ linux mydir]# ls
file1  file2  file3  file4  file5
[root@ linux mydir]# touch file{6..9}
[root@ linux mydir]# ls
file1  file2  file3  file4  file5  file6  file7  file8  file9
[root@ linux mydir]#
```

- Check the content of same directory in **cl5**

```
[root@ cl5 ~]# cd /root/mydir
[root@ cl5 mydir]# ls
file1  file2  file3  file4  file5
[root@ cl5 mydir]#
```

- Use rsync to sync the directory on **cl5** machine, with the one in **linux** machine
- The syntax to rsync a directory is

```
#rsync <options> <encryption> <source dir> <destination IP>:<location of destination dir>
#rsync -rv -e ssh /mydir 192.168.10.95:/root/
```

```
[root@ linux ~]# rsync -rv -e ssh /mydir 192.168.10.95:/root/
sending incremental file list
mydir/  file6
mydir/  file7
mydir/  file8
mydir/  file9
```

Observe that it is only copying the files which are not there in destination's folder.

Note: If you don't want to use ssh just remove –e option from above syntax, but the drawback of it is there will be no encryption

```
[root@ linux ~]# rsync -rv /mydir 192.168.10.95:/root/
sending incremental file list
mydir/  file1
mydir/  file2
mydir/  file3
mydir/  file4
mydir/  file5
mydir/  file6
mydir/  file7
mydir/  file8
mydir/  file9
```

- To compress the data and send it in archive mode use **-avz** instead of **-rv** in rsync

Sync a file using rsync with ssh

- Let's check the file called file1 on both **linux** and **cl5** machines

[root@ linux ~]# cat file1	[root@ cl5 ~]# cat file1
Welcome to Technologies	Welcome to Technologies

- Update the file **file1** in **linux**, sync it with rsync to **cl5** and check the file again.
- The syntax for syncing a file is
#rsync -avz -e ssh <source file> <destination ip>:<location of file >

```
[root@ linux mydir]# vim file1
[root@ linux mydir]# cat file1
Welcome to virtualpathTechnologies
AMEERPET HYDERABAD
[root@ linux mydir]# rsync -avz -e ssh /mydir/ file1 192.168.10.95:/root/mydir/
sending incremental file list
file1

sent 123 bytes received 37 bytes 106.67 bytes/sec
[root@ linux mydir]# cat file1
[root@ cl5 mydir]# cat file1
Welcome to Technologies
AMEERPET HYDERABAD
```

Like this you can use rsync in many ways to transfer the updated file or files/directory to other system.

Other important things in rsync

- **--dry-run** : dry-run is used to see the preview of transfer before doing actual transfer
Syntax: rsync -avz -e ssh /mydir 192.168.104.82:/opt --dry-run
- **Reverse rsync** : reverse rsync is referred to sync folder from destination to source machine. It is exactly same like reverse scp seen in scp chapter
Syntax: rsync -avz -e ssh 192.168.104.82/opt/mydir /mydir
Where 192.168.104.82/opt/mydir is remote machine directory and /mydir is local machine directory

SOFTWARE MANAGEMENT

To manage the software in Linux, two utilities are used,

1. RPM – REDHAT PACKAGE MANAGER
2. YUM – YELLOWDOG UPDATER MODIFIED OR DNF- DANDIFIED YUM

RPM –REDHAT PACKAGE MANAGER

RPM is a package managing system (collection of tools to manage software packages). RPM is a powerful software management tool for installing, uninstalling, verifying, querying and updating software packages. RPM is a straight forward program to perform the above software management tasks.

Features:

- RPM can verify software packages.
- RPM can be served as a powerful search engine to search for software's.
- Components, software's etc can be upgraded using RPM without having to reinstall them
- Installing, reinstalling can be done with ease using RPM
- During updates RPM handles configuration files carefully, so that the customization is not lost.

LAB WORK:-

To check all the installed packages in the system

- To check all the installed packages in the system, the syntax is
- **#rpm -qa** (where q stands for query, and a stands for all)

```
[root@ linux ~]# rpm -qa
Red_Hat_Enterprise_Linux-Release_Notes-6-en-US-1-21.el6.noarch
procps-3.2.8-14.el6.i686
net-snmp-libs-5.5-27.el6.i686
m17n-contrib-urdu-1.1.10-3.el6.noarch
bluez-libs-4.66-1.el6.i686
man-1.6f-29.el6.i686
xorg-x11-fonts-ISO8859-1-100dpi-7.2-9.1.el6.noarch
libXrender-0.9.5-1.el6.i686
nscd-2.12-1.7.el6.i686
dejavu-serif-fonts-2.30-2.el6.noarch
libXfixes-4.0.4-1.el6.i686
libchewing-0.3.2-27.el6.i686
dejavu-sans-mono-fonts-2.30-2.el6.noarch
libXdamage-1.1.2-1.el6.i686
```

Note: The output of above command will be very lengthier.

To check whether a particular package is installed or not

- To check whether a package is installed or not out of the list of installed package, the syntax is

```
#rpm -qa <package name> or  
#rpm -q < package name>  
#rpm -qa vsftpd or #rpm -q vsftpd
```

```
[root@ linux ~]# rpm -qa vsftpd  
vsftpd-2.2.2-6.el6.i686  
[root@ linux ~]# rpm -q vsftpd  
vsftpd-2.2.2-6.el6.i686  
[root@ linux ~]# █
```

- One more method of checking the installed package, when you are not sure about the package name, like whether it starts with capital letter and full name etc.

```
#rpm -qa | grep -i < package name>  
#rpm -qa |grep -i vsft*
```

```
[root@ linux ~]# rpm -qa |grep -i vsf*  
cvs-1.11.23-11.el6.i686  
lklug-fonts-0.6-4.20090803cvs.el6.noarch  
libedit-2.11-4.20080712cvs.1.el6.i686  
vsftpd-2.2.2-6.el6.i686  
xdg-utils-1.0.2-15.20091016cvs.el6.noarch  
[root@ linux ~]# █
```

To check whether a package is consistent or not, before installing it. (Testing the installation)

- To check the package's consistency,
- Move to the directory where you have kept the rpm package which you wish to install

```
[root@ linux ~]# cd /var/ftp/pub/rhel6/Packages/  
[root@ linux Packages]# ls |grep -i finger  
finger-0.17-39.el6.i686.rpm  
finger-server-0.17-39.el6.i686.rpm  
gdm-plugin-fingerprint-2.30.4-21.el6.i686.rpm
```

- The command used to check the package's consistency is

```
#rpm -ivh --test <package name>  
Where i = install, v= verbose view, and h = hash progress.  
#rpm -ivh -- test finger-0.17-39.el7.i686.rpm
```

```
[root@ linux Packages]# rpm -ivh --test finger-0.17-39.el6.i686.rpm  
warning: finger-0.17-39.el6.i686.rpm: Header V3 RSA/SHA256 Signature, key ID fd431d51: NOKEY  
Preparing... #### [100%]  
[root@ linux Packages]# █
```

If the installation status shows 100%, then the package is good or consistent.
But while showing the hash progress if it shows any error, then the package is inconsistent.

To install a package using rpm command and check whether it is installed properly or not.

- To install the package first we need to be in the directory of the package

```
[root@ linux ~]# cd /var/ftp/pub/rhel7/Packages/
[root@ linux Packages]# ls |grep -i finger
finger-0.17-39.el6.i686.rpm
finger-server-0.17-39.el6.i686.rpm
gdm-plugin-fingerprint-2.30.4-21.el6.i686.rpm
```

- To install the package the syntax is

```
#rpm -ivh <package name>
#rpm -ivh finger-0.17-39.el6.i686.rpm
```

```
[root@ linux Packages]# rpm -ivh finger-0.17-39.el6.i686.rpm
warning: finger-0.17-39.el6.i686.rpm: Header V3 RSA/SHA256 Signature, key ID fd431d51: NOKEY
Preparing... ################################################ [100%]
1:finger ################################################ [100%]
[root@ linux Packages]#
```

- To check whether it is installed or not

```
#rpm -qa finger
```

```
[root@ linux Packages]# rpm -qa finger
finger-0.17-39.el6.i686
[root@ linux Packages]#
```

- Check the installed package by using it command; finger is used to check user's details.

```
#finger <user name>
#finger myuser
```

```
[root@ linux Packages]# finger myuser
Login: myuser                                     Name: myuser
Directory: /home/myuser                           Shell: /bin/bash
Never logged in.
No mail.
No Plan.
[root@ linux Packages]#
```

To remove a package or uninstall the package

- To remove a package the syntax is

```
#rpm -evh < package name>
#rpm -evh finger
```

Verify it by #rpm -q or rpm -qa command

```
[root@node1 modules]# rpm -q finger
finger-0.17-52.el7.x86_64
[root@node1 modules]# rpm -evh finger
Preparing... ################################################ [100%]
Cleaning up / removing...
1:finger-0.17-52.el7 ################################################ [100%]
[root@node1 modules]# rpm -q finger
package finger is not installed
```

To see the information about the package before installing

- To see the info about a particular package which is not installed, move to the directory where you have kept the packages.

```
[root@ linux ~]# cd /var/ftp/pub/rhel6/Packages/
[root@ linux Packages]# ls |grep -i finger
finger-0.17-39.el6.i686.rpm
finger-server-0.17-39.el6.i686.rpm
gdm-plugin-fingerprint-2.30.4-21.el6.i686.rpm
```

- To see the info of a package, the syntax is

#rpm -qip <package name> (where **q** is for query, **i** is for install and **p** is for package)
#rpm -qip finger-0.17-39-el6.1686.rpm

```
[root@ linux Packages]# rpm -qip finger-0.17-39.el6.i686.rpm
warning: finger-0.17-39.el6.i686.rpm: Header V3 RSA/SHA256 Signature, key ID fd431d51: NOKEY
Name        : finger                           Relocations: (not relocatable)
Version     : 0.17                            Vendor: Red Hat, Inc.
Release     : 39.el6                          Build Date: Fri 20 Nov 2009 09:03:29 AM IST
Install Date: (not installed)           Build Host: ls20-bc1-14.build.redhat.com
Group       : Applications/Internet      Source RPM: finger-0.17-39.el6.src.rpm
Size        : 25730                           License: BSD
Signature   : RSA/8, Mon 16 Aug 2010 09:36:07 PM IST, Key ID 199e2f91fd431d51
Packager    : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
Summary     : The finger client
Description :
Finger is a utility which allows users to see information about system
users (login name, home directory, name, how long they've been logged
in to the system, etc.). The finger package includes a standard
finger client.
```



To see the information about the installed package

- To see the information or details about the installed package, the syntax is

#rpm -qi < package name >
#rpm -qi vsftpd

```
[root@ linux ~]# rpm -qi vsftpd
Name        : vsftpd                           Relocations: (not relocatable)
Version     : 2.2.2                            Vendor: Red Hat, Inc.
Release     : 6.el6                           Build Date: Wed 26 May 2010 06:16:46 PM IST
Install Date: Wed 12 Oct 2011 05:22:23 PM IST   Build Host: x86-009.build.bos.redhat.com
Group       : System Environment/Daemons      Source RPM: vsftpd-2.2.2-6.el6.src.rpm
Size        : 351576                           License: GPLv2 with exceptions
Signature   : RSA/8, Tue 17 Aug 2010 01:49:04 AM IST, Key ID 199e2f91fd431d51
Packager    : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
URL         : http://vsftpd.beasts.org/
Summary     : Very Secure Ftp Daemon
Description :
vsftpd is a Very Secure FTP daemon. It was written completely from
scratch.
[root@ linux ~]#
```

To check the package related to a command

- To check the package of a particular command, first check the installed location of a command

```
#which <command name>
#which cat
```

```
[root@node1 ~]# which cat
/usr/bin/cat
```

- Now, use the following command,

```
#rpm -qf <path of the command>
#rpm -qf /bin/cat
```

```
[root@node1 ~]# rpm -qf /usr/bin/cat
coreutils-8.22-23.el7.x86_64
```

To install a package forcefully

- Before installing a package forcefully, first understand a situation where we need this force option.
- Let us corrupt one command and show you how to install its package forcefully.
- First check the package of the command we are going to corrupt. Let say **mount** command

```
#which mount, #which date
```

```
[root@node1 ~]# which mount
/usr/bin/mount
[root@node1 ~]# which date
/usr/bin/date
```

- Okay, so we know the package of mount let's copy other commands content over mount command. Let copy **date** command's contents over **mount** command.

```
#cp /usr/bin/date /usr/bin/mount
```

```
[root@node1 ~]# cp /usr/bin/date /usr/bin/mount
cp: overwrite â/usr/bin/mountâ? y
```

- Now when you run mount command it will show date, that means it is corrupted.

```
[root@ linux ~]# mount
Sat Oct 15 03:25:34 IST 2011
[root@ linux ~]# date
Sat Oct 15 03:25:41 IST 2011
[root@ linux ~]#
```

- So, to fix the mount command we need to reinstall its package, let's install the package and check whether mount command is fixed or not. Move to the folder where you kept the packages and install it

```
#rpm -ivh util-linux-ng 2.17.2-6.el6.i686
```

```
[root@node1 ~]# rpm -qf /usr/bin/mount
util-linux-2.23.2-59.el7.x86_64
```

```
[root@node1 rhe17]# cd /var/ftp/pub/rhe17/Packages/
[root@node1 Packages]# ls util-linux*
util-linux-2.23.2-59.el7.i686.rpm  util-linux-2.23.2-59.el7.x86_64.rpm
[root@node1 Packages]# rpm -ivh util-linux-2.23.2-59.el7.x86_64.rpm
warning: util-linux-2.23.2-59.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature
        ID fd431d51: NOKEY
Preparing...                                           #####
package util-linux-2.23.2-59.el7.x86_64 is already installed
```

It says the package is already installed, check by using mount command whether it is working fine.

```
[root@ linux Packages]# mount
Sat Oct 15 03:30:54 IST 2011
[root@ linux Packages]# █
```

- Oops...!!! It isn't fixed yet, now to fix it, force installation is to be done, the syntax is

```
#rpm -ivh <package name> --force
```

```
# rpm -ivh util-linux-ng 2.17.2-6.el7_x86_64 -force
```

```
[root@node1 Packages]# rpm -ivh util-linux-2.23.2-59.el7.x86_64.rpm --force
warning: util-linux-2.23.2-59.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature,
        key ID fd431d51: NOKEY
Preparing...                                           ###### [100%]
Updating / installing...
  1:util-linux-2.23.2-59.el7          ##### [100%]
[root@node1 Packages]# mount |tail
/dev/mapper/rhel-root on / type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
selinuxfs on /sys/fs/selinux type selinuxfs (rw,relatime)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=32,pgrp=1,timeout=5,maxproto=5,direct,pipe_ino=17792)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,seclabel)
mqueue on /dev/mqueue type mqueue (rw,relatime,seclabel)
/dev/sdal on /boot type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,seclabel,size=188264k)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,relatime)
```

Okay then, we've not only installed the package successfully but we have also fixed the command. Congratulations.

To see the configuration files of the installed package

- To see the configuration files of the installed package, the syntax is

```
#rpm -qlc <package name>
```

```
[root@ linux Packages]# rpm -qlc vsftpd
/etc/logrotate.d/vsftpd
/etc/pam.d/vsftpd
/etc/vsftpd/ftpusers
/etc/vsftpd/user_list
/etc/vsftpd/vsftpd.conf
```

To see the directory with which a particular package is associated.

- To see the directory with which a package is associated, the syntax is

```
#rpm -qld <package name>
#rpm -qld vsftpd
```

```
[root@ linux ~]# rpm -qld vsftpd
/usr/share/doc/vsftpd-2.2.2/AUDIT
/usr/share/doc/vsftpd-2.2.2/BENCHMARKS
/usr/share/doc/vsftpd-2.2.2/BUGS
/usr/share/doc/vsftpd-2.2.2/COPYING
/usr/share/doc/vsftpd-2.2.2/Changelog
/usr/share/doc/vsftpd-2.2.2/EXAMPLE/INTERNET_SITE/README
/usr/share/doc/vsftpd-2.2.2/EXAMPLE/INTERNET_SITE/README.configuration
/usr/share/doc/vsftpd-2.2.2/EXAMPLE/INTERNET_SITE/vsftpd.conf
/usr/share/doc/vsftpd-2.2.2/EXAMPLE/INTERNET_SITE/vsftpd.xinetd
/usr/share/doc/vsftpd-2.2.2/EXAMPLE/INTERNET SITE NOINETD/README
```

To install a package without installing dependencies

- Some rpm requires some other packages to be installed before it can be installed; this requirement is termed as '**dependency**'. This means that before installing a package we need to install the required packages first, so that it can work properly.
- But sometimes we can skip installing the dependency, if we don't have that dependent software with us.
- The syntax for it is
`#rpm -ivh <package name> --nodeps`
`#rpm -ivh util-linux-ng 2.17.2-6.el6.i686 --nodeps`

To update a particular package

- To update a package the syntax is
`#rpm -Uvh <package name>`
`#rpm -Uvh vsftpd -2.2.4`

To check the changes are made after installation of package

- First let's make some changes in the configuration file of a package say **vsftpd**
`#vim /etc/vsftpd/vsftpd.conf`

```
# Example config file /etc/vsftpd/vsftpd.conf
#
##### The default compiled in settings are fairly paranoid. This sample file
# loosens things up a bit, to make the ftp daemon more usable.
# Please see vsftpd.conf.5 for all compiled in defaults.
```

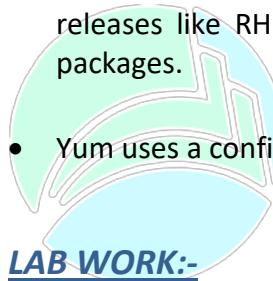
- Now run the following command and check for the result.

```
[root@ linux ~]# vim /etc/vsftpd/vsftpd.conf
[root@ linux ~]# rpm -V vsftpd
S.5....T. c /etc/vsftpd/vsftpd.conf
```

It is showing that some changes have been made. Isn't it cool!!!

YUM – YELLOWDOG UPDATER MODIFIED

- The Yellow dog Updater Modified (YUM) is a package management application for computers running Linux operating systems.
- Yum is a standard method of managing the installation and removal of software. Several graphical applications exist to allow users to easily add and remove packages; however, many are simply friendly interfaces with yum running underneath. These programs present the user with a list of available software and pass the user's selection on for processing. It is yum that actually downloads the packages and installs them in the background.
- Packages are downloaded from collections called **repositories**, which may be online, on a network, and/or on installation media. If one package due to be installed relies on another being present, this **dependency** can usually be resolved without the user needing to know the details. For example, a game being installed may depend on specific software to play its music. The problem of solving such dependencies can be handled by yum because it knows about all the other packages that are available in the repository.
- Yum will work only from Centos 5 / Red hat 5 and latest versions of fedora. For Old releases like RHEL 4 you need to use up2date command to update your rpm based packages.
- Yum uses a configuration file at **/etc/yum.conf**



VIRTUALPATH
TECHNO SOLUTIONS

Configuring a YUM server in RHEL6/7

To configure a YUM server the steps are.

- Make sure that vsftpd package is installed, if not install it.
- Copy entire RHEL6/7 DVD to “/var/ftp/pub/rhel” directory, where rhel dir is to made by us only it is not default dir.
- Make a repo file as “my.repo”in /etc/yum.repos.d directory
- Clean the yum cache and check the package list using yum command

Let's start with the first step

- Checking the vsftpd package is installed or not.

```
[root@node1 Packages]# rpm -q vsftpd  
vsftpd-3.0.2-25.el7.x86_64
```

- If it is not installed, then go to dvd's mount point and navigate to "Packages" directory and install it as shown below.

```
[root@ktlinux ~]# mount
gvfsd-fuse on /run/user/0/gvfs type fuse.gvfsd-fuse
/dev/sr0 on /run/media/root/RHEL-7.1 Server.x86_64 t
 0400,amode=0500,umask=udisks2)
[root@mlinux71 ~]#
```

- As we know the mount point of dvd is **/media/RHEL_6** (in rhel6) & **/run/media/root/RHEL7** (in rhel7), move to its location and enter into **Packages** directory.

```
[root@mlinux71 ~]# cd /run/media/ktuser/RHEL-7.1\ Server.x86_64/
[root@mlinux71 RHEL-7.1 Server.x86_64]# cd Packages/
[root@mlinux71 Packages]# ls vsftpd*
vsftpd-3.0.2-9.el7.x86_64.rpm
[root@mlinux71 Packages]# rpm -ivh vsftpd-3.0.2-9.el7.x86_64.rpm
warning: vsftpd-3.0.2-9.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID fd431d51: NOKEY
Preparing...                                           #####
Updating / installing...
 1:vsftpd-3.0.2-9.el7                                #####
[root@mlinux71 Packages]#
```

As it is already installed, it is not being installed.

Copy entire RHEL7 DVD to "/var/ftp/pub/rhel7" directory, Where rhel7 dir is to be created manually, it is not a default directory

- First make an directory "rhe7" under **/var/ftp/pub**

```
#mkdir /var/ftp/pub/rhel7
```

```
[root@mlinux71 Packages]# cd /var/ftp/pub/
[root@mlinux71 pub]# ls
[root@mlinux71 pub]# mkdir rhel7
[root@mlinux71 pub]#
```

- Now copy the RHEL6/7 DVD to **/var/ftp/pub/rhel** directory with its default permission

```
#cp -rvfp /run/media/root/RHEL-7.1\ Server.x86_64/* /var/ftp/pub/rhel7
```

```
[root@mlinux71 ~]# cp -rvfp /run/media/root/RHEL-7.1\ Server.x86_64/* /var/ftp/pub/rhel7
'/run/media/root/RHEL-7.1 Server.x86_64/addons' -> '/var/ftp/pub/rhel/addons'
'/run/media/root/RHEL-7.1 Server.x86_64/addons/HighAvailability' -> '/var/ftp/pub/rhel/addons/Hi
'/run/media/root/RHEL-7.1 Server.x86_64/addons/HighAvailability/TRANS.TBL' -> '/var/ftp/pub/rhel:/T
ANS.TBL'
'/run/media/root/RHEL-7.1 Server.x86_64/addons/HighAvailability/corosync-2.3.4-4.el7.x86_64.rpm' -
ons/HighAvailability/corosync-2.3.4-4.el7.x86_64.rpm'
'/run/media/root/RHEL-7.1 Server.x86_64/addons/HighAvailability/corosynclib-2.3.4-4.el7.i686.rpm'
```

Note:- it will take around 3-5 minutes copy all the data, based on the DVD

- Check the directory after copying is finished.

```
[root@node1 ~]# cd /var/ftp/pub/rhel7/
[root@node1 rhel7]# ls
addons  extra_files.json  isolinux  Packages          RPM-GPG-KEY-redhat-release
EFI      GPL              LiveOS   repodata        TRANS.TBL
EULA    images           media.repo  RPM-GPG-KEY-redhat-beta
```

Okay, then half of our configuration is completed.

Make a repo file as "my.repo" in /etc/yum.repos.d directory

- The file which we make inside /etc/yum.reops.d, will be functioning as the repository address and configuration file. Create the file with following details.

```
#vim /etc/yum.repos.d/my.repo
```

```
[MYREPO]
name=RHEL7.1
baseurl=file:///var/ftp/pub/rhel
enabled=1
gpgcheck=0
```

I guess there's some explanation requires about the fields we have entered.

- [MYREPO]** is the repo ID, which will be displayed while using yum repository.
- name** is the name given for the repository.
- baseurl** is the location of the dvd dump we have made.
- enabled** is to enable or disable the repository. The possible value for it is **0** and **1**, where **0** means disable and **1** means enabled.
- gpgcheck** gpgcheck= GNU privacy guard. gpgcheck used for signature verification from its central database. If signature verification is successful then you are sure about the security. If you set the value of gpgcheck is 1 then it asks for signature verification else it doesn't.
- Example of gpgkey in RHEL- *RPM-GPG-KEY-redhat-release*

Clean the yum cache and check the package list using yum command

- To clear the cache use the following command

```
#yum clean all
```

```
[root@mlinux71 ~]# yum clean all
Loaded plugins: langpacks, product-id, subscription-manager
This system is not registered to Red Hat Subscription Management.
subscription-manager to register.
Cleaning repos: MYREPO
Cleaning up everything
[root@mlinux71 ~]#
```

If the configuration is correct, then the following output will be displayed, otherwise there will be some errors displayed.

- Now let's check whether our repository is functioning properly or not.

```
#yum repolist (to list all the repositories in the system)
```

```
[root@mlinux71 ~]# yum repolist
Loaded plugins: langpacks, product-id, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register
.
repo id                                repo name                status
MYREPO                                    RHEL7.1                  4,371
repolist: 4,371
[root@mlinux71 ~]#
```

Start the vsftpd service and make it permanent

```
[root@mlinux71 ~]# systemctl start vsftpd; systemctl enable vsftpd
ln -s '/usr/lib/systemd/system/vsftpd.service' '/etc/systemd/system/multi-user.target.wants/vsftpd.service'
[root@mlinux71 ~]#
```

Allow ftp in firewalld in rhel7 if it is running

```
[root@mlinux71 ~]# firewall-cmd --add-service=ftp --permanent
success
[root@mlinux71 ~]# firewall-cmd --reload
success
[root@mlinux71 ~]# firewall-cmd --list-all
```

Configure the yum client and check whether yum server is responding to it.

Configuring a yum client is very simple with just three steps.

- Make a repo file /etc/yum.repo.d/ as “mycl.repo”
- Clean the cache and check whether yum server is responding or not

Make a repo file /etc/yum.repo.d/ as “mycl.repo”

- Just make a repo file like we did for server but with only one change in baseurl as shown below

```
#vim /etc/yum.repos.d/mycl.repo
```

```
[MYCL]
name=rhel7cl
baseurl=ftp://192.168.106.81/pub/rhel7
enabled=1
gpgcheck=0
```

Note: - baseurl =ftp://192.168.10.81/pub/rhel refers to the server's ftp address.

Clean the cache and check whether yum server is responding or not

- Just clean the cache as we have done earlier in server's configuration.

```
[root@ktcl5 ~]# yum clean all
Loaded plugins: refresh-packagekit, rhnplugin
Cleaning up Everything
[root@ktcl5 ~]#
```

- Check whether the server is responding to client's yum request.

```
#yum repolist
```

```
[root@mlinux72 ~]# yum repolist
Loaded plugins: langpacks, product-id, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.

MYCL
(1/2): MYCL/group_gz                                | 4.1 kB   00:00
(2/2): MYCL/primary_db                               | 134 kB   00:01
                                                | 3.4 MB   00:01
repo id                                              repo name          status
MYCL                                                 rhel7cl           4,371

repolist: 4,371

[root@mlinux72 ~]# yum list |more
Loaded plugins: langpacks, product-id, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.

Installed Packages
GConf2.x86_64                                         3.2.6-8.el7          @anaconda/7.1
ModemManager.x86_64                                    1.1.0-6.git20130913.el7    @anaconda/7.1
ModemManager-glib.x86_64                               1.1.0-6.git20130913.el7    @anaconda/7.1
NetworkManager.x86_64                                 1:1.0.0-14.git20150121.b4ea599c.el7
                                                       @anaconda/7.1
NetworkManager-adsl.x86_64                            1:1.0.0-14.git20150121.b4ea599c.el7
                                                       @anaconda/7.1
```

If your output is like this then you have successfully configured a yum client as well.

Congrats!!! Now you can configure as many as clients you want.

INTRODUCTION TO DNF – DANDIFIED YUM FROM RHEL8:

DNF stands for Dandified YUM is a software package manager for RPM-based Linux distributions. It is used to install, update and remove packages in the Fedora/RHEL/CentOS operating system. It is the default package manager of Fedora 22, CentOS8 and RHEL8. DNF is the next generation version of YUM and intended to be the replacement for YUM in RPM-based systems. DNF is powerful and has robust features than you'll find in yum. DNF makes it easy to maintain groups of packages and capable of automatically resolving dependency issues.

Configuring a DNF server in RHEL8

To configure a DNF YUM server the steps are.

- Make sure that vsftpd package is installed, if not install it.
- Copy entire RHEL8 DVD to “/var/ftp/pub/rhel8” directory, where rhel8 dir hast to be created manually by us only, as it is not default dir.
- Make a repo file as “my.repo”in /etc/yum.repos.d directory
- Clean the yum cache and check the package list using yum command

Let's start with the first step

- Checking the vsftpd package is installed or not.

```
[root@localhost ~]# rpm -q vsftpd
package vsftpd is not installed
```

- If it is not installed, mount RHEL8 DVD on some mount point and navigate to “Packages” directory and install it as shown below.

```
[root@localhost ~]# mount /dev/sr0 /media
mount: /media: WARNING: device write-protected, mounted
[root@localhost ~]# mount |grep sr0
/dev/sr0 on /media type iso9660 (ro,relatime,nojoliet,c:
```

- As we know the mount point of DVD is /media, move to its location and enter into **AppStream/ Packages** directory and install vsftpd package using rpm

```
[root@localhost ~]# cd /media
[root@localhost media]# ls
AppStream  EFI  extra_files.json  images  media.repo          RPM-GPG-KEY-redhat-release
BaseOS     EULA  GPL              isolinux  RPM-GPG-KEY-redhat-beta  TRANS.TBL
[root@localhost media]# cd AppStream/
[root@localhost AppStream]# ls
Packages  repodata
[root@localhost AppStream]# cd Packages/
[root@localhost Packages]# ls vsftpd*
vsftpd-3.0.3-28.el8.x86_64.rpm
[root@localhost Packages]#
[root@localhost Packages]# rpm -ivh vsftpd-3.0.3-28.el8.x86_64.rpm
warning: vsftpd-3.0.3-28.el8.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID fd431d51: NOKEY
Verifying...                                              #####
Preparing...                                              #####
Updating / installing...
 1:vsftpd-3.0.3-28.el8                                #####

```

Copy entire RHEL8 DVD to “/var/ftp/pub/rhel8” directory, Where rhel8 dir is to be created manually, as it is not a default directory.

Note: make sure around 8 GB space is free in /var to copy the data

- First make an directory “rhel8” under /var/ftp/pub

```
#mkdir /var/ftp/pub/rhel8
```

```
[root@localhost ~]# cd /var/ftp/pub/
[root@localhost pub]# mkdir rhel8
[root@localhost pub]# ls
rhel8
```

- Now copy the RHEL8 DVD to /var/ftp/pub/rhel directory with its default permission

```
#cp -rvfp /media/* /var/ftp/pub/rhel8
```

```
[root@localhost pub]# cp -rvfp /media/* /var/ftp/pub/rhel8/
'/media/AppStream' -> '/var/ftp/pub/rhel8/AppStream'
'/media/AppStream/repo' -> '/var/ftp/pub/rhel8/AppStream/repo'
'/media/AppStream/repo/0153152556c65c34e8e52a6aeacae37951b697ac654
ar/ftp/pub/rhel8/AppStream/repo/0153152556c65c34e8e52a6aeacae37951
z'
```

Note:- it will take around 3-5 minutes copy all the data, based on the DVD

- Check the directory after copying is finished.

```
[root@localhost ~]# cd /var/ftp/pub/rhel8/
[root@localhost rhel8]# ls
AppStream  EFI  extra_files.json  images  media.repo      RPM-GPG-KEY-redhat-release
BaseOS    EULA  GPL              isolinux  RPM-GPG-KEY-redhat-beta  TRANS.TBL
```

Okay, then half of our configuration is completed.

Make a repo file as “my.repo”in /etc/yum.repos.d directory

- The file which we make inside /etc/yum.repos.d, will be functioning as the repository address and configuration file. Create the file with following details.

```
#vim /etc/yum.repos.d/my.repo
```

```
[BaseOS]
name=rhel8_BaseOS
baseurl=file:///var/ftp/pub/rhel8/BaseOS
enabled=1
gpgcheck=1
gpgkey=file:///var/ftp/pub/rhel8/RPM-GPG-KEY-redhat-release

[App_Stream]
name=rhel8_App_Stream
baseurl=file:///var/ftp/pub/rhel8/AppStream
enabled=1
gpgcheck=1
gpgkey=file:///var/ftp/pub/rhel8/RPM-GPG-KEY-redhat-release
```

I guess there's some explanation requires about the fields we have entered.

- [MYREPO]** is the repo ID, which will be displayed while using yum repository.
- name** is the name given for the repository.
- baseurl** is the location of the dvd dump we have made.

- **enabled** is to enable or disable the repository. The possible value for it is **0** and **1**, where **0** means disable and **1** means enabled.
- **gpgcheck** gpgcheck= GNU privacy guard. gpgcheck used for signature verification from its central database. If signature verification is successful then you are sure about the security. If you set the value of gpgcheck is 1 then it asks for signature verification else it doesn't.
- Example of gpgkey in RHEL- *RPM-GPG-KEY-redhat-release*

Clean the yum cache and check the package list using yum command

- To clear the cache use the following command

```
#dnf clean all
```

```
[root@mlinux3 ~]# dnf clean all
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management.
12 files removed
```

If the configuration is correct, then the following output will be displayed, otherwise there will be some errors displayed.

- Now let's check whether our repository is functioning properly or not.

```
#dnf repolist (to list all the repositories in the system)
```

```
[root@mlinux3 ~]# dnf repolist
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You
rhel8_BaseOS
rhel8_BaseOS
Last metadata expiration check: 0:00:01 ago on Tuesday 31 March 2020
repo id
repo name
rhel8_App_Stream
rhel8_BaseOS
rhel8_BaseOS
```

In order to share files with ftp, make the following changes in ftp config

```
#vim /etc/vsftpd/vsftpd.conf
```

```
#
# Allow anonymous FTP? (Beware - allowed by default if you comment this out).
anonymous_enable=YES
#
```

Start the vsftpd service and make it permanent

```
[root@mlinux71 ~]# systemctl start vsftpd; systemctl enable vsftpd
ln -s '/usr/lib/systemd/system/vsftpd.service' '/etc/systemd/system/multi-user.target.wants/vsftpd.service'
[root@mlinux71 ~]#
```

Allow ftp in firewalld, if it is running

```
[root@mlinux71 ~]# firewall-cmd --add-service=ftp --permanent
success
[root@mlinux71 ~]# firewall-cmd --reload
success
[root@mlinux71 ~]# firewall-cmd --list-all
```

Configure the yum client and check whether yum server is responding to it.

Configuring a yum client is very simple with just three steps.

- Make a repo file /etc/yum.repos.d/ as “mycl.repo”
- Clean the cache and check whether yum server is responding or not

Make a repo file /etc/yum.repos.d/ as “mycl.repo”

- Just make a repo file like we did for server but with only one change in baseurl as shown below

```
#vim /etc/yum.repos.d/mycl.repo
```

```
[BaseOS]
name=rhel8_BaseOS
baseurl=ftp://192.168.10.30/pub/rhel8/BaseOS
enabled=1
gpgcheck=1
gpgkey=ftp://192.168.10.30/pub/rhel8/RPM-GPG-KEY-redhat-release

[App_Stream]
name=rhel8_App_Stream
baseurl=ftp://192.168.10.30/pub/rhel8/AppStream
enabled=1
gpgcheck=1
gpgkey=ftp://192.168.10.30/pub/rhel8/RPM-GPG-KEY-redhat-release
```

Note: - baseurl =ftp://192.168.10.30/pub/rhel8 refers to the server's ftp address.

Clean the cache and check whether yum server is responding or not

- Just clean the cache as we have done earlier in server's configuration.

```
[root@mlinux4 ~]# dnf clean all
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management.
0 files removed
```

- Check whether the server is responding to client's yum request.

