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

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, ME.

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 Linux:-

- ✓ UNIX was developed in 1969 at AT&T Bell Labs by Ken Thompson and Dennis Ritchie.
- ✓ Based on core UNIX then different types of variants were developed like BSD, Solaris, Mac OS X, Linux.
- ✓ Commercial UNIX Variants Apple – MAC OS, HP – HP-UX, IBM – AIX, SUN – SUN OS and Solaris.
- ✓ The Linux kernel was first developed in 1991 by Linus Torvalds, a student at the University of Helsinki in Finland.
- ✓ Linux has been widely adopted for servers and embedded systems.

Advantages of Linux:-

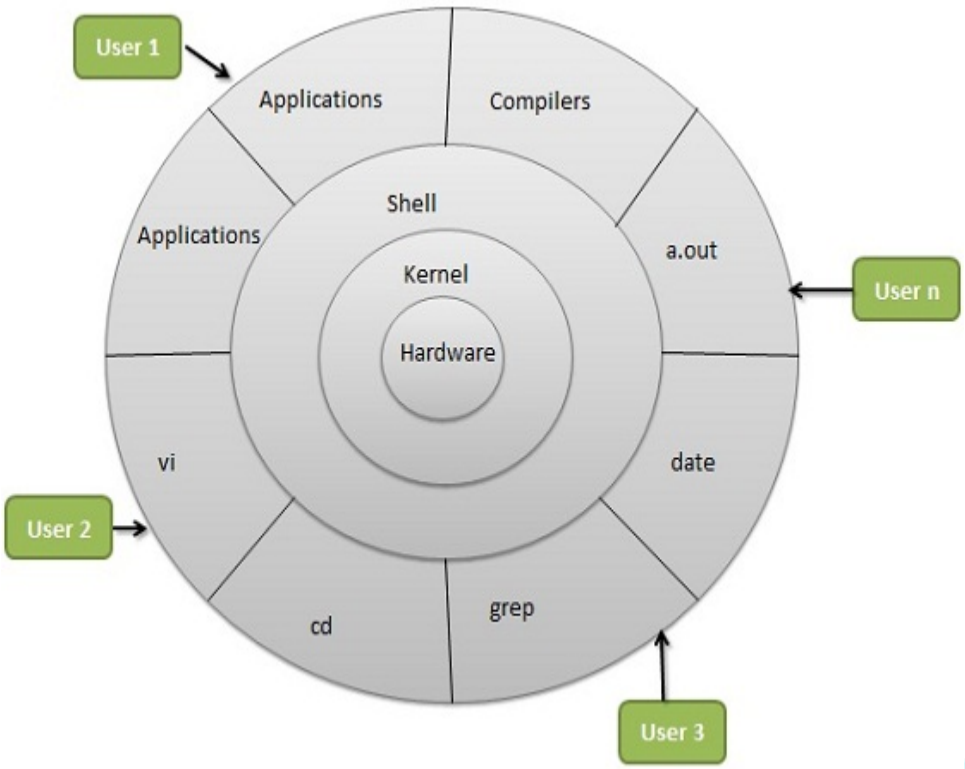
Following are some of the important features of Linux Operating System.

- **Portable** - Portability means software's can works on different types of hardware's in same way. Linux kernel and application programs supports their installation on any kind of hardware platform.
- **Open Source** - Linux source code is freely available and it is community based development project. Multiple teams' works in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
- **Multi-User** - Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
- **Multi programming** - Linux is a multiprogramming system means multiple applications can run at same time.
- **Hierarchical File System** - Linux provides a standard file structure in which system files/ user files are arranged.

- **Shell** - Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs etc.

- **Security** - Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

Architecture



Linux System Architecture is consists of following layers

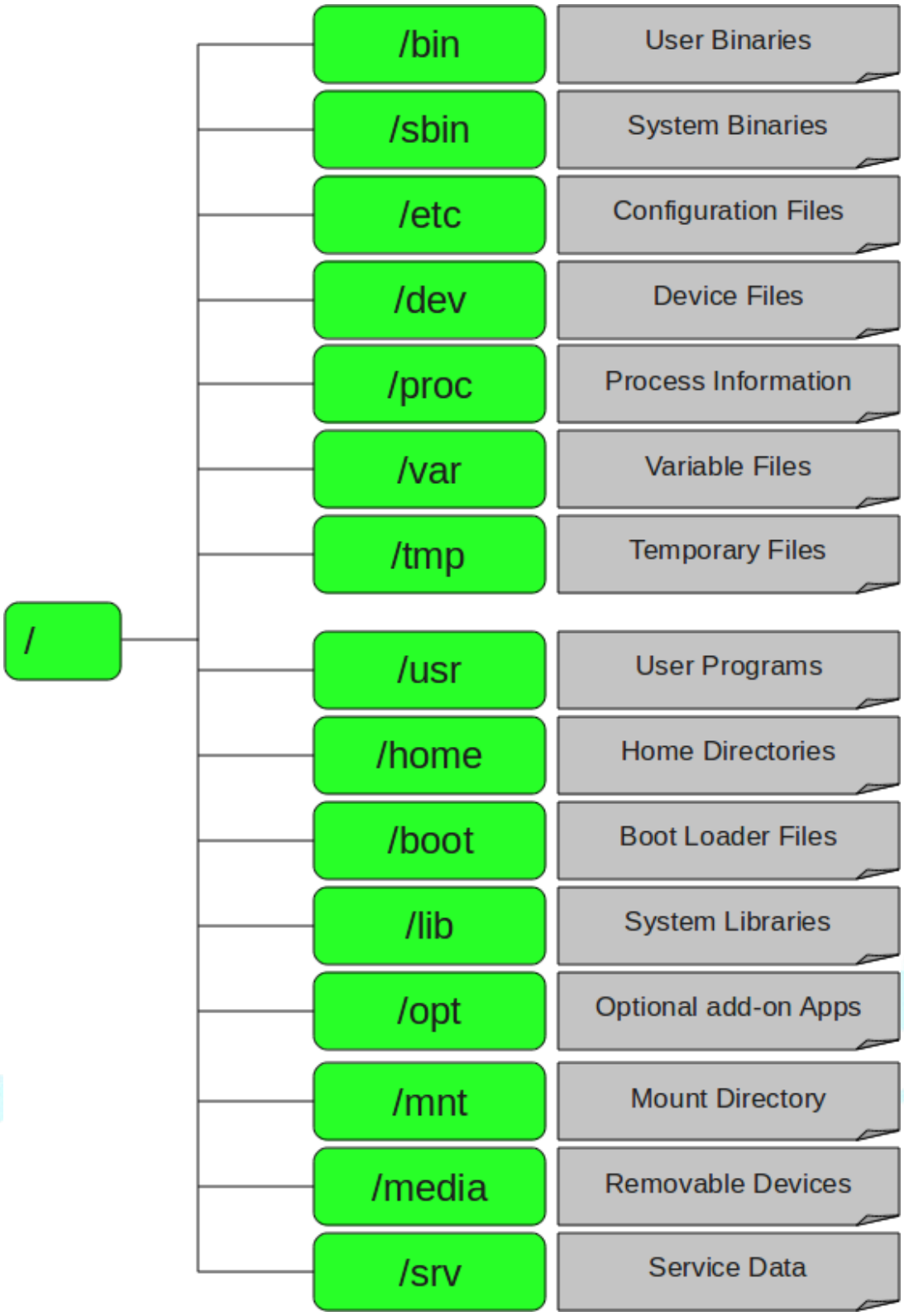
- **Hardware** - Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).

- **Kernel** - Core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.

- **Shell** - An interface to kernel, hiding complexity of kernel's functions from users. Takes commands from user and executes kernel's functions.

- **Utilities** - Utility programs giving user most of the functionalities of an operating systems.

Linux File System Structure:-


/
/bin
User Binaries
/sbin
System Binaries
/etc
Configuration Files
/dev
Device Files
/proc
Process Information
/var
Variable Files
/tmp
Temporary Files
/usr
User Programs
/home
Home Directories
/boot
Boot Loader Files
/lib
System Libraries
/opt
Optional add-on Apps
/mnt
Mount Directory
/media
Removable Devices
/srv
Service Data

All files on a Linux system are stored on file systems which are organized into a single *inverted* tree of directories, known as *file system hierarchy*. This tree is inverted because the root of the tree is said to be at the *top* of the hierarchy, and the branch of directories and sub directories stretch *below* the root.

The directory **/** is the root directory at the top of the file system hierarchy. The **/** character is also used as a *directory separator* in file names. For example, if **var** is a sub directory of the **/** directory, we could call that directory **/var**. Likewise, if the **/var** directory contained a file named **issue**, we could refer to that file as **/var/issue**.

/

- It is a top level parent directory for all other child directories
- the root directory represent with forward slash (/)
- Every single file and directory starts from the root directory.

/root

- only root user has write privilege under this directory.
- Please note that /root is root user's home directory, which is not same as /

/home – Home Directories

- Home directories for all users to store their personal files.
- For example: /home/john, /home/nikita

/boot – Boot Loader Files

- Contains boot loader related files.
- Kernel initrd, vmlinux, grub files are located under /boot

- For example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic

/bin – User Binaries

- Contains binary executable s.
- Common Linux commands you need to use in single-user modes are located under this directory.
- Commands used by all the users of the system are located here.
- For example: ps, ls, ping, grep, and cp.

/sbin – System Binaries

- Just like /bin, /sbin also contain binary executables.
- But, the linux commands located under this directory are used typically by system administrator, for system maintenance purpose.
- For example: iptables, reboot, fdisk, ifconfig, swapon

/etc – Configuration Files

- Contains configuration files required by all programs.
- This also contains startup and shutdown shell scripts used to start/stop individual programs.
- For example: /etc/resolv.conf, /etc/logrotate.conf

/dev – Device Files

- Contains device files.
- These include terminal devices, usb, or any device attached to the system.
- For example: /dev/tty1, /dev/usbmon0

/proc – Process Information

- Contains information about system process.
- This is a pseudo file system contains information about running process. For example: /proc/ {pid} directory contains information about the process with that particular pid.
- This is a virtual file system with text information about system resources. For example: /proc/uptime

/var – Variable Files

- var stands for variable files.
- Content of the files that are expected to grow can be found under this directory.
- This includes – system log files (/var/log); packages and database files (/var/lib); emails (/var/mail); print queues (/var/spool); lock files (/var/lock); temp files needed across reboots (/var/tmp);

/tmp – Temporary Files

- Directory that contains temporary files created by system and users.
- Files under this directory are deleted when system is rebooted.

/usr – User Programs

- Contains binaries, libraries, documentation, and source-code for second level programs.
- /usr/bin contains binary files for user programs. If you can't find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp
- /usr/sbin contains binary files for system administrators. If you can't find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel
- /usr/lib contains libraries for /usr/bin and /usr/sbin
- /usr/local contains users programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2

/lib – System Libraries

- Contains library files that supports the binaries located under /bin and /sbin
- Library file names are either ld* or lib*.so.*
- For example: ld-2.11.1.so, libncurses.so.5.7

/opt – Optional add-on Applications

- opt stands for optional.
- Contains add-on applications from individual vendors.
- Add-on applications should be installed under either /opt/ or /opt/ sub-directory.

/mnt – Mount Directory

- Temporary mount directory where sysadmins can mount file systems.

/media – Removable Media Devices

- Temporary mount directory for removable devices.
- For examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives; /media/cdrecorder for CD writer

/srv – Service Data

- srv stands for service.
- Contains server specific services related data.
- For example, /srv/cvs contains CVS related data.

Properties of the terminal:-

[root @ localhost ~] (# or $)

root : login username

localhost : System Name

~ : User home directory

: Super user privilege

$: local user privilege

Basic Commands:

Man command:-

On Linux and other Unix-like operating systems, **man** is the interface used to view the system's reference manuals.

man is the system's manual viewer; it can be used to display manual pages, scroll up and down, search for occurrences of specific text, and other useful functions.

Each argument given to **man** is normally the name of a program, utility or function.

Syntax:-

```
$ man <command name>
```

Example:-

```
$ man cat
```

Date command:-

Date command is used to display the current date and time of the system. It can also be used by the super user to set the system clock.

Syntax:-

```
$ date

   Sat Apr5 08:13:50 IST 2014
```

Calendar command:-

Cal command is used to displays a simple calendar in traditional format. **ncal** command offers an alternative layout.

Syntax:-

```
$ cal

$ ncal
```

Calendar for particular month & year:

```
$ cal 7 1984

$ cal July 1984
```

Calendar for the whole year

```
$ cal 1984
```

Calculator command:-

bc command is used as calculator in Linux.

Syntax:-

```
$ bc (press enter)

  5*10 (press enter)

  50 (Result)
```

If you have some expressions in a file that are ready to be calculated then you can also use **bc** to calculate those expressions. Here is an example:

Create a file **$cat bcfile** then give the expressions in the file like below

```
10*5

10+5

10-5

     10/5
```

Then execute file with the bc command as follows

```
$ bc < bcfile
```

Present Working Directory:-

pwd command is used to display the Print name of current/working directory.

The **pwd** command displays the full path name of the current location, which helps determine appropriate syntax for reaching files using relative path names.

Syntax:-

```
$ pwd
```

Change Directory:-

Use the **cd** command to Change directories. With working directory of **/home/student**, relative path syntax is shortest to reach the **Documents** subdirectory. The **Downloads** subdirectory is then reached using absolute path syntax.

Syntax:-

```
$ cd Documents(change the directory into /home/student/Documents)
$ pwd
/home/students/Documents
```

The **cd** command has many options.

- The command **cd –** changes directory to the directory where the user was previous to the current directory

```
[student@desktopX ~]$ cd videos
[student@desktopX Videos]$ pwd
/home/student/Videos
[student@desktopX Videos]$ cd /home/student/Documents
[student@desktopX Documents]$ pwd
/home/student/Documents
[student@desktopX Documents]$ cd –
[student@desktopX Videos]$ pwd
/home/student/Videos
```

- The **cd ..** command uses the .. hidden directory to move up one level to the parent directory, without needing to know the exact parent name.

```
[student@desktopX Videos]$ pwd
/home/student/Videos
[student@desktopX Videos]$ cd ..
[student@desktopX]$ pwd
/home/student
```

Listing the files and directories:-

ls command is used to list out the files and directories in present working directory. Color coding for the files and directories are **white** color for **files** (depending up on terminal background color) and **blue** color for the **directories**.

- To list out the files and directories without options

  $ ls

- To long listing the files and directories

  $ ls –l

- To view the hidden files

  $ ls -a [hidden files starts with dot(.)]

- To view the files and directories in human readable format

  $ ls –lh (option h follows the option l)

- To list the files and directories in reverse output order

  $ ls –r

- To list out the recursively sub-directories

  $ ls –R

- To list out the latest modified files and directories

  $ ls –ltr

- To display inode number of files and directories

  $ ls –i

- To display the files and directories by the sizes (maximum size displays at top level)

  $ ls –lS

- To list the directory information of particular directory

  $ ls –l /home

- To list the directories only

  $ ls –ld

- To display the uid and gid of the files and directories

  $ ls -n

File creation methods:

- cat
- touch

Cat command:_

Cat stands for "concatenate." It reads data from files, and outputs their contents. It is the simplest way to display the contents of a file at the command line.

- To Display the content of the file

  Syntax:-

  $cat <filename>

  Example:-

  $cat /etc/passwd

- To create a file

  Syntax:

  $cat > <filename>

  Example:-

```
$cat > mist.txt
```

- To append the data into existing file

Syntax:

```
$cat >> <filename>
```

Example:-

```
$cat >> mist.txt
```

- Display Line Numbers in File

Syntax:-

```
$cat -n <filename>
```

Example:-

```
$cat -n mist.txt
```

Touch command:-

The touch command is the easiest way to create new, empty files. It is also used to change the time stamps (i.e., dates and times of the most recent access and modification) on existing files and directories.

- To create empty files

Syntax:-

```
$touch <filename>
```

Example:-

```
$touch file1
```

- To create multiple files

Syntax:-

```
$touch <first file> <second file> <third file>
```

Example:-

```
$touch mist redhat linux
```

- To create multiple files within a range:-

Syntax:-

```
$touch <filename {range}>
```

Example:-

```
$touch linuxfile {1..10}
$touch linux{a..z}
```

- To change the time stamp of existing file:-

```
Syntax:-
  $ touch <filename>
Example:-
  $ touch /root/anaconda-cfg.ks
```

- To change the date & time of a file

```
Syntax:-
  $ touch –d '< specify date & time >'   < file name >
Example:-
  $ touch –d  '15 July 1984 10:00'  mist
```

- Create a file using a specified time

```
Syntax:-
  $ touch –t YYMMDDHHMM.SS <filename>
Example:-
  $ touch –t 201403150920.01 redhat
```

File Editors:-

- vi/vim (For CLI)
- gedit (For GUI)

VI/VIM:-

The vi editor has three modes, command mode, insert mode and command line mode.

1. **Extended mode:** letters or sequence of letters interactively commands vi. Commands are case sensitive. The ESC key can end a command.
2. **Insert mode:** Text is inserted. The ESC key ends insert mode and returns you to command mode. One can enter insert mode with the "i" (insert), "a" (insert after), and “A” insert at end of line, "o" (open new line after current line) or "O" (Open line above current line) commands.
3. **Command line mode:** One enters this mode by typing ":" which puts the command line entry at the foot of the screen.

Cursor Movement

```
h     -     Move cursor left

j     -     Move cursor down

k     -     Move cursor up

l     -     Move cursor right

w     -     Jump forwards to the start of a word

W     -     Jump forwards to the start of a word

            (Words can contain punctuation)

e     -     Jump forwards to the end of a word

E     -     Jump forwards to the start of a word

            (Words can contain punctuation)

b     -     Jump backwards to the start of a word

B     -     Jump backwards to the start of a word
```

(Words can contain punctuation)

o - Jump to the start of the line

^ - Jump to the first non-blank character of the line

Editing

R - Replace a single character

J - Join line below to the current one

cc - change (replace) entire line

cw - change (replace) to the end of the word

c$ - change (replace) to the end of the line

s - Delete character and substitute text

S - Delete line and substitute text

xp - transpose two letters (delete and paste)

u - Undo

ctrl + r - redo

. - repeat last command

Marking Text (visual mode)

V - Start visual mode mark lines, and then do a command

(Like y-yank) Cut and Paste

yy - yank (copy) a line

nyy - yank (copy) n lines [eg. 2yy - copy 2 lines]

yw - yank (copy) word

y$ - yank (copy) to end of line

p - Put (paste) the clipboard after cursor

P - Put (paste) before cursor

dd - delete (cut) a line

ndd - delete (cut) n lines [e.g. 2dd - delete 2 lines]

dw - delete (cut) word

D - Delete (cut) end of the line

d$ - delete (cut) to the end of the line

x - Delete (cut) character

Exiting

:w - Write (save) the file, But don't exit

:q - Quit the file

:q! - Quit the file forceful

:wq - Save & Quit the file

:wq! - Save & Quit the file forceful:x - Save & Exit the file

:n - go to n^{th} line [eg. :20(go to 20th line)]

/string - search the given string/word in the file

Gedit (For GUI):-

The **gedit** application is a full-featured text editor for the **GNOME** desktop environment. Launch **gedit** by selecting **Applications > Accessories > gedit** from the gnome menu. Like other graphical applications, gedit can be started without navigating the menu. Press **Alt+F2** to open the **Enter a Command** dialog box. Type **gedit** and press **Enter**.

Create a new file:-

To create a new file in gedit click on blank paper toolbar icon, or select File > New (ctrl + n) from the menu.

Save the file:-

To save a file in gedit click the save icon on toolbar or select File > save (ctrl + s) from the menu.

Open the file:-

To open an existing file, click the **open** toolbar icon or select **File > open (ctrl + o)** from the menu. The **Open Files** dialog window will display from which users can locate and select the file to open.

Managing Files Using Command-Line Tools:-

File management involves creating, deleting, copying and moving files. Additionally, directories can be created, deleted, copied and moved to help organize files logically. When working at the command line, file management requires awareness of the current working directory to choose either absolute or relative path syntax as most efficient for the immediate task.