```
#yum repolist
```

```
[root@mlinux4 ~]# dnf repolist
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You
rhel8_BaseOS
rhel8_App_Stream
Last metadata expiration check: 0:00:02 ago on Tuesday 31 March 2020
repo id                                repo name
App_Stream                               rhel8_App_Stream
BaseOS                                  rhel8_BaseOS
```

```
[root@mlinux4 ~]# dnf list |more
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You can use
Last metadata expiration check: 0:03:03 ago on Tuesday 31 March 2020 09:55:37
Installed Packages
GConf2.x86_64                                3.2.6-22.el8
ModemManager.x86_64                            1.10.4-1.el8
ModemManager-glib.x86_64                        1.10.4-1.el8
NetworkManager.x86_64                           1:1.20.0-3.el8
NetworkManager-adsl.x86_64                      1:1.20.0-3.el8
NetworkManager-bluetooth.x86_64                 1:1.20.0-3.el8
```

If your output is like this then you have successfully configured a yum client as well.
 Congrats!!! Now you can configure as many as clients you want

Working with YUM commands

To list the available packages in the repository

- #yum list (or) #yum list all (or) #yum list |more (to view line wise)
As we have seen the first command, second will also give exactly the same output. Let us see the third command
#yum list |more or #dnf list |more

```
[root@ linux ~]# yum list |more
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
Installed Packages
ConsoleKit.i686                               0.4.1-3.el6          @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0                0.4.1-3.el6          @anaconda-RedHatEnterpr
ConsoleKit-libs.i686                            1:0.8.1-5.el6        @anaconda-RedHatEnterpr
NetworkManager.i686                            1:0.8.1-5.el6        @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0                1:0.8.1-5.el6        @anaconda-RedHatEnterpr
NetworkManager-glib.i686                         1:0.8.1-5.el6        @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0                1:0.8.1-5.el6        @anaconda-RedHatEnterpr
NetworkManager-gnome.i686                        1:0.8.1-5.el6        @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0                2.14.17-3.1.el6     @anaconda-RedHatEnterpr
DRBit2.i686                                    0.5.8-13.el6         @anaconda-RedHatEnterpr
iseLinux-201009221732.i386/6.0                -More--
```



To list all the installed packages in the system.

- To view all the installed packages in the system, the syntax is

#yum list installed or #dnf list installed

```
[root@ linux ~]# yum list installed
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
Installed Packages
ConsoleKit.i686                               0.4.1-3.el6          @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0            0.4.1-3.el6          @anaconda-RedHatEnt
ConsoleKit-libs.i686                            0.4.1-3.el6          @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0            0.4.1-3.el6          @anaconda-RedHatEnt
ConsoleKit-x11.i686                            0.4.1-3.el6          @anaconda-RedHatEnt
erpriseLinux-201009221732.i386/6.0            014-1.el6           @anaconda-RedHatEnt
```

To check a particular package is installed or not

- To check whether a package is installed or not the syntax is

#yum list installed <package name> or #dnf list installed <package name>

#yum list installed vsftpd or #dnf installed vsftpd

```
[root@ linux ~]# yum list installed vsftpd
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
Installed Packages
vsftpd.i686           2.2.2-6.el6          @anaconda-RedHatEnterpriseLinux-201009221732.i386/6.0
[root@ linux ~]#
```

To install a package using yum

- Installing a package using yum does not require full package name as in the case of rpm, and it also automatically resolves the dependencies as well.
- The syntax for installing a package is

#yum install <package name> or #dnf install <package name>

#yum install dovecot* or #dnf install dovecot* (where * means anything with name "dovecot")

```
[root@mlinux4 ~]# dnf install dovecot*
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.
Last metadata expiration check: 0:16:23 ago on Tuesday 31 March 2020 09:55:37 PM IST.
Dependencies resolved.
=====
 Package           Architecture Version      Repository
 =====
Installing:
dovecot            x86_64        1:2.2.36-10.el8    App_Stream
dovecot-mysql      x86_64        1:2.2.36-10.el8    App_Stream
dovecot-pgsql      x86_64        1:2.2.36-10.el8    App_Stream
dovecot-pigeonhole x86_64        1:2.2.36-10.el8    App_Stream
Installing dependencies:
clucene-core       x86_64        2.3.3.4-31.20130812.e8e3d20git.el8   App_Stream
libpq              x86_64        10.5-1.el8      App_Stream
mariadb-connector-c x86_64        3.0.7-1.el8      App_Stream
mariadb-connector-c-config noarch       3.0.7-1.el8      App_Stream
=====
Transaction Summary
=====
Install 8 Packages

Total download size: 6.1 M
Installed size: 24 M
Is this ok [y/N]:
```

It will prompt you for y/n (Yes/No) to continue, type y and continue installing the package

- installing a package without being prompt for y or n, the syntax is
#yum install <package name> -y or # dnf install <package name> -y
#yum install dovecot -y or #dnf install dovecot -y

```
[root@mlinux4 ~]# dnf install dovecot -y
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.
Last metadata expiration check: 0:07:49 ago on Tuesday 31 March 2020 10:13:05 PM IST.
Dependencies resolved.
=====
Package           Architecture     Version          Repository
=====
Installing:
dovecot           x86_64          1:2.2.36-10.el8      App_Stream
Installing dependencies:
clucene-core      x86_64          2.3.3.4-31.20130812.e8e3d20git.el8      App_Stream
Transaction Summary
=====
Install 2 Packages

Total download size: 5.2 M
Installed size: 21 M
Downloading Packages:
(1/2): clucene-core-2.3.3.4-31.20130812.e8e3d20git.el8.x86_64.rpm
(2/2): dovecot-2.2.36-10.el8.x86_64.rpm
=====
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing   :
  Installing  : clucene-core-2.3.3.4-31.20130812.e8e3d20git.el8.x86_64
  Running scriptlet: dovecot-1:2.2.36-10.el8.x86_64
  Installing   : dovecot-1:2.2.36-10.el8.x86_64
  Running scriptlet: dovecot-1:2.2.36-10.el8.x86_64
  Verifying    : clucene-core-2.3.3.4-31.20130812.e8e3d20git.el8.x86_64
  Verifying    : dovecot-1:2.2.36-10.el8.x86_64
Installed products updated.

Installed:
  dovecot-1:2.2.36-10.el8.x86_64                           clucene-core-2.3.3.4-31.20130812.e8e3d20git.el8.x86_64

Complete!
[root@mlinux4 ~]#
```

To remove the package with yum command

- To remove the package using yum command, the syntax is
#yum remove <package name> or #dnf remove <package name>
#yum remove finger -y or #dnf remove <package name>

```
[root@mlinux4 ~]# dnf remove dovecot -y
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.
Modular dependency problems:

  Problem 1: conflicting requests
    - nothing provides module/perl:5.26 needed by module perl-DBD-SQLite:1.58:8010020190322125518:073fa5fe-0.x86_64
  Problem 2: conflicting requests
    - nothing provides module/perl:5.26 needed by module perl-DBI:1.641:8010020190322130042:16b3ab4d-0.x86_64
Dependencies resolved.
=====
Package           Architecture     Version
=====
Removing:
dovecot           x86_64          1:2.2.36-10.el8
Removing unused dependencies:
clucene-core      x86_64          2.3.3.4-31.20130812.e8e3d20git.el8
Transaction Summary
=====
Remove 2 Packages
```

To update the package using yum

- To update the package using yum command, the syntax is
#yum update <package name> or #dnf update <package name>
#yum update httpd or #dnf update httpd

```
[root@ linux ~]# yum update httpd
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
Setting up Update Process
No Packages marked for Update
[root@ linux ~]#
```

As there are no updates available for it, it is not showing anything to update

To install a package locally from a folder, pen drive or cd rom

- Move to the package where you have stored the package to be installed

```
[root@ cl5 ~]# cd /
[root@ cl5 /]# ls
bin      etc          lib      mnt      root      sys
boot    finger-0.17-39.el6.i686.rpm lost+found  net      sbin      tmp
cgroup   ftp-0.17-51.1.el6.i686.rpm media     opt      selinux   usr
dev      home         misc     proc     srv      var
```

- The syntax for installing a package locally is
#yum localinstall <package name> -y # dnf localinstall <package name>
#yum localinstall finger* -y (or) #yum localinstall finger-0.17-39.el6.i686.rpm -y
#dnf localinstall finger* -y #dnf localinstall finger-0.17-39.el6.i686.rpm -y

```
[root@ cl5 /]# yum localinstall finger-0.17-39.el6.i686.rpm -y
Setting up Local Package Process
Examining finger-0.17-39.el6.i686.rpm: finger-0.17-39.el6.i686
Marking finger-0.17-39.el6.i686.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package finger.i686 0:0.17-39.el6 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package      Arch      Version       Repository      Size
=====
Installing:
finger      i686      0.17-39.el6      /finger-0.17-39.el6.i686      25 k

Transaction Summary
=====
Install      1 Package(s)
Upgrade      0 Package(s)

Total size: 25 k
Installed size: 25 k
Downloading Packages:
```

To see the information about the package

#yum info <package name> or #dnf info <package name>

#yum info vsftpd or dnf info vsftpd

```
[root@mlinux3 ~]# dnf info vsftpd
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subscription Management. You can use subscrhel8_BaseOS
rhel8_App_Stream
Installed Packages
Name        : vsftpd
Version     : 3.0.3
Release     : 28.el8
Architecture: x86_64
Size        : 356 k
Source      : vsftpd-3.0.3-28.el8.src.rpm
Repository   : @System
Summary     : Very Secure Ftp Daemon
URL         : https://security.appspot.com/vsftpd.html
License     : GPLv2 with exceptions
Description  : vsftpd is a Very Secure FTP daemon. It was written completely from
               : scratch.
```

To list and install a group of packages using yum

- To list the group of package the syntax is

#yum grouplist or dnf grouplist

```
[root@mlinux4 ~]# dnf grouplist
Updating Subscription Management repositories.
Unable to read consumer identity
This system is not registered to Red Hat Subsc
Last metadata expiration check: 0:23:25 ago on
Available Environment Groups:
  Server
  Minimal Install
  Workstation
  Custom Operating System
  Virtualization Host
Installed Environment Groups:
  Server with GUI
Installed Groups:
  Container Management
  Headless Management
Available Groups:
  Legacy UNIX Compatibility
  Development Tools
  .NET Core Development
  Graphical Administration Tools
  Network Servers
  RPM Development Tools
  Scientific Support
  Security Tools
  Smart Card Support
  System Tools
```

- To install package group called “Development Tools”, the syntax is

#yum groupinstall <group name> -y or dnf groupinstall ‘group name’

#yum groupinstall ‘Development Tools’ -y #dnf grouplist ‘Development Tools’ -y

```
[root@node1 ~]# yum groupinstall 'Development Tools' -y
```

Removing a Group package using yum

- To remove a group, the syntax is

#yum groupremove <group name> or dnf groupremove <group name>

#yum groupremove ‘Development Tools’ -y or #dnf groupremove ‘Development Tools’

```
[root@node1 ~]# yum groupremove 'Development Tools' -y
```

```
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
```

Other commands with yum :

To reinstall a package

#yum reinstall <package name> -y or #dnf reinstall <package name>

To see group information

#yum groupinfo <group name> or #dnf groupinfo ‘group name’



**The software management can be learnt more by using manual pages like man yum or dnf
and also man rpm etc.**

MANAGING PROCESS

- A Linux process is a program running in the Linux system. Depending on Linux distributions, it's also known as **service**. In Linux community however, a Linux process is called **daemon**.
- When you start a program or running an application in Linux, you actually execute that program. A Linux process (a **daemon**), running in foreground or in the background, uses memory and CPU resources. That's why we need to manage Linux process. Keeping unused Linux process running in the system is a waste and also exposes your system to security threat.
- In Linux, every running process or daemon is given an identity number called **PID (Process ID)**. The process id is unique. We can terminate unused program in the system by stopping its process id.
- In order to manage Linux processes, we need to identify some process information such as who's responsible for the process, which terminal the process is running from and what command used to run the process.

There are generally three types of processes that run on Linux.

- **Interactive Processes**
- **System Process or Daemon**
- **Automatic or batch**

Interactive Processes

Interactive processes are those processes that are invoked by a user and can interact with the user. VI is an example of an interactive process. Interactive processes can be classified into foreground and background processes. The foreground process is the process that you are currently interacting with, and is using the terminal as its stdin (standard input) and stdout (standard output). A background process is not interacting with the user and can be in one of two states - paused or running.

System Process or Daemon

The second general type of process that runs on Linux is a **System Process or Daemon** (daemon). Daemon is the term used to refer to processes that are running on the computer and provide services but do not interact with the console. Most server software is implemented as a daemon. Apache, Samba, and inn are all examples of daemons.

Any process can become a daemon as long as it is run in the background, and does not interact with the user. A simple example of this can be achieved using the [ls -R] command. This will list all subdirectories on the computer, and is similar to the [dir /s] command on Windows. This command can be set to run in the background by typing [ls -R &], and although technically you have control over the shell prompt, you will be able to do little work as the screen displays the output of the process that you have running in the background. You will also notice that the standard pause (ctrl+z) and kill (ctrl+c) commands do little to help you.

Automatic Processes

Automatic processes are not connected to a terminal. Rather, these are tasks that can be queued into a spooler area, where they wait to be executed on a FIFO (first-in, first-out) basis. Such tasks can be executed using one of two criteria:

At certain date and time: done using the “at” command

At times when the total system load is low enough to accept extra jobs: done using the Cron command. By default, tasks are put in a queue where they wait to be executed until the system load is lower than 0.8. In large environments, the system administrator may prefer cron job processing when large amounts of data have to be processed or when tasks demanding a lot of system resources have to be executed on an already loaded system. Cron job processing is also used for optimizing system performance.

Parent and Child Process

- The Process which starts or creates another process is called **parent process** and the one which got created is known as **child process**.
- Every process will be having a parent process except **init/systemd** process.
- The **init/systemd** process is the parent of all the process in the system. It is the first process which gets started by the kernel at the time of booting
- The PID of init/systemd will be **1**.
- Only after init/systemd process gets started the remaining process are called by it, and hence it is responsible for all the remaining processes in the system.

LAB WORK:-

To monitor the process using ps command

- The ps command gives the running process of the present terminal and present command.
The syntax for ps command is

```
#ps
```

```
[root@ linux ~]# ps
 PID TTY      TIME CMD
 10951 pts/0    00:00:00 bash
 11523 pts/0    00:00:00 ps
[root@ linux ~]#
```

To see total number of processes running in the system

- The possible options which can be used with ps command are

```
#ps -a
```

```
[root@ linux ~]# ps -a
 PID TTY      TIME CMD
 11545 pts/0    00:00:00 ps
[root@ linux ~]#
```

To see the processes running by the logged in user (ex root)

- #ps -u <user name>

```
#ps -u musab
```

#ps -u (if no name is given it will show the processes of the logged in user)

```
[root@ linux ~]# ps -u musab
 PID TTY      TIME CMD
 11566 pts/1    00:00:00 bash
 11591 pts/1    00:00:00 vim
[root@ linux ~]#
```

```
[root@ linux ~]# ps
 PID TTY      TIME CMD
 10951 pts/0    00:00:00 bash
 11523 pts/0    00:00:00 ps
[root@ linux ~]#
```

To see which process are attached with some terminals (tty) and which are not

- #ps -x

```
[root@ linux ~]# ps -x
 PID TTY      STAT   TIME COMMAND
  1 ?        Ss     0:03 /sbin/init
  2 ?        S      0:00 [kthreadd]
  3 ?        S      0:00 [migration/0]
 2015 tty2   Ss+    0:00 /sbin/mingetty /dev/tty2
 2017 tty3   Ss+    0:00 /sbin/mingetty /dev/tty3
```

Note: The process which are showing "?" are not attached to any tty, mostly background processes

To see which process are running by a particular group

- #ps -G <group name> or #pgrep -G <group name>
- #ps -G musab or #pgrep -G musab

```
[root@ linux ~]# ps -G musab
  PID TTY      TIME CMD
11566 pts/1    00:00:00 bash
11591 pts/1    00:00:00 vim
[root@ linux ~]# pgrep -G musab
11566
11591
```

To see the auxiliary information of all the process, like cpu and memory consumptions

#ps -aux

```
[root@ linux ~]# ps -aux
USER      PID %CPU %MEM      VSZ   RSS TTY      STAT START   TIME COMMAND
root        1  0.0  0.3 126580  7420 ?        Ss  05:42  0:09 /usr/lib/systemd/systemctl --switched-root --system --deserialize 21
root        2  0.0  0.0      0     0 ?        S   05:42  0:00 [kthreadd]
root        3  0.0  0.0      0     0 ?        S   05:42  0:01 [ksoftirqd/0]
root        7  0.0  0.0      0     0 ?        S   05:42  0:00 [migration/0]
root        8  0.0  0.0      0     0 ?        S   05:42  0:00 [rcu_bh]
root        9  0.0  0.0      0     0 ?        S   05:42  0:00 [rcuob/0]
root       10  0.0  0.0      0     0 ?        S   05:42  0:00 [rcuob/1]
```

Signals in Linux

- Signals are a way of sending simple messages to processes. Most of these messages are already defined and can be found in <linux/signal.h>. However, signals can only be processed when the process is in user mode. If a signal has been sent to a process that is in kernel mode, it is dealt with immediately on returning to user mode.
- Every signal has a unique signal name, an abbreviation that begins with SIG (SIGINT for interrupt signal, for example). Each signal name is a macro which stands for a positive integer - the signal number for that kind of signal. Your programs should never make assumptions about the numeric code for a particular kind of signal, but rather refer to them always by the names defined. This is because the number for a given kind of signal can vary from system to system, but the meanings of the names are standardized and fairly uniform.
- Signals can be generated by the process itself, or they can be sent from one process to another. A variety of signals can be generated or delivered, and they have many uses for programmers. (To see a complete list of signals in the Linux® environment, uses the command kill -l.)

- There are total 64 signals in Linux, the list of all the signal can be seen by
#kill -l

```
[root@ linux ~]# kill -l
 1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL      5) SIGTRAP
 6) SIGABRT     7) SIGBUS      8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
[root@ktlinux ~]#
```

Few Important Signals with its descriptions:

Signal	Value	Action	Comment
SIGHUP	1	Term	Hangup detected on controlling terminal or death of controlling process
SIGINT	2	Term	Interrupt from keyboard
SIGQUIT	3	Core	Quit from keyboard
SIGILL	4	Core	Illegal Instruction
SIGABRT	6	Core	Abort signal from abort(3)
SIGFPE	8	Core	Floating point exception
SIGKILL	9	Term	Kill signal
SIGSEGV	11	Core	Invalid memory reference
SIGPIPE	13	Term	Broken pipe: write to pipe with no readers
SIGALRM	14	Term	Timer signal from alarm(2)
SIGTERM	15	Term	Termination signal
SIGUSR1	30,10,16	Term	User-defined signal 1
SIGUSR2	31,12,17	Term	User-defined signal 2
SIGCHLD	20,17,18	Ign	Child stopped or terminated
SIGCONT	19,18,25	Cont	Continue if stopped
SIGSTOP	17,19,23	Stop	Stop process
SIGTSTP	18,20,24	Stop	Stop typed at tty
SIGTTIN	21,21,26	Stop	tty input for background process
SIGTTOU	22,22,27	Stop	tty output for background process

The most common signals used are

- 1 for reloading the process
- 9 for killing the process
- 15 for Terminating the process
- 20 for stopping the process

To kill the process completely

- To kill the signal
- First find out the process running in the system, let's say by a user
`#ps -u <user name>`
`#ps -u musab`
`#kill <signal no> <process id>`
`#kill -9 11591`

```
[root@ linux ~]# ps -u musab
 PID TTY      TIME CMD
11566 pts/1    00:00:00 bash
[11591 pts/1]  00:00:00 vim
[root@ linux ~]# kill -9 11591
[root@ linux ~]# ps -u musab
 PID TTY      TIME CMD
11566 pts/1    00:00:00 bash
[root@ linux ~]# █
```

Likewise you can use other signals to kill the process like

```
#kill -15 <pid>
#kill -1 <pid>
```

To stop the process using a signal no. 20

- To stop a process first login as a normal user and start a process
`#su - musab`
`#cat > hello`

```
[musab@ linux ~]$ cat > hello
█
```

- Check its pid and kill it by using 20, `#ps -u musab`
`#kill -20`

```
[root@ linux ~]# ps -u musab
 PID TTY      TIME CMD
11566 pts/1    00:00:00 bash
[12041 pts/1]  00:00:00 cat
[root@ linux ~]# kill -20 12041
```

- check its effect at the user's console

```
[musab@ linux ~]$ cat > hello
[1]+  Stopped                  cat > hello
```

- Restart the process continue working

```
#fg <pid>
#fg 1
```

```
[musab@ linux ~]$ fg 1
cat > hello
Hi Maarij How are you █
```

Setting up the Priority of a Process

- When talking about processes priority is all about managing processor time. The Processor or CPU is like a human juggling multiple tasks at the same time. Sometimes we can have enough room to take on multiple projects. Sometimes we can only focus on one thing at a time. Other times something important pops up and we want to devote all of our energy into solving that problem while putting less important tasks on the back burner.
- In Linux we can set guidelines for the CPU to follow when it is looking at all the tasks it has to do. These guidelines are called ***niceness*** or ***nice value***. The Linux niceness scale goes from **-20 to 20**. **The lower the number the more priority that task gets. If the nice value is high number like 20 the task will be set to the lowest priority and the CPU will process it whenever it gets a chance. The default nice value is zero.**
- By using this scale we can allocate our CPU resources more appropriately. Lower priority programs that are not important can be set to a higher nice value, while high priority programs like daemons and services can be set to receive more of the CPU's focus. You can even give a specific user a lower nice value for all of his/her processes so you can limit their ability to slow down the computer's core services.
- There are two options to reduce/increase value of a process. You can either do it using the ***nice*** command or the ***renice*** command.

LAB WORK:-

To schedule a priority of a process before starting it

- To set a priority to a process before starting it, the syntax is
#nice -n <nice value range (-20 to 20)> <command>
#nice -n 5 cat > myfile

```
[root@ linux ~]# nice -n 5 cat > myfile
Hello World
Welcome to linux Technologies
```

- Log in to other terminal and check the nice value for the above command/ process.
#ps -elf

F S	UID	PID	PPID	C PRI	NI	ADDR	SZ	WCHAN	STIME	TTY	TIME	CMD
4 S	root	1	0	0 80	0 -	707	-	Oct14 ?		00:00:03	/sbin/init	
1 S	root	2	0	0 80	0 -	0	-	Oct14 ?		00:00:00	[kthreadd]	
1 S	root	13152	2	0 80	0 -	0	-	01:56 ?		00:00:00	[flush-253:0]	
0 S	root	13155	10951	0 85	5 -	1010	-	01:56 pts/0		00:00:00	cat	
1 S	root	13163	2	0 80	0 -	0	-	01:58 ?		00:00:00	[flush-253:3]	

To change the nice value of any process while it is running.

- To reschedule the nice value of existing process, first check the PID of that process by running **#ps -elf** command.
- As from previous task we know the PID of cat command i.e. **13155**
- Use the following command to renice the value of a cat command which is still running
#renice <nice value (-20 to 19)> <PID>
#renice 2 13155

```
[root@ linux ~]# renice 2 13155
13155: old priority 5, new priority 2
[root@ linux ~]#
```

Important monitoring commands

There are four critical resources of the system which is also known as 4 critical bottle neck

1. CPU
2. MEMORY
3. I/O (INPUT OUTPUT)
4. NETWORK

1. CPU: To monitor cpu there are following commands

#ps, sar, lscpu, /proc/cpuinfo
#sar (system activity report)

Linux 3.10.0-229.el7.x86_64 (localhost.localdomain) 10/03/2016 _x86_64_ (1 CPU)

07:22:12 AM LINUX RESTART

07:30:01 AM	CPU	%user	%nice	%system	%iowait	%steal	%idle
07:40:01 AM	all	0.07	0.00	0.04	0.00	0.00	99.90
07:50:01 AM	all	0.07	0.00	0.02	0.00	0.00	99.91

To see continuous output of sar

#sar 1 (1 sec is the time interval)

Linux 3.10.0-229.el7.x86_64 (kernel.kt.com) 10/03/2016 _x86_64_ (1 CPU)

01:56:04 PM	CPU	%user	%nice	%system	%iowait	%steal	%idle
01:56:05 PM	all	0.00	0.00	0.00	0.00	0.00	100.00
01:56:06 PM	all	0.00	0.00	0.00	0.00	0.00	100.00
01:56:07 PM	all	0.00	0.00	0.00	0.00	0.00	100.00
01:56:08 PM	all	0.00	0.00	0.00	0.00	0.00	100.00
01:56:09 PM	all	0.00	0.00	0.00	0.00	0.00	100.00
01:56:10 PM	all	0.00	0.00	0.00	0.00	0.00	100.00

To restrict the output for 3 counts for every second

#sar 1 3 (where 1 is time interval and 3 is counts)

Linux 3.10.0-229.el7.x86_64 (mlinux7.mb.com) 10/03/2016 _x86_64_ (1 CPU)

02:01:07 PM	CPU	%user	%nice	%system	%iowait	%steal	%idle
02:01:08 PM	all	0.00	0.00	0.00	0.00	0.00	100.00
02:01:09 PM	all	0.00	0.00	0.00	0.00	0.00	100.00
02:01:10 PM	all	0.00	0.00	0.00	0.00	0.00	100.00
Average:	all	0.00	0.00	0.00	0.00	0.00	100.00

To see the details of cpu

#lscpu

```
[root@ ~]# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                1
On-line CPU(s) list:  0
Thread(s) per core:   1
Core(s) per socket:   1
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 42
Model name:            Intel Xeon E312xx (Sandy Bridge)
Stepping:               1
CPU MHz:               3192.746
BogoMIPS:              6385.49
Hypervisor vendor:    KVM
Virtualization type:  full
L1d cache:             32K
L1i cache:             32K
L2 cache:              4096K
NUMA node0 CPU(s):    0
```

#cat /proc/cpuinfo

```
[root@ ~]# cat /proc/cpuinfo
processor      : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 42
model name    : Intel Xeon E312xx (Sandy Bridge)
stepping       : 1
microcode     : 0x1
cpu MHz       : 3192.746
cache size    : 4096 KB
physical id   : 0
siblings       : 1
core id       : 0
cpu cores     : 1
apicid         : 0
initial apicid: 0
fpu           : yes
fpu_exception : yes
cpuid level   : 13
```

2. Memory: To monitor memory following commands are used

#free, swapon -s , vmstat, /proc/meminfo

#vmstat

```
[root@ ~]# vmstat
procs -----memory----- swap-- io---- system-- cpu-----
 r b  swpd  free   buff  cache   si   so   bi   bo   in   cs us sy id wa st
 2 0    300 443576     32 270840    0    0   343   321   32   56  0  0 98  2  0
[root@ ~]#
```

To continuously see the output of vmstat for every 1 second

#vmstat 1

```
[root@ ~]# vmstat 1
procs -----memory----- ---swap-- -----io---- -system-- -----cpu-----
 r b swpd free buff cache si so bi bo in cs us sy id wa st
 2 0 300 443452 32 270872 0 0 341 319 32 56 0 0 98 2 0
 0 0 300 443436 32 270872 0 0 0 0 22 34 0 0 100 0 0
 0 0 300 443468 32 270872 0 0 0 0 17 32 0 0 100 0 0
```

To restrict the output of vmstat to 3 counts with time interval of 1 second

#vmstat 1 3 (where 1 is time interval and 3 is count)

```
[root@ ~]# vmstat 1 3
procs -----memory----- ---swap-- -----io---- -system-- -----cpu-----
 r b swpd free buff cache si so bi bo in cs us sy id wa st
 2 0 300 442624 32 271312 0 0 338 317 32 56 0 0 98 2 0
 0 0 300 442624 32 271312 0 0 0 0 9 12 0 0 100 0 0
 0 0 300 442600 32 271312 0 0 0 0 12 19 0 0 100 0 0
[root@ ~]#
```

#vmstat -s and #proc/meminfo

```
[root@node1 ~]# vmstat -s
 1882624 K total memory
 349340 K used memory
 348568 K active memory
 158388 K inactive memory
1073192 K free memory
 2116 K buffer memory
 457976 K swap cache
1257468 K total swap
 0 K used swap
1257468 K free swap
 254359 non-nice user cpu ticks
 215 nice user cpu ticks
 30618 system cpu ticks
 913874 idle cpu ticks
 4659 IO-wait cpu ticks
 0 IRQ cpu ticks
 3442 softirq cpu ticks
 0 stolen cpu ticks
304059 pages paged in
152803 pages paged out
 0 pages swapped in
 0 pages swapped out
3926937 interrupts
 2651686 CPU context switches
1547276963 boot time
 22138 forks
```

```
[root@ ~]#
[root@ ~]# cat /proc/meminfo
MemTotal: 1017216 kB
MemFree: 443008 kB
MemAvailable: 543564 kB
Buffers: 32 kB
Cached: 201244 kB
SwapCached: 300 kB
Active: 138320 kB
Inactive: 312216 kB
Active(anon): 16916 kB
Inactive(anon): 239268 kB
Active(file): 121404 kB
Inactive(file): 72948 kB
Unevictable: 0 kB
Mlocked: 0 kB
```

3. I/O: To Monitor I/O devices following commands are used **#fdisk, parted, df -h, iostat, lsblk, lsusb, lspci**

#iostat : to see the statistic of i/o devices

```
[root@ ~]# iostat
Linux 3.10.0-229.el7.x86_64 ( mlinux6.mb.com ) 10/03/2016 _x86_64_ (1 CPU)

avg-cpu: %user %nice %system %iowait %steal %idle
          0.23   0.00   0.24   1.84   0.00  97.68

Device:     tps kB_read/s kB_wrtn/s kB_read kB_wrtn
scd0        2.82    305.39      0.00  7800586      0
sda         2.35     28.08    392.75  717296 10031951
```

To continuously repeat the output of iostat every second and to restrict the same for 3 counts

#iostat 1, #iostat 1 3

```
[root@ ~]# iostat 1
Linux 3.10.0-229.el7.x86_64 ( mlinux7.mb.com ) 10/03/2016 _x86_64_ (1 CPU)

avg-cpu: %user %nice %system %iowait %steal %idle
          0.23   0.00   0.24   1.83   0.00  97.69

Device:     tps kB_read/s kB_wrtn/s kB_read kB_wrtn
scd0        2.81    304.19      0.00  7800586      0
sda         2.34     27.97    391.20  717304 10031981
dm-1        0.07     0.09      4.22   2217  108202
dm-0        0.04     0.07     22.19   1833  569002

avg-cpu: %user %nice %system %iowait %steal %idle
          0.00   0.00   0.00   0.00   0.00 100.00

Device:     tps kB_read/s kB_wrtn/s kB_read kB_wrtn
scd0        0.00     0.00      0.00      0      0
sda         0.00     0.00      0.00      0      0
dm-1        0.00     0.00      0.00      0      0
dm-0        0.00     0.00      0.00      0      0
```

To see a particular device statistics continuously

#iostat -d <dev name> 1

#iostat -d sda3 1

```
[root@ ~]# iostat -d sda3 1
Linux 3.10.0-229.el7.x86_64 ( mlinux7.mb.com ) 10/03/2016 _x86_64_ (1 CPU)

Device:     tps kB_read/s kB_wrtn/s kB_read kB_wrtn
sda3        0.01     0.20      0.02    5072    520

Device:     tps kB_read/s kB_wrtn/s kB_read kB_wrtn
sda3        0.00     0.00      0.00      0      0

Device:     tps kB_read/s kB_wrtn/s kB_read kB_wrtn
sda3        0.00     0.00      0.00      0      0
```

To restrict the same for 3 counts

#iostat -d <dev name> 1 3 (where 1 is time interval and 3 is counts)

#iostat -d sda3 1 3

```
[root@ ~]# iostat -d sda3 1 3
Linux 3.10.0-229.el7.x86_64 (mlinux7.mb.com) 10/03/2016 _x86_64_ (1 CPU)

Device:      tps   kB_read/s   kB_wrtn/s   kB_read   kB_wrtn
sda3        0.01     0.20       0.02     5072       520

Device:      tps   kB_read/s   kB_wrtn/s   kB_read   kB_wrtn
sda3        0.00     0.00       0.00       0          0

Device:      tps   kB_read/s   kB_wrtn/s   kB_read   kB_wrtn
sda3        0.00     0.00       0.00       0          0

[root@ ~]#
```

4. To monitoring network following commands

#ifconfig, ethtool, mii-tool, ping, netstat , route

To see the network configuration

#netstat

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp      0      0  mlinux7.mb.com:ssh      192.168.30.36:63426 ESTABLISHED
tcp      0      64 mlinux7.mb.com:ssh      192.168.30.36:63819 ESTABLISHED
tcp      0      0  mlinux7.mb.com:ssh      192.168.30.36:63279 ESTABLISHED

Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type      State         I-Node      Path
unix    2      [ ]     DGRAM          12914      @/org/freedesktop/systemd/notify
unix    5      [ ]     DGRAM          6609       /run/systemd/journal/socket
```

To see interface statistics

#netstat -i

```
[root@ ~]# netstat -i
Kernel Interface table
Iface      MTU     RX-OK RX-ERR RX-DRP RX-OVR     TX-OK TX-ERR TX-DRP TX-OVR Flg
ens3      1500    63082      0    171 0      42175      0      0      0 BMRU
lo        65536   2124       0      0 0      2124      0      0      0 LRU
[root@ ~]#
```

To see routing table

#netstat -rn

```
[root@ ~]# netstat -rn
Kernel IP routing table
Destination     Gateway            Genmask          Flags  MSS Window irtt Iface
0.0.0.0         192.168.106.1    0.0.0.0         UG        0 0          0 ens3
192.168.106.0  0.0.0.0          255.255.255.0   U        0 0          0 ens3
[root@ ~]#
```

Routing table can also be seen by

#route

```
[root@ ~]# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref Use Iface
default         192.168.106.1   0.0.0.0         UG    100      0      0 ens3
192.168.106.0  0.0.0.0        255.255.255.0  U     100      0      0 ens3
[root@ ~]#
```

Monitoring the process using top command

- When you need to see the running processes on your Linux in real time, you have top as your tool for that.
- top also displays other info besides the running processes, like free memory both physical and swap

Monitoring all process using top command

- To monitor all processes in the system use the following command

#top

```
top - 02:23:18 up 1 day, 13:57, 3 users, load average: 0.01, 0.00, 0.23
Tasks: 273 total, 1 running, 272 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.4%us, 0.5%sy, 0.0%ni, 98.8%id, 0.3%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 543948k total, 526204k used, 17744k free, 11748k buffers
Swap: 2097144k total, 49064k used, 2048080k free, 129928k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	2828	1192	1044	S	0.0	0.2	0:03.15	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.03	kthreadd
3	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
4	root	20	0	0	0	0	S	0.0	0.0	0:00.14	ksoftirqd/0
5	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
6	root	20	0	0	0	0	S	0.0	0.0	0:00.60	events/0
7	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuset
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khelper

The first line in top:

```
top - 02:23:18 up 1 day, 13:57, 3 users, load average: 0.01, 0.00, 0.23
```

- “**02:23:18**” is the current time; “**up 1 day**” shows how long the system has been up for; “**3 user**” how many users are logged in; “**load average: 0.01, 0.00, 0.23**” the load average of the system (1minute, 5 minutes, 15 minutes).

The second line in top:

```
Tasks: 273 total, 1 running, 272 sleeping, 0 stopped, 0 zombie
```

- Shows the number of processes and their current state.

The third line in top:

```
Cpu(s): 0.4%us, 0.5%sy, 0.0%ni, 98.8%id, 0.3%wa, 0.0%hi, 0.0%si, 0.0%st
```

- Shows CPU utilization details. “9.5%us” user processes are using 9.5%; “31.2%sy” system processes are using 31.2%; “27.0%id” percentage of available cpu; “7.6%wa” time CPU is waiting for IO.

The fourth and fifth lines in top:

Mem:	543948k total,	526204k used,	17744k free,	11748k buffers
Swap:	2097144k total,	49064k used,	2048080k free,	129928k cached

- “**543948k total**” is total memory in the system; “**526204K used**” is the part of the RAM that currently contains information; “**17744k free**” is the part of RAM that contains no information; “**17748K buffers and 129928k cached**” is the buffered and cached data for IO.

By default, top starts by showing the following task's property:

Field	Description
PID	Process ID
USER	Effective User ID
PR	Dynamic priority
NI	Nice value, also known as base priority
VIRT	Virtual Size of the task. This includes the size of process's executable binary, the data area and all the loaded shared libraries.
RES	The size of RAM currently consumed by the task. Swapped out portion of the task is not included.
SHR	Some memory areas could be shared between two or more task, this field reflects that shared areas. The example of shared area are shared library and SysV shared memory.
S	Task status
%CPU	The percentage of CPU time dedicated to run the task since the last top's screen update.
%MEM	The percentage of RAM currently consumed by the task.
TIME+	The total CPU time the task has been used since it started. "+" sign means it is displayed with hundredth of a second granularity. By default, TIME/TIME+ doesn't account the CPU time used by the task's dead children.
Command	Showing program names

Interacting with TOP

Now that we are able to understand the output from TOP lets learn how to change the way the output is displayed.

Use the following key while running top and the output will be sorted in real time.

- **h - help**
- **M – Sort by memory usage**
- **P – Sort by CPU usage**
- **T – Sort by cumulative time**
- **z – Color display**
- **L – locate/search a string**
- **r – to renice a process**
- **k – Kill a process**
- **q – quit**

To kill the process with PID 21, then press “k” and a prompt will ask you for the PID number, and enter 21. When asked about singal number give 9 or 15

```
top - 02:54:53 up 1 day, 14:29, 3 users, load average: 0.07, 0.02, 0.01
Tasks: 272 total, 1 running, 271 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.7%us, 0.3%sy, 0.0%ni, 99.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 543948k total, 522892k used, 21056k free, 12232k buffers
Swap: 2097144k total, 49688k used, 2047456k free, 126492k cached
```

PID to kill: 21

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2	root	20	0	0	0	0	S	0.0	0.0	0:00.03	[kthreadd]

```
top - 02:54:53 up 1 day, 14:29, 3 users, load average: 0.07, 0.02, 0.01
Tasks: 272 total, 1 running, 271 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.7%us, 0.3%sy, 0.0%ni, 99.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 543948k total, 522892k used, 21056k free, 12232k buffers
Swap: 2097144k total, 49688k used, 2047456k free, 126492k cached
```

Kill PID 21 with signal [15]: 9

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
-----	------	----	----	------	-----	-----	---	------	------	-------	---------

To renice a process with PID 4, then press “r” and a prompt will ask you for PID enter 4 and press enter. When prompted for renice value give any value .