File management commands:-

- **Copy file**

  cp command is used to copy the files

  Syntax: $ cp <source file> <destination file>

  Example: $ cp file1 file2

- **Move file**

  mv command is used to move the files

  Syntax: $ mv <source file> <directory name> [for single file]

  Example: $ mv file1 dir

  Syntax: $ mv <source files> <dir name> [for multiple files]

  Example: $ mv file1 file2 file3 dir

- **Rename file**

  mv command used to rename the files

  Syntax: $ mv <old file name> <new file name>

  Example: $ mv anil varun

- **Delete/remove fil**e

rm command is used to delete/remove the files

Syntax: $ rm <file name> [it will ask confirmation for delete or not]

Example: $ rm file1

Syntax: $ rm –f <file name> [delete the file force fully]

Example: $ rm –f file1

- **Create directory**

  mkdir command is used to make directories

  Syntax: $ mkdir <directory name>

  Example: $ mkdir dir1

  Syntax: $ mkdir –p <collaborative directory>

  Example: $ mkdir –p /dir1/dir2/dir2/dir4

- **Copy directory**

  Cp command is used to copy the directories

  Syntax: $ cp <options> <source dir> <destination dir>

  Example: $ cp –r dir1 dir2

- **Move directory**

  Mv command is used to move the directories

  Syntax: $ mv <source dir> <destination dir>

  Example: $ mv dir1 dir2

- **Move files into directory**

  Mv command is used to move files into the directory

  Syntax: $ mv <filename> <filename> <filename> <dirname>

  Example: $ mv file1 file2 file3 /dir1

- **Rename directory**

  Mv command is used to rename the directory

Syntax: $ mv <olddirname> <newdirname>

Example: $ mv /dir1 /dir5 (/dir5 should not be present for rename)

- **Delete directory**

Rmdir command is used to delete the empty directories.

Syntax: $ rmdir <dirname>

Example: $ rmdir dir3

rm –rf command is used to delete all files/directories in the directory

Syntax: $ rm –rf <dirname>

Example: $ rm –rf /dir5

User and Group administration:-

User Administration:-

Administering user and group accounts is a fundamental Linux system administration activity. Ordinarily, most people understand user accounts as accounts tied to a particular physical user. Linux systems also have logical user accounts, user accounts that exist for particular applications, such as Apache, or system functions, such as the mail and bin user accounts.

In particular, both actual and logical have user identification numbers (UIDs), numeric values that the kernel and many applications use instead of the account name. Ordinarily each user account has a unique UID (on a given system), but this is not strictly required.

Every user have a user account represented by a username and password for accountability, security, logging, and resource management. Every process on the system runs as a particular user. Every file is owned by a particular user. Access to files and directories are restricted by user.

Linux systems classify users and groups into two categories: System users and Normal users.

1. System Users:-

While we are installing the os, the system users will be created by default. These users are logical users except 'root' user, because root user is the super user in Linux systems. The UIDs of System Users are starts from 0 to 999. Root user UID is set to 0 by default.

2. Normal users:-

The users which are created by the root user are known as Normal users. These users are physical users. The UIDs of Normal Users are starts from 1000 to 60000.

When we are adding a user to the system it will create some key points as follows:

- Red Hat Enterprise Linux uses a *user private group* (*UPG*) scheme, which makes UNIX groups easier to manage.
- When we are creating a user the group also created with the same name, this group is the primary group for that particular user.
- A user can have only one primary group.

User database files:-

When we are adding a user to the system the credentials of a user will stored in the following database files.

- /etc/passwd
- /etc/shadow
- **/etc/passwd:-**

All users created under **/etc/passwd** file, one line per user. The seven colon-separated fields of **/etc/passwd** file as follows.

Username[1]:passwd[2]:UID[3]:GID[4]:comments[5]:directory[6]:shell[7]

Field	**Description**
1. Username	The user's account name on the system
2. Password	user's encrypted passwd or an x
3. UID	user's numeric UID (User ID)
4. GID	user's numeric primary group ID (group ID)
5. Comments	An optional field used for informational purpose
6. Home directory	user's home directory
7. Shell	user's default shell

/etc/shadow

In addition to storing the encrypted password, /etc/shadow file stores password expiration information. As in /etc/passwd, fields are separated by colon (:). It contains the following fields, listed in the order in which they appear in /etc/shadow

Smithj[1]:Ep6mckrOLChF.[2]:10063[3]:0[4]:99999[5]:7[6]:::[7]

Field	Description
Username	The user's account name on the system
Encrypted passwd	user's encrypted passwd from /etc/passw File.
Last password change	Days since Jan 1, 1970 that password was last changed
Minimum	The minimum number of days required between password changes i.e. the number of days left before the user is allowed to change his/her password
Maximum	The maximum number of days the password is valid (after that user is forced to change his/her password)
Warn	The number of days before password i to expire that user is warned that his/her password must be changed
Inactive	
Expiry	
Optional	Reserved for future use.

Create a user:

Syntax: # useradd <username>

Example: # useradd mist

Check whether user added or not

```
# cat /etc/passwd
```

Create a user using with options

Syntax: `# useradd <options> <username>`

Options	Description
-g	The numerical value of the group's ID. This value must be unique.
-G	makes the user a member of each supplemental group.
-d	assign the home directory for the users
-s	sets default login shell for the user
-c	Uses comment for the user account name

Examples:

```
# useradd –u 1975 mistuser

# useradd –s /sbin/nologin mistuser1

# useradd –d /home/mist1   mist1

# useradd –c trainer mist2
```

Use all options in single useradd command.

```
# useradd –u 2015 –d /opt/mist3 –s /bin/false –c student mist3
```

Assign the user password:-

Syntax: # passwd <username>

Example: # passwd mist

Changing password for user mist

New password: XXXX

Retype new password: XXXX

Passwd: all authentication tokens updated successfully

Modify the user accounts:-

The **usermod** command modifies an existing user account. The **usermod** command modifies the system account files to reflect the changes that are specified on the command line.

Modify User

Syntax: #usermod <options> <existing-username>

Options	**Description**
-l	To change the login name of user account.
-L	To lock the user account.
-U	To unlock the user account.

Note: all options of **'useradd'** command can be used with **'usermod'.**

Examples:

```
# usermod –u 1995 mistuser

# usermod –c sysadmins mist2
```

Change login name

```
Syntax:    # usermod -l <new-username> <old-username>

Example:   # usermod -l mistadmin mist
```

Lock the user account

```
Syntax:    # usermod -L <username>

Example:   # usermod -L mistadmin
```

Unlock the user account

```
Syntax:    # usermod -U <username>

Example:   # usermod -U mistadmin
```

Group Administration:-

Group is a collection of user accounts,every time a new user account is added to the system, a group with the same name is created with the username as its only member. Other users can be added to the group later. One of the purposes of groups is to implement a simple access control to files and other system resources by setting the right permissions on those resources.

Group database files:-

When we are adding a group to the system the credentials of a group will stored in the following database files.

- /etc/group
- /etc/gshadow

/etc/group:-

All groups created under **/etc/group** file, one line per group. The four colon-separated fields of **/etc/group** file as follows.

groupname[1]:passwd[2]:gid[3]:groupmembers[4]

Field	**Description**
groupname	The group’s account name on the system
password	group’s encrypted passwd or an x
gid	group’s numeric primary group ID
groupmembers	user's accounts under the group .

/etc/gshadow

In addition to storing the encrypted password, **/etc/gshadow** file stores password expiration information. As in **/etc/group**, fields are separated by colon (:). It contains the following fields, listed in the order in which they appear in **/etc/gshadow**

Mist[1]:Ep6mckrOLChF.[2]:admin[3]:smith,ramesh[4]

	Field	Description
1.	groupname	The group's account name on the system
2.	Encrypted password	group's encrypted password from /etc/group File.
3.	Admin	list of administrative member's
4.	Group members	user account's in inside the group.

Create a group:

Syntax: # group <groupname>

Example: # groupadd mistgrp

Check whether group added or not

cat /etc/group

Create a group using with options

Syntax: # groupadd <options> <groupname>

Options	Description
-g	--gid,The numerical value of the group's ID. This value must be unique.
-h	--help,display options for groupadd command.
-f	--force,exit successfully if the group already exists and cancel -g if the GID is already used
-K	--key KEY=VALUE,override /etc/login.defs defaults
-o	--non-unique,allow to create groups with duplicate(non-unique) GID
-p	--password PASSWORD,use this encrypted password for the new group
-r	--system,create a system account
-R	--root CHROOT_DIR ,directory to chroot into

Examples:

```
# groupadd -g 1020 accountgrp

# groupadd -h

# groupadd -p $6$pYfDJdZU$vcm3RIgkCDlpERgOaVtmax india
```

Assign For Group:

Syntax: # gpasswd <groupname>

Example # gpasswd accountgrp

check the password add or not

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

Modify the group account's

The **groupmod** command modifies an existing group account. The groupmod command modifies the system account files to reflect the changes that are specified on the command line.

Modify group

Syntax: # groupmod <options> <existing-groupname>

Note: The options same (**-g,-h,-o,-p,-R**) like groupadd

Examples: # groupmod -g 1120 accountgrp

groupmod -o -g 1120 india

Modify the groupname

Syntax: # groupmod -n <new-groupnmae> <old-groupname>

groupmod -n bharat india

Manage group Membership.

The **gpasswd** command manage the group membership of existing group account. The gpasswd command modifies the system account files to reflect the changes that are specified on the command line.

Syntax: # gpasswd <options> <arguments> <groupname>

Options	**Description**
-a	add single user account into the group.
-M	add multiple account's into the group.
-A	add a administrator user for group.
-d	deleting user account's from the group.

Examples: # gpasswd -a mistadmin bharat

gpasswd -A mistadmin bharat

```
# gpasswd -M user1,user2,user3 mistgrp
```

User account's and Group account's deletion.

Delete User account's

Syntax: # userdel <username>

Example # userdel mist3

Delete user account's with data

Syntax: # userdel -r <username>

Example # userdel -r mist2

Delete group account.

Syntax: # groupdel <groupname>

Example: # groupdel accountgrp

Linux File types

There is only 1 command you need to know, which will help you to identify and categorize all the seven different file types found on the Linux system.

```
$ ls -ld <file name>
```

Here is an example output of the above command.

```
 $ ls -ld /etc/services
-rw-r--r-- 1 root root 19281 Feb 14  2012 /etc/services
```

ls command will show the file type as an encoded symbol found as the first character of the file permission part. In this case it is "-", which means "regular file". It is important to point out that Linux file types are not to be mistaken with file extensions. Let us have a look at a short summary of all the seven different types of Linux file types and **ls** command identifiers:

1. **-** : regular file
2. **d** : directory
3. **c** : character device file
4. **b** : block device file

5. **s** : local socket file
6. **p** : named pipe
7. **l** : symbolic link

Regular file

The regular file is a most common file type found on the Linux system. It governs all different files such us text files, images, binary files, shared libraries, etc. You can create a regular file with the **touch** command:

```
# touch mist.com
# ls -ld mist.com
-rw-rw-r-- 1 root root 0 Jan 10 12:52 mist.com
```

The first character of the **ls** command, in this case "-", denotes the identification code for the regular file. To remove a regular file you can use the **rm** command:

```
# rm mist.com
```

Directory

Directory is second most common file type found in Linux. Directory can be created with the **mkdir** command:

```
# mkdir mistdir
# ls -ld mistdir/
drwxrwxr-x 2 root root 4096 Jan 10 13:14 mistdir/
```

As explained earlier, directory can be identified by "d" symbol from the **ls** command output. To remove empty directory use the **rmdir** command.

```
# rmdir mistdir
```

When trying to remove directory with the **rmdir** command, which contains additional files you will get an error message:

```
rmdir: failed to remove `FileTypes/': Directory not empty
```

In this case you need to use a command:

```
# rm -r FileTypes/
```

Character device

Character and block device files allow users and programs to communicate with hardware peripheral devices. For example:

```
# ls -ld /dev/vmmon
crw------- 1 root root 10, 165 Jan  4 10:13 /dev/vmmon
```

In this case the character device is the vmware module device.

Block Device

Block devices are similar to character devices. They mostly govern hardware as hard drives, memory, etc.

```
# ls -ld /dev/sda
brw-rw---- 1 root disk 8, 0 Jan  4 10:12 /dev/sda
```

Local domain sockets

Local domain sockets are used for communication between processes. Generally, they are used by services such as X windows, syslog and etc.

```
# ls -ld /dev/log
srw-rw-rw- 1 root root 0 Jan  4 10:13 /dev/log
```

Sockets can be created by socket system call and removed by the **unlink** or **rm** commands.

Named Pipes

Similarly as Local sockets, named pipes allow communication between two local processes. They can be created by the **mknod** command and removed with the **rm** command.

Symbolic Links

With symbolic links an administrator can assign a file or directory multiple identities. Symbolic link can be though of as a pointer to an original file. There are two types of symbolic links:

- hard links
- soft links

The difference between hard and soft links is that soft links use file name as reference and hard links use direct reference to the original

file. Furthermore, hard links cannot cross file systems and partitions. To create symbolic soft link we can use **ln -s** command:

```
# echo file1 > file1
# ln -s file1 file2
# cat file2
file1
# ls -ld file2
lrwxrwxrwx 1 lubos lubos 5 Jan 10 14:42 file2 -> file1
```

To remove symbolic link we can use **unlink** or **rm** command.

Conclusion

As a system administrator you will mostly work with regular files, directories block and character devices. As a software developer you will also work with local sockets and named pipes.

Permissions:-

As you know, Linux is a multi-user operating system that allows hundreds of users the ability to log in and work concurrently. Also, the operating system has hundreds of thousands of files and directories that it must maintain securely in order to warrant a successful system and application operation from a security standpoint. Given these features, it is imperative for us as system administrators to regulate user access to files and directories, and grant them appropriate rights to carry out their designated functions without jeopardizing system security. This control ofpermissions on files and directories for users may also be referred to as user access rights.

Permission's can be dived into below types

- Basic File Permissions
- Advanced permissions
- Access control list

Basic File Permission's

Permission Groups

Each file and directory has three user based permission groups:

- **owner(u)** - The Owner permissions apply only the owner of the file or directory, they will not impact the actions of other users.
- **group(g)** - The Group permissions apply only to the group that has been assigned to the file or directory, they will not effect the actions of other users.
- **others(o)**- The others permissions apply to all users on the system, this is the permission group that you want to watch the most.

Permission Types

Each file or directory has three basic permission types:

- **read** - The Read permission refers to a user's capability to read the contents of the file.
- **write** - The Write permissions refer to a user's capability to write or modify a file or directory.
- **execute** - The Execute permission affects a user's capability to execute a file or view the contents of a directory.

Permission's	**File's**	**Directories**
Read	To view the contents of the file Ex:- cat,more,less	To list the contents of directories Ex:-ls,ll
Write	Allows to Modify contents of the file Ex:- mv,rm,vi	Tocreate,remove and rename file's & sub-directories Ex:-cp,mv,vi,rm
Excute	To execute the file Ex:- Scripting file	To enter into the directories Ex:-cd

Basic file permissions are using two mode's assign the file/ directory perimissions

. Symbolic mode

. Absloute mode

Symbolic mode: We can assign the permission's to file/directory with using the alphabets(Ex: r,w,x)

Syntax: # chmod <permissions> <file/directory>

Example: # chmod u+rwx,g+rw,o+r mist.txt

Absloute mode: We can assigned the permission's to file/directory using with numerical numbers(Ex 4,2,1)

Synatx: # chmod <permissions> <file/directory>

Example: # chmod 775 mist.txt

Permission's	**Symbolic method**	**Absloute method**
read	r	4
write	w	2
excute	x	1
read,write	rw-	6
read,exute	r-x	5
wrrite,excute	-wx	3
read,write,excute	rwx	7
null	---	0

File/Directory Default perimissions:

File/Directory defaults peermissions depend's on **umask (**user mask**)** value Default umask value : 0022

Full file perimissions : 666

umask value : 022

Default file perimissions : 644

Note: If the file having 777 permissions the file become a script file

Directory full permissions : 777

umask value : 022

Default Directory permission's:755

Umask:

umask (user creation mask value)

the default umask value for root user's : 0022

The default umask value for normal user's : 0002

Change the umask value:

Temporary :

Syntax : # umask <value>

Example: # umask 0033

Permanent :

Edit the /root/.bashrc file

Example: # vi /root/.bashrc

<eof>

umask 0033

then save and quit.

Note: Now check defaults permission for file/directory will be changed permanently.

Change File/Directory permissions

can we Changing file/directory permissions using **chmod** command

Syntax: # chmod <permissions> <file/Directory>

Examples:

chmod 744 /mistdir

chmod a+x filename

chmod ugo=rwx filename

To check wheater the permission's changed or not

ls -l for file

ls -ld for directory

Change owner and group membership :

To change the owner and groups membership for files/directories is mandatory in Linux. Every file is associated with an owner and a group. You can use chown and chgrp commands to change the owner or the group of a particular file/directory.

Change Owner ship:

chown command is user for change owner ship of the file/directory

Syntax: # chown <username> <file/directory>

Example: # chown mist mist.txt

to check wheather the owner ship changed or not

ls -l for file

ls -ld for directory

Change Group Membership:

Chgrp command is used for change group membership

of the file/directory

Syntax: # chgrp <groupname> <file/directory>

Example: # chgrp bharat mist.txt

Change Owner & Group Membership:

chown is also used for change owner and group membership at a time

Syntax: # chown <username>:<groupname> <file/dir>

Example: # chown mist1:bharat /dir1

Advanced permission's:-

Linux having not only basic permissions it having advanced permissions. Advanced permissions is also known special permissions

Normally we have read,write and excute perimissions same in

Advanced perimissions we have three types of perimissions

Basic File Permissions

- SUID
- SGID
- STICKY BIT

SUID(Set UserID):

Normal users doesn't have permissions to run admin [(root) or /sbin and /usr/sbin dirtectories contained] commands . We are applying **suid** permissions to admin command's that commnad's used by all normal user's

Check without suid:

useradd antony

su – antony

$ useradd <username>

$ logout (the user doesn't added)

Check with suid:

```
# which useradd

/usr/sbin/useradd

# ls -ld /usr/sbin/useradd

to apply suid to useradd command

# chmod 4755 /usr/sbin/useradd

then login with antony(any normal) user

# su – antony

$ useradd jhon

the user jhon will be created.
```

SGID(Set GroupID)

When SGID permission is set on a directory, files created/modified in the directory belong to the group of which the directory is a group member .example if a user having write permission in the directory creates a file there, that file is a member of the same group as the directory and not the user's group.

Note: we are applying **sgid** permissions to particular directory.only the group member's can access anything and at the same the group membership automaticall when we create files/sub-directory into that directory.

```
# mkdir /mistdir

# ls -ld /mistdir

# chgrp mistgrp /mist (chaged group membership)
```

```
# ls -ld /mistdir
```

applying **sgid** perimissions

```
# chmod 2755 /mistdir

# ls -ld /mistdir

# cd /mistdir (created some files)

# touch a b c
```

now login with **mistgrp** member's and others users

only mistgrp can modify the files but not outside group members.

Sticky Bit:

We are applying **sticky bit** to file/directory only

root user and owner of the file/directoy can access anything but not others users.

Note: if having full permissions for file/directory other users can't deleted

Without sticky bit

```
# useradd abc1

# useradd abc2

# mkdir /mistltd
```

```
# chmod 777 /mistltd

# cd /mistltd

# touch f1 f2 f3

now login with normal user

# su – abc1

$ cd /mistltd

$ ls

$ touch file1 fil2

$ logout

the normal user can do anything into the directory.
```

With Sticky bit:

```
# ls -ld /mistltd

# chmod 1777 /mistltd

now login with normal user

# su – abc2

$ cd /mistltd

$ rm -rf file2 (can't remove file because the file
                owner is  abc1)

$ logout
```

```
# su – abc1

$ cd /mistltd

$ rm -rf file2 ( file successfully removed )

$ logout
```

Access Control List(ACL):

As a **System Admin**, our first priority will be to protect and secure data from unauthorized access. We all are aware of the permissions that we set using some helpful Linux commands like **chmod**, **chown**, **chgrp**… etc. However, these default permission sets have some limitation and sometimes may not work as per our needs. For example, we cannot set up different permission sets for different users on same directory or file. Thus, **Access Control Lists** (**ACLs**) were implemented.

Adding and Modifing acls:

In Order to use ACL perimissions under linux you need to use two commands setfacl and getfacl.

getfacl: If you get the acls perimissions for particular file/directory using the command **getfacl**.

Syntax: # getfacl <file>/<directory>

Example: # getfacl /linux (create /linux dirtectory)

setfacl: If you set(assign) the acls permissions for particular file/directory using the command **setfacl.**

For Users

Syntax: # setfcal -m u:<username>:<permissions> <file>/<directory>

Example: # setfacl -m u:user1:rw /linux

to check # getfacl /linux

For Groups

Syntax: # setfacl -m g:<groupname>:<permissinos> <file/dirtectory>

Example: # setfacl -m g:mistgrp:rwx /linux

to check # getfacl /linux

Remove the acls permissinos:

For File or directory:

Syntax: # setfacl -b <file/directory>

Example # setfacl -b /linux

For User:

Syntax: # setfacl -x u:<username> <file/directory>

Example: # setfacl -x u:user1 /linux

For Group:

Syntax: # setfacl -x g:<groupname> <file/directory>

Example: # setfacl -x g:mistgrp /linux

Configuring sudo Access:

The **sudo** command offers a mechanism for providing trusted users with administrative access to a system without sharing the password of the **root** user. When users given access via this mechanism precede an administrative command with **sudo** they are prompted to enter their own password. Once authenticated, and assuming the command is permitted, the administrative command is executed as if run by the **root** user.

Follow this procedure to create a normal user account and give it sudo access. You will then be able to use the **sudo** command from this user account to execute administrative commands without logging in to the account of the **root** user.

Sudo (super user do command) using with sudoers file we can make normal user will act as root user temporarily.

Configuration file for sudo - **/etc/sudoers,** to edit this configuration

file using **visudo** command. This command is accessed by only super user.

The most two advantages using sudo command are:

- Restricted privileges
- Logs of the actions taken by users

Backup & Restore and File compression tools

File compression is a routine task for most of the Administrators and normal users, to save disk space and to move data from one location to another safer location, this compression utility is used. from historical point of view tar utility was developed to get sequential data backup and it was stored in magnetic tape drives .**tar** is the standard UNIX/Linux archiving application tool. In its early stage it used to be a Tape Archiving Program which gradually is developed into General Purpose archiving package which is capable of handling archive files of every kind. tar accepts a lot of archiving filter with options.

Tar(Tape Archive):

Syntax: tar <options> <destination file path.with extension> <source file/directory path>

Options

-c : Create a new archive file.

-**f** **:** use archive file or device

-**j** : bzip2 the archive

- **-r** : append files to existing archives.
- **-t** : list contents of existing archives.
- **-u** : Update archive
- **-x** : Extract file from existing archive.
- **-z** : gzip the archive
- -**J** : xz the archive

-**v** : (verbose) verbosely list files processed

- **-A** : Append tar files to existing archives.
- **-d** : Compare archive with Specified filesystem.
- **–delete** : Delete files from existing archive.

Example:

To crate backup file

```
# tar -cvf /tmp/mistfile.tar /etc/passwd
# cd /tmp
# ls
# tar -tvf mistfile.tar
```

To Restore the file & directories

```
# tar -xvf mistfile.tar

# ls
```

```
# cat /etc/passwd
```

To crate a backup file with compress using gzip

```
# tar -zcvf  /opt/marc.tar.gz /usr/local

# cd /opt

# ls

# du -sh marc.tar.gz
```

To Restore files and Directories

```
# tar -xvf marc.tar.gz

# ls
```

Note : Similar to bzip2 and xz

cpio

Cpio is an copy input and output, it read a list of file name line by line in input and archive files in output. Its a classic command .

Syntax: cpio <options> > <destionation file path.cpio>

Options

- **o** : create archive file
- **v** : verbose
- **i** : extract archive file
- **d** : crate directories

Example

```
# mkdir /testdir
# cd /testdir
```

```
# touch abc{1..5}

# ls

# ls | cpio -ov >/opt/backup.cpio
```

Un compress the cpio

```
# cd /opt

# cpio -idv <backup.cpio
```

Compress tools

gzip:

The gzip tool is most popular and fast file compression utility in Linux. Gzip tool keep original file name the extension of compressed file .gz and time stamp. Usually comes as inbuilt package for all distributions but can be install easily.

Syntax: # gzip <filename>

Example:# gzip mistfile

```
# ls

# du -sh mistfile.gz
```

Bzip2 :

Bzip2 utility perform more faster then gzip, it compress files and folders more compactly. It required more RAM during

compressing files, to reduce memory consumption.It can compress multiple files at a time.

Syntax:bzip2 <filename> <filename>

Ex:# bzip2 linux

#ls

#du -sh linux.bz2

XZ:

XZ is successor of the lzma utility, it can only compress single file but can not compress multiple file in a single command. It can not be supported on older Linux versions, but it is provided with all of the latest Linux operating systems, it will addd .xz extension to compression file automatically.

Syntax:# xz <file-name>

EX:xz /etc/passwd

cd /etc

ls

du -sh passwd.xz

uncompress tools:

for gzip:

syntax:# gzip -d <file-name.gz>

(or)

```
        # gunzip <file-name.gz>
```

EX:# gzip -d mistfile.gz

ls

du -sh mistfile

for bzip2:

syntax:# bzip2 -d <file-name.bz2>

(or)

bunzip2 <file-name.bz2>

Ex:# bunzip2 linux.bz2

ls

du -sh linux

for XZ:

syntax: # xz -d <file-name.xz>

(or)

unxz <file-name.xz>

Ex:# xz -d /etc/passwd.xz

Managing Partitions with File system:

Data is stored on disk drives that are logically divided into partitions. A partition can exist on a partion of a disk, on an entire disk,or it may span multiple disks. Each partition may contain a file system,a raw data space, a swap space, or a dump space

Disk partitioning is the creation of separate divisions of a hard disk drive using partition editors such as **fdisk**. Once a disk is divided into several partitions, directories and files of different categories may be stored in different partitions.

Many new Linux sys admin (or Windows admin) create only two partitions / (root) and swap for entire hard drive. This is really a bad idea. You need to consider the following points while partitioning disk.

Purposes for Disk Partitioning

Linux can be installed on a single, unpartitioned hard disk. However, the ability to divide a hard disk into multiple partitions offers some important advantages to install multiple partitions in single harddrive.

- **Ease of use** – Make it easier to recover a corrupted file system or operating system installation.
- **Performance** – Smaller file systems are more efficient. You can tune file system as per application such as log or cache files. Dedicated swap partition can also improve the performance (this may not be true with latest Linux kernel 2.6).
- **Security** – Separation of the operating system files from user files may result into a better and secure system. Restrict the growth of certain file systems is possible using various techniques.
- **Backup and Recovery** – Easier backup and recovery.
- **Stability and efficiency** – You can increase disk space efficiency by formatting disk with various block sizes. It depends upon usage. For example, if the data is lots of small files, it is better to use small block size.
- **Testing** – Boot multiple operating systems such as Linux, Windows and FreeBSD from a single hard disk.

Types of Disks

Different type of disks will be having different initials in Linux

IDE drive display with **/dev/hda**

/dev/hda indicate for master disk

/dev/hdb indicate for slave disk

SCSI drive display with **/dev/sda**

/dev/sda indicate for master disk

/dev/sdb indicate for slave disk

Virtual drive display with **/dev/vda**

/dev/vda indicate for master disk

/dev/vdb indicate for slave disk

Partitions:

When a hard drive is installed in a computer, it must be partitioned before you can format and use it. Partitioning a drive is when you divide the total storage of a drive into different pieces. These pieces are called partitions. Once a partition is created, it can then be formatted so that it can be used on a computer.

Partitioning :

Disk partitioning or **disk slicing** is the creation of one or more regions on a hard disk or other secondary storage so that an operating system can manage information in each region separately.

Partition Types:

Primary, Extended, and Logical Partitions:

Primary partition is a partition that is needed to store and boot an operating system, In general, you would install the operating system in a primary partition.

A disk may contain up to four primary partitions (only one of which can be active), or three primary partitions and one extended partition. In the extended partition, the user can create logical drives. Logical partitions are the partitions that are created in the extended partition area.

A disk can be used as a simple entity or broken up into one or more partitions. Disks are generally called /dev/sda, /dev/sdb, etc., in physical servers (s for scsi even though they've got IDE, SATA or SAS interfaces) and /dev/vda, /dev/vdb, etc, in virtual machines.

Partitions get their names from the disk name itself and add a number starting at 1 (/dev/sda1, /dev/sda2,..etc. or /dev/vda1, /dev/vda2,.. etc).

A **partition table** is a special structure containing partitions organization.

MBR partition table (**MBR** stands for Master Boot Record). This organization allows for **4 primary** partitions only. If you want more than that, you need to create

an **extended** partition (using one of the 4 primary slots), and then create **logical** partitions inside.

fdisk:

fdisk stands (for "**fixed disk** or **format disk**") is an most commonly used command-line based disk manipulation utility for a **Linux/Unix** systems. With the help of fdisk command you can view, create, resize, delete, change, copy and move partitions on a hard drive.

Commands:

To View all Disk Partitions:

#fdisk –l

To View Specific Disk Partition in Linux:

#fdisk –l /dev/vdb

To Check all Available fdisk Commands:

#fdisk /dev/vdb

Command (m for help): m (type m)

Command action

a -toggle a bootable flag b - edit bsd disklabel

c - toggle the dos compatibility flag d - delete a partition

g - create a new empty GPT

partition table G - create an

IRIX (SGI) partition table

l - list known

partition types m-

print this menu

n- add a new partition

o- create a new empty DOS parti-

tion table p - print the parti-

tion 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.

Creating a new Partition:

Steps:

1. #fdisk /dev/vdb (now a command mode will start)
2. Type ‘m’ in command mode(for help if needed).

3. Type ‘p’ in command mode to print specific /dev/sda partitions
4. Type ‘n’ to in command mode create new partition.
5. Select option will appear now

   i. Choose ‘p’ for primary partition and(choose available partition no.(1-4)).
   ii. Choose ‘e’ to create extended partition and(choose available partition no.(1-4)).
   iii. Choose any one in steps i, ii at a time.
   iv. First sector leave as default so, press enter key.
   v. Last sector choose size of partition here.(eg: +1G for 1GB, +500M for 500MB.)
   vi. Now partition will be created enter ‘w’ for save and quit. (or ‘q’ for quitting without saving changes).

6. Now we have use a command called ‘partprobe’ to inform os kernel about partition table changes. partprobe is a program that informs the operating system kernel of partition table changes, by requesting that the operating system reread the partition table. For instance, if you create a new partition on one of your disks using parted or fdisk you should run partprobe afterwards to make the kernel aware of the new partition configuration.

```
#partprobe
```

7. If wanna check partition updated to os or not, can view partitions file present in /proc directory.

Create a Filesystem on the New Partition:

After partitioning we have to create a file system on our new partition in order to use that one.

In Redhat-7 default file system was xfs. In ext4,ext3 and ext2 are default file systems in Redhat 6,5 and 4 respectively.

#mkfs command was used to build a linux file system on a device, usually on a hard disk partition.

- #mkfs. (press double TAB) to see available file systems (or)
- #mkfs –t command also returns available file systems

follow below syntax to create file system

- #mkfs –t <filesystemtype> <partition name> Example:

#mkfs –t xfs /dev/vdb1 (/dev/vdb1 is name of partition in my case, and xfs file system) After formatting partition we have to mount that to a mount point.

Mounting:

By using #mount command we can mount partitions to a mount point. Temporary Mounting:

- #mkdir /tempmnt
- #mount /dev/vdb1 /tempmnt

  After rebooting our system we loose our partition if we mount temporarily.

Permanent mounting:

To mount a partition permanently we have write some entries in /etc/ fstab file.

fstab file: fstab is a configuration file that contains information of all the partitions and storage devices in your computer. It has 6 fields. And each field has its importance.

1. First field: device/partition name (eg. /dev/sdb3)
2. Second field: mount point path (eg. /mymntpoint)
3. Third field: File system format(eg.xfs,ext4,etc.)
4. Fourth field: Mount options (use defaults option if not know about different mount options like sync,async, ro,rw, user and nouser , exec and no exec, auto and noauto)
5. Fifth field: Dump value (dump is a backup utility, dump will ignore a filesystem if we insert 0(zero) as dump value).
6. Sixth field: Filesystem check order(fsck), it checks file system consistency at booting process, if we insert 0 here fsck won't check the file system.

Ⓟ Adding our partition to

fstab file: #vim

/etc/fstab

```
/dev/vdb1 /mymntpoint   xfs   defaults   0   0
```

:wq(save and quit)

#mount -a (it mounts all file system that are present in /etc/fstab file, shows error if don't insert fields correctly.)

follow below syntax to create file system

Ⓟ #mkfs –t <filesystemtype> <partition name> Example:

#mkfs –t xfs /dev/vdb1 (/dev/vdb1 is name of partition in my case, and xfs file system) After formatting partition we have to mount that to a mount point.

Mounting:

By using #mount command we can mount partitions to

a mount point. Temporary Mounting:

- #mkdir /tempmnt
- #mount /dev/vdb1 /tempmnt

  After rebooting our system we loose our partition if we mount temporarily.

Permanent mounting:

To mount a partition permanently we have write some entries in /etc/ fstab file.

fstab file: fstab is a configuration file that contains information of all the partitions and storage devices in your computer. It has 6 fields. And each field has its importance.

7. First field: device/partition name (eg. /dev/sdb3)
8. Second field: mount point path (eg. /mymntpoint)
9. Third field: File system format(eg.xfs,ext4,etc.)
10. Fourth field: Mount options (use defaults option if not know about different mount options like sync,async, ro,rw, user and nouser , exec and no exec, auto and noauto)
11. Fifth field: Dump value (dump is a backup utility, dump will ignore a filesystem if we insert 0(zero) as dump value).
12. Sixth field: Filesystem check order(fsck), it checks file system consistency at booting process, if we insert 0 here fsck won't check the file system.

Ⓟ Adding our partition to

fstab file: #vim

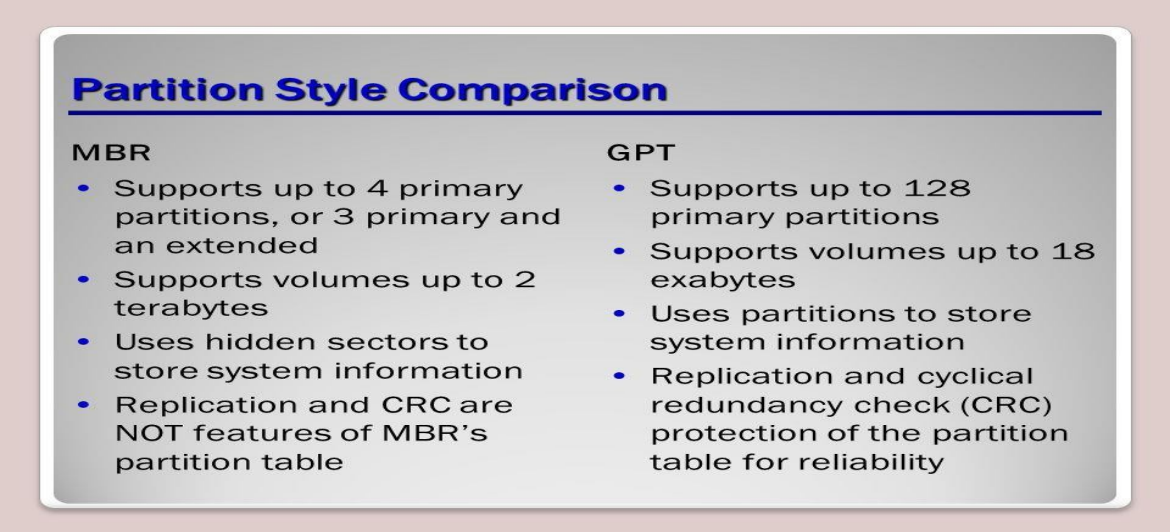
/etc/fstab

```
/dev/vdb1 /mymntpoint   xfs   defaults   0   0
```

:wq(save and quit)

#mount -a (it mounts all file system that are present in /etc/fstab file, shows error if don't insert fields correctly.)

MBR Partition Table	GPT Partition Table
Stands for "Master Boot Record"	Stands for GUID Partition Table (GUID - Globally Unique identifier)
Introduce by IBM For DOS PC in 1980	Introduce By LINUX
It is OLD Partition Table System	It is New Partition Table System
Work with 2 TB (or less) (Cannot handle disks with more the 2 TB)	It will work on less then or more than 2 TB (It has no MBR limitation)
It support 4 Partitions	It Support 128 Partition's
In MBR no way to know about DATA Corruption	In GPT , you can notice the problem and able to attempt it recover
DATA cannot be easily recovered	DATA can be easily recovered



Logical volume management (LVM):

Logical volume management (LVM) is a form of storage virtualization that offers system administrators a more flexible approach to managing disk storage space than traditional partitioning.

LVM makes it easier to manage disk space. If a file system needs more space, it can be added to its logical volumes from the free spaces in its volume group and the file system can be re-sized as we wish. If a disk starts to fail, replacement disk can be registered as a physical volume with the volume group and the logical volumes extents can be migrated to the new disk without data loss.

In a modern world every Server needs more space day by day for that we need to expand depending on our needs. A Physical Disk will be grouped to create a volume Group. Inside volume group we need to slice the space to create Logical volumes. While using logical volumes we can extend across multiple disks, logical volumes and we can reduce logical volumes in size with some commands without reformatting and re-partitioning the current disk. Volumes can stripes data across multiple disks this can increase the I/O stats.

In-order to create LVM's we have to follow below steps:

1. Create some partitions with available disk space (in lab /dev/ vdb)

2. Create physical volumes (in lab /dev/vdb1, /dev/vdb2,/dev/vdb3)

3. With the help of physical volumes cre-

   ate a volume group (in my case I'm

   naming as 'myvg1')

4. In created Volume groups we have to cre-

   ate Logical Volumes (I'm naming them as

   'mylv1', 'mylv2' ,.....).

5. After creating LVM's we have to format and mount them to a mount point to use.

Creating LVM Disk Storage in Linux:

Before creating LVM's we can see Physical volumes(pv), Volume Groups(vg) and LVM's By using following Commands.

1. #pvs or #pvdisplay (to check available physical volumes)

2. #vgs or #vgdisplay (to check volume groups)

3. #lvs or #lvdisplay (to check logical volumes)

Creating Physical Volumes:

Before Creating Physical volumes we have create partitions in our new disk by using fdisk/parted commands.

Partitioning:

1. #fdisk –l (to list available partitions)
2. #fdisk /dev/vdb
3. Type ‘n’ in command mode to create new partition
4. Choose ‘p’ to create a primary partition.
5. Choose which number of partition we need to create (in between 1-4). (/dev/vdb1)
6. Press ‘Enter key’ in first sector.
7. In last sector give 4MB extra than size of partition that you wanted to give. (eg. If you wish to give 1GB(1024MB) size then give as 1028MB and that 4MB is for lvm metadata.)
8. We need to change the type of newly created partition type by using ‘t’.
9. Choose partition number that u want to change . (eg. 1)
10. Here we need to change the type, we need to create LVM so we going to use the hexa code of LVM as ‘8e’ and if u don’t know the code then press ‘l’ then we can see list of available codes.
11. Print ‘p’ the Partition what we created to just confirm.
12. Here we can see the ID as 8e LINUX LVM.
13. ‘w’ for Write the changes and exit fdisk.
14. #partprobe (to update partition table changes to os/kernel)

Now we have '/dev/vdb1', create another one or two partitions(.i.e. /dev/vdb2,/dev/vdb3..) by following above steps.

Creating Physical Volumes:

By using #pvcreate command we can create physical volumes.

1. #pvcreate /dev/vdb1 /dev/vdb2 /dev/vdb3
2. #pvs or #pvdisplay to check pv's

Creating Volume Group:

Now we can create Volume groups by using 'vgcreate' command.

1. #vgcreate myvg1 /dev/vdb1 /dev/vdb2 ('myvg1' is vg name that I'm given)
2. #vgdisplay (to check volume group)
3. #vgextend (to extend vg and vgreduce also

   available to reduce size) #vgextend

   myvg1 /dev/vdb3

Creating LVM's:

By using 'lvcreate' command we can create lvm's in an existing group.

1. #lvcreate -L +1G myvg1 -n mylv1
2. #lvdisplay (to check lvm's) i.e. /dev/myvg1/mylv1 was created.

Like this we can create lvm's

Mounting LVM (xfs):

To use 'mylv1' (lvm we created) we have to format and mount it.