```
Swap: 2097144k total, 49688k used, 2047456k free, 126496k cached
```

PID to renice: 4

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2	root	20	0	0	0	0	S	0.0	0.0	0:00.03	[kthreadd]
3	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	[migration/0]
4	root	20	0	0	0	0	S	0.0	0.0	0:00.15	[ksoftirqd/0]

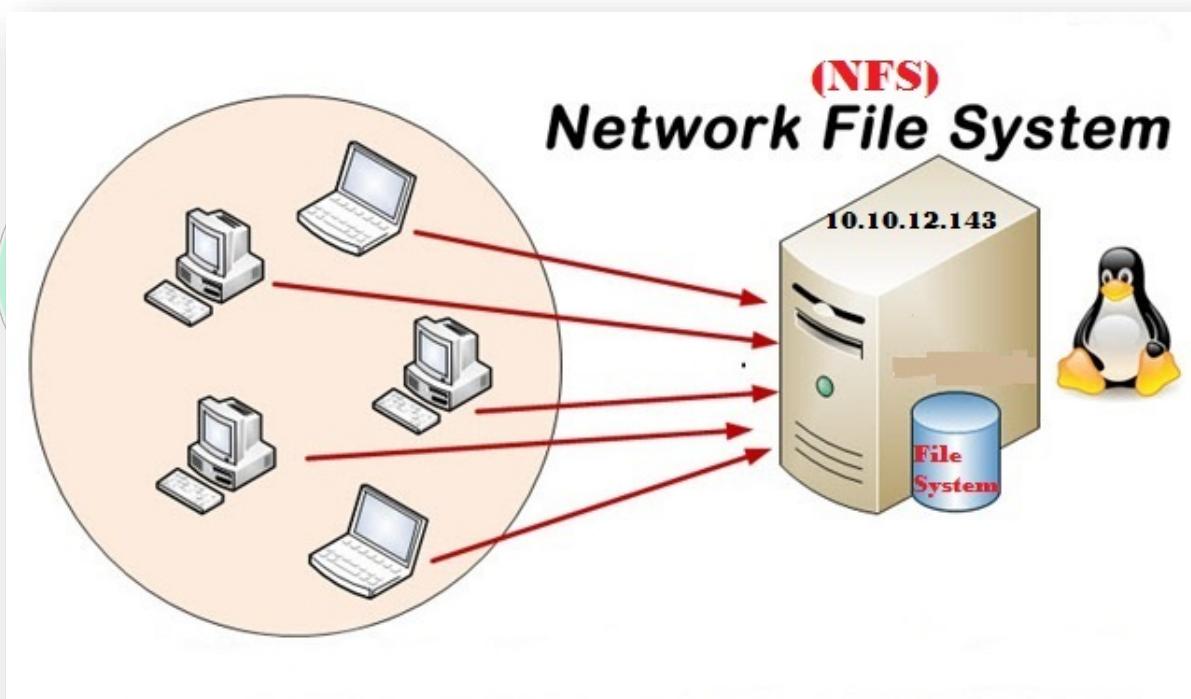
Renice PID 4 to value: -2

4	root	18	-2	0	0	0	S	0.0	0.0	0:00.15	[ksoftirqd/0]
---	------	----	----	---	---	---	---	-----	-----	---------	---------------

Find out more on top command from internet and keep practicing

NFS (NETWORK FILE SYSTEM/SHARING)

- NFS stands for Network File System, and is a way to share files between machines as if they were on your local hard drive. Linux can be both an NFS server and an NFS client, which means that it can export filesystems to other systems, and mount filesystems exported from other machines.
- For example **NFS server** could be a Linux system and UNIX could be a client. But it can't be a window system because windows is not NFS compatible. The NFS server exports one or more directories to the client systems, and the client systems mount one or more of the shared directories to local directories called mount points. After the share is mounted, all I/O operations are written back to the server, and all clients notice the change as if it occurred on the local filesystem.
- A manual refresh is not needed because the client accesses the remote filesystem as if it were local. Because access is granted by IP address, a username and password are not required. However, there are security risks to consider because the **NFS server** knows nothing about the users on the client system.



A Typical view of the NFS structure in Linux/Unix system

Profile for NFS:

- | | | |
|----------------------|---|---|
| • Package | : | nfs-utils |
| • Daemons | : | nfs-server.service
rpc.nfsd, rpc.mountd, rpc.statd, rpc.lockd, rpc.rquotad |
| • Port number | : | 2049 |
| • Configuration File | : | /etc/exports |
| • Other imp files | : | /var/lib/nfs/etab, /var/lib/nfs/rmtab |

Steps to configure NFS server:

Step1: check and if needed install the NFS package using yum or rpm.

Step2: Create a directory and add some data in it.

Step3: Export the directory by editing /etc/exports file and using exportfs command

Step4: Restart the services and make it permanent.

Step1: Install the NFS package.

- Check whether the package is installed

```
#rpm -q nfs-utils
```

```
[root@ cl1 ~]# rpm -q nfs-utils
nfs-utils-1.2.3-7.el6.x86_64
[root@ cl1 ~]#
```

- If it is not installed use following command to install it

```
#yum install nfs-utils* -y
```

Step2: Create a directory or create a partition and mount it and make a mount point and add data to it.

- Create a partition, format it and mount it, access the mount point and add data to it
#fdisk /dev/vda create a partition

/dev/vda1	3812	3927	931738+	8e	Linux LVM
/dev/vda2	3928	3979	417658+	8e	Linux LVM
/dev/vda13	3980	4056	618471	83	Linux

- Update the partition table and format it
#partx -a /dev/vda or #partprobe /dev/vda
#mkfs.ext4/xfs /dev/vda13
- Create a directory and mount the partition over it and also make it permanent in /etc/fstab

```
[root@ cl1 ~]# mkdir /mydir
[root@ cl1 ~]# vim /etc/fstab
[root@ cl1 ~]# mount -a
[root@ cl1 ~]# mount
/dev/mapper/vg_cl1-rootlv on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
/dev/vda2 on /boot type ext4 (rw)
/dev/mapper/vg_cl1-homelv on /home type ext4 (rw)
/dev/mapper/vg_cl1-optlv on /opt type ext4 (rw)
/dev/mapper/vg_cl1-usrlv on /usr type ext4 (rw)
/dev/mapper/vg_cl1-varlv on /var type ext4 (rw)
/dev/mapper/kiranvg-kiranlv on /kiran type ext4 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
gvfs-fuse-daemon on /root/.gvfs type fuse.gvfs-fuse-daemon (rw,nosuid,nodev)
nfsd on /proc/fs/nfsd type nfsd (rw)
/dev/vda13 on /mydir type ext4 (rw)
[root@ cl1 ~]#
```

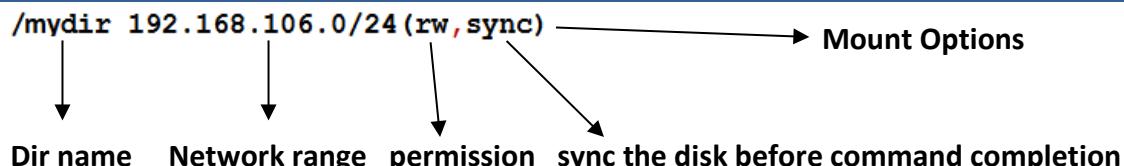
- Access the mount point and add some data in it

```
[root@ cl1 ~]# cd /mydir
[root@ cl1 mydir]# touch file{1..5}
[root@ cl1 mydir]# ls
file1  file2  file3  file4  file5  lost+found
[root@ktcl1 mydir]#
```

Step3: Export the directory by editing /etc/exports file and using exportfs command

- Edit the /etc/exports file

#vim /etc/exports



/mydir : Name of the directory to be exported
 192.168.106.0/24 : Range of network where directory can be mounted
 To give permission to only one node, just give the IP ADDR

(rw, sync)	:	Mount options
<u>The Mount options which can be used</u>		
rw	:	Sets read/write permissions
ro	:	Sets read-only permissions
sync	:	Specifies that all changes must be written to disk before a Command completes
no_wdelay	:	Forces the writing of changes immediately (useful for logs if Something crashes)
root_squash	:	Prevent root user's privilege

- Now run the exportfs command to export the directory

#exportfs -avr

```
[root@mlinux71 ~]# exportfs -rv
exporting 192.168.106.0/24:/mydir
```

Options:

- a Exports or un-exports all directories
- r Reexport all directories
- u Unexports one or more directories
- v Provides verbose output

Step4: Start the services and make it permanent.

- #systemctl start nfs-server.service
- #systemctl enable nfs-server.service

```
[root@mlinux71 ~]# systemctl start nfs-server.service
[root@mlinux71 ~]# systemctl enable nfs-server.service
ln -s '/usr/lib/systemd/system/nfs-server.service' '/etc/systemd/:.
er.target.wants/nfs-server.service'
[root@mlinux71 ~]#
```

Check the directories which is exported in /var/lib/nfs/etab and /var/lib/nfs/rmtab

```
[root@mlinux71 ~]# cat /var/lib/nfs/etab
/mydir 192.168.106.0/24(rw,sync,wdelay,hide,nocrossmnt,secure,1
e_check,secure_locks,acl,anonuid=65534,anongid=65534,sec=sys,rw,
[root@mlinux71 ~]# cat /var/lib/nfs/rmtab
[root@mlinux71 ~]#
```

- Note: Add the nfs in firewalld trusted services
- #firewall-cmd --add-service=nfs --permanently
- #firewall-cmd --reload

```
[root@mlinux71 ~]# firewall-cmd --add-service=nfs --permanent
success
[root@mlinux71 ~]# firewall-cmd --reload
success
[root@mlinux71 ~]# firewall-cmd --list-all
public (default, active)
  interfaces: bond0 ens3 ens8
  sources:
  services: dhcpcv6-client ftp nfs ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```



Client side configuration for NFS mounting

Step1: Check and Install the NFS package if not installed

Step2: Start the NFS services

Step3: Check which directory is exported for this machine using showmount command

Step4: Make a directory and mount the NFS dir over it.

Step5: Add some data to it and check the same is updated on server side.

Step1: Check and Install the package for NFS

#rpm -q nfs-utils

```
[root@ cl1 ~]# rpm -q nfs-utils
nfs-utils-1.2.3-7.el6.x86_64
[root@ cl1 ~]#
```

It will be already installed, if it is not installed use yum install nfs-utils* -y

Step2: check and start the NFS services and make it permanent.

- #systemctl start nfs-server.service
- #systemctl enable nfs-server.service

```
[root@mlinux71 ~]# systemctl start nfs-server.service
[root@mlinux71 ~]# systemctl enable nfs-server.service
ln -s '/usr/lib/systemd/system/nfs-server.service' '/etc/systemd/:.
er.target.wants/nfs-server.service'
[root@mlinux71 ~]#
```

Step3: Check which directory is exported for this machine using showmount command

- To check the exported directories from server the syntax is
#showmount -e <server ip address>

```
[root@mlinux72 ~]# showmount -e 192.168.106.81
clnt_create: RPC: Port mapper failure - Unable to receive: errno 113 (No route
o host)
[root@mlinux72 ~]#
```

Note: At first it may show such error, due to firewall blocking some important services on server side. To resolve it login to server and allow following services in firewall.

- #firewall-cmd --add-service=rpc-bind --permanent
- #firewall-cmd --add-service=mountd --permanent
- #firewall-cmd --reload

```
[root@mlinux71 ~]# firewall-cmd --add-service=rpc-bind --permanent
success
[root@mlinux71 ~]# firewall-cmd --add-service=mountd --permanent
success
[root@mlinux71 ~]# firewall-cmd --reload
success
[root@mlinux71 ~]# firewall-cmd --list-all
public (default, active)
  interfaces: bond0 ens3 ens8
  sources:
  services: dhcpcv6-client ftp mountd nfs rpc-bind ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

Now try showmount -e , it would be working fine

```
[root@mlinux72 ~]# showmount -e 192.168.106.81
Export list for 192.168.106.81:
/mydir 192.168.106.0/24
[root@mlinux72 ~]#
```

Step4: Make a directory and mount NFS over it.

- #mkdir /nfscl
- #mount -t nfs 192.168.106.81:/mydir /nfscl

```
[root@mlinux72 ~]# mkdir /nfscl
[root@mlinux72 ~]# mount -t nfs 192.168.106.81:/mydir /nfscl
[root@mlinux72 ~]# mount |grep /nfscl
192.168.106.81:/mydir on /nfscl type nfs4 (rw,relatime,vers=4.0
ze=262144,namlen=255,hard,proto=tcp,port=0,timeo=600,retrans=2,:
r=192.168.106.82,local_lock=none,addr=192.168.106.81)
[root@mlinux72 ~]# cd /nfscl
[root@mlinux72 nfscl]# ls
nfs1 nfs2 nfs3 nfs4 nfs5
[root@mlinux72 nfscl]#
```

- To make it permanent mount edit /etc/fstab file as follows

UUID=34ef8764-c31e-40e1-ba46-9d0805aa8e0e	swap	swap
ts	0 0	
192.168.106.81:/mydir	/nfscl	nfs defaults 0 0
~		

Auto-mounting the NFS directory

- All the resources of the server is valuable and needs to be available for usage, when we mount a NFS directory over client the network resource gets busy, even when the work is finished the network resource will still be busy as mounting occupy it.
- Autofs automatically mounts file systems for you when they are requested. This has a very handy feature: It's great for handling removable media. Just CD to the right directory, or execute ls or do anything that sends a request to the mount point: and the daemon mounts it. After all, it's the kind of job that's beneath the dignity of a human being First; you need to install the "autofs" package. It should include some appropriate config files. The files you need is /etc/auto.master
- There are two types of Auto-mounting
 - Direct and Indirect Auto-mounting

1. Direct Auto-Mounting:

In direct mounting for each partner server a mount point (dir) needs to be created. For example if there are 10 nfs share to be mounted at client side, there must be 10 directories created and managed manually.

2. Indirect Auto-Mounting:

In this type of mounting for all NFS server shares, only one mother directory needs to be created and for each server a sub-directory will be automatically created and used for mounting.

Note: Before going for Auto-mounting remove all kind of mounting done previously

Steps to configure Indirect auto-mount at client side

Step1: Log into client side and check whether autofs is install or not, if not install autofs

- Check whether autofs is install or not

```
#rpm -q autofs
```

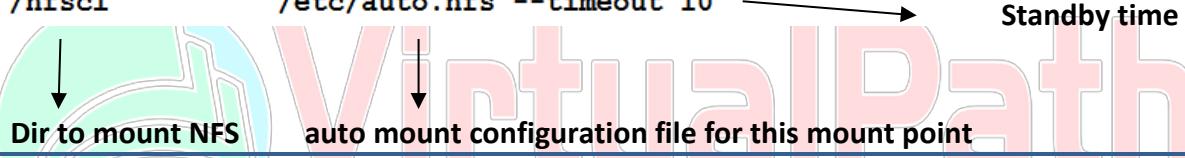
```
[root@ cl1 /]# rpm -q autofs
autofs-5.0.5-31.el6.x86_64
[root@ cl1 /]#
```

- if it is not installed, install it by using yum or rpm
`#yum install autofs* -y`

Step2: Edit the /etc/auto.master as follows

```
#vim /etc/auto.master
```

```
# Note that if there are entries for /net or /misc (as
# above) in the included master map any keys that are the
# same will not be seen as the first read key seen takes
# precedence.
#
+auto.master
/nfscl          /etc/auto.nfs --timeout 10
```



Step3: Create /etc/auto.nfs file and /nfscl directory if not created earlier

- `#vim /etc/auto.nfs`

81	-rw	192.168.106.81:/mydir
Name to be used for sub-dir	Permissions	NFS server and directory name

Step4: Start/Restart the autofs service and make it permanent

- `#systemctl start/restart autofs`
- `#systemctl enable autofs`

```
[root@mlinux72 ~]# systemctl start autofs.service
[root@mlinux72 ~]# systemctl enable autofs.service
ln -s '/usr/lib/systemd/system/autofs.service' '/etc/systemd/system/
[root@mlinux72 ~]#
```

Step5: log into the given directory given in /etc/auto.master i.e. /nfscl and check that if NFS is mounted by mount command

```
[root@mlinux72 ~]# cd /nfscl
[root@mlinux72 nfscl]# ls
[root@mlinux72 nfscl]#
```

Note: Still NFS dir will not be mounted on the client side

Step6: change the directory to the name given in /etc/auto.nfs i.e. 81 and then auto mounting will be done.

```
#cd 81
```

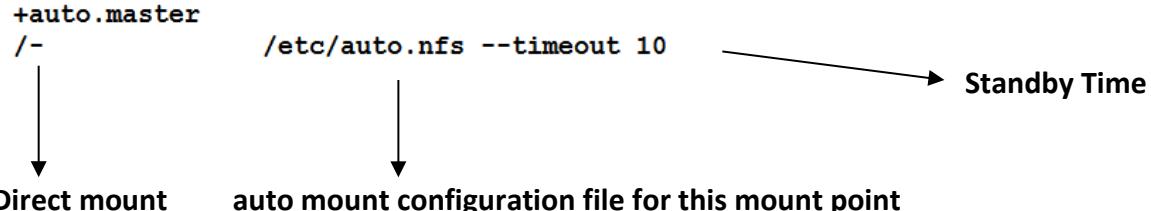
```
#ls
```

```
[root@mlinux72 nfscl]# cd 81
[root@mlinux72 81]# ls
nfs1  nfs2  nfs3  nfs4  nfs5
```

Steps to configure Direct auto-mount at client side

Step1: Edit the auto.master file as follows

```
# Note that if there are entries for /net or /misc (as
# above) in the included master map any keys that are the
# same will not be seen as the first read key seen takes
# precedence.
#
+auto.master
/-           /etc/auto.nfs --timeout 10
```



Step2: Edit /etc/auto.nfs file and /nfscl directory if not created earlier

- #vim /etc/auto.nfs

```
/nfscl  -rw    192.168.106.81:/mydir
```



Step3: Reload the autofs services

- #systemctl reload autofs

```
[root@node1 ~]# systemctl reload autofs
```

Step5: log into the given directory given in /etc/auto.master i.e. /nfscl and check that if NFS is mounted by mount command

```
[root@mlinux72 ~]# cd /nfscl
[root@mlinux72 nfscl]# ls
nfs1  nfs2  nfs3  nfs4  nfs5
[root@mlinux72 nfscl]#
```

Steps for removing NFS

Step1: Remove all autofs details from all configuration files like /etc/auto.master and /etc/auto.nfs

Step2: Delete the entries of the NFS share you want to remove, from /etc/exports on the server and un-export all the directory which was exported earlier using following command

- # exportfs -auv

```
[root@ cl1 ~]# cat /var/lib/nfs/etab
/mydir 192.168.10.0/24(rw,sync,wdelay,hide,nocrossmnt,secure,root_squash,no_all_squash,no_subtree_check,secure_locks,acl,ano
nuid=65534,anongid=65534)
[root@ cl1 ~]# exportfs -auv
[root@ cl1 ~]# cat /var/lib/nfs/etab
[root@ cl1 ~]#
```

Note: - if you don't have DNS and still want to use hostname instead of IP, update hostname with its ip in /etc/hosts file and then you can use hostname instead of IP

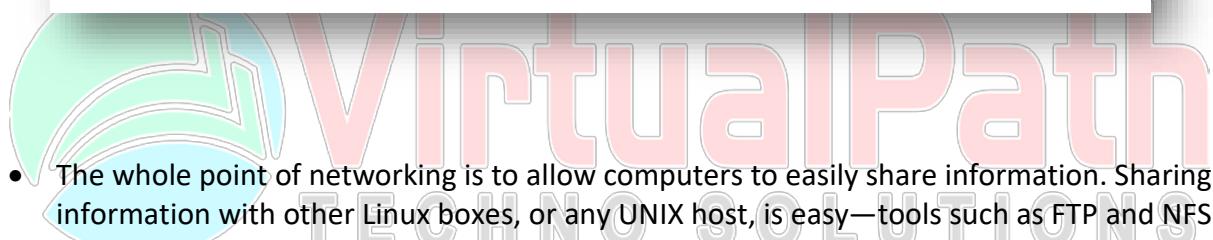
To check ports and protocol on nfs server from client side, the following commands can be used

#rpcinfo -p <server ip>

```
[root@mlinux4 ~]# rpcinfo -p 192.168.10.30
   program  vers  proto   port  service
    100000    4    tcp    111  portmapper
    100000    3    tcp    111  portmapper
    100000    2    tcp    111  portmapper
    100000    4    udp    111  portmapper
    100000    3    udp    111  portmapper
    100000    2    udp    111  portmapper
    100005    1    udp   20048  mountd
    100005    1    tcp   20048  mountd
    100005    2    udp   20048  mountd
    100024    1    udp   36469  status
    100005    2    tcp   20048  mountd
    100024    1    tcp   43027  status
    100005    3    udp   20048  mountd
    100005    3    tcp   20048  mountd
    100003    3    tcp   2049   nfs
    100003    4    tcp   2049   nfs
    100227    3    tcp   2049   nfs_acl
```

Finally we are done with all the NFS practical. Do hands on practice on it, as it is important in real world

SAMBA SERVER



- The whole point of networking is to allow computers to easily share information. Sharing information with other Linux boxes, or any UNIX host, is easy—tools such as FTP and NFS are readily available and frequently set up easily “out of the box”. Unfortunately, even the most die-hard Linux fanatic has to admit the operating system most of the PCs in the world are running is one of the various types of Windows. Unless you use your Linux box in a particularly isolated environment, you will almost certainly need to exchange information with machines running Windows. Assuming you're not planning on moving all of your files using floppy disks, the tool you need is Samba.
- Samba is an implementation of a Common Internet File System (CIFS, also known as SMB) protocol server that can be run on almost every variant of Unix in existence. Microsoft clients will use this protocol to access files and printers located on your Unix box just as if it were a native Windows server.
- **Samba** allows **linux** computers to share files and printers across a network connection. By using its SMB protocol, your **linux** box can appear **in** Windows Network Neighborhood or My Network Places just like any other windows machine. You can share files this way, as well as printers. By using **samba** on my home network, for example, my Windows machines have access to a printer directly hooked up to my **Linux** box, and my **Linux** box has access to a printer directly hooked up to one of my Windows machines. **In** addition, everyone can access everyone else's shared files. You can see how **samba** can be very useful if you have a network of both Windows as well as **Linux** machines.

Profile for SAMBA:

Usage	:	used for sharing files and directories in the network Between different platforms, like Linux-windows
Package	:	SAMBA, SAMBA-common, SAMBA-client.
Daemons	:	smbd, nmbd
Port no	:	137 (net bios –ns{name service}), 138 (net bios–dgm {Datagram}), 139 (net bios-ssn {session service}), 445 (Microsoft –ds {dist sys})
File system	:	CIFS (common internet file system)
Config file	:	/etc/samba/smb.conf
Sample file	:	/etc/samba/smb.conf.example

Steps to configure SAMBA server

Step1: Check and Install the SAMBA package, if not installed

#rpm -q samba

```
[root@ cl1 ~]# rpm -q samba
package samba is not installed
[root@ cl1 ~]#
```

- **Install the package using yum**

#yum install samba* -y

```
[root@ cl1 ~]# yum install samba* -y
Loaded plugins: product-id, refresh-packagekit, subscription-manager
Updating Red Hat repositories.
Setting up Install Process
Package samba-winbind-clients-3.5.6-86.el6.x86_64 already installed and latest version
Package samba-common-3.5.6-86.el6.x86_64 already installed and latest version
Package samba-client-3.5.6-86.el6.x86_64 already installed and latest version
Resolving Dependencies
--> Running transaction check
--> Package samba.x86_64 0:3.5.6-86.el6 will be installed
--> Package samba-winbind.x86_64 0:3.5.6-86.el6 will be installed
--> Finished Dependency Resolution
```

Dependencies Resolved

```
=====
Package           Arch      Version       Repository
=====
Installing:
samba            x86_64   3.5.6-86.el6  RHEL 7
samba-winbind    x86_64   3.5.6-86.el6  RHEL 7
Transaction Summary
=====
Install      2 Package(s)
Installed:
  samba.x86_64 0:3.5.6-86.el6
                           samba-winbind.x86_64 0:3.5.6-86.el6
Complete!
[root@ktcl1 ~]#
```

Step2: Make a directory and assign full permission to it, which will be shared

- #mkdir /samba
- #chmod 777 /samba

```
[root@ cl1 ~]# mkdir /samba
[root@ cl1 ~]# chmod 777 /samba
[root@ cl1 ~]#
```

Step3: Check the context of the directory and change it according to samba

- #ls -ldZ /samba
- #chcon -t samba_share_t /samba

```
[root@ cl1 ~]# ls -ldZ /samba/
drwxrwxrwx. root root unconfined_u:object_r:default_t:s0 /samba/
[root@ cl1 ~]# chcon -t samba_share_t /samba/
[root@ cl1 ~]# ls -ldZ /samba/
drwxrwxrwx. root root unconfined_u:object_r:samba_share_t:s0 /samba/
[root@ cl1 ~]#
```

Step4: Create a user or use any existing user who will be allowed to log in as samba user, add that user to samba user

- As we have a existing user "myuser", let's just make it samba user

#smbpasswd -a <username>

#smbpasswd -a myuser

Give password twice and wait till it add the user

```
[root@ cl1 ~]# smbpasswd -a myuser
New SMB password:
Retype new SMB password:
Added user myuser.
[root@ cl1 ~]#
```

Note: To delete a user from samba use #smbpasswd -x <user name>

- To check all the samba user use

#pdbeedit -L

```
[root@ cl1 ~]# pdbeedit -L
myuser:515:
[root@ cl1 ~]#
```

Step5: Go to the sample configuration file i.e. /etc/samba/smb.conf.example and copy the last paragraph as shown below

- Open the /etc/samba/smb.conf and paste the copied paragraph shown below and edit it.

```
# A publicly accessible directory, but read only, except for people in
# the "staff" group
[public]
comment = Public Stuff
path = /home/samba
public = yes
writable = yes
printable = no
write list = +staff
```

- Once pasted remove ";" mark before it and change it according to following picture

```
#####
[myshare]
comment = Public Stuff
path = /samba
public = no
valid users = myuser
writable = yes
printable = no
hosts allow = 192.168.
```

Explanation about the above fields

- [myshare] : Share Name
- Comment = Public Stuff : Comment
- Path = /samba : Share Directory
- Public = no : Public Access (Every user in network)
- Valid user = myuser : Authorized user
- Writable = yes : Write Permission
- Printable = no : Print permission
- Host allow= 192.168. : Network Range or host range

Note: Use 192.168.10. To allow only 192.168.10 network, in our case we have allowed any machine in 192.168. Network

Step5: Test the samba parameters and restart the service and make it enable after reboot

- To test the parameters us the following command

```
#testparm
```

```
[root@ cl2 /]# testparm
Load smb config files from /etc/samba/smb.conf
rlimit_max: rlimit_max (1024) below minimum Windows limit (16384)
Processing section "[homes]"
Processing section "[printers]"
Processing section "[myshare]"
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions
[

[printers]
comment = All Printers
path = /var/spool/samba
printable = Yes
browseable = No

[myshare]
comment = Public Stuff
path = /samba
valid users = myuser
read only = No
hosts allow = 192.168.

[root@ cl2 /]#
```

- Start the services and make it permanent by enabling it.

```
#systemctl start smb; systemctl enable smb
#systemctl start nmb; systemctl enable nmb
```

```
[root@mlinux71 ~]# systemctl start smb; systemctl enable smb
[root@mlinux71 ~]# systemctl start nmb; systemctl enable nmb
[root@mlinux71 ~]#
```

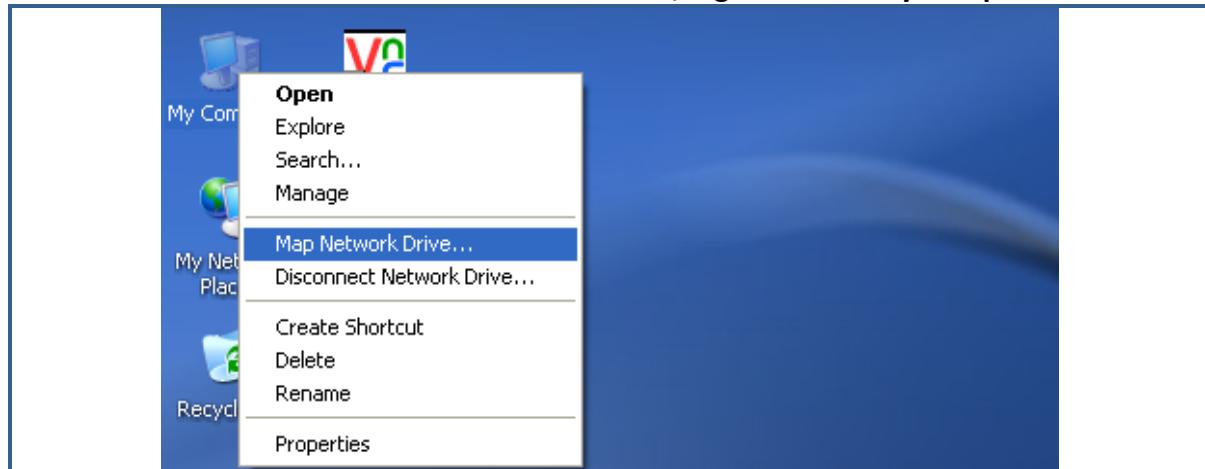
Step6: Add samba service in firewall in RHEL7

```
[root@mlinux71 ~]# firewall-cmd --add-service=samba --permanent
success
[root@mlinux71 ~]# firewall-cmd --reload
success
[root@mlinux71 ~]# firewall-cmd --list-all
public (default)
interfaces:
sources:
services: dhcpcv6-client ftp mountd nfs rpc-bind samba ssh
ports:
masquerade: no
forward-ports:
icmp-blocks:
rich rules:
```

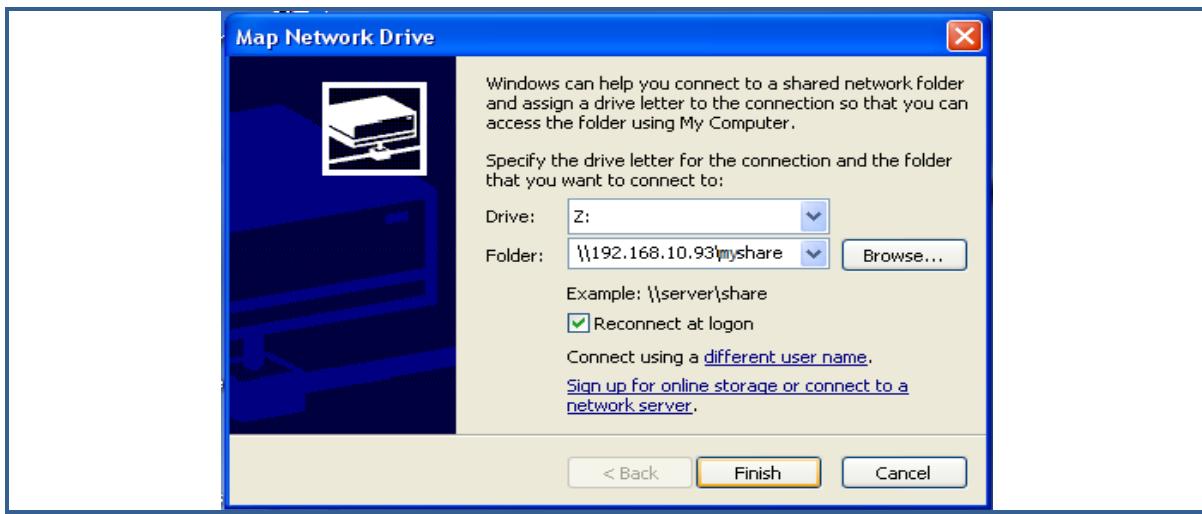


Windows as a samba client:

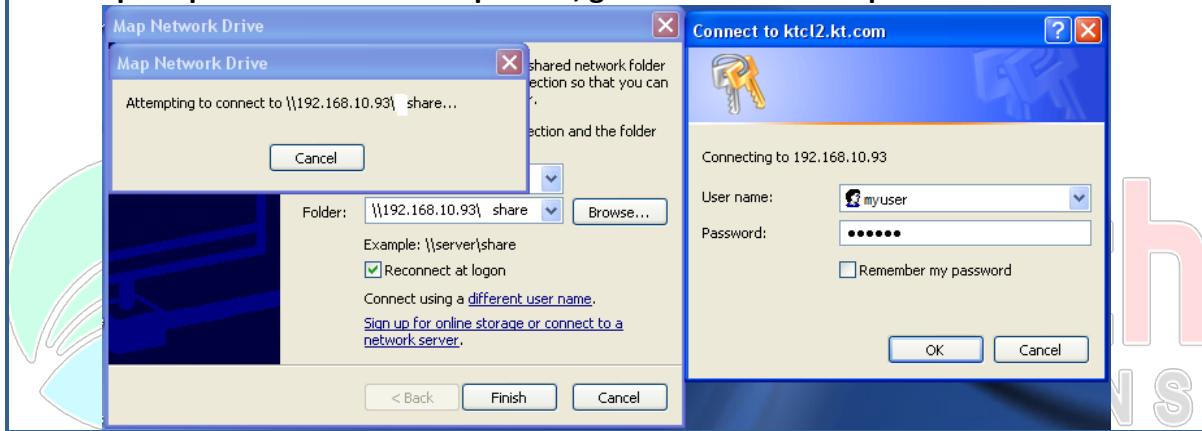
To connect from windows to the samba server, Right click on My Computer icon select



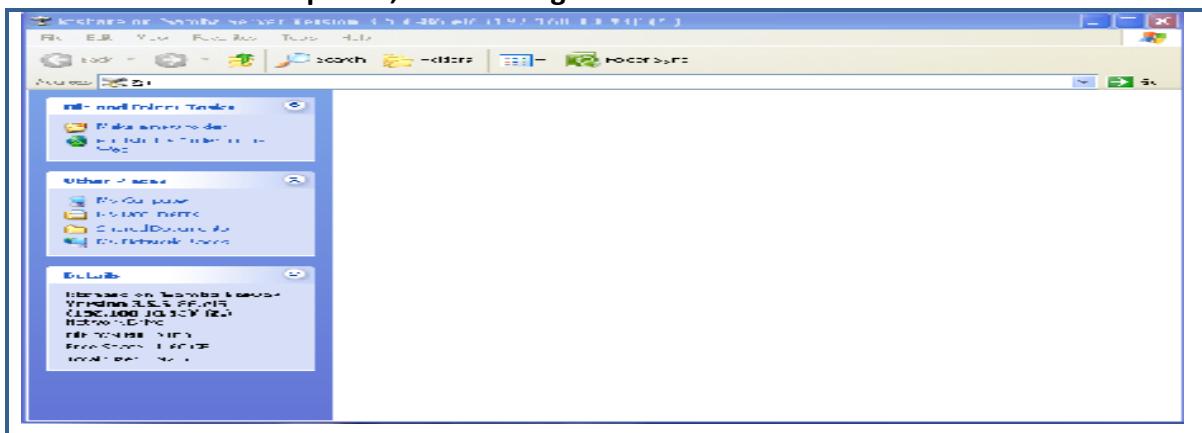
- Give the address of samba server as “\\192.168.10.93\ktshare(sharename), press on finish to continue.

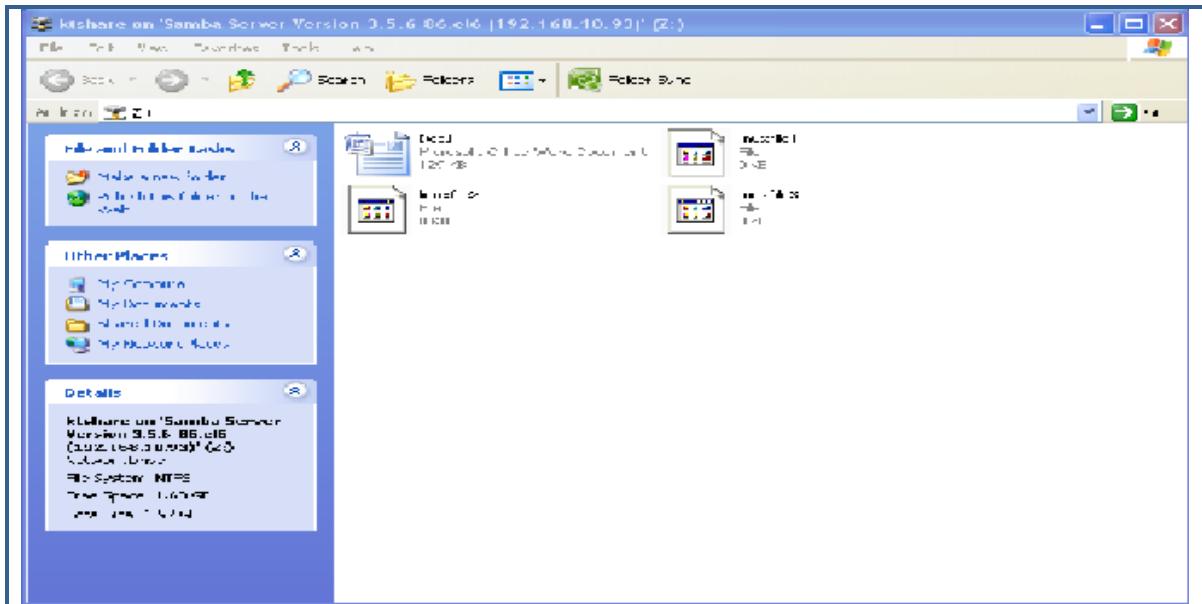


It will prompt for user name and passwd, give samba user and passwd and click on OK



- A window will be opened, start adding some data to it from both sides





Linux as a SAMBA Client

- Log into Linux machine and check how many samba servers are there in your network
#findsmb