- #mkfs.xfs /dev/myvg1/mylv1
- #mkdir /part1
- #vim fstab

  /dev/myvg1/mylv1 /part1 xfs defaults 0 0

  :wq(save&quit).

- #mount –a

Extending LVM:

By using 'lvextend' command we can extend lvm size.

1. Run #vgdisplay command to check if space available or not.
2. #lvextend –L +500M /dev/myvg1/mylv1
3. #lvdisplay will show the space added. (but space will not be added to filesystem, we can check that with command '#df -hT').
4. So, we use another command 'xfs_growfs' to expand existing 'xfs'

filesystem.

5. #xfs_growfs /dev/myvg1/mylv1

6. Now #df -hT will show the newly extended space. **(NOTE: We can not reduce the size of xfs filesystem partitions or logical volumes)**

Creating LVM with ext4 filesystem:

1. #lvcreate -L +500M myvg1 -n mylv2
2. #lvdisplay
3. #mkfs.ext4 /dev/myvg1/mylv2
4. #mkdir /part2
5. #vim /etc/fstab

   /dev/myvg1/mylv2 /part2 ext4 defaults 0 0

   :wq

6. #mount –a

Extending LVM (ext4):

1. #vgdisplay (to make sure space available)
2. #lvextend -L +300M myvg1 /dev/myvg1/mylv2
3. #lvdisplay (will show but #df –hT will not show added space.)
4. #resize2fs is a program to resize ext2,ext3

and ext4 filesystems. #resize2fs /dev/

myvg1/mylv2

5. Run #df -hT command to verify space added.

Reducing LVM size:

We can reduce the size of ext filesystem formatted LVM's

To reduce the size of the LVM typically we should follow below six steps:

1. #umount /dev/myvg1/mylv2 (unmounting mylv2)
2. #e2fsck -f /dev/myvg1/mylv2 ('e2fsck' is used to check ext family filesystems and '–f' for force checking even the file system is clean).
3. #resize2fs /dev/myvg1/mylv2 600M (after reducing mylv2 size will be 600M)
4. #lvreduce -L 600M /dev/myvg1/mylv2 (now size reduced to 600M)
5. #resize2fs /dev/myvg1/mylv2
6. #mount /dev/myvg1/mylv2 (mounting mylv2)

Removing LVM's:

We can remove lvm's by following steps;

1. #umount /dev/myvg1/mylv1 and /dev/myvg1/mylv2
2. Delete entries related to mylv1 and mylv2 in /etc/fstab
3. #mount –a

4. #lvremove /dev/myvg1/

mylv1 #lvremove /dev/

myvg1/mylv2 Removing VG's:

- #vgremove myvg1

Removing PV's:

- #pvremove /dev/vdb1 /dev/vdb2 /dev/vdb3

SWAP:

Swap space/partition is space on a disk created for use by the operating system when memory has been fully utilized. It can be used as virtual memory for the system; it can either be a partition or a file on a disk.

The swap space is the hard disk space which is used to supplement the system RAM by holding idle memory pages. When the kernel runs out of memory, it can move idle/inactive processes into swap creating room for active processes in the working memory. This is memory management that involves swapping sections of memory to and from virtual memory.

We can create swap space/partition by using fdisk/parted commands and we can create swap files with fallocate and dd commands.

SWAP By Using fdisk:

1. #fdisk /dev/sdb
2. Press 'n' for new partition
3. Choose 'p' for primary partion
4. Choose partition number
5. Press enter key at first sector
6. Give size at last sector
7. Now press 't' to change partition's system id
   Here enter '82' as code bcoz swap identifier code is 82(type 'l' to list the types).
8. Press 'w' for save and quit.

Now we a have partition (/dev/sdb1)

Now we can convert that partition into the swap space by following commands.

i. #mkswap /dev/vdb1
ii. #swapon /dev/vdb1
iii. #blkid /dev/vdb1 (here copy the generated 'uuid')
iv. #vim /etc/fstab
uuid=<paste copied uuid here> swap swap defaults 0 0

:wq

v. #mount –a

We can check/verify swap size by using a command

called 'free'. We can check/verify swap size at

/proc/meminfo file also.

Creating Swap File:

By using 'dd' command. First check at which partition have space by using "df -hT" command.

Now run this command to create a file of 1GB with

"dd" command. #dd if=/dev/zero of=/swap_file

bs=1GiB count=1

if=input file

/dev/zero is a special file in linux which returns

endless zero bytes. of=output file

/swap_file is file name given in

our case bs=size of bytes

count=(bs value * given_count value We have a 1GB /swap_file file.

Follow below steps to Convert this file into Swap space.

1. #chmod 600 /swap_file
2. #mkswap /swap_file
3. #swapon /swap_file
4. Generate uuid with command #blkid /swap_file
5. Copy and add UUID in /etc/fstab file as below
   UUID=<paste copied uuid here> swap swap defaults 0 0
6. Run #mount – a command

Now 1GB /swap_file was converted to Swap space to verify use #free command.

Creating file with 'fallocate' command

fallocate - preallocate or

deallocate space to a file

#fallocate –l 1GiB /swapfile

Here A /swapfile called 1GB file created and we can use that file as swap space by following above procedure we used for /swap_file which was created by using dd command.

scheduling jobs

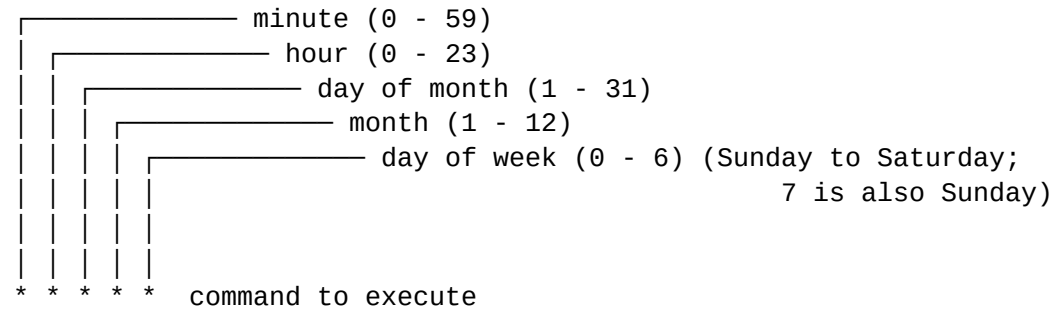
In Linux operating system we can schedule jobs automatically by using **cronjobs.**

It is used to reduce the administrative task in Unix and Unix-like operating systems and used to schedule commands to be executed periodically.

We can automate the process like backup,schedule updates,synchronization of files like more.cron is a deamon to run schedule tasks.

Generally, crontab uses a daemon, crond, which runs constantly in the background and checks once a minute to see if any of the scheduled jobs need to be executed. These jobs are generally referred to as **cron jobs**. Cron jobs run as the user who creates them, as though that user typed the command into their shell.

The Crontab configuration file is **/etc/crontab** and this file contains 6 fields



Important files related to cron:

/etc/cron.monthly – This file is used to scheduling the jobs for monthly

/etc/cron.weekly – This file is used to scheduling the jobs for weekly

/etc/cron.daily – This file is used to scheduling the jobs for daily

/etc/cron.hourly – This file is used to scheduling the jobs for hourly

/etc/cron.deny – This file is used to restrict users to scheduling the jobs

Note: By default all users can scheduling jobs using crontabs

if one user have entries in two files [i.e /etc/cron.deny and /etc/cron.allow] by defaults system choose /etc/cron.allow file.

To scheduling the cronjobs

syntax:-# crontab -e

ex:scheduling job at 5:30 everday;

[mintues] [hours] [DoM] [MoY] [DoW] [Cmd]

30 05 * * * echo“Welcome to Mist” >> /dev/pts/0

:wq

to check all asigned cronjobs

crontab -l

To remove all the scheduling jobs

crontab -r

To schedule cronjobs for particular user

crontab -u <username> -e

To Check cronjobs for particular user

crontab -u <username> -l

To Remove cronjobs for particular user

crontab -u <username> -r

options

***-** means for every

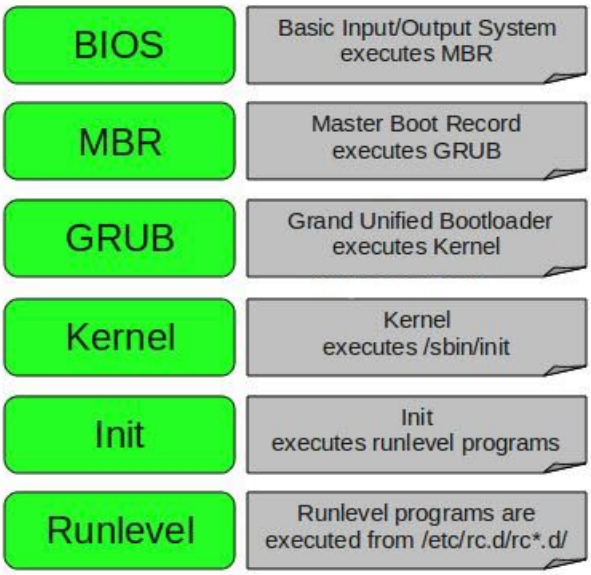
*/2-for every two

All assigned cronjobs details stored under /var/spool/cron

Booting Procedure

booting procedure?

In Linux systems the booting is done in stages.



BIOS :

BIOS stands for Basic Input and Output System. Whenever we power on the system , the system runs self diagnostic checks and detects all the connected input and out peripherals. This process is called POST (Power On Self Test). If any errors found it displays on the screen. Then BIOS locates the booting disk in the system and

locates and loads the Primary boot loader nothing but MBR (Master Boot Record) into the memory. So, in simple terms the BIOS loads the MBR into memory and executes the MBR.

MBR :

MBR stands for Master Boot Record. It is located in the 1st sector of the bootable disk (it may be /dev/hda or /dev/sda). The size of the MBR is 512 bytes and it contains three components.

(i) Primary boot loader information and its size is 446 bytes.

(ii) Partition table information and its size is 64 bytes.

(iii) MBR validation check and its size is 2 bytes. Its main purpose is whether the MBR is valid or not.

The primary boot loader contains the secondary boot loader nothing but GRUB or LILO (in old systems).

Then primary boot loader locates and loads the secondary boot loader into memory.

So, in simple terms the MBR loads and executes the GRUB boot loader.

GRUB or LILO :

GRUB stands for Grand Unified Boot loader. LILO stands for Linux Loader and is used in old Linux systems. If we have multiple kernel images installed in our system, we can choose which one to be

executed. GRUB displays a splash screen, waits for few seconds. If we do not enter anything, it loads the default kernel image as specified in the grub configuration file. GRUB has the knowledge of the file system (the old LILO didn't understand the system). GRUB configuration file is **/boot/grub/grub.conf** (**/etc/grub.conf** is a link to this). This file contains kernel and initrd images. So, in simple terms GRUB just loads and executes kernel and initrd images.

Kernel :

Kernel initialises itself and loads the kernel modules and mounts the root file system as specified in the "root=" in grub.conf and then kernel executes the **/sbin/init** program. Since init was the 1st program to be executed by Linux kernel, it has the process ID (PID) of 1. We can see this id by **# ps -ef | grep init** command. initrd stands for initial RAM Disk. initrd is used by kernel as temporary file system until kernel is booted and the real root the file system is mounted. It also contains necessary drivers compiled inside which helps it to access the hard drive partitions and other hardware.

init level :

In this init program reads the **/etc/inittab** file and put the system into specified run level. init identifies the default run level from **/etc/inittab** file and we can change the this default run level whenever we needed. We can find the default run level by **# grep "initdefault" /etc/inittab** command on our system. Normally the default run level in Linux is 3 in CLI (Command Line Interface) mode and 5 in GUI (Graphical User Interface) mode.

Run Level Programs :

The following run levels are available in Linux systems.

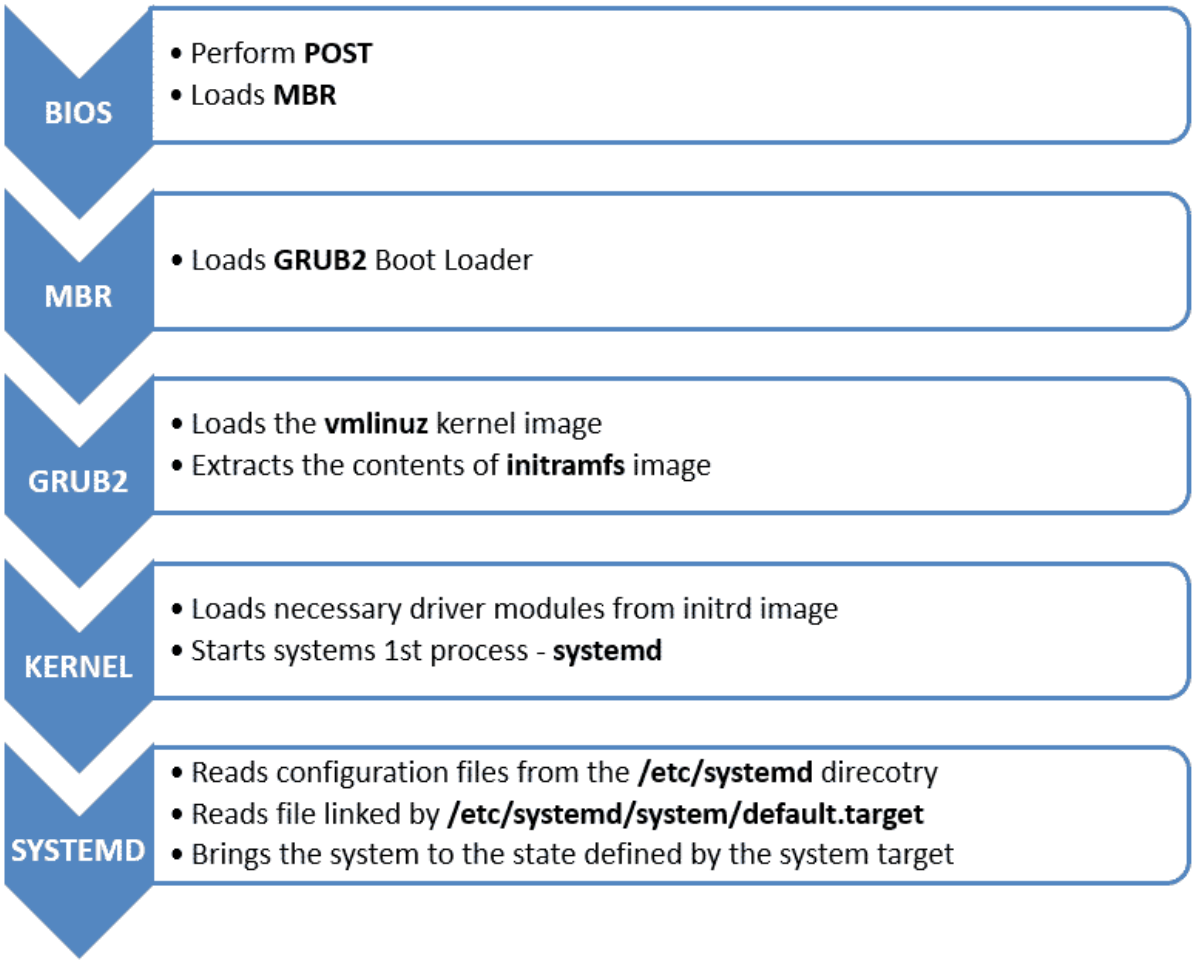
0 -----> halt or shutdown the system
1 -----> Single user mode
2 -----> Multi user without NFS
3 -----> Full multi user mode but no GUI and only CLI mode
4 -----> Unused
5 -----> Full multi user mode with GUI (X11 system)
6 -----> reboot the system

Whenever we start the Linux system is booting we can see various services getting started. Those services are located in different run levels programs executed from the run level directory as defined by our default run level. Depending on our default init level setting, the system will execute the programs from one of the following directories.

Run level 0 -----> /etc/rc.d/rc0.d
Run level 1 -----> /etc/rc.d/rc1.d
Run level 2 -----> /etc/rc.d/rc2.d
Run level 3 -----> /etc/rc.d/rc3.d
Run level 4 -----> /etc/rc.d/rc4.d
Run level 5 -----> /etc/rc.d/rc5.d
Run level 6 -----> /etc/rc.d/rc6.d

The above directories are also having symbolic links available for those directories under **/etc/rc0.d, /etc/rc1.d,** ....etc., So, the /etc/rc0.d is linked to /etc/rc.d/rc0.d

Booting procedure in RHEL – 7:



Upto kernel the booting process is same as the above.
/boot/grub2/grub.conf is the GRUB configuration file in RHEL - 7.
systemd is the initial process in RHEL - 7 and its process ID is **1**.
linux16 read the root (/) file system and then **initrd16**process will mount the root (/) file system in read & write mode and starts the **systemd**process. And the **systemd** process will read the **/etc/fstab** file and mount all the file systems. Then it reads the file **/etc/systemd/system/default.target** file and brings the system into the default run level according to the scripts the processes will start or stop.

How to check the current run level of the system?
who -r (to see the present run level of the system)
How to change the default run level?

First open the **/etc/inittab** file by **# vim /etc/inittab** command and go to last line change the run level number as we required and then reboot the system by **# init 6** command. After rebooting the system check the current run level by **# who -r** command.

difference between # reboot and # init 6 commands?

Both commands are used to restart or reboot the system.

 # reboot command will not send the kill signals to the system and it will kill all the running processes and services forcefully and then restart the system.

 # init 6 command will send the kill signals to the system and it will stop all the processes and services one by one and then restart the system.

console port and how to connect to the console port?

Console port is used to connect the system even though the system is not booted with the main O/S. This port is used to connect the system for troubleshooting purpose only. We can connect the console port as same as connect to systems LAN port and it is also having IP address, user name and password to connect to the console.

There are different types of console ports for different types of servers. They are given below.

Server Name	Name of the Console port	Expansion name
DELL	DRAC or i-DRAC	DRAC ---> DELL Remote Access Controllers i-DRAC ---> Integrated DELL Remote Access Controllers
IBM Power series	HMC	Hardware Management Console
HP	ILO	Integrated Light Out

differences between run level 2 and run level 3?

Run Level 2 :

(i) It supports multiuser operations.

(ii) Multiple users can access the system.

(iii) All the system deamons will run except NFS and some other network service related deamons.

(iv) So, without NFS we can use all other services.

Run Level 3 :

(i) It is also supports Multi user operations.

(ii) Multiple users can access the system.

(iii) All the system deamons including NFS and other network related service deamons will run.

(iv) So, we can avail all the services including NFS also.

Package Management:-

The Red Hat software management system was originally called Red hat Package Manager but is now known a **RPM Package Manager (RPM)**. RPM also refers to one or more files that are packaged together in a special format and stored in files with the .rpm extension. These rpm files (also called rpms, rpm packages, or packages) are manipulated by the RPM package management system. And another utility is **Yellowdog Updater, Modified** (**yum**) and is an interactive, automated update program which can be used for maintaining systems using rpm

RPM:

Rpm is a powerful Package Manager for Red Hat Linux. It can be used to build, install, query, verify, update, and remove/delete individual software packages. A Package consists of an archive of files, and package information, including name, version, and description. But using with rpm it will install only Packages but not all Dependencies. And no need to create any repository file.

Syntax : rpm <options> <Package-name>

Options:

q – querying the Package

i – install the Package

v – verbose

h – hash able

e – remove /erase Package

U – update the Package

qa – querying all (q-quire a-all)

qi – querying information

qc – querying with configuration files

- Querying / verifying Package

Syntax: rpm -q <Package-name>

Ex: #rpm -q openssh

Syntax: rpm -qa | grep <Package-name>

Ex: #rpm -qa | grep openssh

Verifying all available Packages

Syntax: rpm -qa

To know the information of the Package

Syntax: rpm -qi <Package-name>

Ex: rpm -qi httpd

To Install the Package

Syntax: rpm -i <Package-name>

Note: Full Package name required and will not get any confirmation for that use (vh) options

Ex: rpm -i httpd-2.4.6-45.el7.redhat.x86_64

Ex: rpm -ivh httpd-2.4.6-45.el7.redhat.x86_64

To Install Package without Dependencies

Syntax : rpm -ivh <Package-name> --nodeps

Ex: rpm -ivh httpd-2.4.6-45.el7.redhat.x86_64

To remove Package

Syntax: rpm -e <Package-name>

Ex: rpm -e httpd-2.4.6-45.el7.redhat.x86_64

To remove Package with dependencies

Syntax: rpm -ivh <Package-name> --deps

Ex: rpm -ivh httpd-2.4.6-45.el7.redhat.x86_64 --deps

Ex: rpm -evh httpd-2.4.6-45.el7.redhat.x86_64

To Update the Package

Syntax: rpm -Uvh <Package-name>

Ex: rpm -Uvh openssh-*

YUM

The yum command (yellowdog updater, modified) is the front-end to the rpm command and is the preferred tool for package management.Yum is a powerful Package Manager for Red Hat Linux. It can be used to build, install, query, verify, update, and remove/delete individual software packages.A Package consists of an archive of files, and package information, including name, version, and description. By using yum we can install packages along with dependencies and we need to create repository file.