```
[root@ cl5 ~]# findsmb

*=DMB
+=LMB
IP ADDR      NETBIOS NAME      WORKGROUP/OS/VERSION
-----
192.168.10.83  CL3          +[CL] [Windows Server 2003 R2 3790 Service Pack 2]
[Windows Server 2003 R2 5.2]
192.168.10.93  CL2          [MYGROUP] [Unix] [Samba 3.5.6-86.el6]
192.168.10.98  LINUX         +[WORKGROUP] [Windows Server 2003 R2 3790 Service]
```

- Check the share name of that samba server by using following command

#smbclient -L //192.168.10.93

when prompted for passwd just press enter without giving any passwd

```
[root@ cl5 ~]# smbclient -L //192.168.10.93
Enter root's password:
Anonymous login successful
Domain=[MYGROUP] OS=[Unix] Server=[Samba 3.5.6-86.el6]

        Sharename      Type      Comment
        -----
        IPC$          IPC       IPC Service (Samba Server Version 3.5.6-86.el6
)
        myshare        Disk      Public Stuff
Anonymous login successful
Domain=[MYGROUP] OS=[Unix] Server=[Samba 3.5.6-86.el6]

        Server          Comment
        -----
        CL2            Samba Server Version 3.5.6-86.el6
        LINUX          Samba Server Version 3.5.4-68.el6

        Workgroup      Master
        -----
        CL             CL3
        TS             ADS
        MYGROUP        LINUX
        WORKGROUP     LINUX
```

- To connect to the samba server use the following syntax

```
#smbclient //<server IP>/<share name> -U <User name>
#smbclient //192.168.10.93/myshare -U myuser
```

```
[root@ cl5 ~]# smbclient //192.168.10.93/myshare -U myuser
Enter myuser's password:
Domain=[MYGROUP] OS=[Unix] Server=[Samba 3.5.6-86.el6]
smb: \> ls
.
..
linuxfile3
Doc1.docx
linuxfile1
linuxfile2

D          0  Wed Nov  9 20:40:18 2011
DR         0  Wed Nov  9 20:00:43 2011
          0  Wed Nov  9 20:40:18 2011
A   131602 Sat Oct  1 15:26:29 2011
          0  Wed Nov  9 20:40:18 2011
          0  Wed Nov  9 20:40:18 2011

62994 blocks of size 32768. 52515 blocks available
smb: \> ■
```

- To mount the SAMBA directory on remote Linux client
- Make sure *cifs-utils* package is installed, if not install it first before using *mount* command.
- The syntax for mounting samba share is as follows

```
#mount -t <type of fs> //<server IP address>/<share name> /<mount point> -o user=<user name>.
#mount -t cifs //192.168.10.93/myshare /mnt -o user=myuser
```

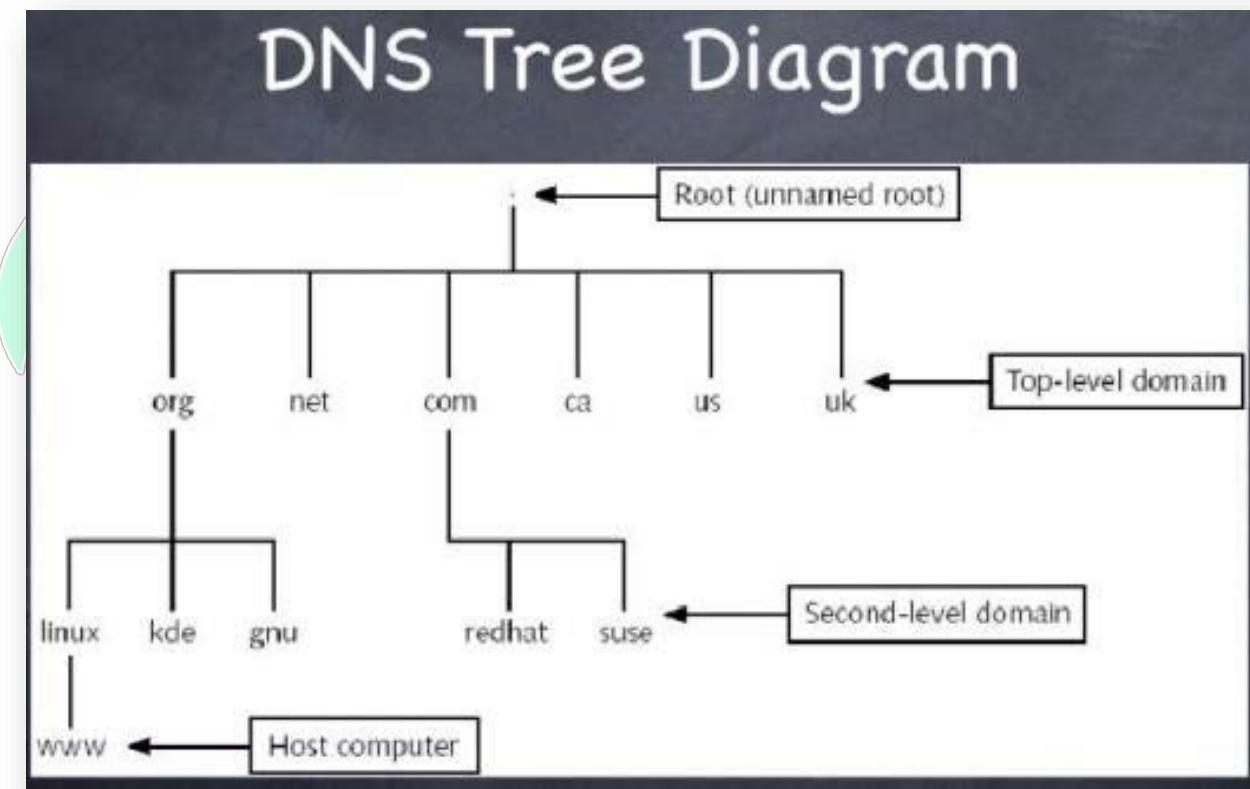
```
[root@ cl5 ~]# mount -t cifs //192.168.10.93/myshare /mnt -o user=myuser
Password:
[root@ cl5 ~]# mount
/dev/mapper/rootvg_ cl5-LogVol00 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw,rootcontext="system_u:object_r:tmpfs_t:s0")
/dev/sda2 on /boot type ext4 (rw)
/dev/mapper/rootvg_ cl5-LogVol01 on /home type ext4 (rw)
/dev/mapper/rootvg_ cl5-LogVol04 on /opt type ext4 (rw)
/dev/mapper/rootvg_ cl5-LogVol05 on /tmp type ext4 (rw)
/dev/mapper/rootvg_ cl5-LogVol02 on /usr type ext4 (rw)
/dev/mapper/rootvg_ cl5-LogVol03 on /var type ext4 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
gvfs-fuse-daemon on /root/.gvfs type fuse.gvfs-fuse-daemon (rw,nosuid,nodev)
//192.168.10.93/myshare/ on /mnt type cifs (rw,mand)
[root@ cl5 ~]# ■
```

Note: To learn how to make samba permanent mount and auto-mount visit my website <http://musab.in/2014/02/making-samba-permanent-and-auto-mount-in-linux/>

DNS (Domain Name System) SERVER

Domain Name System

The Domain Name System (DNS) is the crucial glue that keeps computer networks in harmony by converting human-friendly hostnames to the numerical IP addresses computers require to communicate with each other. DNS is one of the largest and most important distributed databases the world depends on by serving billions of DNS requests daily for public IP addresses. Most public DNS servers today are run by larger ISPs and commercial companies but private DNS servers can also be useful for private home networks.



Like the telephone system, every device attached to the Internet has a unique number, its IP address. Also like the telephone system there is a directory services to help you find those numbers called DNS.

If you have someone's name and address you can call a directory services, give them the details you know and they will (usually) give you the telephone number to call them. Likewise, if you know a server's host name (maybe <http://www.google.co.in/>) you can give that name to a DNS server and it will give you the IP address of that server.

The format of a domain name

Like a physical address, Internet domain names are hierarchical (only a little stricter), so while your address might look like:

House name:	TM Residency
	Ameerpet
Town:	Hyderabad
County:	Telangana
Country:	India

An Internet domain name looks like:

Host name	www
Domain	google
Second level domain	co
Top-level domain	In

A database is made up of records and the DNS is a database. Therefore, common resource record types in the DNS database are:

- **A** - Host's IP address. Address record allowing a computer name to be translated into an IP address. Each computer must have this record for its IP address to be located. These names are not assigned for clients that have dynamically assigned IP addresses, but are a must for locating servers with static IP addresses.
- **PTR** - Host's domain name, host identified by its IP address
- **CNAME** - Host's canonical name allows additional names or aliases to be used to locate a computer.
- **MX** - Host's or domain's mail exchanger.
- **NS** - Host's or domain's name server(s).
- **SOA** - Indicates authority for the domain (Start of Authority)
- **TXT** - Generic text record
- **SRV** - Service location record
- **RP** - Responsible person
- **HINFO** - Host information record with CPU type and operating system

The package which is used in Linux for performing DNS activity is BIND (Berkeley Internet Name Domain)

Profile for DNS Server

Usage	:	To Resolve IP into hostname and vice-versa
Package	:	bind, caching-name
Daemon	:	named
Port	:	53
Configuration File	:	/etc/named.conf, /etc/named.rfc1912.zones
Document root	:	/var/named/

Step by Step configuration of DNS server

Step1: Check and Install the package for DNS

- The package which is to be installed for DNS is bind and caching
#rpm -q bind

```
[root@ adm ~]# rpm -q bind
package bind is not installed
[root@ adm ~]#
```

- To install the package use yum or rpm command

```
[root@ adm ~]# yum install bind* caching* -y
Loaded plugins: product-id, refresh-packagekit, subscription-manager
Updating Red Hat repositories.
Setting up Install Process
Package 32:bind-utils-9.7.3-2.el6.x86_64 already installed and latest version
Package 32:bind-libs-9.7.3-2.el6.x86_64 already installed and latest version
Resolving Dependencies
--> Running transaction check
--> Package bind.x86_64 32:9.7.3-2.el6 will be installed
--> Package bind-chroot.x86_64 32:9.7.3-2.el6 will be installed
--> Package bind-dyndb-ldap.x86_64 0:0.2.0-1.el6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch    Version        Repository      Size
=====
Installing:
bind              x86_64  32:9.7.3-2.el6   RHEL6          3.9 M
bind-chroot       x86_64  32:9.7.3-2.el6   RHEL6          67 k
bind-dyndb-ldap   x86_64  0.2.0-1.el6    RHEL6          49 k

Installed:
bind.x86_64 32:9.7.3-2.el6                  bind-chroot.x86_64 32:9.7.3-2.el6
bind-dyndb-ldap.x86_64 0:0.2.0-1.el6

Complete!
```

Step2: Update the /etc/hosts file with the server's ip address, and change the hostname with fully qualified domain name.

- Change the hostname by adding fully qualified domain name

```
#hostnamectl set-hostname mlinux3.vpts.com
```

```
[root@mlinux3 ~]# hostnamectl set-hostname mlinux3.vpts.com
[root@mlinux3 ~]# hostname
mlinux3.vpts.com
```

Note:- change the hostname on all clients by making it FQDN

- Update /etc/hosts on DNS server with hostname and IP address

```
#vim /etc/hosts
```

```
[root@mlinux3 named]# more /etc/hosts
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1          localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.10.30  mlinux3.vpts.com
[root@mlinux3 named]#
```

Step3: Edit the configuration file “/etc/named.conf

- Edit the /etc/named.conf file with our name server's IP address and network range for clients.

```
#vim /etc/named.conf
```

```
options {
    listen-on port 53 { 127.0.0.1; };
    listen-on-v6 port 53 { ::1; };
    directory      "/var/named";
    dump-file      "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    allow-query     { localhost; };
    recursion yes;
```

Note: Need to add our systems details in highlighted lines

```
options {
    listen-on port 53 { 127.0.0.1; 192.168.10.30; };
    listen-on-v6 port 53 { ::1; };
    directory      "/var/named";
    dump-file      "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    secroots-file   "/var/named/data/named.secroots";
    recursing-file  "/var/named/data/named.recurse";
    allow-query     { localhost; 192.168.10.0/24; };
```

Where 192.168.10.31 is our Name server's IP Address

And 192.168.10.0/24 is the network's range from where clients can query the Name server. If you want it be open for all network, your “any” instead of network addr.

Step4: Edit the other zone configuration file i.e. “/etc/named.rfc1912.zones”

- To add the details of the zones i.e. forward lookup zone and reverse lookup zone we need to edit the /etc/named.rfc1912.zones file as shown below
#vim /etc/named.rfc1912.zones

Copy the following 11 lines and paste it at the end of the file

```
zone "localhost.localdomain" IN {
    type master;
    file "named.localhost";
    allow-update { none; };
};

zone "localhost" IN {
    type master;
    file "named.localhost";
    allow-update { none; };
};
```

Once pasted, edit the fields as follows

```
#####
zone "vpts.com" IN {
    type master;
    file "my.flz";
    allow-update { none; };
};

zone "10.168.192.in-addr.arpa" IN {
    type master;
    file "my.rlz";
    allow-update { none; };
};
```

Where “vpts.com” is the name of the domain

And “10.168.192.in-addr.arpa” is the reverse order of our domain network.

“my.flz” is the name of the forward lookup zone file and...

“my.rlz” is the name of the reverse lookup zone file.

Note: extensions like flz or rlx are not required it is only used here to demonstrate forward and reverse zones



Step5: Navigate to /var/named/ directory and create a forward and reverse zone files.

- Navigate to /var/named/ directory and copy the named.localhost file with its permissions as my.flz and edit it.

#cd /var/named

#cp -p named.localhost my.flz

```
[root@mlinux1 ~]# cd /var/named/
[root@mlinux1 named]# ls
chroot  dynamic      named.ca      named.localhost  slaves
data     dyndb-ldap   named.empty   named.loopback
[root@mlinux1 named]# cp -p named.localhost my.flz
[root@mlinux1 named]# vim my.flz
```

- Edit my.flz file as follows

```
$TTL 1D
@ IN SOA mlinux3.vpts.com. root.vpts.com. (
    202004101 ; serial
    1D         ; refresh
    1H         ; retry
    1W         ; expire
    3H )       ; minimum

NS      mlinux3.vpts.com.
mlinux3 A   192.168.10.30
mlinux4 A   192.168.10.40
mlinux5 A   192.168.10.50
mlinux6 A   192.168.10.60
```

- Copy again named.localhost, this time as my.rlz or copy my.flz to my.rlz to avoid re-typing common entries in both files, and edit it as shown below.

```
#cp -p my.flz my.rlz
#vim my.rlz
```

```
[root@mlinux1 named]# cp -p my.flz my.rlz
[root@mlinux1 named]# vim my.rlz
```

```
$TTL 1D
@ IN SOA mlinux3.vpts.com. root.vpts.com. (
    202004101 ; serial
    1D         ; refresh
    1H         ; retry
    1W         ; expire
    3H )       ; minimum

NS      mlinux3.vpts.com.
30     PTR     mlinux3.vpts.com.
40     PTR     mlinux4.vpts.com.
50     PTR     mlinux5.vpts.com.
60     PTR     mlinux6.vpts.com.
```

Step6: check whether the zone files are consistent or not

- To check the consistency of zone files the command is
`#named-chkzone <domain name> zone file`
`#named-chkzone my.com my.flz (if you are not in named dir give absolute path)`

```
[root@mlinux3 named]# named-checkzone vpts.com my.flz
zone vpts.com/IN: loaded serial 202004101
OK
```

```
#named-chkzone network addr my.rlz
```

```
[root@mlinux3 named]# named-checkzone 192.168.10. my.rlz
zone 192.168.10/IN: loaded serial 202004101
OK
```

Step7: Restart the appropriate services

- Start the named service and make it permanent

```
#systemctl start named; systemctl enable named
```

```
[root@mlinux1 ~]# systemctl start named; systemctl enable named
ln -s '/usr/lib/systemd/system/named.service' '/etc/systemd/system/multi-user.target.wants/named.service'
```

Step8: Add DNS in firewall trust services to avoid blockage by firewall

```
#firewall-cmd --add-service=dns --permanent
```

```
#firewall-cmd --reload
```

```
#firewall-cmd --list-all
```

```
[root@mlinux1 ~]# firewall-cmd --add-service=dns --permanent
success
[root@mlinux1 ~]# firewall-cmd --reload
success
[root@mlinux1 ~]# firewall-cmd --list-all
public (default)
  interfaces:
  sources:
  services: dhcpcv6-client dns ftp mountd nfs rpc-bind samba ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

Step9: Add the address of DNS server in /etc/resolv.conf on server side

- Edit the /etc/resolv.conf and add the IP of DNS server

```
#vim /etc/resolv.conf
```

```
[root@mlinux3 named]# cat /etc/resolv.conf
# Generated by NetworkManager
search vpts.com
nameserver 192.168.10.30
```

Okay, now we've done with DNS server configuration, check whether it is resolving IP to hostname and hostname to IP using various commands.

- Using dig command to check the DNS resolution
- Check with giving hostname of server

```
#dig <FQDN> of server/#dig mlinux3.vpts.com
```

```
[root@mlinux3 named]# dig mlinux3.vpts.com

; <>> DiG 9.11.4-P2-RedHat-9.11.4-26.P2.el8 <>> mlinux3.vpts.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 8378
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 8ada506a60227eef31a0fd405e9080f22b2602dff3b0132d (good)
; QUESTION SECTION:
;mlinux3.vpts.com.      IN      A

;; ANSWER SECTION:
mlinux3.vpts.com.    86400   IN      A       192.168.10.30

;; AUTHORITY SECTION:
vpts.com.            86400   IN      NS     mlinux3.vpts.com.

;; Query time: 0 msec
;; SERVER: 192.168.10.30#53(192.168.10.30)
;; WHEN: Fri Apr 10 19:51:38 IST 2020
;; MSG SIZE rcvd: 103
```

- Check with giving IP of hostname

```
#dig -x 192.168.10.30
```

```
[root@mlinux3 named]# dig -x 192.168.10.30

; <>> DiG 9.11.4-P2-RedHat-9.11.4-26.P2.el8 <>> -x 192.168.10.30
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23967
; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: udp: 4096
; COOKIE: f487e96a25f94e129ae63cf75e9085a37420441a41db0fd3 (good)
; QUESTION SECTION:
;30.10.168.192.in-addr.arpa. IN PTR

; ANSWER SECTION:
30.10.168.192.in-addr.arpa. 86400 IN PTR mlinux3.vpts.com.

; AUTHORITY SECTION:
10.168.192.in-addr.arpa. 86400 IN NS mlinux3.vpts.com.

; ADDITIONAL SECTION:
mlinux3.vpts.com. 86400 IN A 192.168.10.30

; Query time: 0 msec
; SERVER: 192.168.10.30#53(192.168.10.30)
; WHEN: Fri Apr 10 20:11:39 IST 2020
; MSG SIZE rcvd: 143
```

- Check the same with client's IP and Host name

```
#dig mlinux4.vpts.com
```

```
[root@mlinux3 named]# dig mlinux4.vpts.com

; <>> DiG 9.11.4-P2-RedHat-9.11.4-26.P2.el8 <>> mlinux4.vpts.com
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49732
; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: udp: 4096
; COOKIE: d887fc2d28536b466285585c5e9082bdb4abe4654eafa0ab (good)
; QUESTION SECTION:
;mlinux4.vpts.com. IN A

; ANSWER SECTION:
mlinux4.vpts.com. 86400 IN A 192.168.10.40

; AUTHORITY SECTION:
vpts.com. 86400 IN NS mlinux3.vpts.com.

; ADDITIONAL SECTION:
mlinux3.vpts.com. 86400 IN A 192.168.10.30

; Query time: 0 msec
; SERVER: 192.168.10.30#53(192.168.10.30)
; WHEN: Fri Apr 10 19:59:17 IST 2020
; MSG SIZE rcvd: 127
```

- With IP address:

```
#dig -x 192.168.10.40
```

```
[root@mlinux3 named]# dig -x 192.168.10.40

; <>> DiG 9.11.4-P2-RedHat-9.11.4-26.P2.el8 <>> -x 192.168.10.40
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38188
; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: udp: 4096
; COOKIE: 7716f92f999692f7519a97425e9085f4d8884b8797d2debf (good)
; QUESTION SECTION:
;40.10.168.192.in-addr.arpa. IN PTR

; ANSWER SECTION:
40.10.168.192.in-addr.arpa. 86400 IN PTR mlinux4.vpts.com.

; AUTHORITY SECTION:
10.168.192.in-addr.arpa. 86400 IN NS mlinux3.vpts.com.

; ADDITIONAL SECTION:
mlinux3.vpts.com. 86400 IN A 192.168.10.30

; Query time: 0 msec
; SERVER: 192.168.10.30#53(192.168.10.30)
; WHEN: Fri Apr 10 20:13:00 IST 2020
; MSG SIZE rcvd: 151
```

Using ping to test the resolution

- Try pinging with hostname both server and client

```
#ping -c2 mlinux4
```

```
[root@mlinux3 named]# ping -c2 mlinux4
PING mlinux4.vpts.com (192.168.10.40) 56(84) bytes of data.
64 bytes from mlinux4.vpts.com (192.168.10.40): icmp_seq=1 ttl=64 time=0.449 ms
64 bytes from mlinux4.vpts.com (192.168.10.40): icmp_seq=2 ttl=64 time=1.20 ms

--- mlinux4.vpts.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 41ms
rtt min/avg/max/mdev = 0.449/0.822/1.195/0.373 ms
```

Using host command to check resolution

- Checking the DNS resolution with host command for both server as well as clients

```
#host <hostname>
#host mlinux3
#host mlinux4
```

```
[root@mlinux3 named]# host mlinux3
mlinux3.vpts.com has address 192.168.10.30
[root@mlinux3 named]# host mlinux4
mlinux4.vpts.com has address 192.168.10.40
```

- Using host command with IP address of server as well as client

```
#host 192.168.10.3
#host 192.168.10.4 (or) 5, 6 any client
```

```
[root@mlinux3 named]# host 192.168.10.30
30.10.168.192.in-addr.arpa domain name pointer mlinux3.vpts.com. S
[root@mlinux3 named]# host 192.168.10.40
40.10.168.192.in-addr.arpa domain name pointer mlinux4.vpts.com.
```

Using nslookup command to check the DNS resolution

- Use nslookup command with server and clients hostname and check it

```
#nslookup mlinux2
#nslookup mlinux3
```

```
[root@mlinux3 named]# nslookup mlinux3
Server:          192.168.10.30
Address:         192.168.10.30#53

Name:   mlinux3.vpts.com
Address: 192.168.10.30
```

```
[root@mlinux3 named]# nslookup mlinux4
Server:          192.168.10.30
Address:         192.168.10.30#53

Name:   mlinux4.vpts.com
Address: 192.168.10.40
```

- Check the same thing with IP addresses

```
#nslookup 192.168.10.30
#nslookup 192.168.10.40
```

```
[root@mlinux3 named]# nslookup 192.168.10.30
30.10.168.192.in-addr.arpa      name = mlinux3.vpts.com.
```

```
[root@mlinux3 named]# nslookup 192.168.10.40
40.10.168.192.in-addr.arpa      name = mlinux4.vpts.com.
```

Client side configuration for DNS

- Log into any client machine and add the DNS server's information in /etc/resolv.conf file.

```
#vim /etc/resolv.conf
```

```
[root@mlinux3 named]# cat /etc/resolv.conf
# Generated by NetworkManager
search vpts.com
nameserver 192.168.10.30
```

- Now check with any of the options used previously like dig, ping, host or nslookup for dns resolution

```
[root@mlinux4 ~]# nslookup mlinux3
Server:          192.168.10.30
Address:         192.168.10.30#53
```

```
Name:   mlinux3.vpts.com
Address: 192.168.10.30
```

```
[root@mlinux4 ~]# nslookup mlinux4
Server:          192.168.10.30
Address:         192.168.10.30#53
```

```
Name:   mlinux4.vpts.com
Address: 192.168.10.40
```

```
[root@mlinux4 ~]# nslookup 192.168.10.30
30.10.168.192.in-addr.arpa      name = mlinux3.vpts.com.
```

```
[root@mlinux4 ~]# nslookup 192.168.10.40
40.10.168.192.in-addr.arpa      name = mlinux4.vpts.com.
```

**Do the same for every client and check it with various commands on every client
Also make sure that hostname should Fully Qualified Domain Name.**

MAIL SERVER

Electronic mail is one of the best way to communicate for computer users anywhere in the world. If I wanted to write an email message to my friend who is sitting somewhere in the world, I simply open up my outlook-click on compose-type my friends email address in the “to box-mention” the subject-draft the message-attach files (if needed)-click on send. That's it. This is what I do to send an email to my friends. Not only me, all the computer users will do the exact same thing. But for most of the time i didn't know how the mail flow takes place. How the transfer takes place and how will it reach the recipient and the intermediate process and so on....

There are a few new keywords we need to look into.....

Mail User Agent: MUA is the email client which we use to create-draft-send emails. Generally Microsoft Outlook, Thunderbird, Kmail and so on..... are examples of MUA's

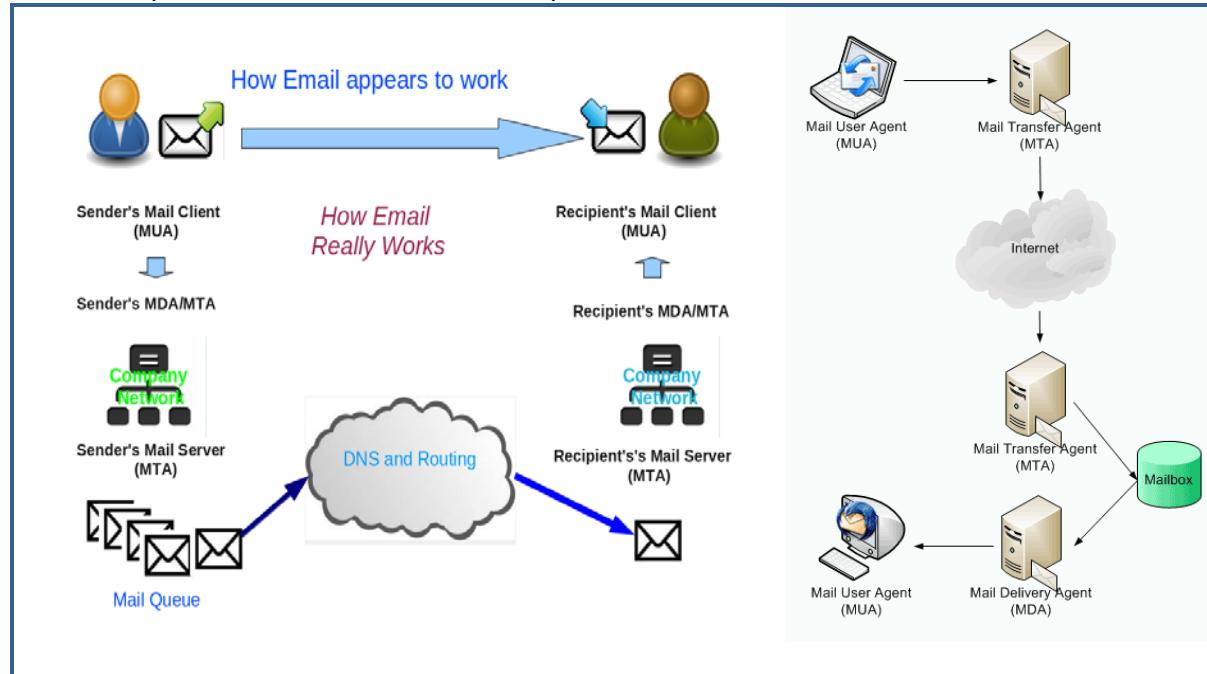
Mail Transfer Agent: Message, Mail transfers between sender(s) and recipient(s) will take place between the MTA's. Exchange, Qmail, Sendmail, PostFix and so on.... are example of MTA's

Mail Delivery Agent: It is an agent which is responsible for delivery of mail to the devices like laptop, desktop, mobiles and tabs etc. Imtp, pop3, imap4 etc., are the examples of MDA

SMTP

Simple Mail Transfer Protocol will transfer the mails between the MTA's

Let's take a deeper look into this with a small example. The below picture will depict how mail flow takes place between sender and recipient.



From the above picture, when the sender clicks on Send in his MUA, the mail will be transferred to the MTA of the sender which exists in the Mail server of the Sender. The MTA of the sender will check for the recipients address (MX records-Mail Exchange Records) and if it finds the recipients address then the mail will be flowed from Senders MTA to Recipient MTA using the SMTP via TCP Port 25. Once the Recipients MTA receives the email, it will be transferred to MUA of recipient. Once the Recipient Clicks on the Send/Receive button then the email will be once click away from him residing in his inbox.

In addition to the above processes there is another agent called as MDA-Mail delivery agent. MDA will receive the email from the MTA and will deliver it to the recipients MUA.

Profile for MAIL server

Usage	:	To send and receive emails
Package	:	Postfix, Dovecot
Configuration file	:	/etc/postfix/main.cf, /etc/dovecot/dovecot.conf
Port no	:	25 – SMTP (Simple Mail Transfer Protocol)
Daemon	:	postfix, dovecot

Lab Work:

Pre-requisite: DNS should be configured and server as well as client should be participant of it

Step 1: Check the hostname of the system

#hostname

```
[root@myrhel73 ~]# hostname
myrhel73.my.com
```

Step 2: Install the Packages

```
[root@myrhel73 ~]# yum install postfix* -y
[root@myrhel73 ~]# yum install dovecot* -y
```

Note: Most probably postfix will be pre-install, so only dovecot is needed to be installed

Step 3: Search and Edit the configuration file /etc/postfix/main.cf as shown:

#vim /etc/postfix/main.cf with following line numbers

```
94 myhostname = mlinux3.vpts.com
102 mydomain = vpts.com
118 myorigin = $mydomain
132 inet_interfaces = all
133 #inet_interfaces = $myhostname
134 #inet_interfaces = $myhostname, localhost
135 #inet_interfaces = localhost
183 mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain
283 mynetworks = 192.168.10.0/24, 127.0.0.0/8
```

Step 4: Edit the configuration file /etc/dovecot/dovecot.conf as shown

#vim /etc/dovecot/dovecot.conf ; line no. 24

```
24 protocols = imap pop3 lmtp
```

Step 5: Start/Restart the services and make them permanent

```
#systemctl restart postfix; systemctl enable postfix
#systemctl start dovecot; systemctl enable dovecot
```

```
[root@myrhel73 ~]# systemctl restart postfix.service; systemctl enable postfix.service
[root@myrhel73 ~]# systemctl start dovecot; systemctl enable dovecot
```

Step 6: Allow smtp in firewall as below

```
[root@myrhel73 ~]# firewall-cmd --add-service=smtp --permanent
success
[root@myrhel73 ~]# firewall-cmd --reload
success
```

Step 7: Add the following entries with the MX record in DNS

```
#vim /var/named/my.flz (forward zone)
```

```
[root@mlinux3 ~]# cat /var/named/my.flz
$TTL 1D
@ IN SOA mlinux3.vpts.com. root.vpts.com. (
    202004101 ; serial
    1D         ; refresh
    1H         ; retry
    1W         ; expire
    3H )       ; minimum
NS      mlinux3.vpts.com.
mlinux2 A   192.168.10.20
mlinux3 A   192.168.10.30
mail    CNAME  mlinux3
mlinux4 A   192.168.10.40
mlinux5 A   192.168.10.50
mlinux6 A   192.168.10.60
vpts.com. A  192.168.10.30
vpts.com. MX 10 mlinux3
[root@mlinux3 ~]#
```

Step 8: Reload the named service

```
[root@myrhel7 ~]# systemctl reload named
```

Client side configuration for RHEL machine

Step1: Make the following change in /etc/postfix/main.cf

```
132  inet_interfaces = all
133  #inet_interfaces = $myhostname
134  #inet_interfaces = $myhostname, localhost
135  #inet_interfaces = localhost
338  relayhost = [mlinux3.vpts.com]
```

Note: Line no. varies from rhel6, 7 and 8 versions.

Step2: Restart the postfix service

```
[root@myrhel72 ~]# systemctl restart postfix
```

Step 3: Allow smtp in firewall as below

```
[root@myrhel73 ~]# firewall-cmd --add-service=smtp --permanent
success
[root@myrhel73 ~]# firewall-cmd --reload
success
```

Send the mail from one machine to other as following:

```
#mail -s (subject) user@hostname
#mail -s TESTMAIL root@mlinux4.com
```

```
[root@m... 3 ~]# mail -s TESTMAIL root@mlinux4.vpts.com
HI THERE,
THIS IS A TEST MAIL
EOT ctrl+d to send the mail
[root@myrhel73 ~]#
```

Check the mail que, to see if the mail has flown out or not

```
#mailq
```

```
[root@myrhel73 ~]# mailq
Mail queue is empty
```

Note: If mailq is empty, that means the mail has been sent

Check the mailbox on destination server, whether mail is received

```
#mail
```

```
[root@myrhel72 ~]# mail
Heirloom Mail version 12.5 7/5/10. Type ? for help.
"/var/spool/mail/root": 1 message 1 new
>N 1 root Wed Oct 19 22:35 22/793 "TESTMAIL"
&
```

Read the mail number 1 and reply

```
& 1
From: root@myrhel73.my.com (root)
Status: R

HI THERE,
THIS IS A TEST MAIL
```

Reply it using “r” key

```
& r
To: root@myrhel72.my.com root@myrhel73.my.com
Subject: Re: TESTMAIL

root@myrhel73.my.com (root) wrote:

> HI THERE,
> THIS IS A TEST MAIL
Rec'd it successfully
EOT
```

Check back on first Machine whether it received a mail back or not

#mail

```
[root@myrhel73 ~]#
You have new mail in /var/spool/mail/root
[root@myrhel73 ~]# mail
Heirloom Mail version 12.5 7/5/10. Type ? for help.
"/var/spool/mail/root": 2 messages 1 new
  1 Mail Delivery System  Wed Oct 19 22:29  75/2394 "Undelivered Mail Returned to Sender"
>N  2 root           Wed Oct 19 22:38  27/996 "Re: TESTMAIL"
& [2]
Message 2:
From root@myrhel72.my.com  Wed Oct 19 22:38:14 2016

Return-Path: <root@myrhel72.my.com>
X-Original-To: root@myrhel73.my.com
Delivered-To: root@myrhel73.my.com
Date: Wed, 19 Oct 2016 22:38:15 +0530
To: root@myrhel73.my.com, root@myrhel72.my.com
Subject: Re: TESTMAIL
User-Agent: Heirloom mailx 12.5 7/5/10
Content-Type: text/plain; charset=us-ascii
From: root@myrhel72.my.com (root)
Status: R

root@myrhel73.my.com (root) wrote:

> HI THERE,
> THIS IS A TEST MAIL
Rec'd it successfully

& [ ]
```

Also test mails from between two different clients to test Relay host functionality.

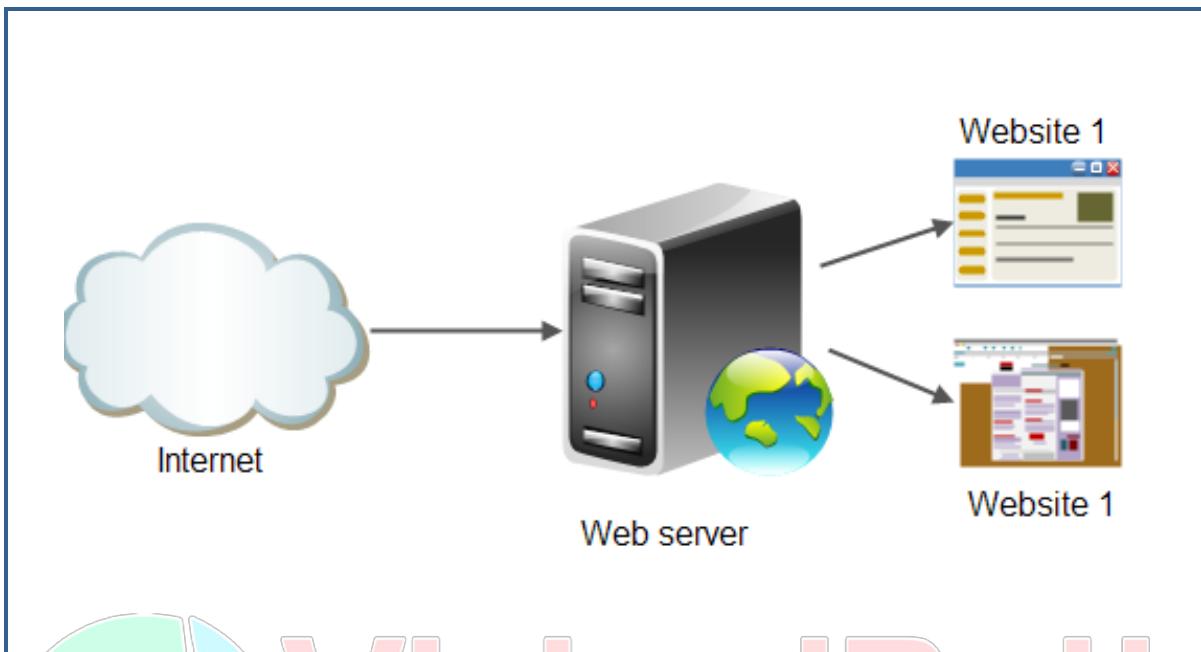
To delete the mails stucked in mailq use the following command

#postsuper -d ALL

Note: If internet connection is available and you have a registered domain on internet, then you can send the mails directly to any mail address in the world, but unregistered domains mail will be rejected..

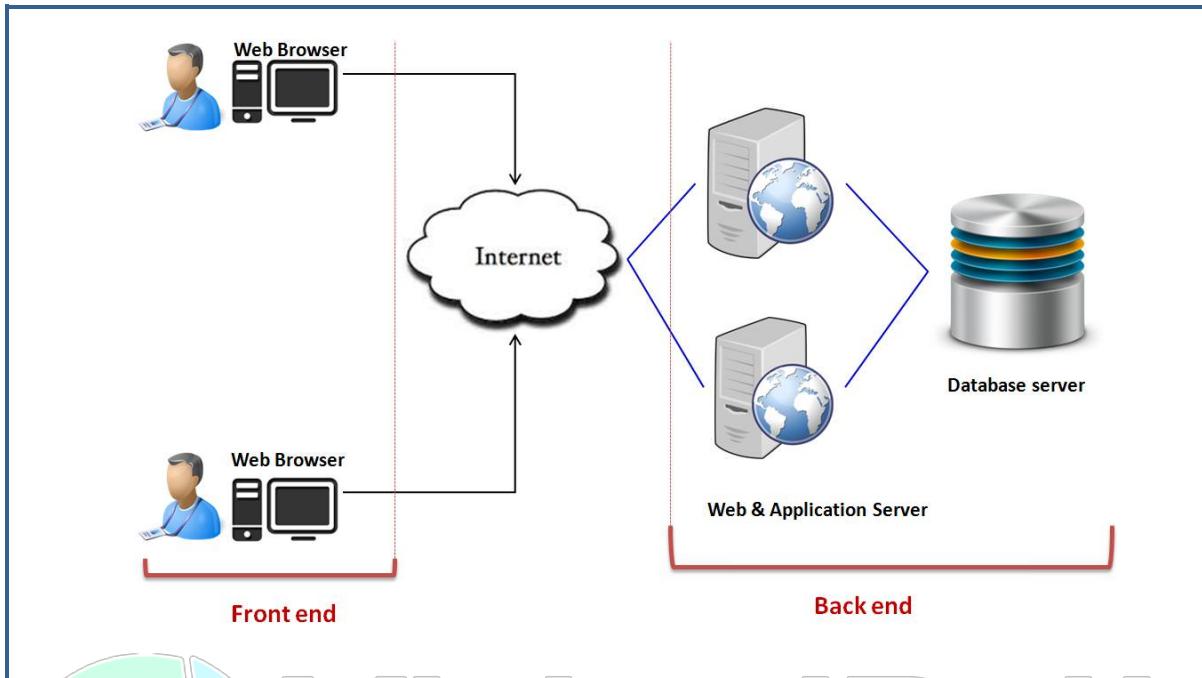
That's it, keep working

WEB SERVER (APACHE)



- Every Web site sits on a computer known as a Web server. This server is always connected to the internet. Web servers are computers that deliver (serves up) Web pages. Every Web server has an IP address and possibly a domain name.
- A web server can mean two things - a computer on which a web site is hosted and a program that runs on such a computer. So the term web server refers to both hardware and software.
- A web server is what makes it possible to be able to access content like web pages or other data from anywhere as long as it is connected to the internet. The hardware houses the content, while the software makes the content accessible through the internet.
- The most common use of web servers is to host websites but there are other uses like data storage or for running enterprise applications. There are also different ways to request content from a web server. The most common request is the Hypertext Transfer Protocol (HTTP), but there are also other requests like the Internet Message Access Protocol (IMAP) or the File Transfer Protocol (FTP).

How a Web Server Works



A simple exchange between the client machine and Web server goes like this:

1. The client's browser dissects the URL into a number of separate parts, including address, path name and protocol.
2. A Domain Name Server (DNS) translates the domain name the user has entered into its IP address, a numeric combination that represents the site's true address on the Internet (a domain name is merely a "front" to make site addresses easier to remember).
3. The browser now determines which protocol (the language client machines use to communicate with servers) should be used. Examples of protocols include FTP, or File Transfer Protocol, and HTTP, Hypertext Transfer Protocol.
4. The server sends a GET request to the Web server to retrieve the address it has been given. For example, when a user types <http://www.example.com/1.jpg>, the browser sends a GET 1.jpg command to example.com and waits for a response. The server now responds to the browser's requests. It verifies that the given address exists, finds the necessary files, runs the appropriate scripts, exchanges cookies if necessary, and returns the results back to the browser. If it cannot locate the file, the server sends an error message to the client.
5. The browser translates the data it has been given into HTML and displays the results to the user.

Profile for Apache Server

Use	:	Hosting a web site.
Package	:	httpd
Port	:	80/http
Configuration File	:	/etc/httpd/conf/httpd.conf /etc/httpd/conf.d/ssl.conf
Sample Config File	:	/usr/share/doc/httpd/httpd-vhosts.conf
Configuration directory	:	/etc/httpd/conf.d
Document Root	:	/var/www/html
Daemon	:	httpd

Steps to configure a simple web server

Step1: Install the package

- The package for apache web server is httpd.

```
#yum install httpd* -y
```

```
[root@ adm ~]# yum install httpd* -y
Loaded plugins: refresh-packagekit, rhnplugin
This system is not registered with RHN.
RHN support will be disabled.
rhel                                         | 3.7 kB     00:00 ..
Setting up Install Process
Package httpd-2.2.15-5.el6.i686 already installed and latest version
Package httpd-tools-2.2.15-5.el6.i686 already installed and latest version
Resolving Dependencies
--> Running transaction check
-->> Package httpd-devel.i686 0:2.2.15-5.el6 set to be updated
-->> Processing Dependency: apr-util-devel for package: httpd-devel-2.2.15-5.el6
Installed:
  httpd-devel.i686 0:2.2.15-5.el6          httpd-manual.noarch 0:2.2.15-5.el6

Dependency Installed:
  apr-devel.i686 0:1.3.9-3.el6            apr-util-devel.i686 0:1.3.9-3.el6
  cyrus-sasl-devel.i686 0:2.1.23-8.el6    db4-cxx.i686 0:4.7.25-16.el6
  db4-devel.i686 0:4.7.25-16.el6         expat-devel.i686 0:2.0.1-9.1.el6
  openldap-devel.i686 0:2.4.19-15.el6

Complete!
```



Step2: Copy a sample file to /etc/httpd/conf.d folder and edit it.

- #cp /usr/share/doc/httpd/httpd-vhosts.conf /etc/httpd/conf.d/msw.conf

```
[root@mlinux3 ~]# cp /usr/share/doc/httpd/httpd-vhosts.conf /etc/httpd/conf.d
```

- Edit the copied file
- #vim /etc/httpd/conf.d/msw.conf

```
<VirtualHost *:@@Port@@>
  ServerAdmin webmaster@dummy-host.example.com
  DocumentRoot "@@ServerRoot@@/docs/dummy-host.example.com"
  ServerName dummy-host.example.com
  ServerAlias www.dummy-host.example.com
  ErrorLog "/var/log/httpd/dummy-host.example.com-error_log"
  CustomLog "/var/log/httpd/dummy-host.example.com-access_log" common
</VirtualHost>
```

Keep these lines in file and edit it with your preferences.

- Edit the file as below

```
#####MY SIMPLE WEBSITE#####
<VirtualHost 192.168.106.81:80>
    ServerAdmin root@mlinux1.my.com
    DocumentRoot "/var/www/html"
    ServerName mlinux1.my.com
    ServerAlias mlinux1.my.com
    ErrorLog "/var/log/httpd/msw-error_log"
    CustomLog "/var/log/httpd/msw-access_log" common
</VirtualHost>
```

Step2: Navigate to the document root folder i.e. /var/www/html/ and create an index.html file which will be accessed through a web browser

- #vim /var/www/html/index.html

```
<h1>MY SIMPLE WEBSITE</h1>
<h2>WELCOME TO WEB SERVER </h2>
```

Step3: Start the service and enable it in boot configuration

```
#systemctl start httpd; systemctl enable httpd
```

```
[root@mlinux1 html]# systemctl start httpd; systemctl enable httpd
ln -s '/usr/lib/systemd/system/httpd.service' '/etc/systemd/system/m
```

Step3: Allow http in firewall trusted list

```
[root@mlinux1 html]# firewall-cmd --add-service=http --permanent
success
[root@mlinux1 html]# firewall-cmd --reload
success
[root@mlinux1 html]# firewall-cmd --list-all
public (default, active)
  interfaces: bond0 ens3 ens8
  sources:
  services: dhcpcv6-client dns ftp http mounted nfs rpc-bind samba ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

Step4: Access the Server via web browser like Firefox, etc.

- Open Firefox web browser and type the IP Address of the web server
<http://192.168.106.81>



Same thing with hostname

- <http://mlinux1.my.com>
- (note: the client must be integrated with DNS, to get this thing work)



- To open the website from command line use the following command

```
#curl <IP/HOSTNAME of web server>
```

```
#curl 192.168.106.81
```

```
[root@mlinux2 Desktop]# curl 192.168.106.81
<h1>MY SIMPLE WEBSITE</h1>
<h2>WELCOME TO WEBSERVER </h2>
[root@mlinux2 Desktop]#
```

```
[root@mlinux2 Desktop]# curl mlinux1.my.com
<h1>MY SIMPLE WEBSITE</h1>
<h2>WELCOME TO WEBSERVER </h2>
[root@mlinux2 Desktop]#
```

Also Try #elinks --dump 192.168.10.95 and check the output

DNS configuration to get the feel of “www”

- Open the DNS configuration file and add the canonical name as “www”, so that we can use our domain as full fledged website.

```
#vim /var/named/my.rlz
```

```
[root@mlinux1 html]# vim /var/named/my.rlz
                                201610121      ; serial
                                1D            ; refresh
                                1H            ; retry
                                1W            ; expire
                                3H )          ; minimum

        NS      mlinux1.my.com.
mlinux1 A      192.168.106.81
www    CNAME   mlinux1
mlinux2 A      192.168.106.82
mlinux3 A      192.168.106.83
mlinux4 A      192.168.106.84
```

- Restart the DNS services

```
#systemctl restart/reload named
```

```
[root@node1 ~]# systemctl reload named
```

- Okay! now we are ready, point the browser to the address as follows
www.my.com



Note: This will only work in your DNS range, from others who are not in DNS use ip addresses

To create an Alias/other page in the same website

- Navigate to the document root i.e. /var/www/html/ and create a folder

```
[root@mlinux1 ~]# cd /var/www/html
[root@mlinux1 html]# mkdir sec
[root@mlinux1 html]# cd sec
[root@mlinux1 sec]# vim index.html
```

- Create an index.html

```
<h1>MY SIMPLE WEBSITE SECOND PAGE</h1>
<h2>WELCOME TO WEB SERVER </h2>
```

- Open the configuration file /etc/httpd/conf.d/msw.conf and add a line as alias
#vim /etc/httpd/conf/httpd.conf (RHEL6)
#vim /etc/httpd/conf.d/httpd-vhosts.conf (RHEL7)

```
#####MY SIMPLE WEBSITE#####
<VirtualHost 192.168.106.81:80>
    ServerAdmin root@mlinux1.my.com
    DocumentRoot "/var/www/html"
    Alias      /2 /var/www/html/sec
    ServerName mlinux1.my.com
    ServerAlias mlinux1.my.com
    ErrorLog "/var/log/httpd/msw-error_log"
    CustomLog "/var/log/httpd/msw-access_log" common
</VirtualHost>
```

- Reload the httpd service
#systemctl reload httpd

```
[root@node1 ~]# systemctl reload httpd
```

- Open the Firefox web browser and type the following url
<http://192.168.106.81/2> or <http://mlinux1.my.com/2>



MY SIMPLE WEBSITE SECOND PAGE

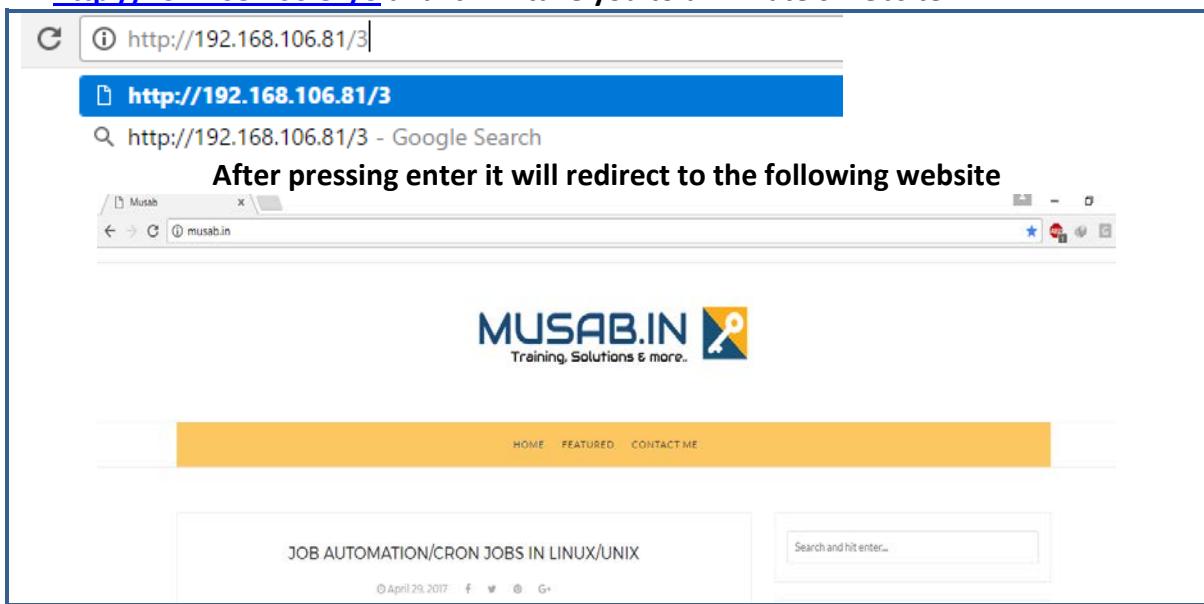
WELCOME TO WEB SERVER

Redirecting the page to other website:

- Redirecting is to take the visitor to other website while they visit a particular page in our website
- To redirect a website, navigate and open the configuration file of http i.e. /etc/httpd/conf.d/msw.conf and add the following line in it at the end.
`#vim /etc/httpd/conf.d/msw.conf`

```
#####MY SIMPLE WEBSITE#####
<VirtualHost 192.168.106.81:80>
    ServerAdmin root@mlinux1.my.com
    DocumentRoot "/var/www/html"
    Alias /2 /var/www/html/sec
    Redirect /3 "http://www.musab.in"
    ServerName mlinux1.my.com
    ServerAlias mlinux1.my.com
    ErrorLog "/var/log/httpd/msw-error_log"
    CustomLog "/var/log/httpd/msw-access_log" common
</VirtualHost>
```

- Reload the httpd service
`#systemctl reload httpd`
- Take a browser and type the following website address
<http://192.168.106.81/3> and it will take you to unixmate's website



After pressing enter it will redirect to the following website

<http://www.virtualpathtech.com>

VirtualPath Techno Solutions

Virtual Web hosting

Virtual hosting is a method for hosting multiple domain names on a server using a single IP address. This allows one server to share its resources, such as memory and processor cycles, in order to use its resources more efficiently.

Port based Hosting:

- The default port number for HTTP is 80. However, most web servers can be configured to operate on almost any port number, provided the port number is not in use by any other program on the server.
- For example, a server may host the website www.example.com. However, if the owner wishes to operate a second site, and does not have access to the domain name configuration for their domain name, and/or owns no other IP addresses which could be used to serve the site from, they could instead use another port number, for example, www.example.com:81 for port 81, www.example.com:8000 for port 8000, or www.example.com:8080 for port 8080.

Steps to configure a port based web hosting

Step1: Make a directory for port based web hosting in document root i.e. /var/www/ say port.

```
#mkdir /var/www/port
```

```
[root@ktadm ~]# mkdir /var/www/port
[root@ktadm ~]# cd /var/www/
[root@ktadm www]# ls
cgi-bin  error  html  icons  manual  port
[root@ktadm www]#
```

Step2: Navigate to port directory and create an index.html file there

```
[root@ adm ~]# cd /var/www/port/
[root@ adm port]# vim index.html
<h1>MY PORT BASED WEBSITE</h1>
<h2>WELCOME TO WEB SERVER</h2>
```

Step3: Navigate to /etc/httpd/httpd.conf and copy the old file i.e., msw.conf with new name for new website

```
#cd /etc/httpd/conf.d/httpd
```

```
#cp msw.conf port.conf
```

```
[root@node1 ~]# cd /etc/httpd/conf.d/
[root@node1 conf.d]# ls
autoindex.conf  manual.conf  msw.conf  README  userdir.conf  welcome.conf
[root@node1 conf.d]# cp msw.conf port.conf
```

```
#vim /etc/httpd/conf.d/port.conf
```

```
#####
PORT BASED WEBSITE #####
Listen 8080
<VirtualHost 192.168.106.81:8080>
    ServerAdmin root@mlinux1.my.com
    DocumentRoot "/var/www/port"
    ServerName mlinux1.my.com
    ErrorLog "/var/log/httpd/port-error_log"
    CustomLog "/var/log/httpd/port-access_log" common
</VirtualHost>
```

Step4: Reload the httpd service

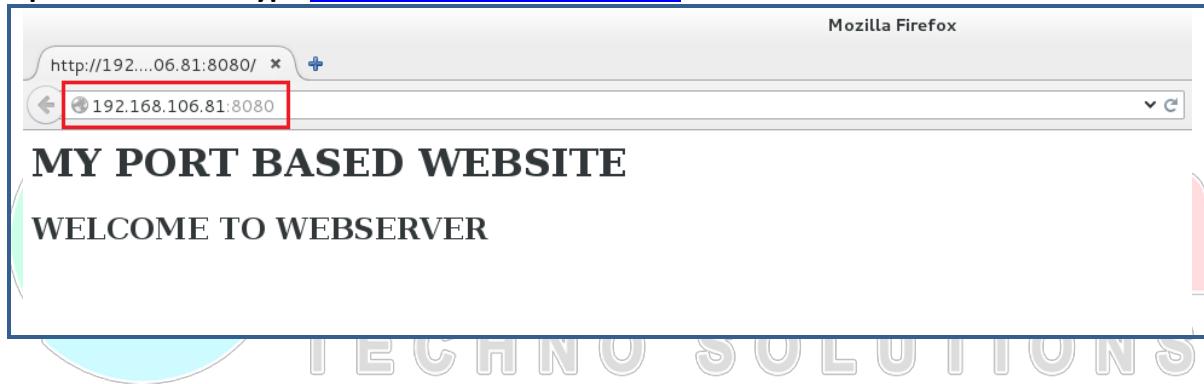
```
#systemctl reload httpd
```

```
[root@node1 ~]# systemctl reload httpd
```

- Allow the port no 8080 in firewall in RHEL7

```
[root@mlinux1 port]# firewall-cmd --add-port=8080/tcp --permanent
success
[root@mlinux1 port]# firewall-cmd --reload
success
[root@mlinux1 port]# firewall-cmd --list-all
public (default, active)
  interfaces: bond0 ens3 ens8
  sources:
  services: dhcpc6-client dns ftp http mountd nfs rpc-bind samba ssh
  ports: 8080/tcp
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

Open Firefox and type <http://192.168.106.81:8080>



Name Based Virtual web Hosting

- Name-based virtual hosts use multiple host names for the same web server IP address.
- With web browsers that support HTTP/1.1 (as nearly all now do), upon connecting to a webserver, the browsers send the hostname from the address that the user typed into their browser's address bar along with the requested resource itself to the web server. The server can use the Host header field to determine which web site (or *virtual host*), as well as page, to show the user. The browser specifies the address by setting the Host HTTP header with the host specified by the user. The Host header is required in all HTTP/1.1 requests.
- For instance, a server could be receiving requests for two domains, www.example.com and www.example.net, both of which resolve to the same IP address. For www.example.com, the server would send the HTML file from the directory /var/www/user/Joe/site/, while requests for www.example.net would make the server serve pages from /var/www/user/Mary/site/.
- Example: A blog server can be hosted using Name base hosting. blog1.example.com and blog2.example.com

Steps to configure name based web hosting:

Step1: Make a directory in document root i.e. /var/www/ with some name say "ktname"

```
#mkdir /var/www/name
```

- Add an index page in it

```
#vim /etc/var/www/name/index.html
```

```
<h1>MY NAME BASED WEBSITE</h1>
<h2>WELCOME TO WEB SERVER</h2>
```

Step2: Update the DNS zone configuration files with the new hostname of the web server

```
#vim /var/named/my.flz
```

```
$TTL 1D
@ IN SOA mlinux1.my.com. root.my.com. (
    201610121 ; serial
    1D ; refresh
    1H ; retry
    1W ; expire
    3H ) ; minimum

    NS mlinux1.my.com.
mlinux1 A 192.168.106.81
www CNAME mlinux1
prod A 192.168.106.81
mlinux2 A 192.168.106.82
mlinux3 A 192.168.106.83
mlinux4 A 192.168.106.84
```

```
#vim /var/named/my.rlz
```

```
$TTL 1D
@ IN SOA mlinux1.my.com. root.my.com. (
    201610121 ; serial
    1D ; refresh
    1H ; retry
    1W ; expire
    3H ) ; minimum

    NS mlinux1.my.com.
mlinux1 A 192.168.106.81
81 PTR mlinux1.my.com.
81 PTR prod.my.com.
82 PTR mlinux2.my.com.
83 PTR mlinux3.my.com.
84 PTR mlinux4.my.com.
```

- Reload the DNS services

```
#systemctl reload named
```

```
[root@node1 ~]# systemctl reload named
```

Step5: Navigate to /etc/httpd/httpd.conf and copy the old file i.e., port.conf with new name for new website

```
#cd /etc/httpd/conf.d/httpd
#cp port.conf name.conf
```

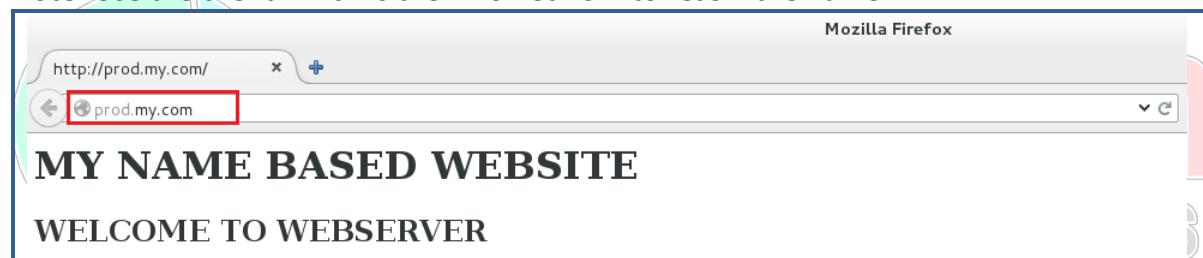
```
[root@node1 ~]# cd /etc/httpd/conf.d
[root@node1 conf.d]# ls
autoindex.conf manual.conf msw.conf port.conf README userdir.conf welcome.conf
[root@node1 conf.d]# cp port.conf name.conf
[root@node1 conf.d]# vim name.conf
#####
# NAME BASED WEBSITE #####
NameVirtualHost 192.168.106.81:80
<VirtualHost 192.168.106.81:80>
    ServerAdmin root@prod.my.com
    DocumentRoot "/var/www/name"
    ServerName prod.my.com
    ErrorLog "/var/log/httpd/name-error_log"
    CustomLog "/var/log/httpd/name-access_log" common
</VirtualHost>
```

Step6: Reload the httpd services

```
#systemctl reload httpd
```

http://prod.my.com now it will navigate to our name based web page

Note: Use the client which is the DNS network to resolv the name.

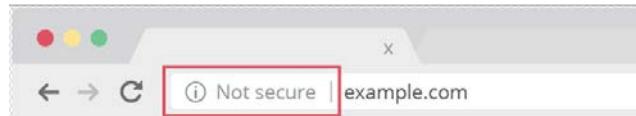


CONFIGURING HTTPS SERVER USING SELF-SIGNED CERTIFICATE

What is HTTPS?

Hypertext transfer protocol secure (HTTPS) is the secure version of HTTP, which is the primary protocol used to send data between a web browser and a website. HTTPS is encrypted in order to increase security of data transfer. This is particularly important when users transmit sensitive data, such as by logging into a bank account, email service, or health insurance provider.

Any website, especially those that require login credentials, should use HTTPS. In modern web browsers such as Chrome, websites that do not use HTTPS are marked differently than those that are. Look for a green padlock in the URL bar to signify the webpage is secure. Web browsers take HTTPS seriously; Google Chrome and other browsers flag all non-HTTPS websites as not secure.



How does HTTPS work?

HTTPS uses an encryption protocol to encrypt communications. The protocol is called Transport Layer Security (TLS), although formerly it was known as Secure Sockets Layer (SSL). This protocol secures communications by using what's known as an asymmetric public key infrastructure. This type of security system uses two different keys to encrypt communications between two parties:

1. The private key - this key is controlled by the owner of a website and it's kept, as the reader may have speculated, private. This key lives on a web server and is used to decrypt information encrypted by the public key.
2. The public key - this key is available to everyone who wants to interact with the server in a way that's secure. Information that's encrypted by the public key can only be decrypted by the private key.

Why is HTTPS important? What happens if a website doesn't have HTTPS?

HTTPS prevents websites from having their information broadcast in a way that's easily viewed by anyone snooping on the network. When information is sent over regular HTTP, the information is broken into packets of data that can be easily "sniffed" using free software. This makes communication over the unsecure medium, such as public Wi-Fi, highly vulnerable to interception. In fact, all communications that occur over HTTP occur in plain text, making them highly accessible to anyone with the correct tools, and vulnerable to on-path attacks.

With HTTPS, traffic is encrypted such that even if the packets are sniffed or otherwise intercepted, they will come across as nonsensical characters. Let's look at an example:

Before encryption:

This is a string of text that is completely readable

After encryption:

**ITM0IRyiEhVpa6VnKyExMiEgNveroyWBPIgGyfkfIYjDaaFf/Kn3bo3OfghBPDWo6AfSHINtL
8N7ITEwlXc1gU5X73xMsJormzzXlwOyrCs+9XCPk63Y+z0=**

In websites without HTTPS, it is possible for Internet service providers (ISPs) or other intermediaries to inject content into webpages without the approval of the website owner. This commonly takes the form of advertising, where an ISP looking to increase revenue injects paid advertising into the webpages of their customers. Unsurprisingly, when this occurs, the profits for the advertisements and the quality control of those advertisements are in no way shared with the website owner. HTTPS eliminates the ability of unmoderated third parties to inject advertising into web content.

Steps to configure SSL based website:

1. Install httpd openssl mod_ssl packages using yum or dnf

```
[root@mlinux4 ~]# dnf install httpd openssl mod_ssl -y
rhel8-localBaseOS                                     2.1 MB/s | 2.8 kB
rhel8-localAppStream                                  2.5 MB/s | 3.2 kB
Package httpd-2.4.37-21.module+el8.2.0+5008+cca404a3.x86_64 is already installed.
Package openssl-1:1.1.1c-15.el8.x86_64 is already installed.
Dependencies resolved.
=====
 Package          Architecture      Version           Repository
=====
 Installing:
 mod_ssl          x86_64          1:2.4.37-21.module+el8.2.0+5008+cca404a3   AppStream
 Transaction Summary
=====
 Install 1 Package
```

Note: in this case httpd and openssl are already installed during httpd server configuration.