Yum configuration file /etc/yum.repos.d

Syntax: yum <options> <package-name>

To check all packages list

Syntax:yum list all

To install package

Syntax:yum install <package-name>

Ex:yum install httpd

To install the package ask without confirmation

Syntax:yum install <package-name> -y

Ex:yum install elinks -y

To remove the package

Syntax:yum remove <package-name>

Ex:yum remove httpd

To update all packages

Syntax:yum update

To update the particular package

Syntax:yum update <package-name>

Ex:yum update openssh

To know the information of the package

Syntax:yum info <package-name>

Ex:yum info vsftpd

To check the repo list

Syntax:yum repolist

To search the exact package

Syntax:yum search <package-name>

Ex:yum search vsftpd

To create a repository file /yum client configuration:

#cd /etc/yum.repos.d

#ls

Note:rename or remove existing repo files

To create a repo file using yum-config-manager

Syntax: # yum-config-manager -–add-repo=”baseurl”

Ex: #yum-config-manager–-add-repo=”http://content.example.com/rhel7.0/x86_64/dvd”

vim content.example.com.repo

<eof>

gpgcheck=0

```
:wq

# yum clean all

# yum update

# yum repolist
```

To create a yum repository file manually

Note: Remove or rename existing repository file.Create a new repo file using with any name with extension **.repo**

```
# vim mist.repo

[repoid]

name=mist repo

baseurl=http://content.example.com/rhel7.0/x86_64/dvd

enabled=1

gpgcheck=0

:wq

#yum clean all ; yum repolist
```

Managing services and Daemones

These are programs or processes which are run at boot time. Some remain in memory to execute various tasks when required (daemons). Most are started and stopped with scripts in the /usr/lib/systemd/system directory. The exact contents of this directory will depend on which packages from a particular distribution are installed. For example, installing the Apache package will cause an httpd script to be placed in /usr/lib/systemd/system.

Services:

service is a shell script available on Redhat systems which allows you to perform various tasks on services. It performs the starting and stopping of services temporally (with in the session), these setting are not saved. If you start Apache this way but it is not set to start on boot using the above method then it will continue to run but on next boot will not start automatically.

Daemons:

A “daemon” is a system process which runs in the background performing a particular task. Daemon name normally end with “d” (ex:vsftpd) either listen for certain events or perform a system task.

Controlling services and daemons in rhel7 using **systemctl** command

To start the service

Syntax:# systemctl <command> <service-name>/<service-name>.service

Ex:# systemctl start httpd/httpd.service

To restart the service

Syntax:# systemctl <command> <service-name.service>

Ex:systemctl restart httpd.service

To reload the service

Syntax:# systemctl <command> <service-name>

Ex: # systemctl reload vsftpd

To stop the service

Ex:# systemctl stop httpd.service

To check the status of service

Ex:# systemctl status httpd.service

Note: These services perform temporally

To enable the service

Syntax:# systemctl enable <service-name>

Ex:systemctl enable httpd.service

To disable the service

Syntax: # systemctl disable <service-name>

Ex:# systemctl disable httpd.service

TO check the status whether the service enabled or disabled

Syntax:# systemctl is-enabled <service-name>

Ex:#systemctl is-enabled httpd.service

Note:difference between restart and reload in services

Restart: If restart any service the service should be off and on. It means it will kill old process id(pid) and generates new process id to that service.

Reload: If reload the service changes will be updated.It means it works on the same process id(pid).

Process Management

An instance of a computer program in execution and also called as **job** or **task**. Processes on modern systems can have several threads of execution or several different pieces of the program running in parallel and every process on system has some memory allocated and every process has own unique identified number I.e process id (pid).

To display all running process in terminal

```
# ps
```

To display all running process particular user

Syntax:# ps -u <username>

Example:# ps -u root

To display all running process particular group

Syntax: # ps -g <groupname>

Example:# ps -g mistgrp

To to check process id particular service wit user

Syntax: # pgrep -u <username> daemon

Examples: # pgrep -u root sshd

To display running process in system

ps -aux | less

To see every process on the system

ps -ef

ps -elf

Controlling Processes with Signals.

Linux having number of signals(64 signals) available for use but we mostly deal with only a few of them. Each signal is associated with a unique numeric identifier, a name, and an action. A list of available signals can be displayed with the kill command using the –l option.

kill -l

some mostly used signals

1 – SIGHUP - for reloading the process

2 – SIGINT – for interrupting the process

9 – SIGKILL - for killing the process

15 – SIGTERM - for Terminating the process

20 – SIGTSTP - for stopping the process

Zombie process?

When we start parent process, it will start some child processes. After some time the child processes will died because of not knowing the parent processes. These parent processes (which are running without child processes) are called Zambie processes. These are also called as defaunct processes.

top command

top is a command to see the processes states and statuses information continuously until we quit by pressing " **q** ". By default top command will refresh the data for every **3** seconds.

When we need to see the running processes on our Linux in real time, the top command will be very useful. Besides the running processes the top command also displays other information like free memory both physical and swap.

The first line shows the current time, "**up 1 day**" shows how long the system has been up for, "**3 user**" how many users login, "**load average : 0.01, 0.00, 0.23**" the load average of the system **1, 5 and 15 minutes**.

The second line shows the no of processes and their current states.

The third line shows CPU utilization details like % of the users processes, % of the system processes, % of available CPU and % of CPU waiting time for I/O (input and output).

The fourth and fifth lines shows the total physical memory in the system, used physical memory, free physical memory, buffered physical memory, the total swap memory in the system, used swap memory, free swap memory and cached swap memory, ... etc.,

From sixth line onwards the fields are as follows.

PID Process ID
USER Owner of the process ie., which user executed that process
PR Dynamic Priority
NI Nice value, also known as base value
VIRT Virtual size of the task includes the size of processes executable binary
RES The size of RAM currently consumed by the task and not included the swap portion
SHR Shared memory area by two or more tasks
S Task Status
% CPU The % of CPU time dedicated to run the task and it is dynamically changed
% MEM The % of memory 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+ does not account the CPU time used by the task's dead children
COMMAND Showing program name or process name.

* While running the top command, just press the following keys woks and the output will be stored in real time.

1 -----> 2nd CPU information Shift + > -----
>Page up
h ----->Help Shift + < -----> Page
down
Enter -----> Refresh immediately n ----->
Number of tasks
k -----> Kill the process u -----
>user processes
M -----> Sort by memory usage P ----->
Sort by CPU usage
T -----> Sort by cumulative time z -----
>Color display
r -----> To reschedule the priority by renice d -----
>Change the delay time (refresh time)
b -----> Highlight the running process W -----
>Write the information in **/root/.toprc** file
q -----> quit the top command

The status of the processes :

r -----> Running process s ----->
Sleeping process
z -----> Zombie process T ----->
Stopped process
D -----> Uninterrupted sleeping process R < -----
>High priority
N > ----> Low priority o ----->
Orphan process
+ -----> Foreground process ? ----->
Background process
renice -n 10 5453 (to change the
specified running process priority on line)
nice -n -15 firefox (to start the
firefox process with priority level -15)

"sosreport" and how to generate it?

Sosreport is a command in linux (**RHEL / CentOS**) which collects **system configuration** and diagnostic information of your linux box like running kernel version, loaded modules, and system and service configuration files. This command also runs external programs to collect further information, and stores this output in the resulting archive.

Sosreport is required when you have open a case with redhat for technical support. Redhat support Engineers will require sosreport of your server for troubleshooting purpose.

To run sosreport ,**sos** package should be installed. Sos package is part of default installation in most of linux. If for any reason this package is no installed , then use below yum command to install **sos package** :

yum install sos -y

Generate the sosreport :

Open the terminal and type sosreport command :

sosreport

This command will normally complete within a **few minutes**. Depending on local configuration and the options specified in some cases the command may take longer to finish. Once completed, sosreport will generate a compressed a file under **/tmp** folder. Different versions use different compression schemes (**gz, bz2, or xz**). The file should be provided to Redhat support representative (normally as an attachment to an open case).

Note: sosreport requires root permissions to run.

Different Options used in sosreport command :

The sosreport command has a **modular structure** and allows the user to enable and disable modules and specify module options via the command line. To **list available modules** (plug-ins) use the following command:

sosreport -l

To **turn off** a module include it in a comma-separated list of modules passed to the -n/–skip-plugins option. For instance to disable both the kvmand amd modules:

sosreport -n kvm,amd

Individual modules may provide additional options that may be specified via the **-k option.** For example on Red Hat Enterprise Linux 5 installations the sos rpm module collects "rpm -Va" output by default. As this may be **time-consuming** the behaviour may be disabled via:

 # sosreport -k rpm.rpmva=off

port numbers for different services?

The Port no. list :

FTP (For data transfer)	20	HTTP	80
FTP (For connection)	21	POP3	110
SSH	22	NTP	123
Telnet	23	LDAP	389
Send Mail or Postfix	25	Log Server	514
DNS	53	HTTPS	443
DHCP (For Server)	67	LDAPS (LDAP + SSL)	636
DHCP (For Client)	68	NFS	2049
TFTP (Trivial File transfer)	69	Squid	3128
Samba shared name verification	137	Samba Data Transfer	138
Samba Connection Establishment	138	Samba Authentication	445
MySQL	3306	ISCSI	3260

* Ping is not used any port number. It is used ICMP (Internet Control Message Protocol) only.

Security Levels

Linux have some security inbuilt level there i.e

SELinux

Iptables

Firewalls

Tcp Wrappers

SELinux:

Although necessary as first level permissions and access control mechanisms, they have some limitations that are addressed by Security Enhanced Linux (SELinux)

With those limitations in mind, the United States National Security Agency (NSA) first devised SELinux, a flexible mandatory access control method, to restrict the ability of processes to access or perform other operations on system objects (such as files, directories, network ports, etc) to the least permission model, which can be modified later as needed. In few words, each element of the system is given only the access required to function.

Managing SELinux

What is SELinux?

It is a one type of security that enhances the security that allows users and administrators more control over which users and applications can access which resources, such as files, Standard Linux access controls etc.,

It is mainly used to protect internal data (not from external data) from system services. In real time SELinux is disabled and instead of this IP tables are used. It protects all the services, files and directories by default if SELinux is enabled.

In how many ways we can implement the SELinux? Explain them.

We can implement the SELinux mainly in 2 modes.

(i) Enabled

(ii) Disabled (default mode)

Enabled :

Enabled means enabling the SELinux policy and this mode of SELinux is divided into two parts.

(a) Enforcing

(b) Permissive

Disabled :

Disabled means disabling the SELinux policy.

Enforcing mode in SELinux?

Enforcing means SELinux is on. It checks SELinux policy and stored a log. No can access the services by default but we can change the policy whenever we needed.

Permissive mode in SELinux?

SELinux is on and it don't check SELinux policy and stored the log. Everybody can access the services by default and we can also change the SELinux policy. It is also called as debugging mode or troubleshooting mode. In this mode SELinux policies and rules are applied to subjects and objects but actions are not affected.

Disabled mode in SELinux?

SELinux is turned off and no warning and log messages will be generated and stored.

What are Booleans?

Booleans are variables that can either be set as true or false. Booleans enhance the effect of SELinux policies implemented by the System Administrators. A policy may protects certain deamons or services by applying various access control rules.

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 process. It uses roles to limit the domains that can be entered and the user identities to specify the role that can be attained.

required files for SELinux?

vim /etc/selinux/config -----> It is main file for SELinux.

vim /etc/sysconfig/selinux -----> It is a link file to the above file.

vim /var/log/audit/audit.log -----> SELinux log messages will be stored in this file.

command to see the SELinux mode?

getenforce (to check the SELinux mode)

command to set the SELinux mode temporarily?

setenforce 0 or 1 (to set the SELinux mode. Where ' 0 ' -----> permissive and ' 1 ' -----> Enforcing)

Note :

(i) To change the SELinux mode from Permissive to Enforcing or Enforcing to Permissive modes the system restart is not required.

(ii) To change Enforcing mode to Disabled mode or Disabled mode to Enforcing mode the system restart is required.

(iii) The above commands are changed the SELinux mode temporarily only.

To make the selinux changes permanently then open **/etc/selinux/config** and go to ,

SELINUX=Enforcing or Permissive or Disabled

(save and exit this file)

command to see the SELinux policy details?

sestatus (to see the SELinux policy details)

Other useful commands :

ls -Z <file name> (to see the SELinux context of the file)

ls -ldZ <directory name> (to see the SELinux context of the directory)

Network Configuration and Troubleshooting

Network:

Combination of two more computers connected together to share their resources each other by means of communication like cable is called Network.

Networking:

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

basic requirements for networking?

(a) NIC (Network Interface Card or controller)
(b) Media (nothing but cables)
(c) Topology
(d) Protocol
(e) IP Addresses

NIC card:

A Network Interface Card or controller is hardware component that connects a computer to a computer network. Each NIC card will be having MAC (Media Access Controller) address to avoid conflicts between same NIC adapters. In Linux these NIC adapter is represented by the word **"eth"** . For example if two NIC cards are there in a system then it will be denoted as **"eho","eth1",** .....etc.,

media:

Media is nothing but cable to connect two or systems. Example : **RJ 45, CAT 5** and **CAT 6**, ...etc.,

topology:

Topology is a design in which the computers in network will be connected to each other. Example for topologies are Bus, Ring, Star, Mesh, Tree topologies.

Protocol:

A **Network Protocol** defines rules and conventions for communication between the network devices. Protocols are generally use packet switching techniques to send and receive messages in the form of packets.

Example for protocols are **TCP/IP** (Transmission Control Protocol and Internet Protocol), **UPD** (User Datagram Protocol) and **HTTP** (Hyper Text Transfer Protocol), ...etc.,

Differences between TCP/IP and UDP protocols?

TCP/IP	UDP
Transmission Control Protocol	User Datagram Protocol
It is connection oriented	It is connection less
Reliable	Non-Reliable
TCP Acknowledgement will be sent / received	No Acknowledgement
Slow communication	Fast communication
Protocol No. for TCP is 6	Protocol No. for UDP is 17
HTTP, FTP, SMTP, ...etc., uses TCP	DNS, DHCP, ...etc., uses UDP

IP address:

Every Computer will be assigned an IP address to identify each one to communicate in the network. The IP address sub components are Classes of an IP address, Subnet masks and Gateway.

Classes of IP address :

The IP addresses are further divided into classes. The classes are A, B, C, D, E and the ranges are given below.

Class	Start	End	Default Subnet mask	Classless Inter Domain Routing
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	224.0.0.0	239.255.255.255		
Class E	240.0.0.0	255.255.255.255		

Loopback address?

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

Multicasting:

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

Subnet mask:

A subnet mask allows the users to identify which part of an IP address is reserved for the network and which part is available for host use.

Gateway:

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

Omportant configuration files in network configuration:

 # cat /etc/sysconfig/network (This file keeps the information about the hostname assigned to the system and if we want to change the hostname permanently, we need to change the hostname in this file)

 # cat /etc/sysconfig/network-scripts/ (This directory keeps the configuration of network devices connected to the system. Examples are **ifcfg-eht0, ifcfg-eth1, ifcfg-eth2,** .....etc.,)

 # cat /etc/hosts (This file is responsible for resolving hostname into IP address locally. ie., local DNS if DNS server is not available)

 # cat /etc/resolve.conf (This file keeps the address of the DNS server to which the clients will be accessing to resolve IP address to hostname and hostname to IP address)

Differences between MAC and IP addresses:

MAC Address	**IP Address**
It is a permanent address. So we cannot change this address.	It is a temporary address. So, we can change this address any no. of times.
It stands for Media Access Control Address.	Internet Protocol address.
It is a physical address.	It is a logical address.

It is divided into 6 parts. --- : --- : --- : --- : --- : --- (each 8 bits. So, 8 X 6 = 48 bits	It is two types. **IPV4** :(It is divided into 4 parts) --- . --- . --- . --- (each 8 bits. So, 8 X 4 = 32 bits **IPV6** : (It is divided into 16 parts) --- . --- . --- . --- . --- . --- . --- . --- . --- . --- . --- . --- . --- . --- . --- . --- (each 8 bits. So, 8 X 16 = 128 bits.
ifconfig (to see the MAC address)	**# ifconfig** (to see the IP address)

Types of NIC cards available:

(a) eth0 (1st NIC card)
(b) eth1 (2nd NIC card)
(c) br0 (Bridge -----> used for communication from physical to virtual)
(d) lo (loopback device name and IP address is 127.0.0.1)
ifconfig (to see all the NIC devices connected to the system)

Types of cable connections available:

(i) Cross cable (to connect two systems directly)
(ii) Straight cable (to connect more systems with the help of switch)
ethtool <device name> (to check the network cable is connected or not)
miitool <device name> (It is also used to check the network cable but it will not supports RHEL - 7 and only supports RHEL - 6 and it also works on physical system only not on virtual system)

ways we can configure the network

There are two ways to configure the network.
(a) Static Network.
(b) Dynamic Network.

Static Network :

In this way we assign the IP address and hostname manually. Once we configure the IP address, it will not change.

Dynamic Network :

In this way we assign the IP address and hostname dynamically. This means the IP address will change at every boot.

Assign the static IP address to the NIC card?

In RHEL - 6 :

setup
(Move the cursor to Network configuration and press Enter key)
(Move the cursor to Device configuration and press Enter key)
(Select the NIC adapter ie., eth0 and press Enter key)
(Assign the above IP address and other details as per our requirements and move the cursor to "OK" and press

Enter key)

(Move the cursor to "Save" to save the changes in device configuration and press Enter key)

(Once again move the cursor to "Save & Quit" button and press Enter key)

(Finally move the cursor to "Quit" button and press Enter key to quit the utility)

(Then restart the network service and check for the IP address by **# service network restart** command)

(If the change is not reflected with the above service, then restart the network manager by **# service NetworkManager restart** command)

ifconfig (to see the IP address of the NIC card)

ping < IP address > (to check whether the IP is pinging or not)

In RHEL - 7 :

nmcli connection show (to see all the network connections)

nmcli device show (to see the network details if already configured manually or dynamically)

nmcli connection add con-name "System eth0" ifname eth0 type ethernet (to add the network connection)

nmcli connection modify "System eth0" ipv4.addresses ' < IP address >/< netmask >< gateway > ' ipv4.dns < dns server IP address > ipv4.dns-search < domain name> ipv4.method <static or manually> (to assign IP address, gateway, dns, domain name and configure the network as static or manually)

nmcli connection up "System eth0" (to up the connection)

systemctl restart network (to restart the network service)

systemctl enable network (to enable the network service)

ifconfig (to see the IP address of the NIC card)

ping < IP address > (to check whether the IP is pinging or not)

Differences between RHEL - 6 and RHEL - 7 network configuration files

RHEL - 6	**RHEL - 7**
/etc/sysconfig/network-scripts is the directory which contains the NIC configuration information.	**/etc/sysconfig/network-scripts** is the directory which contains the NIC configuration information.
/etc/sysconfig/network-scripts/ifcfg-<device name> is the file which contains the NIC configuration details.	**/etc/sysconfig/network-scripts/ifcfg-<device name>** is the file which contains the NIC configuration details.
/etc/resolve.conf is the file which contains DNS server IP and domain name	**/etc/resolve.conf** is the file which contains DNS server

location.	IP and domain name location.
/etc/sysconfig/network is the hostname configuration file.	**/etc/hostname** is the hostname configuration file.
/etc/hosts is the file which contains the local DNS server IP address.	**/etc/hosts** is the file which contains the local DNS server IP address.

differences between Dynamic and Static configuration information

Dynamic configuration information	**Static configuration information**
Device =<NIC device name>	**Device =**<NIC device name>
HWADDR=02:8a:a6:30:45	**HWADDR=**02:8a:a6:30:45
Bootproto=DHCP	**Bootproto=**none (means static network)
Onboot=yes (**yes** means whenever we restart the system this connection will be activated and **no** means whenever we restart the system the connection will be deactivated)	**Onboot=**yes
Type=Ethernet	**Type=E**thernet
Userctl=yes/no ----> If it is **yes** all normal users can disable the NIC card and If it is **no** except root user nobody can disable the NIC card.	**Userctl=**yes/no ----> If it is **yes** all normal users can disable the NIC card and If it is **no** except root user nobody can disable the NIC card.

to set the hostname temporarily and permanently

RHEL - 6 :

hostname <fully qualified domain name> (to set the hostname temporarily)

vim /etc/sysconfig/network (to set the hostname permanently)

HOSTNAME=<fully qualified domain name>

(save and exit this file)

service network restart (to update the hostname in the network)

chkconfig network on (to enable the connection at next reboot)

RHEL - 7 :

hostname <fully qualified domain name> (to set the hostname temporarily)

hostnamectl set-hostname <fully qualified domain name> (to set the hostname permanently)

systemctl restart network (to update the hostname in the network)

systemctl enable network (to enable the connection at next reboot)

troubleshoot if the NIC is notworking

(a) First check the NIC card is present or not by **# ifconfig** command.

(b) If present thencheck the status of the NIC card is enabled or disabledby click on System menu on the status bar, then select Network Connections menu.

(c) Click on IPV4 settings tab, select the device eth0 or any other and select Enable button, then Apply and OK.

(d)Open **/etc/sysconfig/network-scripts/ifcfg-eth0** file check Userctl=yes or no. If it is yes make it as no, then check Onboot= yes or no. If it is no make it as yes and save that file.

(e) If not present thencheck the status of the NIC card is enabled or disabled by click on System menu on the status bar, then select Network Connections menu.

(f) Click on IPV4 settings tab, select the device eth0 or any other and select Enable button, then Apply and OK.

(g) Using **# setup** (in RHEL - 6) or **# nmcli** (in RHEL - 7) commands assign the IP address to the system and restart the network service by **# service network restart** (in RHEL - 6) or **# systemctl restart network** (in RHEL - 7) commands and enable the service at next reboot by **# chkconfig network on** (in RHEL - 6) or **# systemctl enable network** (in RHEL - 7) commands.

(h) Then up the connection by **# ifconfig eth0 up** (in RHEL - 6) or **# nmcli connection up <connection name>** commands.

(i) Even though it is not working may be the fault in NIC card. If so, contact the hardware vendor by taking the permissions from higher authorities.

difference between TCP and UDP protocol

TCP is a connection oriented protocol and contain the information of sender as well as receiver.

Example : HTTP, FTP, Telnet

TCP is slower than UDP due to its error checking mechanism

UDP protocols are connection less packets have no information to where they are going. These type of ports are generally used for broadcasting.

For example : DNS, DHCP

UDP are faster

/etc/resolve.conf

It contains the details of nameserver, i.e., details of your DNS server which helps us connect to Internet.

/etc/hosts file

To map any hostname to its relevant IP address.

command to check all the open ports of your machine

#nmap localhost

command to check all the open ports of remote machine

nmap <IP address or hostname of the remote system>

command to check all the listening ports and services of your machine

netstat -ntulp

make a service run automatically after boot

chkconfig <service name> on

possible ways to check if your system is listening to port 67?

nmap localhost | grep 67

netstat -ntulp | grep 67

IPV6?

It's length is **128 bits.** It's netmask is 64

nmcli connection modify "System eth0" ipv6.addresses 2005:db8:0:1::a00:1/64 ipv6.method static
(to add the IPV6 version of IP address to the connection "System eth0")

nmcli connection modify "System eth0" ipv4.addresses '172.25.5.11/24 172.25.5.254' ipv4.dns 172.25.254.254 ipv4.dns-search example.com ipv4.method static ipv6. addresses 2005:ac18::45/64
ipv6.method static (to assign ipv4 and ipv6 IP addresses to "System eth0 connection)

nmcli connection down "System eth0" (to down the "System eth0" connection)

nmcli connection up "System eth0" (to up the "System eth0" connection)

troubleshoot if the network is not reaching?

(i) First check the network cable is connected or not by **# ethtool <NIC device name>** command. if connected then check the IP address is assigned or not by **# ifconfig <NIC device name>** command.

(ii) Then check the system uptime by **# uptime** command.

(iii) Then check the network services status by **# service network status** and **# service NetworkManager status** commands.

(iv) Then check the network service at Run Level by **# Chkconfig --list network** command.

(v) Then check whether the source network and destination network are in the same domain or not.

(v) Then finally check the routing table by **# route -n** command.

<u>Other useful commands</u> :

ping <IP address or hostname> (to check the pinging)

Normally the ping command pings continuously until a stop signal reaches by Ctrl + c, so to avoid continuous pinging by

ping -c <number><IP address> (to ping upto the specified no of times)

ipcalc -m <IP address) (to find the subnet mask for that specified IP address)

Normally IP addresses are assigned by ISP (Internet Service Provider) and managed by IANA (Internet Assign Number Authority)

ifconfig (to see or check all the NIC device names and IP addresses)

```
     # ethtool    <NIC device name>              (to check the network
cable is connected or not)
     # miitool  <NIC device name>               (It is also used to
check the network cable but it works on
     physical system not on virtual system and supports  in RHEL - 6
only)
     # ip addr  show                           (to show all NIC devices
present on the system)
     # hostname                          (to see the hostname with fully
qualified domain name)
     # hostname  -i              (to see the IP address of the system)
     # hostname  -d              (to check the domain name of the
system)
     # hostname  -s              (to check the hostname without domain
name)
     # netstat  -r               (to check the default gateway and
routing table)
     # route                     (to check the default gateway with
routing table)
     # ip route                  (to display the NIC device with
default gateway)
     # dig   or   # host   or   #nslookup             (all are used to
resolve the name to IP and IP to name)
     # nslookup   <IP address>             (to resolve IP to name)
     # nslookup  <hostname>                (to resolve name to IP)
     # host   <IP address>                 (to resolve IP to name)
     # host   <fully qualified domain name>      (to resolve name to
IP address)
     # dig  -x   <IP address>         (to resolve IP address to name)
     # dig   <fully qualified domain name>       (to resolve name to
IP address)
     # nmcli    (Network Manager Command Line Interface used to
                    configure the network setup in RHEL - 7)
     # setup                                     (to setup the static
network in RHEL - 2, 3, 4, 5 and 6)
     # nmtui                                     (to setup the static
network in GUI mode for RHEL - 7)
     # nmcli device show             (It displays all the NIC devices
network information of the system )
     # nmcli device show eth0                    (to see all the
network devices information of the eth0)
     # nmcli connection  or nmcli connection show     (to see all the
network connection names)
     # nmcli connection add con-name  <connection name>  ifname  <NIC
device name>  type ethernet
                                               (to create a new connection
name for eth0)
     # nmcli connection show  --activate              (it shows which
connection is active currently)
     # nmcli connection add con-name   <connection name>   ifname
<NIC device name>   type  ethernet
```

(to add a connection name to NIC device)

nmcli connection modify <connection name> ipv4.addresses '<IP address>/<netmask><default gateway> ' ipv4.dns <dns server IP address> ipv4.dsn-search <domain name> ipv4.method <static/manual>
(to modify the connection as static and assign the IP, gateway, dns IP, domain name)

nmcli connection delete <connection name> (to delete the specified connection)

nmcli connection modify <connection name> ipv4.method <static/manual> (to modify dynamic connection to static connection)

nmcli connection up <connection name> (to activate or up the specified connection)

nmcli connection down <connection name> (to disable or down the specified connection)

nmcli connection show <connection name> (to see the information about the specified NIC device)

ping -I <NIC device name><IP address> (to check the connection from NIC device to IP address)

hostname <fully qualified domain name> (to set the hostname temporarily)

hostnamectl set-hostname <fully qualified domain name> (to set the hostname permanently in RHEL - 7)

NOTE: Whenever we change any parameters in **/etc/sysconfig/network-scripts/ifcfg-<NIC device name>** file, then we have to reload that file and again we have to up the connection (nothing but activate the connection by **# nmcli connection reload** command.

nmcli connection reload (to reload the configuration of the connection if any changes on it and it reloads all configuration files)

nmcli connection reload /etc/sysconfig/network-scripts/ifcfg-<NIC device name> (to reload a single file)

hostnamectl status (it displays full details of the hostname and works in RHEL - 7 only)

nmcli networking off (to disable all the connections at a time)

nmcli device status (to display all NIC device connections statuses)

nmcli connection modify <connection name> + ipv4.dns <secondary dns server IP> (to add a secondary dns server IP to the existing connection)

netstat -ntulp (to check how many open ports are there in local system)

ss -ntulp (" ")

nmap (to check how many open ports are there in remote system)

```
      # tracepath                                (it displays the routing
information)
      # miitool   <NIC device name>                    (to check the network
cable is connected or not)
      # ethtool  <NIC device name>                     (          "
      "          )
      # ifconfig                                 (to check the NIC card is
enable or not)
      # ifup   <NIC  device name>                      (to enable or up the
NIC card)
      #ifdown   <NIC device name>                (to disable or down the NIC
card)
      # route  -n                                      (to check the
gateway)
      # cat  /etc/resolve.conf                         (to check the dns
server information)
      # cat  /etc/sysconfig/network-scripts/ifcfg-<NIC device name>
      (to  see the NIC device information)
      # hostname   or  cat  /etc/sysconfig/network     (to check the
hostname in RHEL - 6)
      # hostnamectl  status   or   cat  /etc/hostname  (to check the
hostname in RHEL - 7)
      # ping   <IP address>                      (to check the connection
communication)
      # chkconfig    --list                      (to list all the services
which are running at boo time in RHEL - 6 & 7)
      # systemctl   list-unit-files                    (to list all the
processes which are running at boot time in RHEL - 7)
      # chkconfig    --level   <service name>     (it will set the
service at run level 3 when the system is booting)
      # service   --status-all                   (to see the list of all the
processes which are currently running)
      # ls  /etc/init.d                          (is the location of all the
services and deamons  in RHEL - 6)
      # ls  /usr/lib/systemd/system              (is the location of all the
services and deamons  in RHEL - 7)
      #  /etc/rc.local                    (is the last script to be run
when the system is booting)
           (If we enter as   sshd    stop  at the last line of the
script file then   sshd   will be stopped even though that
                                                      sshd   is
enabled)
      # service sshd status              (to check the sshd status)
      # service   --service -all                 (to see the process  ID  of
all the services)
      # netstat   -ntulp                         (to see all the services
with port no., status, process  ID and all open
           ports in local system, routing table and NIC device
information)
                -n   ----->   port no.  (numeric  no)      -t   -----
>tcp protocol
```

-u -----> upd protocol -l -----> port is listening or not
-p -----> display the process ID

netstat -r (to see all routing table information)

netstat -i (to see all the NIC cards information)

nmap (to see the network mapping ie., open ports list on remote system)

Note : By default this command will not available. So, first install the **nmap** package by **# you install nmap -y**

nmap <remote system IP address> (to see all the services which are running in the specified remote system)

nmap <remote IP 1><remote IP 2><remote IP 3> (to see the running services on specified remote systems)

nmap 172.25.0.11 - 50 (to see the running service on 172.25.0.11 to 172.25.0.50 systems)

nmap -p 80 <remote IP> (to see the http port is running or not on specified remote system)

nmap -p 80 - 90 <remote IP> (to see port no's 80 to 90 are running or not on remote systems)

nmap -sp 172.25.0.0/24 (to see all the systems which are in upstate ie., 172.25.0.1, 172.25.0.2, 172.25.0.3, ...upto 172.25.0.254 systems)
(where s -- scan & p -- ping)

Open a file, write all the systems IP addresses, save & exit the file. Example has given below,

vim coss
172.25.2.50
172.25.3.50
172.25.4.50 ....etc., (save and exit this file)

nmap -iL coss (to scan all the IP addresses by reading the coss file)(where -i ----> input, -L ----> list)

nmap --iflist (to see all the routing table information in the network)

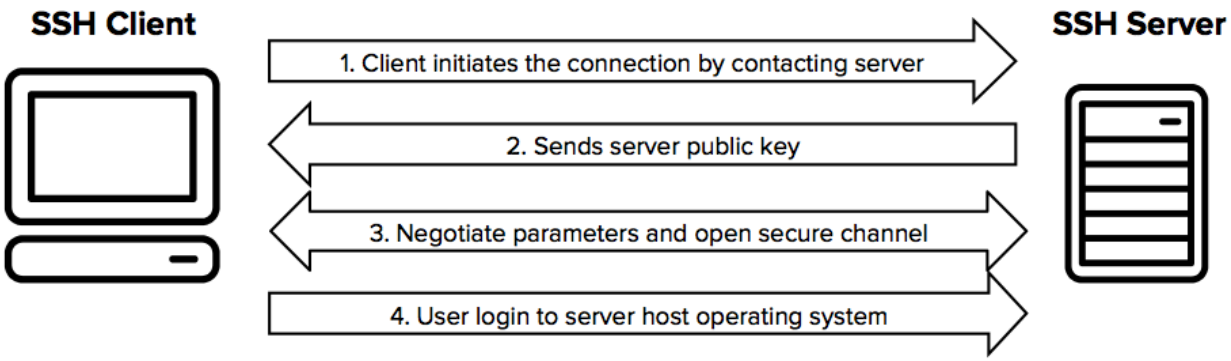
nmap 172.25.0.10 - 20 --exclude 172.25.0.15 (to scan all the systems from 172.25.0.10 to 172.25.0.20 systems and excluding 172.25.0.15 system)

nmcli connection show --active (to control the network connections)

ip link (to check the network connection)

ping -I eth1 <IP address> (to check the 2nd NIC card connection)

SSH :



1. SSH stand for Secure Shell.
2. SSH is a network protocol for secure data communication.
3. SSH protocol allows remote command line login.
4. SSH protocol enables remote command execution.
5. To use SSH you need to deploy SSH Server and SSH Client program respectively.
6. OpenSSH is a FREE version of the SSH.
7. Telnet, rlogin, and ftp transmit unencrypted data over internet.
8. OpenSSH encrypt data before sending it over insecure network like internet.
9. OpenSSH effectively eliminate eavesdropping, connection hijacking, and other attacks.
10. OpenSSH provides secure tunneling and several authentication methods.
11. OpenSSH replace Telnet and rlogin with SSH, rcp with scp, ftp with sftp.

- **Package name: openssh-server,openssh-clients or openssh***
- **Port number: 22**
- **Configuration file: /etc/ssh/sshd_config**
- **Service name: ssh**
- **Daemon : sshd**

LAB WORK:

Server ip address: 172.25.1.10

Guest ip address: 172.25.1.11

ON SERVER:

- # yum install openssh*

Enabling firewalls

- # firewall-cmd --permanent --add-service=ssh
- # firewall-cmd --reload

Starting service and deamon

- # systemctl enable sshd
- # systemctl start sshd

(On client also install and start the service.)
Pinging guest machine to check reachable or not

- # ping 172.25.1.11

Connecting with guest machine

- # ssh 172.25.1.11

root@172.25.1.11 type password: xxxxxxx

```
[root@server ~]# rpm -qa openssh*
openssh-cavs-7.4p1-11.el7.x86_64
openssh-7.4p1-11.el7.x86_64
openssh-askpass-7.4p1-11.el7.x86_64
openssh-clients-7.4p1-11.el7.x86_64
openssh-server-sysvinit-7.4p1-11.el7.x86_64
openssh-ldap-7.4p1-11.el7.x86_64
openssh-keycat-7.4p1-11.el7.x86_64
openssh-server-7.4p1-11.el7.x86_64
[root@server ~]# ping 172.25.1.11 -c 2
PING 172.25.1.11 (172.25.1.11) 56(84) bytes of data.
64 bytes from 172.25.1.11: icmp_seq=1 ttl=64 time=0.443 ms
64 bytes from 172.25.1.11: icmp_seq=2 ttl=64 time=0.664 ms

--- 172.25.1.11 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.443/0.553/0.664/0.112 ms
[root@server ~]# ssh 172.25.1.11
The authenticity of host '172.25.1.11 (172.25.1.11)' can't be established.
ECDSA key fingerprint is SHA256:98+vkPbxl2rku2wdCQNvr7Clr4CqZeyKYUM+i+cRI8o.
ECDSA key fingerprint is MD5:72:94:e7:73:d3:46:18:ac:65:51:d3:2a:ca:51:9b:09.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.25.1.11' (ECDSA) to the list of known hosts.
root@172.25.1.11's password:
Last login: Mon Jul  9 21:46:54 2018
[root@client ~]#
```

Ssh with Hostname:

```
[root@server ~]# #connecting to client machine with hostname
[root@server ~]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
172.25.1.10 sever.mist.com
172.25.1.11 client.mist.com
[root@server ~]# ping client.mist.com #client hostname should be present in /etc/hosts file
PING client.mist.com (172.25.1.11) 56(84) bytes of data.
64 bytes from client.mist.com (172.25.1.11): icmp_seq=1 ttl=64 time=0.472 ms
64 bytes from client.mist.com (172.25.1.11): icmp_seq=2 ttl=64 time=0.662 ms
^C
--- client.mist.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1004ms
rtt min/avg/max/mdev = 0.472/0.567/0.662/0.095 ms
[root@server ~]# ssh client.mist.com
The authenticity of host 'client.mist.com (172.25.1.11)' can't be established.
ECDSA key fingerprint is SHA256:98+vkPbxl2rku2wdCQNvr7Clr4CqZeyKYUM+i+cRI8o.
ECDSA key fingerprint is MD5:72:94:e7:73:d3:46:18:ac:65:51:d3:2a:ca:51:9b:09.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'client.mist.com' (ECDSA) to the list of known hosts.
root@client.mist.com's password:
Last login: Mon Jul  9 22:36:43 2018 from 172.25.1.10
[root@client ~]#
```

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.

LAB WORK:

Generating SSH key pair

To generate the SSH key pair, the syntax is

ssh-keygen

```
[root@server ~]# 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):
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:KsH6OBO63fCq9HFngsnfR3KHXUEIg05JqRWoiwladG0 root@server.mist.com
The key's randomart image is:
+---[RSA 2048]----+
|      .oo=o. o.  |
|   . ..E*  .. .  |
|  . ...=      .  |
| . .o . .     .  |
|o.o +   S o .    |
|.oooo. o + o     |
| oo*.o.o+ .      |
|o.+*+.=  .       |
|oo+=+. ..        |
+----[SHA256]-----+
```

After generating keys by using #ssh-keygen these keys will store under user's home directory in a hidden directory " .ssh".

We can by using following commands.

#cd /root

ls -a

.ssh

cd /root/.ssh/

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 Keys to guest machine:

Syntax: ssh-copy-id <username>@<guest ip or hostname>

```
[root@server ~]# ssh-copy-id client.mist.com
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@client.mist.com's password:

Number of key(s) added: 1

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

[root@server ~]#
```

Connecting to the client macine without entering password

```
[root@server ~]# ssh client.mist.com
Last login: Mon Jul  9 22:46:35 2018 from 172.25.1.10
[root@client ~]#
```

We can allow and deny the user's for remote login.

For example to deny root login in client machine, we have to edit configuration file

In client machine:

```
#vim /etc/ssh/sshd_config
#
#
#PermitRootLogin no (uncomment and set value as 'no')
systemctl restart sshd
vim /etc/ssh/sshd_config
systemctl restart sshd
```

On Server machine

```
[root@server etc]# ssh root@172.25.1.11
root@172.25.1.11's password:
Permission denied, please try again.
root@172.25.1.11's password:
```

SSH with Normal users:

Create a user on both server and client.

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

Connecting to client machine with normal user:

```
[root@server ~]# su - feroz
[feroz@server ~]$ ssh feroz@client.mist.com
The authenticity of host 'client.mist.com (192.168.163.134)' can't be established.
ECDSA key fingerprint is SHA256:98+vkPbxl2rku2wdCQNvr7Clr4CqZeyKYUM+i+cRI8o.
ECDSA key fingerprint is MD5:72:94:e7:73:d3:46:18:ac:65:51:d3:2a:ca:51:9b:09.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'client.mist.com,192.168.163.134' (ECDSA) to the list of known hosts.
feroz@client.mist.com's password:
[feroz@client ~]$
```

If we know the password of a user present in guest machine machine we can connect to guest machine.