2. Create an self-signed SSL certificate and RSA private key to be used for securing website

```
#openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout server.key -out server.crt
```

```
[root@mlinux4 ~]# openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout server.key -out server.crt
Generating a RSA private key
.....+++++
.....+++++
writing new private key to 'server.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:IN
State or Province Name (full name) []:TS
Locality Name (eg, city) [Default City]:HYD
Organization Name (eg, company) [Default Company Ltd]:VPTS
Organizational Unit Name (eg, section) []:IT DEPT
Common Name (eg, your name or your server's hostname) []:mlinux4
Email Address []:root@mlinux4
```

Note: answer the questions asked to complete certificate and key creation

Check in the current folder you are, the key and certificate will be available

```
[root@mlinux4 ~]# ls -l ser*
-rw-r--r--. 1 root root 1383 Sep 23 17:33 server.crt
-rw-----. 1 root root 1704 Sep 23 17:28 server.key
```

3. Copy cert file and key file in following locations

```
[root@mlinux4 ~]# cp server.crt /etc/pki/tls/certs/
[root@mlinux4 ~]# cp server.key /etc/pki/tls/private/
```

4. Add the key and certificate location in /etc/httpd/conf.d/ssl.conf file

```
SSLCertificateFile /etc/pki/tls/certs/server.crt

# Server Private Key:
# If the key is not combined with the certificate, use this
# directive to point at the key file. Keep in mind that if
# you've both a RSA and a DSA private key you can configure
# both in parallel (to also allow the use of DSA ciphers, etc.)
# ECC keys, when in use, can also be configured in parallel
SSLCertificateKeyFile /etc/pki/tls/private/server.key
```

5. Create a configuration file for a new website in /etc/httpd/conf.d folder

```
[root@mlinux4 ~]# vim /etc/httpd/conf.d/myweb.conf
[root@mlinux4 ~]# vim /etc/httpd/conf.d/myweb.conf
<VirtualHost *:443>
SSLEngine on
SSLCertificateFile /etc/pki/tls/certs/server.crt
SSLCertificateKeyFile /etc/pki/tls/private/server.key
ServerName www.myweb.com
DocumentRoot /var/www/html/myweb
</VirtualHost>
```

6. Create a new directory in /var/www/html with the name myweb to hold website data and create index.html file.

```
[root@mlinux4 ~]# cd /var/www/html
[root@mlinux4 html]# mkdir myweb
[root@mlinux4 html]# cd myweb
[root@mlinux4 myweb]# vim index.html
<h1>My secure website with ssl</h1>
```

7. Start the service of httpd and make it enabled to be started at boot time automatically

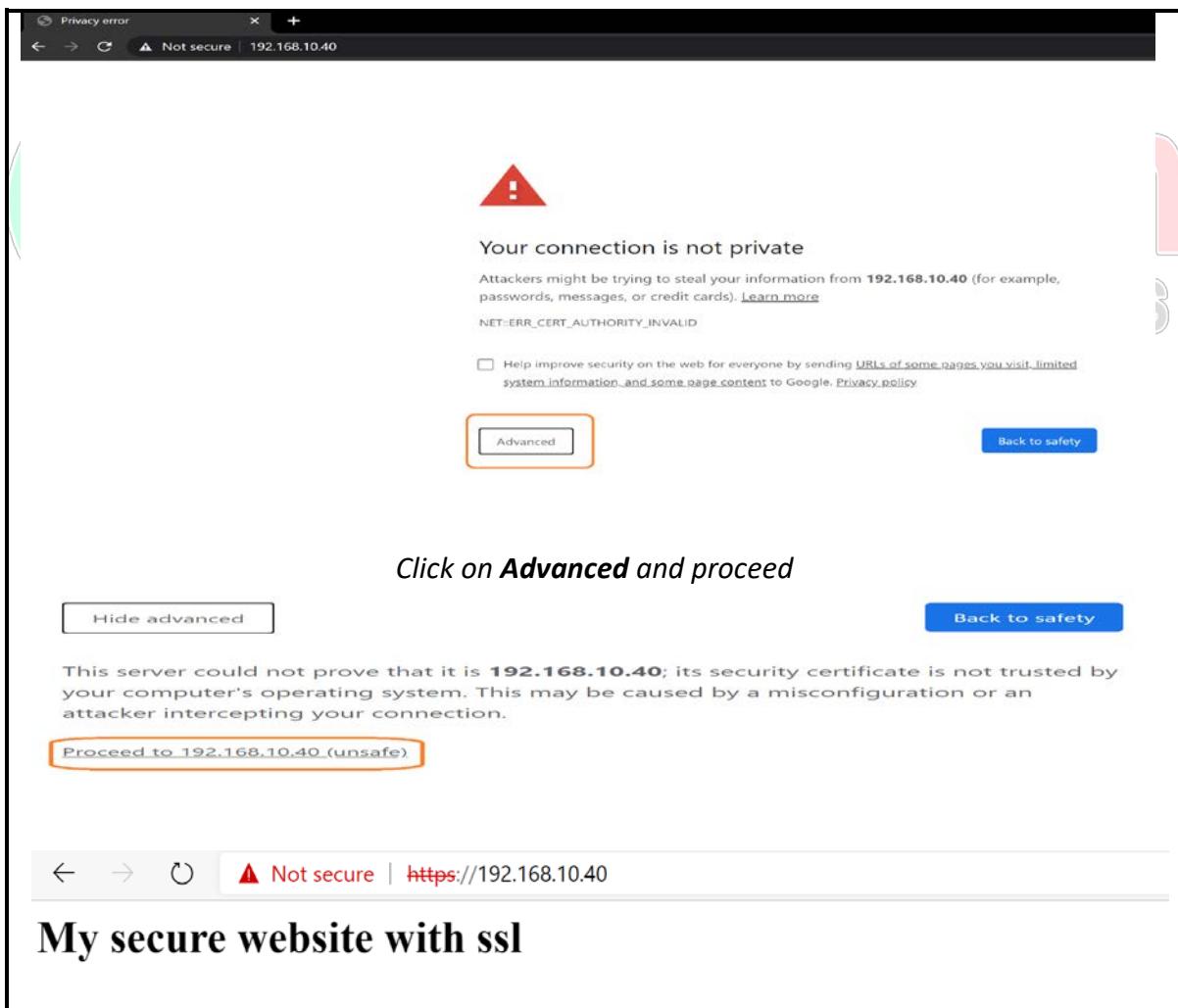
```
[root@mlinux4 ~]# systemctl enable httpd --now
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr
[root@mlinux4 ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset
   Active: active (running) since Wed 2020-09-23 22:16:19 IST; 4s ago
```

8. Add http https and or port 443 in firewall to allow access

```
[root@mlinux4 ~]# firewall-cmd --add-service=http --add-service=https --permanent
success
[root@mlinux4 ~]# firewall-cmd --reload
[root@mlinux4 ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3 enp0s8 enp0s9 team0
  sources:
  services: cockpit dhcpcv6-client dns ftp http https mountd nfs rpc-bind samba smtp ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

9. Try connecting from a browser using name or IP of the website

<https://192.168.10.40> or <https://www.myweb.com> (name can be used if dns or /etc/hosts file is updated on client side)



BONUS STUFF FOR SELF-LEARNING

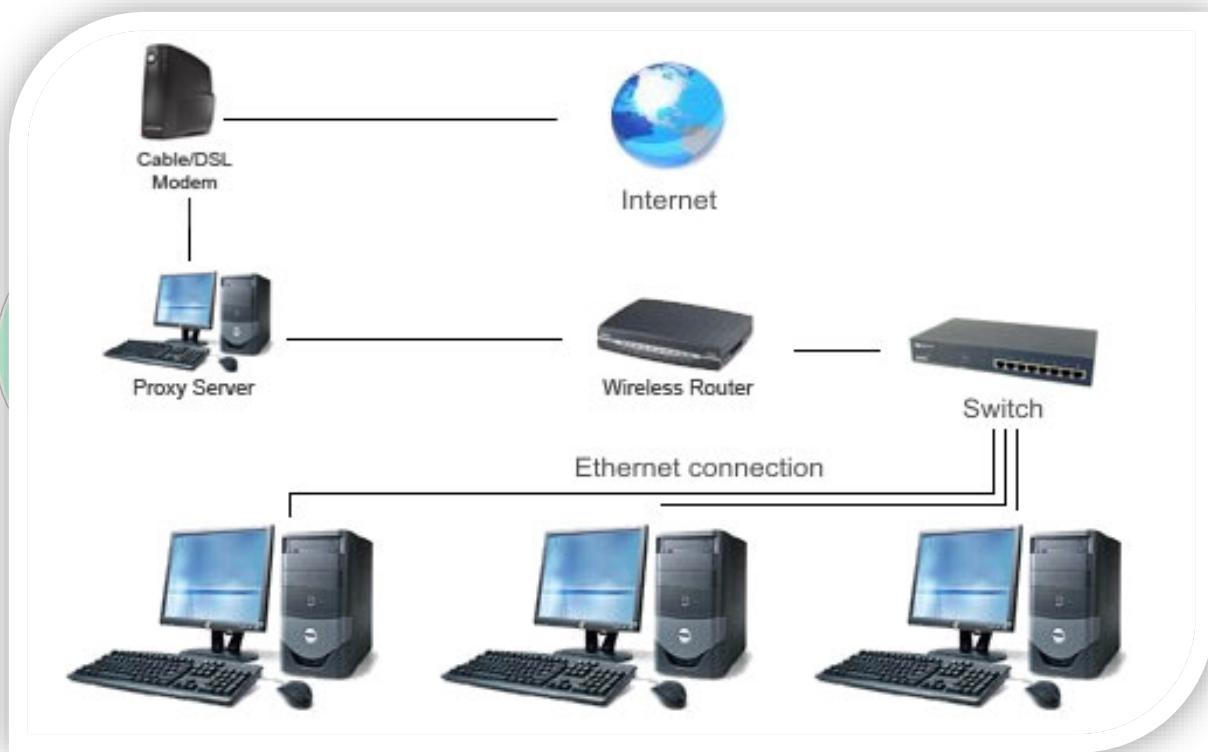


**For video tutorials of the following topics please visit my
blog <http://www.musab.in>**

<http://musab.in/2018/03/configuring-proxy-and-dhcp-on-centos-rhel7/>

SQUID PROXY SERVER

A proxy server is one that receives requests intended for another server and that acts on the behalf of the client (as the client proxy) to obtain the requested service. It is often used when the client and the server are incompatible for direct connection. For example, the client may be unable to meet the security authentication requirements of the server but may be required to access some services. It may also be used for screening purposes to enable the administrator to control access to undesirable sites. The proxy server may also be used for caching purposes, which enables faster access to frequently used websites. All the computers connected to the LAN access the Internet through a single IP address, resulting in improved security simply because the number of ports exposed is reduced.



Profile for Squid proxy server

Use	:	To Share Internet, Restrict unwanted websites.
Package	:	squid
Port	:	3128 (default)
Configuration Files	:	/etc/squid/squid.conf
Daemon	:	squid

Configuring a proxy server for internet sharing:

Step1: Check and Install the squid package

```
[root@ linux ~]# rpm -qa squid
[root@ linux ~]# yum install squid* -y
Loaded plugins: product-id, refresh-packagekit, security, subscription-manager
Updating certificate-based repositories.
ftp://192.168.10.96/pub/rhel6/repo/epel/repomd.xml: [Errno 14] PYCURL ERROR 7 - "
couldn't connect to host"
Trying other mirror.
Setting up Install Process
Resolving Dependencies
--> Running transaction check
--> Package squid.x86_64 7:3.1.10-1.el6_1.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====


| Package     | Arch   | Version            | Repository | Size  |
|-------------|--------|--------------------|------------|-------|
| Installing: |        |                    |            |       |
| squid       | x86_64 | 7:3.1.10-1.el6_1.1 | KTREPO     | 1.7 M |


Transaction Summary
```

```
[root@ adm ~]# rpm -qa squid
squid-3.1.10-1.el6_1.1.x86_64
```

Step2: Edit the configuration file for squid i.e. “/etc/squid/squid.conf”, Add the network range from where the clients can connect to proxy server.

#vim /etc/squid/squid.conf

```
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
#
acl mynet src 192.168.104.0/24
http_access allow mynet
```

Step3: Start the squid services and make it permanent

#systemctl start squid; systemctl enable squid (RHEL7)

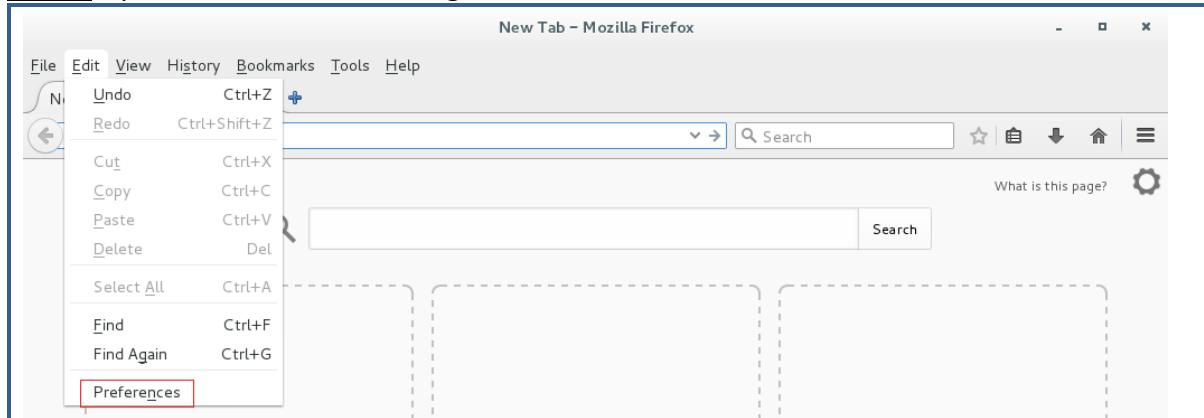
```
[root@myrhel73 ~]# systemctl start squid; systemctl enable squid
Created symlink from /etc/systemd/system/multi-user.target.wants/squid.service
[root@myrhel73 ~]#
```

Step4: Allow the squid service in firewall in RHEL7

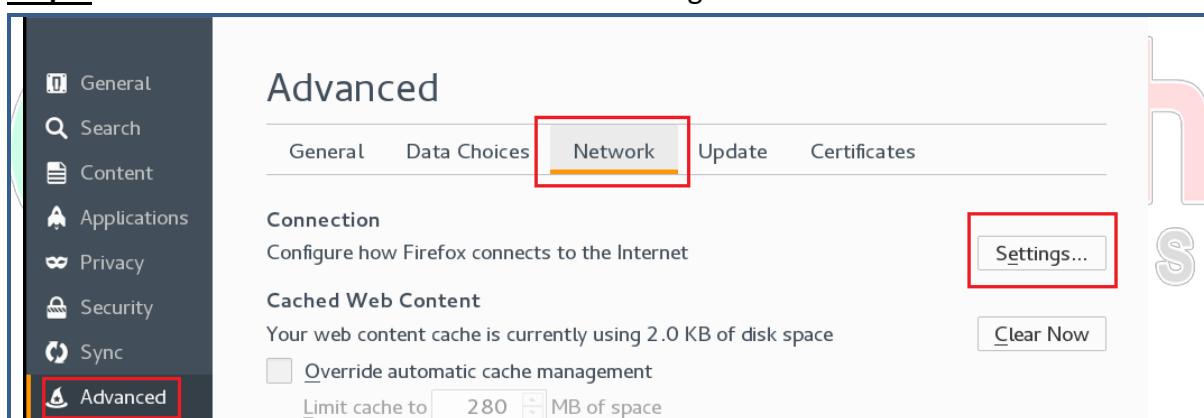
```
[root@myrhel73 ~]# firewall-cmd --add-service=squid --permanent
success
[root@myrhel73 ~]# firewall-cmd --reload
success
```

Client side configuration for receiving internet:

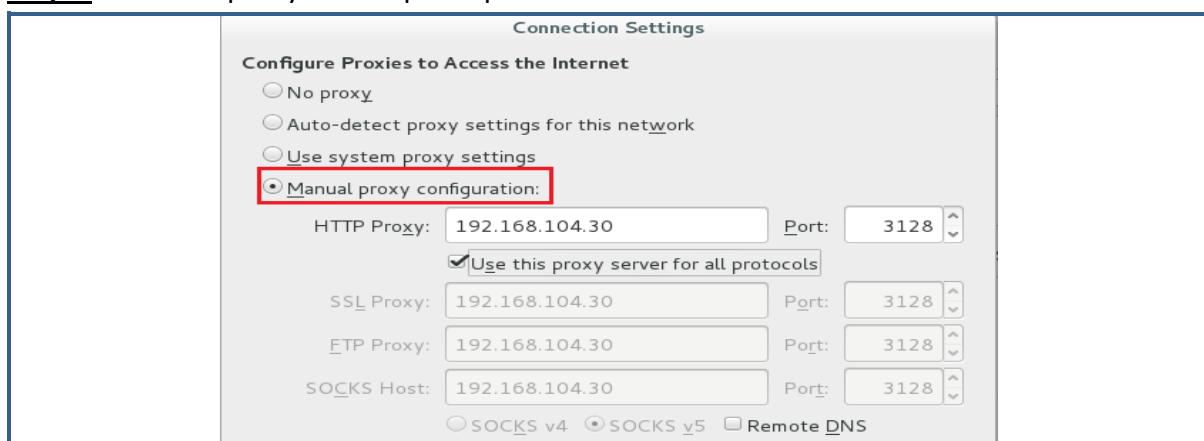
Step1: Open Browser, ex: firefox, go to edit/tool → Preferences



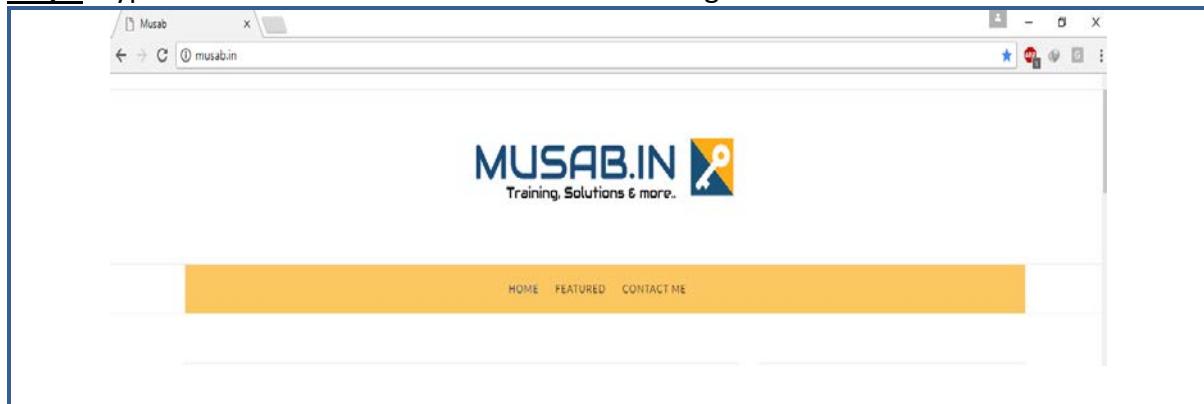
Step2: Go to Advanced → Network and select Settings



Step3: Enter the proxy server ip and port number as shown below



Step4: Type the website address to see if it connects



Blocking websites through proxy:-

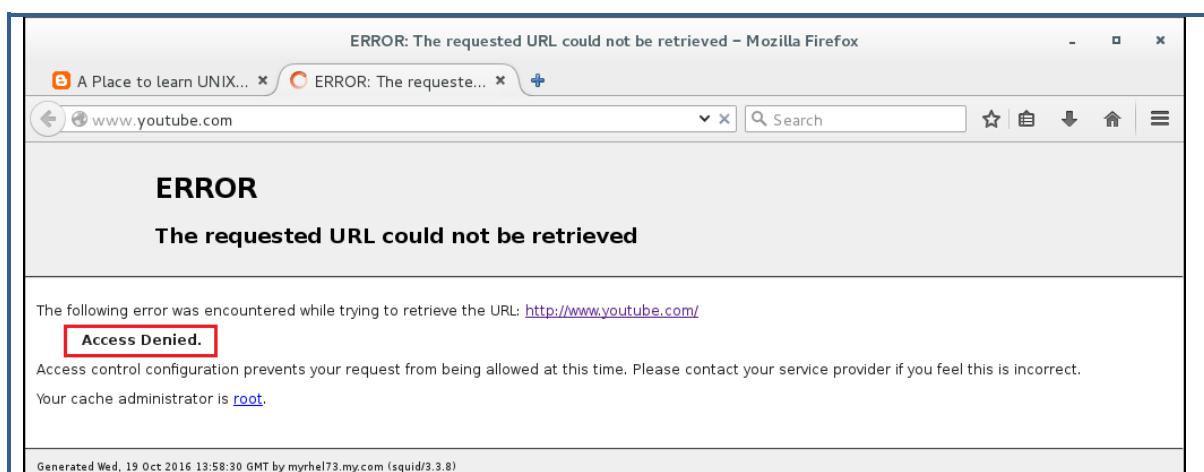
Step1: Go to the configuration file, **/etc/squid/squid.conf** and add the following parameters.

```
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
#
acl block url_regex youtube
http_access deny block
acl mynet src 192.168.104.0/24
http_access allow mynet
```

Step2: Restart/Reload the squid services (preferably reload)

```
[root@myrhel73 ~]# systemctl reload squid
[root@myrhel73 ~]#
```

Step3: Check with the browser can you access www.hotmail.com through your browser



Blocking multiple sites using proxy:

Step1: Create a file in /etc/squid with any name and add the phrase of the website, which you want to block

```
#vim /etc/squid/block
```

```
hotmail
youtube
facebook
twitter
yahoo
~
```

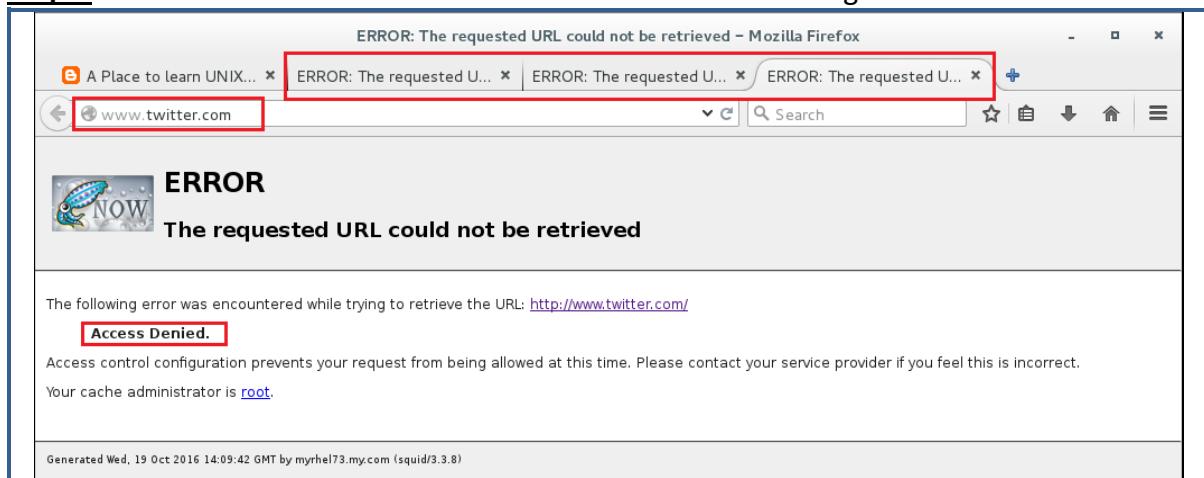
Step2: Add the same file info in configuration file, i.e., /etc/squid/squid.conf

```
#
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
#
acl block url_regex "/etc/squid/block"
http_access deny block
acl mynet src 192.168.104.0/24
http_access allow mynet
```

Step3: Restart/Reload the squid services (preferably reload)

```
[root@myrhel73 ~]# systemctl reload squid
[root@myrhel73 ~]#
```

Step4: Go to client browser and check whether the sites are being blocked



Changing the default port of Proxy:

Step1: By default the port no. for proxy is 3128, which can be changed by making a small change in the configuration file as shown below and change it to 8000.

```
[root@ktlinux2 ~]# vim /etc/squid/squid.conf
```

```
# Squid normally listens to port 3128
http_port 8000
```

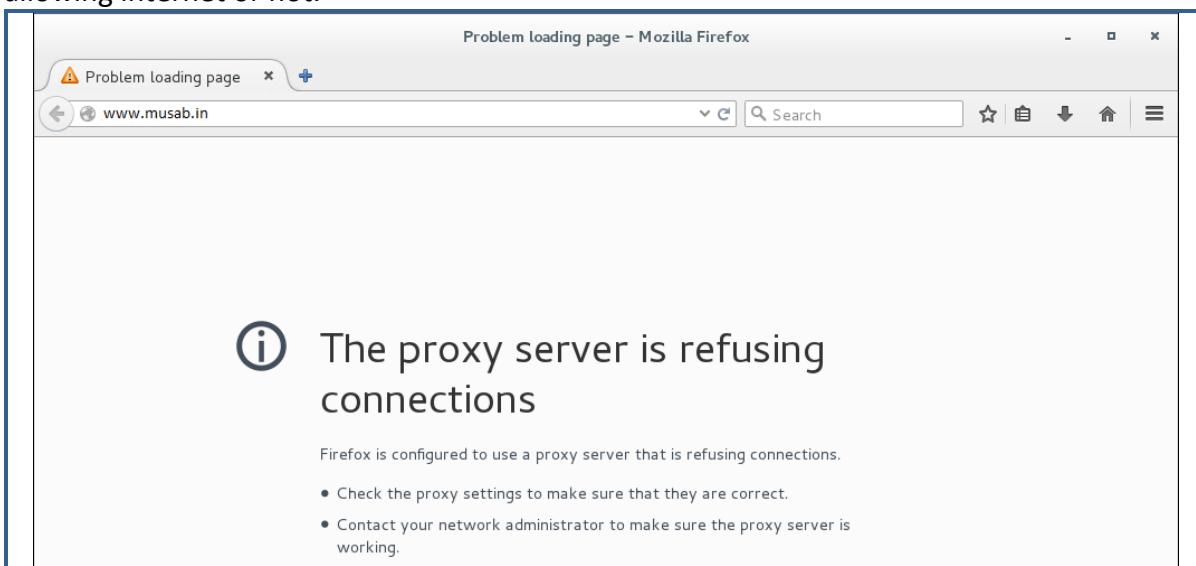
Step2: Restart/Reload the squid services (preferably reload)

```
[root@myrhe173 ~]# systemctl reload squid
[root@myrhe173 ~]#
```

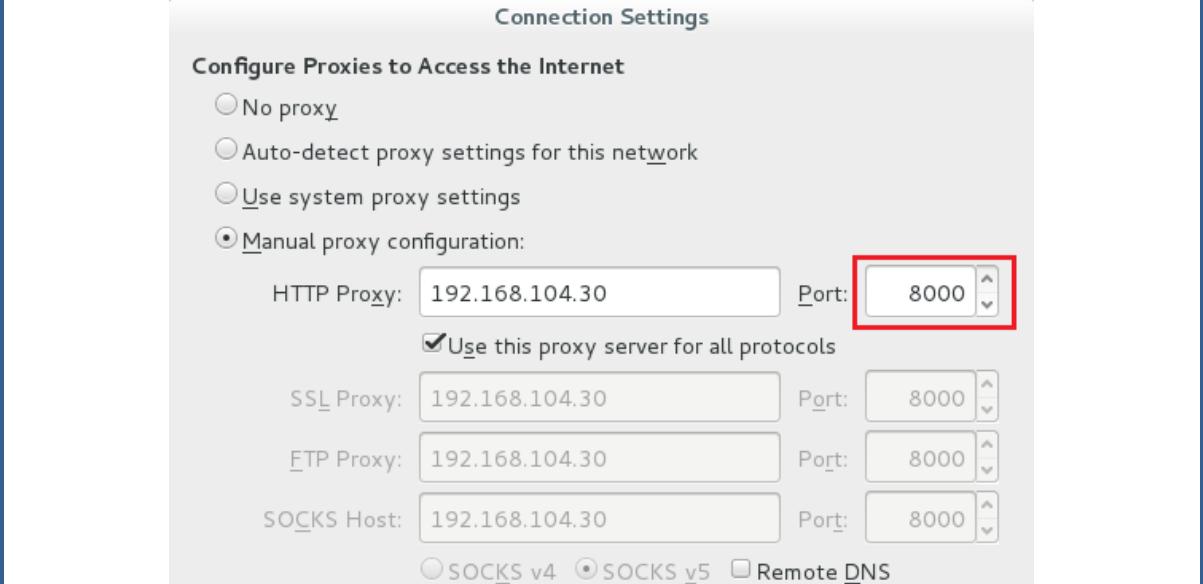
Step3: Allow the new port into firewall

```
[root@myrhe173 ~]# firewall-cmd --add-port=8000/tcp --permanent
success
[root@myrhe173 ~]# firewall-cmd --reload
success
[root@myrhe173 ~]# firewall-cmd --list-all
public (default, active)
  interfaces: enp0s3 enp0s8
  sources:
  services: dhcp dhcpcv6-client squid ssh
  ports: 8000/tcp
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

Step4: Go to client's browser and check whether with default port, i.e. 3128, whether it is allowing internet or not.



Step 4: change the port to 8000 and check whether internet is allowed or not.



Connection Settings

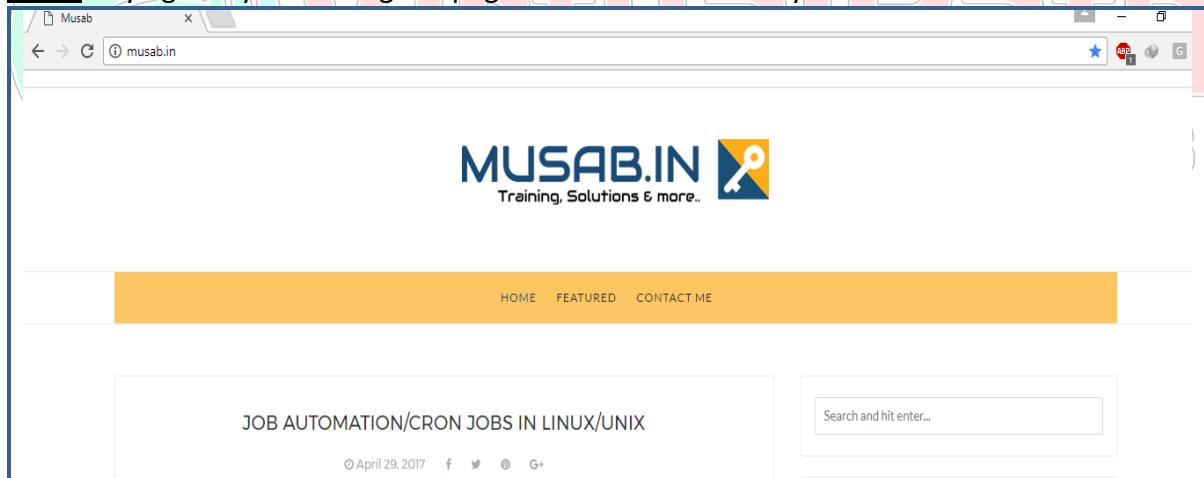
Configure Proxies to Access the Internet

- No proxy
- Auto-detect proxy settings for this network
- Use system proxy settings
- Manual proxy configuration:

HTTP Proxy:	<input type="text" value="192.168.104.30"/>	Port:	<input type="text" value="8000"/>
<input checked="" type="checkbox"/> Use this proxy server for all protocols			
SSL Proxy:	<input type="text" value="192.168.104.30"/>	Port:	<input type="text" value="8000"/>
FTP Proxy:	<input type="text" value="192.168.104.30"/>	Port:	<input type="text" value="8000"/>
SOCKS Host:	<input type="text" value="192.168.104.30"/>	Port:	<input type="text" value="8000"/>

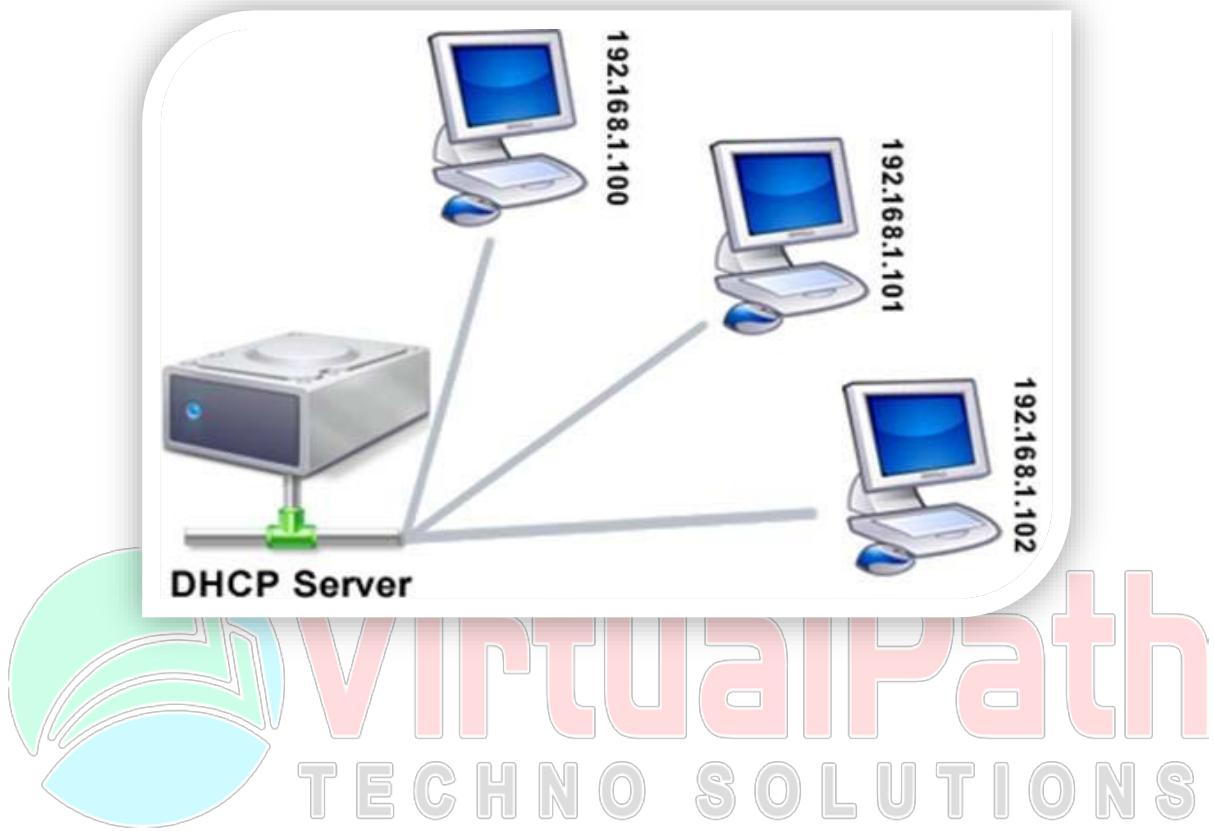
SOCKS v4 SOCKS v5 Remote DNS

Step6: Try again by refreshing the page and it would certainly work



Note: Squid Proxy is only the basic proxy, to learn more on proxy google for the third party tools like; **Squidguard, Untangle and Smoothwall**. There is lot to do with squid, try doing google and read the /etc/squid/squid.conf for more information.

DHCP SERVER



What is DHCP?

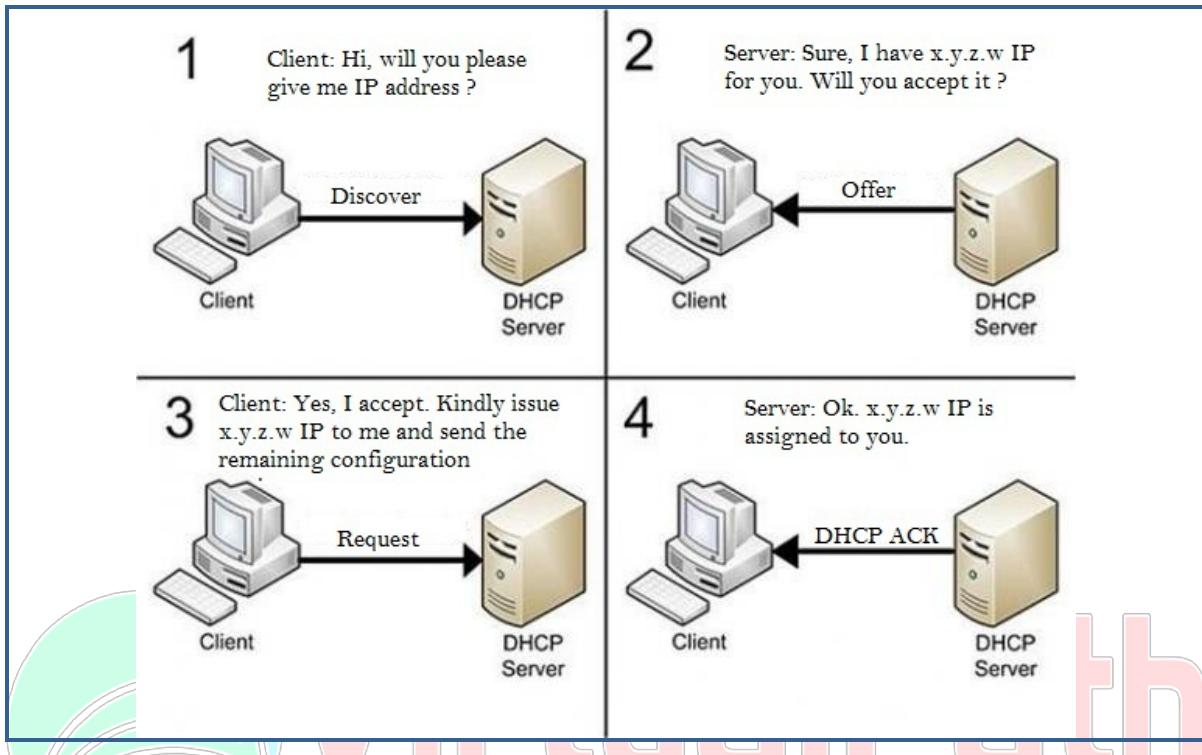
Dynamic Host Configuration Protocol (DHCP) is a network protocol that enables a server to automatically assign an IP address to a computer from a defined range of numbers (i.e., a scope) configured for a given network.

DHCP allows a computer to join an IP-based network without having a pre-configured IP address. DHCP is a protocol that assigns unique IP addresses to devices, then releases and renews these addresses as devices leave and re-join the network.

Internet service providers usually use DHCP to help customers join their networks with minimum setup effort required. Likewise, home network equipment like broadband routers offer DHCP support for added convenience in joining home computers to local area networks (LANs).

How does DHCP works?

The “DORA” process in DHCP



DHCP assigns an IP address when a system is started, for example:

1. A user turns on a computer with a DHCP client.
2. The client computer sends a broadcast request (called a **DISCOVER** or **DHCPODISCOVER**), looking for a DHCP server to answer.
3. The router directs the DISCOVER packet to the correct DHCP server.
4. The server receives the DISCOVER packet. Based on availability and usage policies set on the server, the server determines an appropriate address (if any) to give to the client. The server then temporarily reserves that address for the client and sends back to the client an **OFFER** (or **DHCPOFFER**) packet, with that address information. The server also configures the client's DNS servers, WINS servers, NTP servers, and sometimes other services as well.
5. The client sends a **REQUEST** (or **DCHPREQUEST**) packet, letting the server know that it intends to use the address.
6. The server sends an **ACK** (or **DHCPACK**) packet, confirming that the client has been given a lease on the address for a server-specified period of time.

When a computer uses a static IP address, it means that the computer is manually configured to use a specific IP address. One problem with static assignment, which can result from user error or inattention to detail, occurs when two computers are configured with the same IP address. This creates a conflict that results in loss of service. Using DHCP to dynamically assign IP addresses minimizes these conflicts.