Restricting particular users:

We can restrict any particular user.

On server :

Lab Work:

To restrict any user or group we need add them into the ssh configuration file.(/etc/ssh/sshd_config).

Syntax : to deny users

DenyUsers <username1> <username2> … …

Syntax : to deny groups

DenyGroups <groupname1> <groupname2> … …

Adding two users to restrict remote login…

```
[root@server ~]# useradd shaik
[root@server ~]# useradd shah
[root@server ~]# passwd shaik
Changing password for user shaik.
New password:
BAD PASSWORD: The password is shorter than 8 character
Retype new password:
passwd: all authentication tokens updated successfully
[root@server ~]# passwd shah
Changing password for user shah.
New password:
BAD PASSWORD: The password is shorter than 8 character
Retype new password:
passwd: all authentication tokens updated successfully
[root@server ~]#
```

Next edit the configuration file.

#vim /etc/ssh/sshd_config

```
#LoginGraceTime 2m
#PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
DenyUsers shaik shah  #add this line and users wanna deny here
#DenyGroup <group_name> to deny a group
```

Now restart the service

#systemctl restart sshd

From Client machine to login with restricted users

```
[tom@client ~]$ ssh shaik@server.mist.com
shaik@server.mist.com's password:
Permission denied, please try again.
shaik@server.mist.com's password:

[tom@client ~]$ ssh shah@server.mist.com
shah@server.mist.com's password:
Permission denied, please try again.
shah@server.mist.com's password:
```

Here, we got permission denied message.. Like this we can restrict users or group ssh connection.

Restricting client:

We can able to restrict particular ip address by using firewalls and tcp wrappers.

- ⇨ With firewall:
- ⇨ #fiirewall-cmd --permanent --add-rich-rule='rule family="ipv4" source address="172.25.1.11" service name="ssh" reject'
- ⇨ # firewall-cmd --reload

(Note: we have to give ip-address what we want restrict as source ip address.)

Removing Rich Rule:

Use following commands to remove rich rule that we added above.

- ⇨ #firewall-cmd --permanent --remove-rich-rule='rule family="ipv4" source address=172.25.1.11 service name="ssh" reject'
- ⇨ #firewall-cmd –reload
- ⇨ #systemctl restart sshd.

TCP WRAPPERS:

To allow or deny the clients in a particular domain or a particular ip.

- ⇨ #vim /etc/hosts.allow (To deny #vim /etc/hosts.deny)
- ⇨ At the end of the file add

  sshd: <ip address>
- ⇨ Now restart ssh service

```
[root@server ~]# vim /etc/hosts.deny
[root@server ~]# cat /etc/hosts.deny
#
# hosts.deny    This file contains access rules which are used to
#               deny connections to network services that either use
#               the tcp_wrappers library or that have been
#               started through a tcp_wrappers-enabled xinetd.
#
#               The rules in this file can also be set up in
#               /etc/hosts.allow with a 'deny' option instead.
#
#               See 'man 5 hosts_options' and 'man 5 hosts_access'
#               for information on rule syntax.
#               See 'man tcpd' for information on tcp_wrappers
#
sshd:172.25.1.11 #client ip address which one we want to restrict.
[root@server ~]# systemctl restart sshd
[root@server ~]#
```

On Client machine:

Try to login to server/host machine

```
[root@client ~]# ssh 172.25.1.10
root@172.25.1.10's password:
Last login: Fri Jul 13 23:27:23 2018 from 172.25.1.11
[root@server ~]# exit
logout
Connection to 172.25.1.10 closed.
[root@client ~]# ssh 172.25.1.10
ssh_exchange_identification: read: Connection reset by peer
[root@client ~]#
```

After denying a ip-address of ssh service we can't login to the that system with the restricted ip.

If we want to allow that ip again then comment '# sshd: 172.25.1.11' line in /etc/hosts.deny file and restart service.

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.

On server: copying a file

```
[root@server ~]# cat > file1.txt
copying a file to client using scp
[root@server ~]# scp file1.txt client.mist.com:/root/.
file1.txt                                     100%   35    27.3KB/s   00:00
[root@server ~]#
```

On Client: received a file from server

```
[root@client ~]# ls
anaconda-ks.cfg  Desktop  Documents  Downloads  file1.txt
[root@client ~]#
```

With scp we can transfer files and directories also.

To copy/transfer a directory

mkdir data

cd data

touch a b c d

scp -r /root/data root@client.mist.com:/root/.

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 suppose 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

Transferring files:

⇨ Create file and transfer it.

```
[root@server ~]# cat > filessh.txt
this is a test file to transfering over ssh
by using rsync method.
[root@server ~]# rsync filessh.txt root@172.25.1.11:/root/.
[root@server ~]#
```

Transferring Directories:

#mkdir data

#cd data

touch a b c d a{11..20}

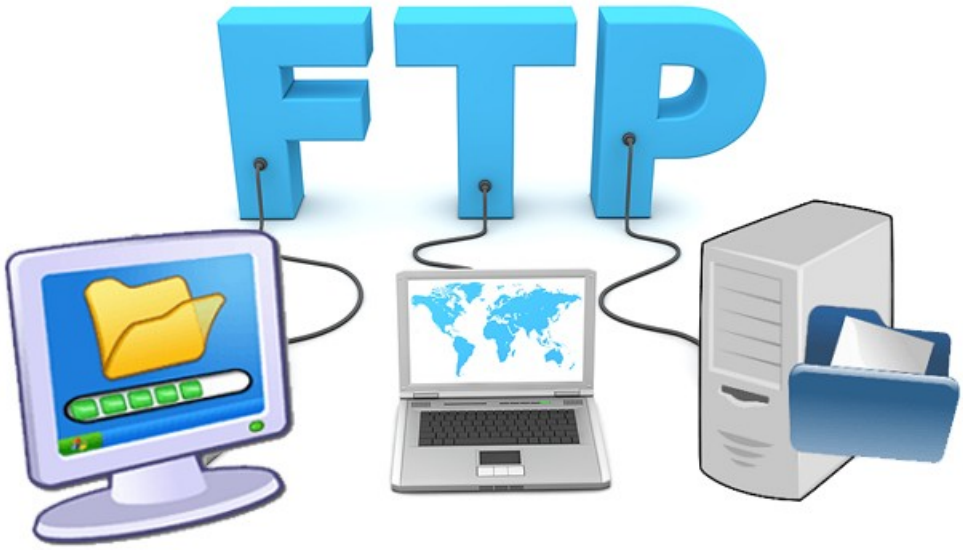
```
root@server ~]# rsync -rv data root@client.mist.com:/root/.
ending incremental file list
lata/a
lata/a11
lata/a110
lata/a12
lata/a13
lata/a14
lata/a15
lata/a16
lata/a17
lata/a18
lata/a19
lata/b
lata/c
lata/d

ent 725 bytes  received 279 bytes  669.33 bytes/sec
otal size is 0  speedup is 0.00
```

-r to transfer directories.

FTP (File Transfer Protocol) SERVER

File Transfer Protocol (FTP) is a standard network protocol used to transfer files from one host to another host over a TCP-based network, such as the Internet.



Profile of ftp server

- **Use : Ftp is used for uploading and downloading the files.**
- **Disadvantage : Directory cannot be uploaded or downloaded.**
- **Package : vsftpd**
- **Daemon : vsftpd (Very Secure Ftp daemon)**
- **Port no : 20,21**
- **Configuration files : /etc/vsftpd/vsftpd.conf**
- **Document Root : /var/ftp/pub**

INSTALLATION:

- # yum install vsftpd.x86_64 –y

```
File  Edit  View  Search  Terminal  Help
[root@server1 ~]# yum install vsftpd.x86_64 -y
Loaded plugins: fastestmirror, langpacks
base                                                     | 3.6 kB
extras                                                   | 3.4 kB
updates                                                  | 3.4 kB
extras/7/x86_64/primary_db                               | 187 kB
Loading mirror speeds from cached hostfile
 * base: centos.mirror.net.in
 * epel: epel.mirror.angkasa.id
 * extras: centos.mirror.net.in
 * updates: centos.mirror.net.in
Package vsftpd-3.0.2-22.el7.x86_64 already installed and latest version
Nothing to do
[root@server1 ~]#
```

After Installation Enable firewalls and starting the service.

- # firewall-cmd --permanent --add-service=ftp
- # firewall-cmd --reload
- # systemctl start vsftpd
- # systemctl enable vsftpd

```
File  Edit  View  Search  Terminal  Help
[root@server1 ~]# firewall-cmd --permanent --add-service=ftp
success
[root@server1 ~]# firewall-cmd --reload
success
[root@server1 ~]# systemctl start vsftpd
[root@server1 ~]# systemctl enable vsftpd
Created symlink from /etc/systemd/system/multi-user.target.wants/vsftpd.se
/lib/systemd/system/vsftpd.service.
[root@server1 ~]#
```

Now Open the configuration file of vsftp

- # vim /etc/vsftpd/vsftpd.conf
- Uncomment a line call “anon_upload_enable=yes”.

```
[root@server1 ~]# vim /etc/vsftpd/vsftpd.conf
[root@server1 ~]#
```

```
# obviously need to create a directory writable by the FTP user.
# When SELinux is enforcing check for SE bool allow_ftpd_anon_write, allo
cess
anon_upload_enable=YES
#
```

Now Check the SELinux mode and Booleans of FTP

- # getenforce
- # getsebool -a | grep ftp

getsebool reports where a particular SELinux boolean or all SELinux Booleans are on or off In certain situations a boolean can be in one state with a pending change to the other state.

```
[root@server1 ~]# getenforce
Enforcing
[root@server1 ~]# getsebool -a | grep ftp
ftpd_anon_write --> off
ftpd_connect_all_unreserved --> off
ftpd_connect_db --> off
ftpd_full_access --> off
ftpd_use_cifs --> off
ftpd_use_fusefs --> off
ftpd_use_nfs --> off
ftpd_use_passive_mode --> off
httpd_can_connect_ftp --> off
httpd_enable_ftp_server --> off
tftp_anon_write --> off
tftp_home_dir --> off
[root@server1 ~]#
```

Now change the Boolean values.

setsebool sets the current state of a particular SELinux boolean or a list of booleans to a given value. The value may be 1 or true or on to enable the boolean, or 0 or false or off to disable it. Without the -P option, only the current boolean value is affected; the boot-time default settings are not changed.

- # setsebool -P ftp_home_dir on
- # setsebool -P ftp_anon_write on
- # setsebool -P ftpd_full_access on

```
[root@server1 ~]# setsebool -P ftpd_anon_write on
[root@server1 ~]# setsebool -P ftpd_full_access on
[root@server1 ~]# 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
ftpd_use_nfs --> off
ftpd_use_passive_mode --> off
httpd_can_connect_ftp --> off
httpd_enable_ftp_server --> off
tftp_anon_write --> off
tftp_home_dir --> on
[root@server1 ~]#
```

- Now add some content to /var/ftp/pub to share with clients.
- # cd /var/ftp/pub/
- # cat > file1
- #cat > file2

```
[root@server1 ~]# cd /var//ftp/pub/
[root@server1 pub]# ls
[root@server1 pub]# cat > file1
this is content of file1
[root@server1 pub]# cat > file2
this is content of file2
[root@server1 pub]# ls
file1  file2
[root@server1 pub]#
[root@server1 pub]# cd
[root@server1 ~]# systemctl restart vsftpd
[root@server1 ~]#
```

After adding content restart the server.

Give the write permission for on /var/ftp/pub to accept uploads from client machine.

- # chmod o+w /var/ftp/pub
- # systemctl restart vsftpd

On Client Machine:

- Now on client machine install ftp package and access data shared by ftp server.
- # yum install ftp.x86_64 -y

```
[root@localhost ~]# yum install ftp.x86_64 -y
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: centos.excellmedia.net
 * extras: centos.excellmedia.net
 * updates: centos.excellmedia.net
Package ftp-0.17-67.el7.x86_64 already installed and latest version
Nothing to do
[root@localhost ~]# rpm -q ftp
ftp-0.17-67.el7.x86_64
[root@localhost ~]#
```

Now connect to ftp server by using ftp server ip address.

- ftp <ip-address >

- ftp 172.25.0.11 (it's server ip)
- Enter username: anonymous
- Password: (Just press on enter key).
- ftp> cd pub
- ftp> ls

```
[root@localhost ~]# ftp 172.25.0.11
Connected to 172.25.0.11 (172.25.0.11).
220 (vsFTPd 3.0.2)
Name (172.25.0.11:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd pub
250 Directory successfully changed.
ftp> ls
227 Entering Passive Mode (172,25,0,11,238,239).
150 Here comes the directory listing.
-rw-r--r--    1 0        0              25 Sep 24 18:38 file1
-rw-r--r--    1 0        0              25 Sep 24 18:38 file2
```

Now use ftp commands to download those files

Use command 'help' to get some help.

```
ftp> help
Commands may be abbreviated.  Commands are:

!               debug           mdir            sendport        site
$               dir             mget            put             size
account         disconnect      mkdir           pwd             status
append          exit            mls             quit            struct
ascii           form            mode            quote           system
bell            get             modtime         recv            sunique
binary          glob            mput            reget           tenex
bye             hash            newer           rstatus         tick
case            help            nmap            rhelp           trace
cd              idle            nlist           rename          type
cdup            image           ntrans          reset           user
chmod           lcd             open            restart         umask
close           ls              prompt          rmdir           verbose
cr              macdef          passive         runique         ?
delete          mdelete         proxy           send
ftp>
```

- use command ''get'' to download files.

```
ftp> ls
227 Entering Passive Mode (172,25,0,11,131,203).
150 Here comes the directory listing.
-rw-r--r--    1 0        0              25 Sep 24 18:38 file1
-rw-r--r--    1 0        0              25 Sep 24 18:38 file2
226 Directory send OK.
ftp> get file1
local: file1 remote: file1
227 Entering Passive Mode (172,25,0,11,58,226).
150 Opening BINARY mode data connection for file1 (25 bytes).
226 Transfer complete.
25 bytes received in 4.7e-05 secs (531.91 Kbytes/sec)
ftp> !cat file1
this is content of file1
ftp>
```

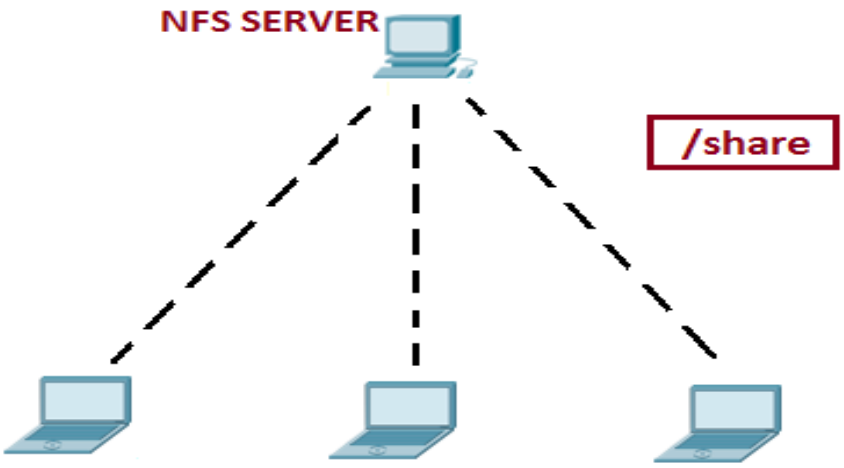
- use '!' to run linux commands
- Ex: to create a file " !cat > filename"

To check “! ls”

- Uploading a file from client to server
- ftp> !cat > file_from_client
- ftp> put file_from_client (put is the ftp command used to upload a file.)

```
ftp> !cat file_from_client
Hi, I am Client machine...
I'm uploading a file to server.
ftp> put file_from_client
local: file_from_client remote: file_from_client
227 Entering Passive Mode (172,25,0,11,104,111).
150 Ok to send data.
226 Transfer complete.
59 bytes sent in 0.000103 secs (572.82 Kbytes/sec)
ftp> ls
227 Entering Passive Mode (172,25,0,11,69,133).
150 Here comes the directory listing.
-rw-r--r--    1 0        0              25 Sep 24 18:38 file1
-rw-r--r--    1 0        0              25 Sep 24 18:38 file2
-rw-------    1 14       50             59 Sep 24 19:18 file_from_client
226 Directory send OK.
ftp>
```

NFS(NETWORK FILE SYSTEM/SHARING)



Here /share is a folder that is been shared over NFS server with its client .

- 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 window 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.

Profile for NFS:

- Package : nfs-utils
- Daemons : nfs-server (nfsd in RHEL 6,5,4)
- Port number : 2049
- Configuration File : /etc/exports

Steps to configure NFS server:

Step1: Install the NFS package using yum or rpm.

Step2: Create a dir or directory on partition 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.

- #yum install nfs-utils*
- To Check whether the package is installed

```
#rpm –q nfs-utils
```

```
[root@server ~]# yum install nfs-utils* -y
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
Package 1:nfs-utils-1.3.0-0.48.el7.x86_64 already installed and latest version
Nothing to do
[root@server ~]# rpm -qa nfs*
nfsometer-1.7-1.el7.noarch
nfs4-acl-tools-0.3.3-15.el7.x86_64
nfstest-2.1.5-1.el7.noarch
nfs-utils-1.3.0-0.48.el7.x86_64
[root@server ~]#
```

- ⇨ Add firewalls and enable the nfs service.
- ⇨ #firewall-cmd --permanent --add-service=nfs
- ⇨ #firewall-cmd --reload
- ⇨ #systemctl enable nfs-server
- ⇨ #systemctl start nfs-server

Step2: Create a directory or create a partition and mount it and make a mount point and add data to it.

Here I am creating two directories under '/ ' to export with different mount options(ex. rw and ro).

```
[root@server ~]# mkdir /private
[root@server ~]# cd /private/
[root@server private]# touch file{1..5}
[root@server private]# ls
file1  file2  file3  file4  file5
[root@server private]# mkdir /public
[root@server private]# cd /public/
[root@server public]# touch a b c d
[root@server public]# ls
a  b  c  d
[root@server public]#
```

Step3: Export the directory by editing /etc/exports file and using exportfs command. Edit the /etc/exports file

- ⇨ #vim /etc/exports

```
[root@server ~]# vim /etc/exports
[root@server ~]# cat /etc/exports
/public         172.25.1.0/24(rw,sync)
/private        172.25.1.0/24(ro,sync)
[root@server ~]#
```

/private and /public: Names of the directories to be exported

172.25.1.0/24 : Range of network where directory can be mounted

To give permission to only one node, just give the IP ADDR Of that node (ex: 172.25.1.11)

(rw, sync) and (ro,sync): Mount options

The Mount options which can be used

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 users

Now run the exportfs command to export the directory

  exportfs - maintain table of exported NFS file systems

#exportfs -avr

  'a' for exporting, 'v' for verbose, 'r' for re-exporting all directories

Step4: Restart the service.

   ⇨ #systemctl restart nfs-server

```
[root@server ~]# firewall-cmd --permanent --remove-service=nfs
success
[root@server ~]# firewall-cmd --reload
success
[root@server ~]# vim /etc/exports
[root@server ~]# cat /etc/exports
/public          172.25.1.11(rw,sync,no_root_squash)
/private         172.25.1.11(ro,sync,no_root_squash)
[root@server ~]# systemctl restart nfs-server
[root@server ~]# exportfs -avr
exporting 172.25.1.11:/private
exporting 172.25.1.11:/public
[root@server ~]# showmount -e
Export list for server.mist.com:
/private 172.25.1.11
/public  172.25.1.11
[root@server ~]#
```

On Client

Step1: Install nfs (same as server).

Step2: Check exported list for server using 'showmount' command

- ⇨ #showmount -e <server-ip addr> or <server-hostname>
- ⇨ #showmount -e server.mist.com (in our case.)

```
[root@client ~]# #showmount -e <server ip-address>
[root@client ~]# showmount -e 172.25.1.10
Export list for 172.25.1.10:
/private 172.25.1.11
/public  172.25.1.11
[root@client ~]# mkdir /private
[root@client ~]# mkdir /public
[root@client ~]#
```

We are going to mount our nfs shared directories in '/private' and '/public' directories on client machine.

Step3: mounting exported directories.