Profile for DHCP server

Usage	:	To assign IP's to the computers in the network dynamically.
Package	:	Dhcp
Configuration file	:	/etc/dhcp/dhcpd.conf
Port no	:	67, 68
Daemon	:	dhcpd

Note: Pl watch the video tutorial to configure dhcp server on my website using following url

<http://musab.in/2018/03/configuring-proxy-and-dhcp-on-centos-rhel7/>

Configuring a DHCP server:

Step1: Check whether the package is installed or not

```
[root@ cl5 ~]# rpm -q dhcp
package dhcp is not installed
```

Step2: Install the package using yum,

```
[root@ cl5 ~]# yum install dhcp* -y
Loaded plugins: product-id, refresh-packagekit, security, subscription-manager
Updating certificate-based repositories.
Setting up Install Process
Package 12:dhcp-common-4.1.1-25.P1.el6.x86_64 already installed and latest version
Resolving Dependencies
--> Running transaction check
--> Package dhcp.x86_64 12:4.1.1-25.P1.el6 will be installed
--> Finished Dependency Resolution
```

Step3: Copy the example file for dhcp configuration over dhcp configuration file, i.e., </etc/dhcp/dhcpd.conf>

```
# cp -p /usr/share/doc/dhcp*/dhcpd.conf.sample /etc/dhcp/dhcpd.conf
```

Step4: Open the configuration file and edit it as per the requirement.

```
[root@ cl5 ~]# vim /etc/dhcp/dhcpd.conf

# A slightly different configuration for an internal subnet.
subnet 192.168.106.0 netmask 255.255.255.0 {
    range 192.168.106.10 192.168.106.30;
    option routers 192.168.106.1;
    default-lease-time 600;
    max-lease-time 7200;
}
```

Subnet	: The subnet of the network
Netmask	: The netmask of the network
Range	: The range of IP address to be assigned to the clients, in short "Scope"
Option routers	: gateway address (optional)
Default-lease-time	: The minimum lease time of the ip assigned to the clients
Max-lease-time	: The maximum lease time of the IP assigned to the clients

Step5: Make sure the dhcp server contains same range static IP as follows

```
[root@mlinux1 ~]# ifconfig ens3
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.106.81 netmask 255.255.255.0 broadcast 192.168.106.255
inet6 fe80::5054:ff:fe92:fd24 prefixlen 64 scopeid 0x20<link>
ether 52:54:00:92:fd:24 txqueuelen 1000 (Ethernet)
RX packets 6888 bytes 809163 (790.1 KiB)
RX errors 0 dropped 65 overruns 0 frame 0
TX packets 1950 bytes 283016 (276.3 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Step6: Start/Restart the dhcp services and make it permanent

```
[root@mlinux1 ~]# systemctl start dhcpcd; systemctl enable dhcpcd
```

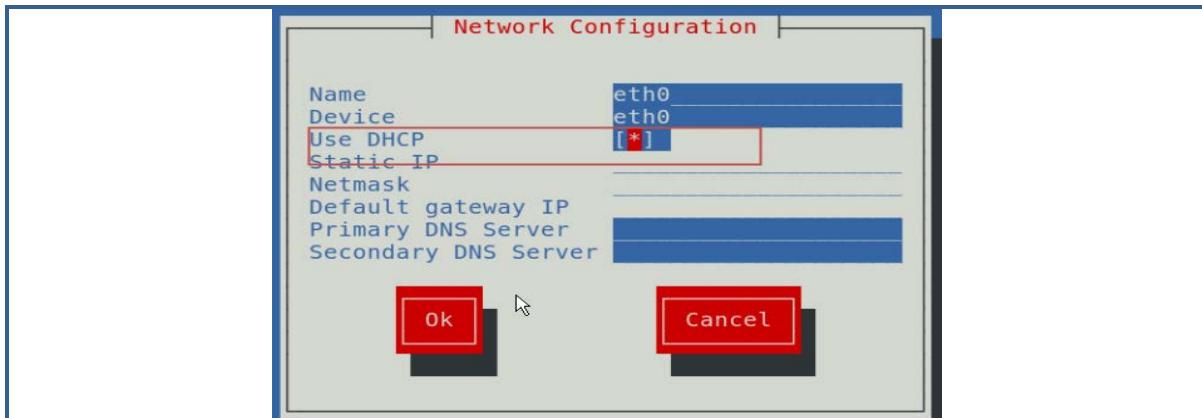
Step 7: Allow dhcp in firewall

```
[root@mlinux1 ~]# firewall-cmd --add-service=dhcp --permanent
success
[root@mlinux1 ~]# firewall-cmd --reload
success
```

Client side configuration for DHCP:

RHEL6 as a client

Step1: Make the dhcp option enabled in network configuration using #setup command.



Step2: Restart the network services and check the IP address is in dhcp scope.

```
[root@ cl6 Desktop]# setup
[root@ cl6 Desktop]# service network restart
Shutting down interface eth0: Device state: 3 (disconnected)
[ OK ]
Shutting down loopback interface: [ OK ]
Bringing up loopback interface: [ OK ]
```

Step3: Check the IP address using #ifconfig command

```
[root@ cl6 Desktop]# ifconfig eth0
eth0      Link encap:Ethernet HWaddr 52:54:00:5A:40:95
          inet addr:192.168.100.10 Bcast:192.168.100.255 Mask:255
                      inet6 addr: fe80::5054:ff:fe5a:4095/64 Scope:Link
                      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                      RX packets:141 errors:0 dropped:0 overruns:0 frame:0
                      TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
                      collisions:0 txqueuelen:1000
                      RX bytes:8914 (8.7 KiB) TX bytes:1730 (1.6 KiB)
                      Interrupt:11 Base address:0x4000
```

RHEL7 as a client

Step1: Make the dhcp option enabled in network configuration using *nmcli* command

```
#nmcli con mod ens#(connection name) ipv4.method auto (dhcp)
```

```
[root@mlinux1 ~]# nmcli con mod ens8 ipv4.method auto
```

Step2: activate the connection

```
#nmcli con up <con-name>
```

```
[root@mlinux1 ~]# nmcli con up ens8
Connection successfully activated (D-Bus active path:
[root@mlinux1 ~]#
```

Step3: Check the IP address using *#ifconfig* command

```
[root@mlinux1 ~]# ifconfig ens8
ens8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.106.14 netmask 255.255.255.0 broadcast 192.168.106.255
inet6 fe80::5054:ff:fe12:8546 prefixlen 64 scopeid 0x20<link>
ether 52:54:00:12:85:46 txqueuelen 1000 (Ethernet)
RX packets 6554 bytes 697510 (681.1 KiB)
RX errors 0 dropped 65 overruns 0 frame 0
TX packets 131 bytes 23863 (23.3 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```



Ansible Automation for Linux Admin

By Musab Syed

[REDHAT CERTIFIED]

VirtualPath Techno Solutions
Contact/whatsapp: +91 799 309 6092

Email: info@virtualpathtech.com
musabsyd@gmail.com

Website: www.virtualpathtech.com, www.musab.in



ANSIBLE AUTOMATION FOR LINUX ADMINIS LAB MANUAL:

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied.



Our Address:

VirtualPath Techno Solutions:

Phone or WhatsApp:+91 799 309 6092

Email: info@virtualpathtech.com

E-mail : musabsyd@gmail.com

website: www.virtualpathtech.com

website: www.musab.in

FOREWORD

Words to the Students

Though we have taken utmost efforts to present you this book error free, but still it may contain some errors or mistakes. Students are encouraged to bring, if there are any mistakes or errors in this document to our notice. So that it may be rectified in the next edition of this document.

This document provides good information on every topic and lab practices. This could become more effective if equally good practice is done. I urge the readers/students to do rigorous practice to polish your skill sets.

You can reach us on the following email address

info@virtualpathtech.com

musab@virtualpathtech.com

musabsyd@gmail.com

Go through our website and blog

www.virtualpathtech.com

www.musab.in

MUSAB.IN 
Training. Solutions & more.

OTHER COURSES AT VIRTUALPATH

LINUX 8 ADMINISTRATION

LINUX CLUSTERS

DEVOPS

AWS

OPENSTACK

IBM & EMC SAN

NETAPP &
NETAPP CLUSTER

VMWARE

ORACLE DBA, RAC
& GOLDEN GATE

IBM AIX, POWER
HA, LPAR VIO

SHELL SCRIPTING

And Many More...

Table of Contents

1. Introduction	05-06
2. Installation of ANSIBLE using EPEL repository	07-08
3. SETTING UP CONTROLLER/MASTER SERVER	09-10
4. UNDERSTANDING THE CORE COMPONENTS.....	11-19
5. Ansible Modules	20-23
6. Ansible plays and playbooks.....	24-24
7. ANSIBLE VARIABLES WITH YAML	25-35
8. ANSIBLE FACTS	36-40
9. CONFIGURING ANSIBLE MANAGED NODES.....	41-45
10. USEFUL SYSTEM ADMIN TASKS USING ANSIBLE PLAYBOOK.....	46-54



ANSIBLE AUTOMATION FOR LINUX ADMIN



ANSIBLE

INTRODUCTION:

According to the official definition on docs.ansible.com, Ansible is an IT automation tool. It can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments or zero downtime rolling updates.

Ansible's main goals are simplicity and ease-of-use. It also has a strong focus on security and reliability, featuring a minimum of moving parts, usage of OpenSSH for transport (with other transports and pull modes as alternatives), and a language that is designed around auditability by humans—even those not familiar with the program.

What it means is that it lets you manage software installations, updates, configurations and tasks within your environment. Managing a huge environment that have various LINUX, windows, and cloud and virtualization instances can be managed in an automated way.

Ansible was developed by **Michael DeHaan** and acquired by Red Hat in 2015. The latest version of ansible is 2.10 and stable version is 2.9

The other tools which are similar to ANSIBLE are chef, puppet, salt etc. The thing that makes ansible different from others is ANSIBLE is an agent less tool, which does not require any special agent to be installed and running on client sides. Usually ANSIBLE uses SSH to connect and manage UNIX/LINUX systems and WINDOWS remote management and using PowerShell execution. It is developed on python and most Ansible modules are written in Python, including the ones central to letting Ansible work.

Ansible is installed and configured on LINUX and it is used as central control station or master node. For non-linux users ansible came up with **Ansible Tower**, a web based UI tool to control ansible without the need of linux command line.

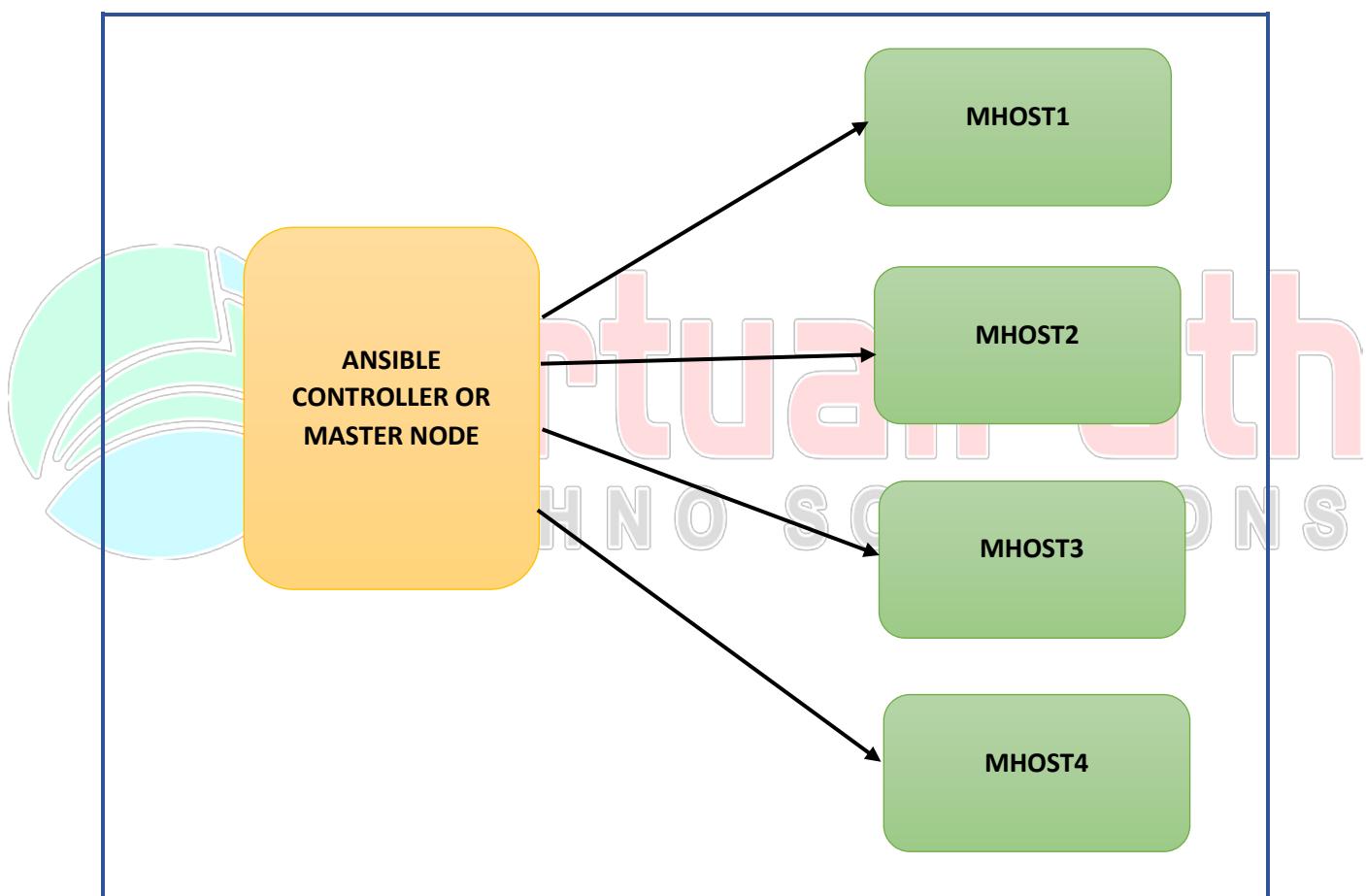


Installation of ANSIBLE using EPEL repository

To install ansible on a CENTOS OR Red Hat OS, we need to configure EPEL (Extra Package of Enterprise Linux) repository which provides ansible package.

Note: Make sure the machine is able to connect to internet.

In the labs we are going to work with RHEL 8.3 or CentOS 8.3
1 machine as control node and 4 machines as managed hosts.



On Controller node login and configure EPEL repository using following command.

```
# yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm -y
```

Or

```
# dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm -y
```

```
[root@ansible-c ~]# dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm -y
```

Check the status of repositories by using repolist command, make sure BaseOS and Appstream local repositories are also configured in addition to EPEL.

```
[root@ansible-c ~]# dnf repolist
repo id          repo name
AppStream        RHEL8_LOCALREPO-APPSTREAM
BaseOS          RHEL8_LOCALREPO-BASEOS
epel            Extra Packages for Enterprise Linux 8 - x86_64
epel-modular    Extra Packages for Enterprise Linux Modular 8 - x86_64
```

Check whether ansible is installed on this machine so far or not

```
[root@ansible-c ~]# rpm -q ansible
package ansible is not installed
```

As local and EPEL both repositories are configured, its time now to install ansible

```
[root@ansible-c ~]# dnf install ansible -y
Dependencies resolved.
=====
Package           Architecture Version       Repository
=====
Installing:
ansible          noarch      2.9.18-1.el8   epel
Installing dependencies:
libsodium         x86_64     1.0.18-2.el8   epel
python3-asn1crypto noarch     0.24.0-3.el8   BaseOS
python3-babel     noarch     2.5.1-5.el8   AppStream
python3-bcrypt    x86_64     3.1.6-2.el8.1 epel
python3-cffi      x86_64     1.11.5-5.el8  BaseOS
python3-cryptography x86_64  2.3-3.el8    BaseOS
python3-jinja2    noarch     2.10.1-2.el8_0 AppStream
python3-jmespath  noarch     0.9.0-11.el8  AppStream
python3-markupsafe x86_64     0.23-19.el8  AppStream
python3-pyasn1    noarch     0.3.7-6.el8  AppStream
python3-pyparser  noarch     2.14-14.el8  BaseOS
python3-pynaol   x86_64     1.3.0-5.el8  epel
sshpass          x86_64     1.06-9.el8   epel
Installing weak dependencies:
python3-paramiko noarch     2.4.3-1.el8   epel
```

Check ansible version and configuration details after installation

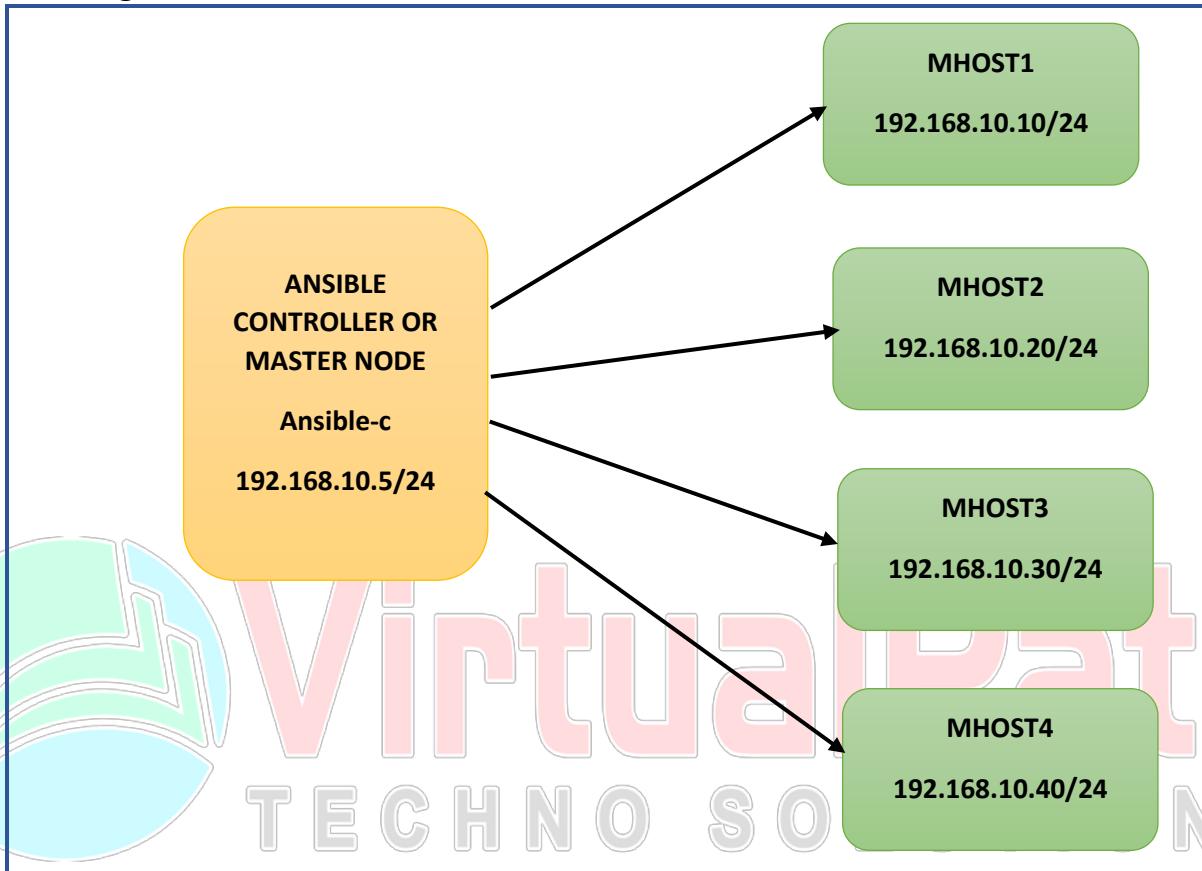
```
[root@ansible-c ~]# rpm -q ansible
ansible-2.9.18-1.el8.noarch

[root@ansible-c ~]# ansible --version
ansible 2.9.18
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules',
  '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.6.8 (default, Aug 18 2020, 08:33:21) [GCC 8.3.1 20191121 (Red Hat 8.3.1-5)]
```

It's confirmed now that ansible is installed perfectly on master/controller node.

SETTING UP CONTROLLER/MASTER SERVER

In this course we are going to use 1 machine as controller node or master with 4 managed nodes/hosts.



Configure /etc/hosts with all nodes IP and hostname for name resolution
#vim /etc/hosts

```

192.168.10.5 ansible-c.example.com master
192.168.10.10 mhost1.example.com mhost1
192.168.10.20 mhost2.example.com mhost2
192.168.10.30 mhost3.example.com mhost3
192.168.10.40 mhost4.example.com mhost4
  
```

To verify the resolution use getent or ping command

```

[root@ansible-c ~]# getent hosts mhost1
192.168.10.10 mhost1.example.com mhost1

[root@ansible-c ~]# getent hosts mhost2
192.168.10.20 mhost2.example.com mhost2

[root@ansible-c ~]# getent hosts
  
```

SETTING UP SSH BASED PASSWORD LESS LOGIN:

- Setup password less login or SSH based trust relationship from control node and managed hosts.
- To setup SSH based password logins, either you can copy id_rsa.pub key manually or using a “**for loop**” to automate your work.
- First of all we need to generate SSH Key pair as root user using **ssh-keygen** command

```
[root@ansible-c ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:mK1/1PY2ccGClu9a+RpPnsoCG6W3tdbkdwxY4PgkPyg root@ansibl
The key's randomart image is:
+---[RSA 3072]----+
|                               |
|                               |
|                               |
|                               = o |
| + B + + |
| + S= B + . |
| . E = O .+ |
| . . B =+O.o |
| . o ooB*o.+ |
| .. .=++=..|
+--- [SHA256]-----+
```

Copying SSH public keys on all managed hosts using *for loop*

```
[root@ansible-c ~]# ls .ssh
id_rsa id_rsa.pub
[root@ansible-c ~]# for i in 1 2 3 4
> do
> ssh-copy-id mhost$i
> done
```

Now check ssh connectivity to managed hosts

```
[root@ansible-c ~]# ssh mhost1
[root@mhost1 ~]#
[root@mhost1 ~]#exit
[root@ansible-c ~]#
```

Do the same for all clients

UNDERSTANDING THE CORE COMPONENTS OF ANSIBLE CONFIGURATION

Introducing Ansible core components

- **Ansible Configuration Files** - Defines various default values to be used by Ansible. Almost all default values can be overwritten in ansible ad-hoc commands and playbooks.
- **Ansible Inventories** - Defines hosts and groups of hosts on which ansible operates.
- **Ansible Modules** - Module is small python program which is designed to execute specific task on ansible managed nodes or local node.
- **Ansible Variables** - Variables used in playbook.
- **Ansible Facts** - Represent data about remote systems gathered by Ansible which can be used in playbooks as variables. Ansible facts are also very important for conditional playbook execution.
- **Ansible Plays** - Ansible play basically defines target hosts and specific task(s) to be executed on target hosts. Ansible play is written in YAML.
- **Ansible Playbooks** - List of ansible plays

Ansible Config file:

- Different Ansible settings can be configured using Ansible configuration file named as ansible.cfg.
- When installing ansible using package manager (dnf or yum) , configuration file is found at path /etc/ansible which is system wide directory.
- Ansible configuration file can be present at multiple locations as described below and they are searched in same order.

ANSIBLE_CONFIG	Environmental variable
ansible.cfg	In current working directory
~/.ansible.cfg	User's home directory
/etc/ansible/ansible.cfg	System wide default directory

Ansible searches for configuration file in the above order. Config file found first is considered and others are simply ignored

Lab Exercise:

1. Create a user ansible and logged in as ansible user on control node

```
[root@ansible-c ~]# useradd ansible
[root@ansible-c ~]# echo redhat |passwd ansible --stdin
Changing password for user ansible.
passwd: all authentication tokens updated successfully.
[root@ansible-c ~]#
[root@ansible-c ~]# su - ansible
[ansible@ansible-c ~]$ whoami
Ansible
```

2. Check ansible config file status #ansible --version

```
[ansible@ansible-c ~]$ ansible --version
ansible 2.9.18
  config file = /etc/ansible/ansible.cfg
  configured module search path      =      ['/home/ansible/.ansible/plugins/modules',
 '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.6.8 (default, Aug 18 2020, 08:33:21) [GCC 8.3.1 20191121 (Red Hat 8.3.1-5)]
```

3. Create a file in /tmp with name ansible.cfg

Create a file in /home/ansible with name ansible.cfg

Finally create a hidden file /home/ansible with a name .ansible.cfg

Export a variable ANSIBLE_CONFIG=/tmp/ansible.cfg

```
[ansible@ansible-c ~]$ whoami
ansible
touch /tmp/ansible.cfg

[ansible@ansible-c ~]$ pwd
/home/ansible

[ansible@ansible-c ~]$ touch ansible.cfg
[ansible@ansible-c ~]$ touch .ansible.cfg

[ansible@ansible-c ~]$ export ANSIBLE_CONFIG=/tmp/ansible.cfg

[ansible@ansible-c ~]$ ansible --version
ansible 2.9.18
  config file = /tmp/ansible.cfg
  configured module search path      =      ['/home/ansible/.ansible/plugins/modules',
 '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.6.8 (default, Aug 18 2020, 08:33:21) [GCC 8.3.1 20191121 (Red Hat 8.3.1-5)]
```

4. Un-export the variable and check the status of config, delete each file created and check the status of config file escalation.

```
[ansible@ansible-c ~]$ ansible --version
ansible 2.9.18
  config file = /tmp/ansible.cfg
  configured module search path = ['/home/ansible/.ansible/plugins/modules'],
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.6.8 (default, Aug 18 2020, 08:33:21) [GCC 8.3.1 20191121]
[ansible@ansible-c ~]$
[ansible@ansible-c ~]$ rm -f /tmp/ansible.cfg
[ansible@ansible-c ~]$
[ansible@ansible-c ~]$ ansible --version
ansible 2.9.18
  config file = /home/ansible/ansible.cfg
  configured module search path = ['/home/ansible/.ansible/plugins/modules'],
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.6.8 (default, Aug 18 2020, 08:33:21) [GCC 8.3.1 20191121]

[ansible@ansible-c ~]$ rm -f ansible.cfg
[ansible@ansible-c ~]$ ansible --version
ansible 2.9.18
  config file = /home/ansible/.ansible.cfg
  configured module search path = ['/home/ansible/.ansible/plugins/modules'],
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.6.8 (default, Aug 18 2020, 08:33:21) [GCC 8.3.1 20191121]

[ansible@ansible-c ~]$ rm -f .ansible.cfg
[ansible@ansible-c ~]$ ansible --version
ansible 2.9.18
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/ansible/.ansible/plugins/modules'],
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.6.8 (default, Aug 18 2020, 08:33:21) [GCC 8.3.1 20191121]
```

Note: check /etc/ansible/ansible.cfg file for more information

Some Important Default Settings of Ansible

Below is list of important defaults which are used by Ansible if we don't define them in ansible.cfg file considered by Ansible.

[Defaults]

Option /Directive Name	Default Value/Behaviour	Explanation
inventory	/etc/ansible/hosts	Default inventory file, ansible looks for this file by default
remote_port	22	By default SSH port 22 is used to connect, can be changed to desired if required
forks	5	By default, Ansible starts 5 forks or actions at a time, this can also be changed
roles_path	/etc/ansible/roles	Ansible looks for roles in this directory

gathering	Implicit	By default, Facts are always gathered unless we set this directive to explicit in config file.
module_name	Command	Default module is command module which can be changed to some other module.

Sample config file for a specific user

A different settings can be set as per user requirement using ansible.cfg

Let's say

- Location of ansible inventory file
- Location of roles directory
- User to be used to connect to remote hosts
- Enabling/Disabling privilege escalation and other settings

A sample config file looks like this

```
vim /home/ansible/tasks/ansible.cfg
[defaults]
inventory = /home/ansible/tasks/hosts
roles_path = /home/ansible/tasks/roles
remote_user = ansible

[privilege_escalation]
become = yes
become_user = root
become_method = sudo
become_ask_pass = False
```

ANSIBLE INVENTORIES

- Ansible inventory file defines the hosts to be managed by ansible. Different managed hosts can be grouped together in different groups.
- Ansible ad-hoc commands and playbooks are executed on inventory hosts defined on command line and in playbook, respectively.
- Host and group specific variables are also defined in inventory file.
- Ansible inventory file default path is defined in ansible configuration file.
- Ansible inventory file can be defined in INI, YAML or JSON format. Most used format is INI format.
- Ansible inventory file name is not a standard name, you can define any file in ansible config file to be used as ansible inventory e.g. hosts or inventory or mhosts.

An Example of Ansible Inventory file in INI format

```
vim /home/ansible/tasks/hosts
```

```
[mygroup1]
```

```
host1
```

```
host2
```

```
192.168.10.10
```

```
host1.example.com
```

```
host[3:5]
```

```
[mygroup2]
```

```
webserver.example.com
```

```
db.example.com
```

Lab Exercise:

1. Add the all the hosts name in default inventory file /etc/ansible/hosts

```
[root@ansible-c ~]# vim /etc/ansible/hosts
```

```
## [dbservers]
```

```
##
```

```
## db01.intranet.mydomain.net
```

```
## db02.intranet.mydomain.net
```

```
## 10.25.1.56
```

```
## 10.25.1.57
```

```
# Here's another example of host ranges, this time there are no
# leading 0s:
```

```
## db-[99:101]-node.example.com
```

```
[mygroup]
```

```
mhost1
```

```
mhost2
```

```
mhost3
```

```
mhost4
```

```
:wq!
```

2. Now run ansible command list hosts from 'mygroup' group

```
[root@ansible-c ~]# ansible mygroup --list-hosts
```

```
hosts (4):
```

```
 mhost1
```

```
 mhost2
```

```
 mhost3
```

```
 mhost4
```

3. Re-edit the /etc/ansible/hosts file in the form of range

```
root@ansible-c ~]# vim /etc/ansible/hosts
```

```
## db-[99:101]-node.example.com
```

```
[mygroup]
```

```
mhost[1:4]
```

```
:wq!
```

4. Now check the hosts list in different ways as shown below

```
[root@ansible-c ~]# ansible mygroup --list-hosts
```

hosts (4):

- mhost1
- mhost2
- mhost3
- mhost4

```
[root@ansible-c ~]# ansible my* --list-hosts
```

hosts (4):

- mhost1
- mhost2
- mhost3
- mhost4

5. Let's do a ping test to check connectivity to all hosts using ping module

```
[root@ansible-c ~]# ansible mygroup -m ping or [root@ansible-c ~]# ansible my* -m ping
```

```
mhost1 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/libexec/platform-python"
  },
  "changed": false,
  "ping": "pong"
}
mhost4 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/libexec/platform-python"
  },
  "changed": false,
  "ping": "pong"
}
mhost2 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/libexec/platform-python"
  },
  "changed": false,
  "ping": "pong"
}
mhost3 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/libexec/platform-python"
  },
  "changed": false,
  "ping": "pong"
}
```

To see condensed one liner output

```
[root@ansible-c ~]# ansible mygroup -m ping -o
mhost1 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/libexec/platform-python"}, "changed": false, "ping": "pong"}
mhost3 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/libexec/platform-python"}, "changed": false, "ping": "pong"}
mhost4 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/libexec/platform-python"}, "changed": false, "ping": "pong"}
mhost2 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/libexec/platform-python"}, "changed": false, "ping": "pong"}
```

6. Edit the config file as shown below and observe the results of ping test

```
[root@ansible-c ~]# vim /etc/ansible/ansible.cfg
[defaults]

# some basic default values...

#inventory      = /etc/ansible/hosts
#library        = /usr/share/my_modules/
#module_utils   = /usr/share/my_module_utils/
#remote_tmp     = ~/.ansible/tmp
#local_tmp      = ~/.ansible/tmp
#plugin_filters_cfg = /etc/ansible/plugin_filters.yml
forks          = 1
#poll_interval  = 15
#sudo_user      = root
#ask_sudo_pass  = True
#ask_pass        = True
#transport      = smart
#remote_port    = 22
#module_lang    = C
#module_set_locale = False

[root@ansible-c ~]# ansible mygroup -m ping -o
mhost1 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/libexec/platform-python"}, "changed": false, "ping": "pong"}
mhost3 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/libexec/platform-python"}, "changed": false, "ping": "pong"}
mhost4 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/libexec/platform-python"}, "changed": false, "ping": "pong"}
mhost2 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/libexec/platform-python"}, "changed": false, "ping": "pong"}
```

Note: This time ping test will be one at a time, due to fork value =1

7. Restore fork value back to 5, comment the line, re-test the ping test and observe the result in output
8. Same way change the remote_port = 222 and see whether it will be able to connect remote hosts in ping command or not. Restore back to original value and comment back after test.

Ansible Host and Group variables (Example)

```
vim /home/ansible/tasks/nodes
[mygroup1]
host1 ansible_user=ansible ansible_port=222
host2
192.168.10.10
host1.xyz.com
host[3:5]

[mygroup2]
webserver.xyz.com
db.xyz.com

[mgroup2:vars]
ansible_user=ansible
```

Lab Exercise:

1. Create host groups and test with ansible command

```
[root@ansible-c ~]# vim /etc/ansible/hosts
[mygroup1]
mhost1 ansible_user=ansible ansible_port=222
mhost2

[mygroup2]
mhost[3:4]

[mygroup2:vars]
ansible_user=ansible

:wq!
```

Perform ping test with all hosts

```
[root@ansible-c ~]# ansible mygroup1 -m ping -o
mhost1 | UNREACHABLE!: Failed to connect to the host via ssh: ssh: connect to host mhost1 port 222: No route to host
mhost2 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/libexec/platform-python"}, "changed": f
}
```

Observe mhost1 is trying to connect via port 222

Go back to hosts file and remove ansible_port variable

```
[root@ansible-c ~]# ansible mygroup1 -m ping -o
mhost1 | UNREACHABLE!: Failed to connect to the host via ssh: ansible@mhost1: Permission denied
,password).
mhost2 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/libexec/platform-p
}
```

Observe mhost1 is trying to connect with ansible user

Now test mygroup2 and observe group vars is working

```
[root@ansible-c ~]# ansible mygroup2 -m ping -o
mhost4 | UNREACHABLE!: Failed to connect to the host via ssh: ansible@mhost4: Permission denied
,password).
mhost3 | UNREACHABLE!: Failed to connect to the host via ssh: ansible@mhost3: Permission denied
,password).
```

Change group variable value to root user and ping test again

```
[root@ansible-c ~]# vim /etc/ansible/hosts
[mygroup2:vars]
ansible_user=root

[root@ansible-c ~]# ansible mygroup2 -m ping -o
mhost4 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/libexec/platform-
python"}, "changed"
mhost3 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/libexec/platform-
python"}, "changed"
```

Organizing Host and Group Variables

Up till now we have learnt how we can mention host and group variables in inventory files. In this chapter we will learn we can organize hosts and group variables by putting them in a separate files for specific host and group.

Organizing variables is a cleaner approach when you have many hosts (or host groups) in inventory file. To organize them, we must create directories with name **host_vars** and **group_vars** at the same path where inventory file is present.

To define host vars:

Create file with same as that of host (for example mhost1) under host_vars directory and specify host vars specific to that host in this file.

For Example:

```
vim /etc/ansible/hosts/host_vars/mhost1 (Valid extensions are .yml,.yaml, .json and  
no extension)  
---  
ansible_user: ansible  
ansible_port: 222  
:wq
```

To define group vars:

Create file with same name as that of group name and specify group vars in this file.

Note: You must create directories with standard names host_vars and group_vars to specify host and group specify variables. Also, you must create these directories at path where inventory file is present.

For more information
visit https://docs.ansible.com/ansible/2.3/intro_inventory.html

ANSIBLE MODULES

Modules (also referred to as “task plugins” or “library plugins”) are discrete units of code that can be used from the command line or in a playbook task. Ansible executes each module, usually on the remote target node, and collects return values. Basically, a module is a command or set of similar Ansible commands meant to be executed on the client-side.

Commonly used Modules are:

Most used modules are:

- ping
- setup
- yum
- yum_repository
- service
- package
- systemd
- cron
- lvg
- lvol
- parted
- template
- fetch
- register
- user
- group
- authorized_key
- lineinfile
- firewalld
- nmcli
- sefcontext
- selinux
- command
- file
- filesystem
- mount
- debug
- and many others

To check details about any module use **#ansible-doc -l**

- To view only windows related modules

```
#ansible-doc -l |grep ^win
```

- To view non-windows modules

```
#ansible-doc -l |grep -v ^win
```

- To view file modules to manipulate files

```
#ansible-doc -l |grep -v ^win |grep ^file
```

Or

```
#ansible-doc -l |grep ^file
```

- To copy files modules

```
#ansible-doc -l |grep ^copy (same way go for lvol, yum, dnf etc)
```

- To view detailed help about a module

```
#ansible-doc -l <module name> (for e.g ping, copy, file, yum etc)
```

To get detailed information about a module

```
#ansible-doc <module name>
```

e.g.

```
#ansible-doc ping
```

```
#ansible-doc copy
```

```
#ansible-doc user
```

Ansible ad-hoc Commands

Ansible ad-hoc commands are the simplest way to execute some command on a bunch of servers. Ad-hoc commands are not stored for future uses but represent a fast way to interact with the desired servers. For example -To Shutdown/Restart remote host or to quickly verify some configuration.

Syntax of ad-hoc command is:

```
#ansible [pattern] -m [module] -a “[module options]”
```

Examples of some ad-hoc commands are shown below:

ansible mhost1 -m command -a “cat /etc/hosts”	To display hosts file of mhost1
ansible mhost2 -m copy -a “src=/etc/hosts dest=/tmp/hosts_bkp”	To copy hosts file to /tmp/hosts_bkp file on mhost2
ansible mhost3 -m file -a “path=/tmp/test state=touch”	To create test file on mhost3
ansible mhost4 -m user -a “name=ansible state=present”	To create user ansible on mhost4

We must keep in mind ad-hoc command would be successfully executed if user we are using has permissions to perform that action. For example, normal user cannot create user, so we must execute command to create user as root user.

If default remote_user is normal user as defined in ansible.cfg, the use --become flag to execute command on remote host with sudo provided remote host must be configured to allow to use sudo for this user.

```
ansible mhost4 -m user -a "name=ansible state=present" --become
```

Lab Exercise:

Make sure that ansible hosts file has proper hosts added as shown below

```
[root@ansible-c ~]# tail /etc/ansible/hosts
```

```
# Here's another example of host ranges, this time there are no
# leading 0s:
```

```
## db-[99:101]-node.example.com
[mygroup1]
mhost[1:2]

[mygroup2]
mhost[3:4]
```

Check connectivity using ansible all -m ping command before proceeding

1. Reading a file on mhost1 using ansible ad-hoc command

```
#ansible mhost1 -m command -a "cat /etc/hosts"
```

2. To read a particular file on all managed hosts

```
#ansible all -m command -a "cat /etc/fstab"
```

3. To read a particular file on all nodes, with 1 fork at a time

```
# ansible all -m command -a "cat /etc/fstab" -f 1
```

4. To copy a file/dir on a specific host

```
#ansible mhost2 -m copy -a "src=/etc/hosts dest=/tmp/hosts_bkp"
```

Note: Repeat one more time and check status color (as it is already copied, it won't overwrite it)

5. Creating a new file on a specific host

Check if file is there or not

```
#ansible mhost3 -m file -a "path=/tmp/test state=file" (for dir state=dir)
```

6. Creating a file

```
#ansible mhost3 -m file -a "path=/tmp/test state=touch"
```

7. Check the presence of file

```
#ansible mhost3 -m file -a "ls -l /tmp/test" (or -a "cat /tmp/test")
```

8. Creating a user on a specific host

```
#ansible mhost4 -m user -a "name=test state=present" (to remove  
state=absent)
```

9. To check user details

```
#ansibe mhost4 -m user -a "id test" Or #ansible mhost4 -a "id test"
```

Note: Always use **#ansible-doc <module name>** for details to use that module



ANSIBLE PLAYS AND PLAYBOOKS

Ansible playbook is a set of plays you want to execute on remote machines. A play consist of the tasks we want to perform on client machine(s). A typical playbook will have instructions like, hosts on which the play has to be performed with what user privilege and tasks to be performed using different modules on remote clients.

If Ansible modules are the tools in your workshop, playbooks are your instruction manuals, and your inventory of hosts are your raw material. Ansible playbooks are written in YAML format which is human readable and this makes ansible easy to use and understand than other programming languages.

A typical playbook consists of following elements

Hosts	To define target of play and other directives like remote_user ,become, gather_facts etc
Vars	To define variables
Tasks	To define list of tasks to be executed
handlers	To define some task to be executed on successful change due to some other task in tasks section
roles	Roles to be included in the play

Ansible Playbook format example in YAML

```

---
- hosts: mhost1
  become: True
  tasks:
    - name: Adding firewall rule
      firewalld:
        port: 80/tcp
        state: enabled
        permanent: yes
      notify: Reload firewall
  handlers:
    - name: Reload firewall
      systemd:
        name: firewalld
        state: reloaded
...

```

ANSIBLE VARIABLES WITH YAML

Variable in general is the name given to a set of data, whenever in a script or in general admin tasks we need to repeatedly recall a set of data, we create a variable with some name storing some data inside it. Whenever we need to call the data we'll use the variable name.

- YAML stands for “YAML Ain’t Markup Language”
- Ansible playbooks are written in YAML. YAML supports dictionary, list and complex variables using dictionaries and lists both.
- Valid variable names should start with letter and can contain only underscore
- Dictionary maps keys to values
- In general, most things in a YAML file are a form of key-value pair where the key represents the pair’s name and the value represents the data linked to that name.

A simple dictionary variable example:

vars:

key: value

The defined variables in playbooks can be called using Jinja2 templating system. For example, we can reference the variable defined above in playbook with jinja2 expression “{{ key }}”.

We can display the variable using debug module in simple playbook to understand this.

Lab Exercise:

Run **#ansible-doc debug** to let more about debug module to use it in a playbook.

1. Creating a simple playbook to print a message

```
[root@ansible-c ~]# vim vars.yml
---
- hosts: mhost1
  vars:
    sub: ansible
  tasks:
    - name: Printing the value of a variable
      debug:
        msg: The value of the key sub is " {{ sub }}"
...
```

Let's check the playbook for any syntax errors

```
[root@ansible-c ~]# ansible-playbook vars.yml --syntax-check
```

playbook: vars.yml

Note: Observe there are no errors reported in the playbook and it is good to execute

Let's run the playbook and see the results

```
[root@ansible-c ~]# ansible-playbook vars.yml
```

PLAY [mhost1] *****

TASK [Gathering Facts] *****

ok: [mhost1]

TASK [Printing the value of a variable] *****

ok: [mhost1] => {

 "msg": "The value of the key sub is \" ansible\""

}

PLAY RECAP *****

mhost1 : ok=2 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

2. Disabling the gathering of facts for faster execution of a playbook

Facts collection will take longer time and then execution of playbook will happen.

Run a playbook with time command and check execution time

```
[root@ansible-c ~]# time ansible-playbook vars.yml
real 0m3.743s
user 0m1.584s
sys 0m0.178s

[root@ansible-c ~]# vim vars.yml
---
- hosts: mhost1
gather_facts: no
vars:
  sub: ansible
tasks:
  - name: Printing the value of a variable
    debug:
      msg: The value of the key sub is " {{ sub }}"
...
msg: The value of the key sub is " ansible"
```

```
[root@ansible-c ~]# time ansible-playbook vars.yml
real 0m1.106s
user 0m0.954s
sys 0m0.053s
```

Observe the time difference between both executions

Named Dictionary variable with multiple key value pair:

vars:

```
dict:
  key1: value1
  key2: value2
```

Jinja2 expressions to refer variable in playbook.

“{{ dict }}” - To display dictionary variable.

“{{ dict.key1 }}” - To display value mapped to key1 using dot notation.

“{{ dict['key1'] }}” - To display value mapped to key1 using bracket notation.

1. Creating a playbook with using a dict with multiple key value pair

```
[root@ansible-c ~]# vim vars2.yml
---
- hosts: mhost1
  gather_facts: no
  vars:
    dict:
      key1: value1
      key2: value2
  tasks:
    - name: Printing the value of a variable
      debug:
        msg: Example of dictionary " {{ dict }}"
...
```

Perform syntax-check and execute the playbook

```
[root@ansible-c ~]# ansible-playbook vars2.yml --syntax-check
playbook: vars2.yml
[root@ansible-c ~]#
[root@ansible-c ~]# ansible-playbook vars2.yml
PLAY [mhost1]
*****
TASK [Printing the value of a variable]
*****
ok: [mhost1] => {
    "msg": "Example of dictionary \" {'key1': 'value1', 'key2': 'value2'}\""
}

PLAY RECAP
*****
*
mhost1 : ok=1  changed=0  unreachable=0  failed=0  skipped=0
rescued=0  ignored=0
```

Make the change in playbook `msg` option “`{{ dict.key1 }}`” or “`{{ dict['key1'] }}`” to display value mapped to key1 and re-run the playbook to observe the changes in output

Named List variable example:

vars:

list:

- item1
- item2
- item3

Jinja2 expressions to refer a variable in playbook.

“{{ list }}” - To display list variable.

“{{ list[0] }}” - To display first item in the list.

“{{ list[1] }}” - To display second item in the list.

Lab Exercise:

Create a playbook to use YAML lists as variable

```
[root@ansible-c ~]# vim vars3.yml
---
- hosts: mhost1
gather_facts: no
vars:
  list:
    - item1
    - item2
    - item3
tasks:
  - name: Printing the values of the variable
    debug:
      msg: Example of list " {{ list }}"
...
```

```
[root@ansible-c ~]# ansible-playbook vars3.yml --syntax-check
```

```
playbook: vars3.yml
[root@ansible-c ~]# ansible-playbook vars3.yml
PLAY [mhost1] ****
TASK [Printing the values of the variable] ****
ok: [mhost1] => {
  "msg": "Example of list \" [item1, 'item2', 'item3']\""
}

PLAY RECAP ****
mhost1 : ok=1  changed=0  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
```

Modify the playbook msg option to “{{ list[X] }}”, where x can be 0,1,2 to access item1,2 or 3.

Dictionary+list mixed variable example:

In this example we are going to learn using both dictionary and list variables together.

Ex:

vars:

users:

- name: Steve
- age: 35
- name: tina
- age: 30

Jinja2 expressions to refer a variable in playbook.

“{{ users }}” - To display list variable.

“{{ users[0] }}” - To display first item in the list.

“{{ users[0].name }}” - To display name in first list item.

“{{ users[0]['name'] }}” - To display name in first list item.

Lab Exercise:

Create a playbook to use both dictionary and list as variables

```
[root@ansible-c ~]# vim vars4.yml
---
- hosts: mhost1
gather_facts: no
vars:
    users:
        - name: Steve
          age: 35

        - name: Tina
          age: 30
tasks:
    - name: Printing the values of the variable
      debug:
        msg: Example of dic+list " {{ users }}"
...
```

```
[root@ansible-c ~]# ansible-playbook vars4.yml --syntax-check
playbook: vars4.yml
```

```
[root@ansible-c ~]# ansible-playbook vars4.yml
PLAY [mhost1]
*****
TASK [Printing the values of the variable]
*****
ok: [mhost1] => {
    "msg": "Example of dic+list \" [{name': 'Steve', 'age': 35}, {'name': 'Tina', 'age': 30}]\""
}

PLAY RECAP
*****
mhost1 : ok=1  changed=0  unreachable=0  failed=0  skipped=0
rescued=0  ignored=0
```

See the syntax and modify playbook accordingly

Defining a variable in a different file and accessing it in a playbook

Create a file with .yml or .yaml extension and store vars in it.

```
[root@ansible-c ~]# vim varfile.yml
---
users:
  - name: Steve
    age: 35
  - name: Tina
    age: 30
...
```

Create a playbook to access variable from other file & execute it

```
[root@ansible-c ~]# vim vars5.yml
---
- hosts: mhost1
gather_facts: no
vars_files:
  - varfile.yml
tasks:
  - name: Printing the values of the variable
    debug:
      msg: Example of dic+list " {{ users }}"
...
```

```
[root@ansible-c ~]# ansible-playbook vars5.yml --syntax-check
```

```
playbook: vars5.yml
[root@ansible-c ~]# ansible-playbook vars5.yml
```

```
PLAY [mhost1]
*****
```

```
TASK [Printing the values of the variable]
*****
```

```
ok: [mhost1] => {
    "msg": "Example of dic+list \" [{\"name\": 'Steve', 'age': 35}, {"name": 'Tina', 'age': 30}]\\""
}
```

```
PLAY RECAP
*****
```

```
mhost1 : ok=1  changed=0  unreachable=0  failed=0  skipped=0
rescued=0  ignored=0
```

Repeat the different vars we went through in previous example



Prompting value for the variable while executing play book

Syntax:

`vars_prompt:`

```
- name: var
  prompt: Enter variable
  private: no
```

Jinja2 expressions to refer a variable in playbook.

`"{{ var }}"` - To display variable.

Create a playbook with above mentioned syntax

```
[root@ansible-c ~]# vim vars6.yml
---
- hosts: mhost1
  gather_facts: no
  vars_prompt:
    - name: var
      prompt: Enter Some Value
  tasks:
    - name: Printing the values of the variable
      debug:
        msg: Example of prompt "{{ var }}"
...
[root@ansible-c ~]# ansible-playbook vars6.yml --syntax-check
playbook: vars6.yml
[root@ansible-c ~]# ansible-playbook vars6.yml
Enter Some Value:
PLAY [mhost1]
*****
TASK [Printing the values of the variable]
*****
ok: [mhost1] => {
    "msg": "Example of prompt \"abc\""
}
PLAY RECAP
*****
mhost1 : ok=1  changed=0  unreachable=0  failed=0  skipped=0
rescued=0  ignored=0
```

Let's add an option **private: no** and see the change

```
[root@ansible-c ~]# vim vars6.yml
---
- hosts: mhost1
  gather_facts: no
  vars_prompt:
    - name: var
      prompt: Enter Some Value
      private: no
  tasks:
    - name: Printing the values of the variable
      debug:
        msg: Example of prompt "{{ var }}"
...
```

```
[root@ansible-c ~]# ansible-playbook vars6.yml --syntax-check
```

```
playbook: vars6.yml
```

```
[root@ansible-c ~]# ansible-playbook vars6.yml
```

```
Enter Some Value: abcd
```

```
PLAY [mhost1]
```

```
*****
```

```
TASK [Printing the values of the variable]
```

```
*****
```

```
ok: [mhost1] => {
```

```
  "msg": "Example of prompt \"abcd\""
```

```
}
```

```
PLAY RECAP
```

```
*****
```

```
*****
```

```
mhost1 : ok=1  changed=0  unreachable=0  failed=0  skipped=0
```

```
rescued=0  ignored=0
```

Observe this time the value is displayed on the screen

Let's create a playbook to create a user using prompt option

```
[root@ansible-c ~]# vim user.yml
---
- hosts: mhost1
  gather_facts: no
  vars_prompt:
    - name: username
      prompt: Enter Username
      private: no
  tasks:
    - name: Creating a user on mhost1
      user:
        name: "{{ username }}"
        state: present  # to delete a user state: absent, remove: yes & force: yes
should be used
...

```

Execute the playbook and check the result;

#ansible-playbook user.yml



ANSIBLE FACTS

Ansible collects pretty much all the information about the remote hosts as it runs a playbook. The task of collecting this remote system information is called as **Gathering Facts** by ansible and the details collected are generally known as facts or variables

This information can be obtained manually using Ansible ad-hoc command and a specialized module named **setup**. In fact, ansible playbooks call this setup module by default to perform Gathering Facts task

Ansible facts are data/information related to your remote systems like operating systems, IP addresses assigned on different interfaces, hostname, disks, filesystems and more.

You can access this data in the **ansible_facts** variable. By default, you can also access some Ansible facts as top-level variables with the **ansible_** prefix.

We can display ansible facts using setup module using ansible ad-hoc command:
#ansible managed_host -m setup

Note: As normal user all the facts cannot be collected, so facts should be collected using root user to fetch all important facts.

To check facts about specific hosts

[root@ansible-c ~]# ansible mhost1 -m setup

```
mhost1 | SUCCESS => {
  "ansible_facts": {
    "ansible_all_ipv4_addresses": [
      "192.168.10.10",
      "192.168.122.1"
    ],
    "ansible_all_ipv6_addresses": [
      "fe80::4da1:36ea:cc45:2b38",
      "fe80::5dc9:fd0:9ce9:d81c"
    ],
    "ansible_apparmor": {
      "status": "disabled"
    },
    "ansible_architecture": "x86_64",
    "ansible_bios_date": "12/01/2006",
    "ansible_bios_version": "VirtualBox",
    "ansible_cmdline": {
      "BOOT_IMAGE": "(hd0,msdos1)/vmlinuz-4.18.0-240.el8.x86_64",
      "crashkernel": "auto",
      "initrd": "(hd0,msdos1)/initramfs-4.18.0-240.el8.x86_64"
    }
  }
}
```

```

    "quiet": true,
    "rd.lvm.lv": "rhel/swap",
    "resume": "/dev/mapper/rhel-swap",
    "rhgb": true,
    "ro": true,
    "root": "/dev/mapper/rhel-root"
},

```

Note: ansible facts will be displayed in the form of dictionary and list, key value pair variables

Lab Exercise:

1. Create a playbook to display all facts of a specific host

```
[root@ansible-c ~]# vim facts.yml
---
- hosts: mhost1
gather_facts: True
tasks:
  - name: Displaying the facts
    debug:
      msg: "{{ ansible_facts }}"
...

```

```
[root@ansible-c ~]# ansible-playbook facts.yml
```

Run the playbook and you'll see all the facts of mhost1 will be displayed

2. Create or make changes in previous playbook to view only a key in dictionary **all_ipv4_addresses**.

```
[root@ansible-c ~]# vim facts.yml
---
- hosts: mhost1
gather_facts: True
tasks:
  - name: Displaying the facts
    debug:
      msg: "{{ ansible_facts.all_ipv4_addresses }}"
...

```

```
[root@ansible-c ~]# ansible-playbook facts.yml
PLAY [mhost1] ****
TASK [Gathering Facts] ****
ok: [mhost1]
TASK [Displaying the facts] ****
ok: [mhost1] => {
  "msg": [
    "192.168.10.10",
    "192.168.122.1"
  ]
}
PLAY RECAP ****
mhost1 : ok=2  changed=0  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
```

3. Create a playbook to view only first IP in the list from previous example

```
[root@ansible-c ~]#vim facts.yml
---
- hosts: mhost1
  gather_facts: True
  tasks:
    - name: Displaying the facts
      debug:
        msg: "{{ ansible_facts.all_ipv4_addresses[0] }}"
...
```

*Note: To view all the ansible variables that can be used in playbook, use
#ansible mhost1 -m setup|grep ansible_*



4. Create a playbook to check hostname using `ansible_nodename`

```
[root@ansible-c ~]#vim facts.yml
---
- hosts: mhost1
  gather_facts: True
  tasks:
    - name: Displaying the facts
      debug:
        msg: "{{ ansible_nodename }}"
...
Note: the same can be called as following
      msg: "{{ ansible_facts.nodename }}"
      or
      msg: "{{ ansible_facts[nodename] }}"
```

5. Create a playbook to check the details about the disks connected to mhost1

```
[root@ansible-c ~]#vim facts.yml
---
- hosts: mhost1
  gather_facts: True
  tasks:
    - name: Displaying the facts
      debug:
        msg: "{{ ansible_devices }}"
...

```

To view info only about sda disk change msg to

```
msg: "{{ ansible_devices.sda }}"
```

To view info about all the partitions in sda disk

```
msg: "{{ ansible_devices.sda.partitions }}"
```

To view info only about sda1 partition in sda disk

```
msg: "{{ ansible_devices.sda.partitions.sda1 }}"
```

To view the size of sda1 partition in sda disk

```
msg: "{{ ansible_devices.sda.partitions.sda1.size }}"
```

Same can be viewed in bracket notations

```
msg: "{{ ansible_devices['sda']['partitions']['sda1']['size'] }}"
```

or

```
msg: "{{ ansible_facts['devices']['sda']['partitions']['sda1']['size'] }}"
```

Filtering Ansible facts

To filter Ansible Facts, We can use gather_subset and filter parameters. We can use them using -a option with setup module.

If the command is being run by a normal user, use --become option to get privilege of root user.

Examples of using ‘gather_subset’ directive

To gather Network related facts:

```
ansible target_host(s) -m setup -a "gather_subset=network" --become
```

To gather hardware related facts:

```
ansible target_host(s) -m setup -a "gather_subset=hardware" --become
```

To gather Virtual environment related facts:

```
ansible target_host(s) -m setup -a 'gather_subset=virtual,!min' --become  
Default is all (All facts).
```

To gather facts except network facts:

```
ansible target_host(s) -m setup -a 'gather_subset=!network' --become
```

Examples of using 'filter' directive

```
ansible target_host(s) -m setup -a "filter=ansible_devices" --become  
ansible target_host(s) -m setup -a "filter=ansible_lvm" --become  
ansible target_host(s) -m setup -a "filter=ansible_*name" --become  
ansible target host(s) -m setup -a "filter=ansible_*family" --become
```

Note 1: When using ! Symbol to filter facts, always use single quotes ('single quotes'). With double quotation marks it will not work.

Note 2: Minimum default facts are always gathered with each subset of facts until we exclude them (!min)

Ansible Documentation

Ansible documentation is available on website <https://docs.ansible.com/>

Using ansible-doc command line tool, we can conveniently check information about modules, various directives that can be used and with some playbook examples.

For example, to display all information about copy module, execute below command.

#ansible-doc copy

In similar way you can check information about other modules, but you must know exact name of module.

To display all modules, execute **#ansible-doc -l**

CONFIGURING ANSIBLE MANAGED NODES

In this chapter we are going to learn configuring managed hosts to be used with a normal user called ansible. We will setup ansible user to get privileges of root using sudo powers on all managed hosts.

Let's first learn some interesting options in ansible command line to edit a file on managed hosts.

Task. Configure 'mhost4' to listen on non-default SSH port 222.

- ansible user should be able to connect to mhost4 on new SSH port as well as standard SSH port.
- Update the inventory file to tell ansible to use port 222 to connect to mhost4
- Setup inventory file for ansible user as shown below

Task1: Configure ansible configuration file with following default values.

- Ansible config file must be created in subdirectory tasks in ansible user's home directory.
- Roles directory path should be /home/ansible/tasks/roles, in addition default path should also be considered.
- Inventory file with name mnodes should exist on path /home/ansible/tasks.
- Remote port 22 should be used for SSH connection.
- User ansible should be used to connect to remote hosts.
- Privilege escalation must be disabled.
- Default module must be command module.
- Ansible should start 5 forks.

Execute commands as ansible user:

mkdir /home/ansible/tasks

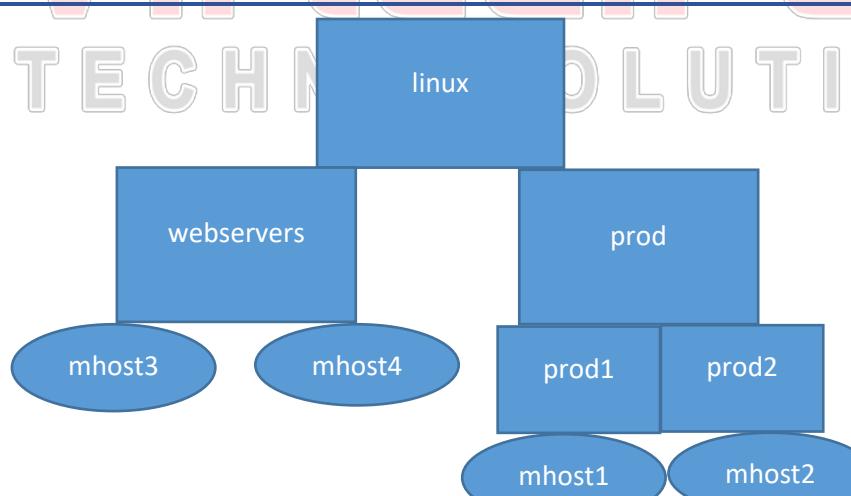
```
vim /home/ansible/tasks/ansible.cfg
[defaults]
inventory = /home/ansible/tasks/mnodes
roles_path = /etc/ansible/roles:/home/ansible/tasks/roles
remote_port = 22
remote_user = ansible
module_name = command
forks = 5
[privilege_escalation]
become = False
```

Task2: Create inventory file 'mnodes' on the inventory path defined in ansible config file with below requirements.

- mhost1 must be part of host group **prod1**.
- mhost2 must be part of host group **prod2**.
- mhost3 and mhost4 must be part of **webservers** group.
- prod1 and prod2 must be part of **prod** group.
- group **linux** should include all managed hosts.
- Make sure all hostnames used can be resolved to IP Address.

```
vim /home/ansible/tasks/mnodes
```

```
[prod1]
mhost1
[prod2]
mhost2
[webservers]
mhost[3:4]
[prod:children]
prod1
prod2
[linux:children]
prod
webservers
```



Test inventory using following commands as ansible user

```
$ ansible all --list-hosts
$ ansible prod1 --list-hosts
$ ansible prod2 --list-hosts
$ ansible webservers --list-hosts
$ ansible prod --list-hosts
$ ansible linux --list-hosts
```

Login as root and perform ping test (note: go to /home/ansible/tasks folder and run commands)

```
#ansible all -m ping -k  
#ansible all -m ping -k -u root
```

Tasks to manage hosts we need to authorize ansible user to connect all hosts and also have sudo privileges.

- Create and distribute SSH keys to managed nodes
- Configure privilege escalation on managed nodes
- Validate a working configuration using ad hoc Ansible commands

Task1: Configure 'mhost4' to listen on non-default SSH port 222.

- ansible should be able to connect to mhost4 on new SSH port as well as standard SSH port.
- Update the inventory file to tell ansible to use port 222 to connect to mhost4.

Execute commands as root user :

```
ansible mhost4 -m lineinfile -a "path=/etc/ssh/sshd_config regexp='^#Port' line='Port 22'" -u root
```

```
ansible mhost4 -m lineinfile -a "path=/etc/ssh/sshd_config insertafter='^Port' line='Port 222'" -u root
```

```
ansible mhost4 -m seport -a "ports=222 proto=tcp setype=ssh_port_t state=present" -u root
```

```
ansible mhost4 -m firewalld -a "port=222/tcp state=enabled permanent=yes" -u root
```

```
ansible mhost4 -m service -a "name=firewalld state=reloaded" -u root
```

```
ansible mhost4 -m service -a "name=sshd state=restarted" -u root
```

Modify inventory file:

```
vim /home/ansible/tasks/mnodes
```

```
---
```

```
mhost4 ansible_port=222
```

```
---
```

```
:wq
```

Note: if required we need to install policycoreutils package to use seport module. Install same using
#dnf install policycoreutils**

Task2: Generate SSH Keys for user 'ansible' on Ansible control node.

- Use ansible ad-hoc command to create user **ansible** on all managed nodes and copy the public key for ansible user to managed nodes.
- Execute this task as root user.
- Use **redhat** as password for this user.

Execute command as ansible user:

```
ssh-keygen
```

Execute commands as root user:

```
ansible all -m user -a "name=ansible state=present password='{{ 'redhat' |  
password_hash('sha256') }}'" -u root  
ansible all -m authorized_key -a "user='ansible' state='present' key='{{ lookup('file',  
'/home/ansible/.ssh/id_rsa.pub') }}' path='/home/ansible/.ssh/authorized_keys'" -u root
```

Note: observe in above task to assign password in encrypted form, we are using sha256 algorithm, you can also use sha512 algorithm if required for stronger encryption.

Task3: Using ansible ad-hoc commands, Configure privilege escalation (sudo) for user 'ansible' on all Managed hosts.

- User **ansible** should be able to use **sudo** without providing password.

Execute this command as root user:

```
ansible all -m lineinfile -a "path=/etc/sudoers state=present line='ansible ALL=(ALL)  
NOPASSWD: ALL' backup=yes validate='/usr/sbin/visudo -cf %s'" -u root
```

Task4: Use ansible ad-hoc command to configure MOTD on all managed hosts as "Welcome to Ansible managed host"

Execute this command as ansible user:

```
ansible all -m copy -a "content='Welcome to Ansible managed host' dest=/etc/motd" --  
become
```

Note: --become will ansible user sudo privilege

Task5: Use ansible adhoc-commands to configure all managed nodes to use 'BaseOS' and 'AppStream' repos with following information

For BaseOS Repository:

- name=BaseOS
- description= DNF BaseOS Repo
- baseurl=ftp://192.168.10.5/pub/rhel8/BaseOS
- gpgcheck=1
- gpgkey=/etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
- enabled=1

For AppStream Repository:

- name=AppStream
- description= DNF AppStream Repo
- baseurl= ftp://192.168.10.5/pub/rhel8/AppStream
- gpgcheck=1
- gpgkey=/etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
- enabled=1

Now, execute following ad-hoc command as ansible user

Execute commands as ansible user:

```
ansible all -m yum_repository -a "name=BaseOS description='DNF BaseOS Repo'
baseurl=ftp://192.168.10.5/pub/rhel8/BaseOS gpgcheck=1 gpgkey=file:///etc/pki/rpm-
gpg/RPM-GPG-KEY-redhat-release enabled=1 file=BaseOS" --become
```

```
ansible all -m yum_repository -a "name=AppStream description='DNF AppStream Repo'
baseurl=ftp://192.168.10.5/pub/rhel8/AppStream gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release enabled=1 file=AppStream"
--become
```

Check on all nodes if it is done

To Disable GPG Key Check

Execute commands as ansible user:

```
ansible all -m yum_repository -a "name=BaseOS description='DNF BaseOS Repo'
baseurl=ftp://192.168.10.5/pub/rhel8/BaseOS gpgcheck=0 enabled=1 file=BaseOS" --
become
```

```
ansible all -m yum_repository -a "name=AppStream description='DNF AppStream Repo'
baseurl=ftp://192.168.10.5/pub/rhel8 gpgcheck=0 enabled=1 file=AppStream" --become
```

USEFUL SYSTEM ADMIN TASKS USING ANSIBLE PLAYBOOK

Perform all the tasks as ansible user.

Task 1: Create a playbook named 'services.yml' under tasks directory to perform below tasks.

- Install httpd service on webservers nodes.
- Install mariadb service on prod nodes.
- Make sure services are started and enabled.

```
[ansible@ansible-c ~]$ vim services.yml
```

```
---
-
hosts: webservers
become: True
tasks:
  - name: Installing httpd service
    yum:
      name: httpd
      state: present
  - name: Starting and enabling httpd service
    service:
      name: httpd
      state: started
      enabled: yes
-
hosts: prod
become: True
tasks:
  - name: Installing mariadb service
    yum:
      name:
        - mariadb-server
        - mariadb-common
      state: present
  - name: Starting and enabling mariadb service
    service:
      name: mariadb
      state: started
      enabled: yes
...
```

Execute services.yml playbook

```
[ansible@ansible-c ~]$ ansible-playbook services.yml --syntax-check
```

```
[ansible@ansible-c ~]$ ansible-playbook services.yml
```

Task 2: Create a playbook 'user.yml' to create user on all managed hosts with below information.

- Use username as mark.
- Set redhat as password.
- Password must be encrypted with Sha512.

```
[ansible@ansible-c ~]$ vim users.yml
---
-
hosts: all
become: True
gather_facts: False
tasks:
  - name: Creating user
    user:
      name: mark
      password: "{{ 'redhat' | password_hash('sha512') }}"
      state: present
...
```

Execute the playbook and see the result

```
[ansible@ansible-c ~]$ ansible-playbook users.yml --syntax-check
[ansible@ansible-c ~]$ ansible-playbook users.yml
```

Task 3: Create a playbook named 'file.yml' to create file '/root/mark_file' on all managed nodes.

- User and group ownership must be set to mark.
- Configure full permissions for user, read/write at group level and no permissions for others on this file.
- Set GiD bit.

```
[ansible@ansible-c ~]$ vim file.yml
---
-
hosts: all
become: True
gather_facts: False
tasks:
  - name: Creating file, setting permissions and gid bit
    file:
      path: /root/mark_file
      owner: mark
      group: mark
      mode: '2760'
      state: touch
...
```

Execute the playbook and see the result

```
[ansible@ansible-c ~]$ ansible-playbook file.yml --syntax-check
[ansible@ansible-c ~]$ ansible-playbook file.yml
```

Task4: Using ansible ad-hoc commands, create file '/root/file1.txt' on all managed nodes.

- File should contain text “This text file is created using Ansible.”
- Remove all permissions for others on this file

Execute Commands as ansible user:

```
ansible all -m file -a "path=/root/file1.txt mode=0-rwx state=touch" --become
ansible all -m copy -a "content='This text file is created using Ansible' dest=/root/file1.txt"
--become
```

Task 5: Using ansible playbook ‘archive.yml’, archive contents of ‘/etc’ directory into ‘etc.tar’ file under ‘/root’ directory.

- Playbook should be executed on webservers nodes.
- Compress the archive using bzip2.

```
[ansible@ansible-c tasks]$ vim archive.yml
---
-
hosts: webservers
become: Yes
gather_facts: False
tasks:
  - name: Archiving /etc directory
    archive:
      path: /etc
      dest: /root/etc.tar.bz2
      format: bz2
...

```

Execute the playbook and see the result

```
[ansible@ansible-c tasks]$ ansible-playbook archive.yml --syntax-check
[ansible@ansible-c tasks]$ ansible-playbook archive.yml
```

Task 6: Create a playbook 'cronjobs.yml' to schedule below tasks.

- Restart rsyslog service at 23h00 and 06h00 on prod nodes every day.
- Restart rsyslog service at 02h00 on webservers nodes on every Monday.

```
[ansible@ansible-c tasks]$ vim cron.yml
---
-
hosts: prod
become: Yes
gather_facts: False
tasks:
  - name: Scheduling restart of rsyslog on prod nodes
    cron:
      name: "Scheduling cron job on prod nodes"
      hour: "23,6"
      minute: "0"
      job: /usr/bin/systemctl restart rsyslog

  -
hosts: webservers
become: True
gather_facts: False
tasks:
  - name: Scheduling restart of rsyslog on webservers nodes
    cron:
      name: "Scheduling cron job on webservers nodes"
      hour: "2"
      minute: "0"
      weekday: "1"
      job: /usr/bin/systemctl restart rsyslog
...
```

Task 7: Create a playbook 'update.yml' to update all packages on prod1 node.

```
[ansible@ansible-c tasks]$ vim update.yml
---
-
hosts: prod1
become: True
gather_facts: False
tasks:
  - name: Update all packages on prod1 node
    yum:
      name: '*'
      state: latest
...
...
```

Task 8: Create a playbook 'firewall.yml' to configure firewall on all 'webservers' nodes.

- Inbound traffic for http service should be accepted.
- Setting should be persistent and reload firewall to enforce this

```
[ansible@ansible-c tasks]$ vim firewall.yml
---
-
hosts: webservers
become: Yes
gather_facts: False
tasks:
  - name: Configuring firewall on webservers nodes
    firewalld:
      service: http
      state: enabled
      permanent: yes
    notify: Reload firewall
handlers:
  - name: Reload firewall
    service:
      name: firewalld
      state: reloaded
...
...
```

Task 9: Create a playbook 'group.yml' to perform below tasks.

- Create directory path /web/html on webservers nodes.
- Create group testing on webservers nodes and group networks on prod nodes.

```
[ansible@ansible-c tasks]$ vim group.yml
```

```
---
```

```
- hosts: webservers
  become: Yes
  gather_facts: False
  tasks:
    - name: Creating directory
      file:
        path: /web/html
        state: directory
```

```
    - name: Creating group
      group:
        name: testing
        state: present
```

```
- hosts: prod
  become: True
  gather_facts: False
  tasks:
    - name: Creating group
      group:
        name: networks
        state: present
```

```
...
```

Task 10. Create a playbook 'context.yml' to set selinux context type

'httpd_sys_content_t' on '/web/html' directory on all webservers nodes.

- Setting should be persistent, and context should be restored.
- Verify the context type using ansible ad-hoc command.

```
[ansible@ansible-c tasks]$ vim context.yml
---
-
hosts: webservers
become: Yes
gather_facts: False
tasks:
  - name: Setting Context type
    sefcontext:
      target: '/web/html(/.*)?'
      setype: httpd_sys_content_t
      state: present
  - name: Restoring context type
    command: restorecon -irv /web/html
```

Task 11: Create a playbook 'parted.yml' to create extended partition on all managed nodes.

- Use all remaining space for extended partition (container for logical partitions).
- Create one logical partition of size 200 MiB on all managed nodes.

```
[ansible@ansible-c tasks]$ vim parted.yml
---
-
hosts: all
become: Yes
gather_facts: True
tasks:
  - name: Read device information
    parted: device=/dev/sda unit=MiB
    register: sda_info
  - name: Creating Extended partition
    parted:
      device: /dev/sda
      number: "4"
      part_type: extended
      part_start: "{{ sda_info.partitions[2].end + 1 }}MiB"
      state: present
  - name: Creating logical partition
    parted:
      device: /dev/sda
      number: "5"
      part_type: logical
      part_start: "{{ sda_info.partitions[2].end + 2 }}MiB"
      part_end: "{{ sda_info.partitions[2].end + 202 }}MiB"
      state: present
...
...
```

Task 12: Create a playbook 'mount.yml' to format the device '/dev/sda5' with 'ext4' filesystem.

- Mount the file system on /mnt/partition directory.
- Mount should be persistent

```
[ansible@ansible-c tasks]$ vim mount.yml
```

```
---
```

```
-
```

```
hosts: all
```

```
become: Yes
```

```
gather_facts: False
```

```
tasks:
```

```
  - name: Creating filesystem
```

```
    filesystem:
```

```
      dev: /dev/sda5
```

```
      fstype: ext4
```

```
  - name: Mounting filesystem
```

```
    mount:
```

```
      src: /dev/sda5
```

```
      path: /mnt/partition
```

```
      fstype: ext4
```

```
      state: mounted
```

```
...
```