Here, we have different kinds of mount options.. like below

- ⇨ Temporary mounting
- ⇨ Permanent mounting
- ⇨ Automounting

Temporary Mounting:

We can mount nfs temporarily, but a system reboot will make that nfs mount unavailable.

```
[root@client ~]# #temporary mounting
[root@client ~]# mount.nfs 172.25.1.10:/private /private
[root@client ~]# mount.nfs 172.25.1.10:/public /public
[root@client ~]# cd /public/
[root@client public]# ls
a     a13  a2   a26  a32  a39  a45  a51  a58  a64  a70  a77  a83  a9   a96  e
a1    a14  a20  a27  a33  a4   a46  a52  a59  a65  a71  a78  a84  a90  a97
a10   a15  a21  a28  a34  a40  a47  a53  a6   a66  a72  a79  a85  a91  a98
a100  a16  a22  a29  a35  a41  a48  a54  a60  a67  a73  a8   a86  a92  a99
a101  a17  a23  a3   a36  a42  a49  a55  a61  a68  a74  a80  a87  a93  b
a11   a18  a24  a30  a37  a43  a5   a56  a62  a69  a75  a81  a88  a94  c
a12   a19  a25  a31  a38  a44  a50  a57  a63  a7   a76  a82  a89  a95  d
[root@client public]# cd /private/
[root@client private]# ls
file1  file2  file3  file4  file5
[root@client private]#
```

Permanent Mounting:

We can mount nfs permanently with the help of /etc/fstab file.

```
[root@client ~]# ##permanent mounting##
[root@client ~]# vim /etc/fstab
[root@client ~]# tail -4 /etc/fstab #add following fields in fstab

172.25.1.10:/private    /private        nfs     defaults        0       0

172.25.1.10:/public     /public         nfs     defaults        0       0
[root@client ~]# mount -a
[root@client ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda2              15G  3.4G   11G  24% /
devtmpfs              560M     0  560M   0% /dev
tmpfs                 575M     0  575M   0% /dev/shm
tmpfs                 575M  8.3M  566M   2% /run
tmpfs                 575M     0  575M   0% /sys/fs/cgroup
/dev/sda1             488M  126M  327M  28% /boot
tmpfs                 115M  4.0K  115M   1% /run/user/42
tmpfs                 115M   36K  115M   1% /run/user/0
/dev/sr0              8.1G  8.1G     0 100% /run/media/root/CentOS 7 x86_64
172.25.1.10:/private   15G   13G  1.8G  88% /private
172.25.1.10:/public    15G   13G  1.8G  88% /public
[root@client ~]#
```

Automounting

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.

We have to install a package called ‘autofs*’ to auto mount nfs.

```
[root@client ~]# ##auto mounting##
[root@client ~]# yum install autofs* -y
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
Package 1:autofs-5.0.7-69.el7.x86_64 already installed and latest version
Nothing to do
[root@client ~]# systemctl enable autofs
[root@client ~]# systemctl start autofs
[root@client ~]#
```

After installing autofs edit the file ‘/etc/auto.master’.

```
[root@client ~]# vim /etc/auto.master
[root@client ~]# head /etc/auto.master
#
# Sample auto.master file
# This is a 'master' automounter map and it has the following format:
# mount-point [map-type[,format]:]map [options]
# For details of the format look at auto.master(5).
#
/misc   /etc/auto.misc
/mnt    /etc/MyAutofs  #(## ADD THIS LINE ##)#
#
# NOTE: mounts done from a hosts map will be mounted with the
[root@client ~]#
```

⇨ Add the “/mnt /etc/MyAutofs” to configure our nfs exported directories.

⇨ Now, create /etc/MyAutofs and configure nfs exports

⇨ #vim   /etc/MyAuotfs

```
[root@client ~]# vim /etc/MyAutofs
[root@client ~]# head /etc/MyAutofs ##add following in MyAutofs file##
private -ro,sync        172.25.1.10:/private
public  -ro,sync        172.25.1.10:/public
[root@client ~]# ##now restart the service##
[root@client ~]# systemctl restart autofs.service
[root@client ~]#
```

Private,public are autofs names that we have given and “–ro” permissions and next nfs directories following by server ip address.

Now restart the autofs.

⇨ #systemctl restart autofs
⇨ Now navigate to autofs directory
⇨ #cd /mnt

```
[root@client ~]# systemctl restart autofs.service
[root@client ~]# cd /mnt
[root@client mnt]# ls
[root@client mnt]# cd public
[root@client public]# ls
a     a13  a2   a26  a32  a39  a45  a51  a58  a64  a70  a77  a83  a9   a96  e
a1    a14  a20  a27  a33  a4   a46  a52  a59  a65  a71  a78  a84  a90  a97
a10   a15  a21  a28  a34  a40  a47  a53  a6   a66  a72  a79  a85  a91  a98
a100  a16  a22  a29  a35  a41  a48  a54  a60  a67  a73  a8   a86  a92  a99
a101  a17  a23  a3   a36  a42  a49  a55  a61  a68  a74  a80  a87  a93  b
a11   a18  a24  a30  a37  a43  a5   a56  a62  a69  a75  a81  a88  a94  c
a12   a19  a25  a31  a38  a44  a50  a57  a63  a7   a76  a82  a89  a95  d
[root@client public]# df -h
Filesystem           Size  Used Avail Use% Mounted on
/dev/sda2             15G  3.4G   11G  24% /
devtmpfs             560M     0  560M   0% /dev
tmpfs                575M     0  575M   0% /dev/shm
tmpfs                575M  8.2M  566M   2% /run
tmpfs                575M     0  575M   0% /sys/fs/cgroup
/dev/sda1            488M  126M  327M  28% /boot
tmpfs                115M  4.0K  115M   1% /run/user/42
tmpfs                115M   28K  115M   1% /run/user/0
/dev/sr0             8.1G  8.1G     0 100% /run/media/root/CentOS 7 x86_64
172.25.1.10:/public   15G   13G  1.8G  88% /mnt/public
[root@client public]# ##here we can see/mnt/public automounted
```

Above, we used “/mnt/public” only that’s why when we check mounted filesystems using ‘df –h’ then it showing /mnt/public only.

```
[root@client mnt]# ls
public
[root@client mnt]# cd private
[root@client private]# ls
file1  file2  file3  file4  file5
[root@client private]# cd ..
[root@client mnt]# ls
private  public
[root@client mnt]# df -h | grep 172.25.1.10
172.25.1.10:/public    15G   13G  1.8G  88% /mnt/public
172.25.1.10:/private   15G   13G  1.8G  88% /mnt/private
[root@client mnt]# ##now both automounted##
```

We configured /private as read-only(ro) and /public as read-write(rw).

```
[root@client ~]# cd /mnt
[root@client mnt]# cd private/
[root@client private]# ls
file1  file2  file3  file4  file5
[root@client private]# touch newfile1
touch: cannot touch ‘newfile1’: Read-only file system
[root@client private]# ###we can not modify/add new data to private
[root@client private]# ###becoz it's read only
[root@client private]# cd ../public
[root@client public]# ls
a     a13  a2   a26  a32  a39  a45  a51  a58  a64  a70  a77  a83  a9   a96
a1    a14  a20  a27  a33  a4   a46  a52  a59  a65  a71  a78  a84  a90  a97
a10   a15  a21  a28  a34  a40  a47  a53  a6   a66  a72  a79  a85  a91  a98
a100  a16  a22  a29  a35  a41  a48  a54  a60  a67  a73  a8   a86  a92  a99
a101  a17  a23  a3   a36  a42  a49  a55  a61  a68  a74  a80  a87  a93  b
a11   a18  a24  a30  a37  a43  a5   a56  a62  a69  a75  a81  a88  a94  c
a12   a19  a25  a31  a38  a44  a50  a57  a63  a7   a76  a82  a89  a95  d
[root@client public]# touch newfile2
[root@client public]# ##we can insert/modify content in public
[root@client public]# ##becoz we configured it as read-write(rw)
[root@client public]#
```

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.

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 : smb

Portno : 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

Steps to configure SAMBA server:

Step1: Installing samba server

⇨ #yum install samba* -y

```
[root@client ~]# rpm -q samba
samba-4.6.2-8.el7.x86_64
[root@client ~]# firewall-cmd --permanent --add-service=samba
success
[root@client ~]# firewall-cmd --reload
success
[root@client ~]# systemctl enable smb.service
Created symlink from /etc/systemd/system/multi-user.target.wants/smb.service to
b/systemd/system/smb.service.
[root@client ~]# systemctl start smb.service
[root@client ~]#
```

⇨ Add firewalls and start the service.

Step2: Make a directory and assign full permission to it ,which will be shared

⇨ #mkdir /share

⇨ Now we have to change the SELinux context values of our created directory '/share' from default to samba by using 'semanage fcontext command' and to make this effective we need to run another command called 'restorecon' (in previous versions it is 'chcon').

⇨ #ls -ldZ /share (to check context of our directory.)

```
[root@client ~]# mkdir /share
[root@client ~]# ls -ldZ /share
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 /share
[root@client ~]# cd /share
[root@client share]# touch a b c d
[root@client share]# chmod 777 /share
[root@client share]# ls -ldZ /share
drwxrwxrwx. root root unconfined_u:object_r:default_t:s0 /share
[root@client share]# ###now Change default context to samba###
[root@client share]# semanage fcontext -a -t samba_share_t "/share(/.*)?"
[root@client share]# ls -ldZ /share
drwxrwxrwx. root root unconfined_u:object_r:default_t:s0 /share
[root@client share]# ###still it didn't changed###
[root@client share]# ###to apply changes we use another command #restorecon###
[root@client share]# restorecon -vRF /shares/
[root@client share]# ###Done###
[root@client share]#
```

```
[root@client ~]# !ls
ls -ldZ /share
drwxrwxrwx. root root unconfined_u:object_r:default_t:s0 /share
[root@client ~]# restorecon -vRF /share
restorecon reset /share context unconfined_u:object_r:default_t:s0->system_u:object_r:s
amba_share_t:s0
restorecon reset /share/c context unconfined_u:object_r:default_t:s0->system_u:object_r
:samba_share_t:s0
restorecon reset /share/b context unconfined_u:object_r:default_t:s0->system_u:object_r
:samba_share_t:s0
restorecon reset /share/a context unconfined_u:object_r:default_t:s0->system_u:object_r
:samba_share_t:s0
restorecon reset /share/d context unconfined_u:object_r:default_t:s0->system_u:object_r
:samba_share_t:s0
[root@client ~]# ls -ldZ /share
drwxrwxrwx. root root system_u:object_r:samba_share_t:s0 /share
[root@client ~]#
```

Step3: Create a user or use any existing user who will be allowed to log in as samba user, add that user to samba user.

useradd feroz

```
[root@client ~]# #Creating A Samba User#
[root@client ~]# useradd feroz
```

⇨ Add a password to that user using command ‘smbpasswd’
⇨ #smbpasswd -a <username>

```
[root@client ~]# ##give passwd to samba user##
[root@client ~]# vipw

vipw: /etc/passwd is unchanged
[root@client ~]# smbpasswd -a feroz
New SMB password:
Retype new SMB password:
Added user feroz.
[root@client ~]# systemctl restart smb
[root@client ~]#
```

Note: To delete a user from samba user #smbpasswd –x <user name>

⇨ To check all the samba users use
⇨ #pdbedit -L

Step4: Go to the configuration file i.e. /etc/samba/smb.conf and make the following changes.

```
[shares]
  path=/share
  read=yes
  writeable=yes
  browseable=yes
  valid users=feroz
  valid hosts=192.168.0.0/24
```

⇨ #vim /etc/samba/smb.conf

```
[root@client ~]# vi /etc/samba/smb.conf
[root@client ~]# tail -8 /etc/samba/smb.conf
###Samba Configuration###
[shares]
  path=/share
  read=yes
  writeable=yes
  browseable=yes
  valid users=feroz
  valid hosts=192.168.0.0/24
[root@client ~]#
```

Explanation about the above fields

⇨ [shares] : Share Name
⇨ Path = /share : Share Directory
⇨ read = yes : Read permission
⇨ Valid user = feroz : Authorized user
⇨ Writable = yes : Write Permission
⇨ valid hosts= 192.168.0.0/24 Network Range or host range

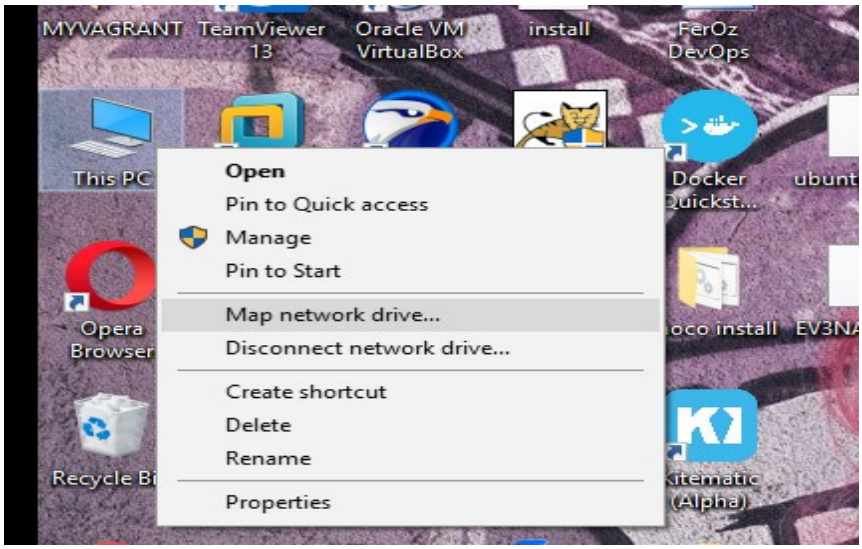
Step5: Test the samba parameters and restart the service.

⇨ To test the parameters use the following command
⇨ #testparm
⇨ #systemctl restart smb.service

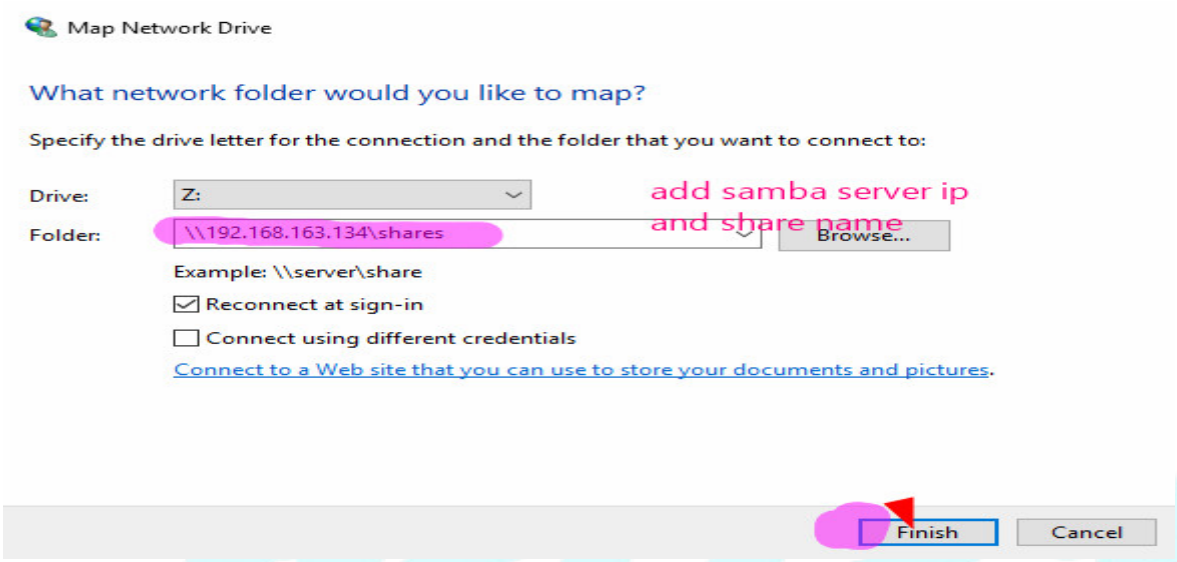
Windows as a samba client:

To connect from windows to the samba server, Right click on “My Computer” or “This PC” icon select

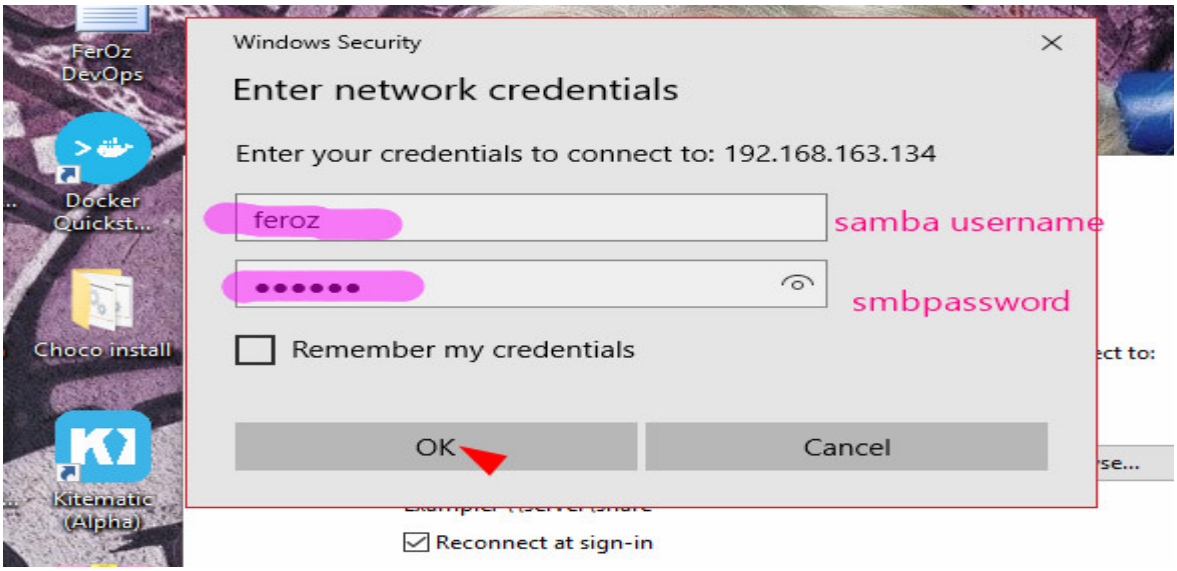
“”Map Network Drive…””



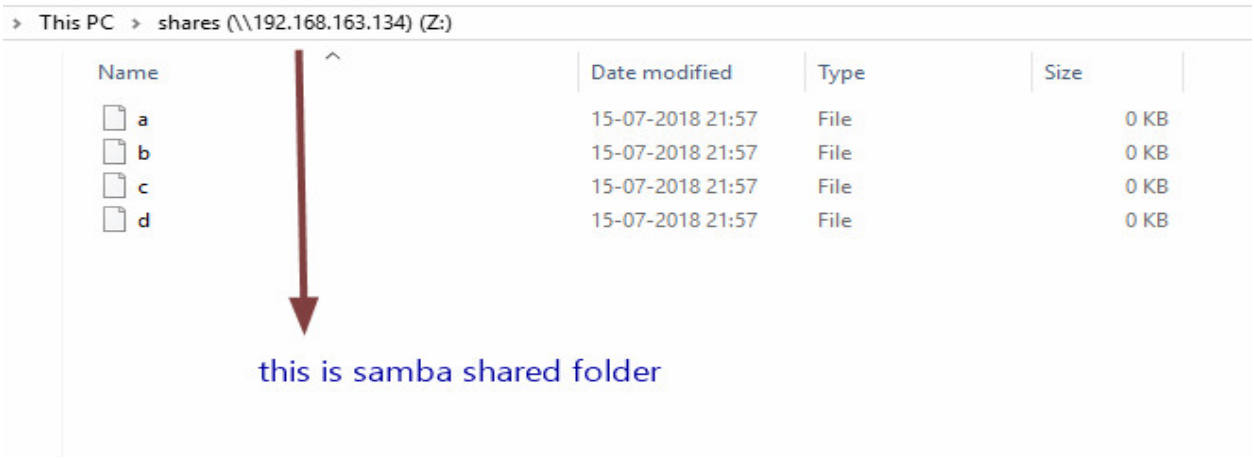
⇨ Give the address of samba server as “\\192.168.163.134\shares”(\\ <ip-address>\<share-name>s), press on finish to continue.



⇨ It will prompt for user name and passwd, give samba user and passwd and click on OK.



Now a window will be opened..



We can use Linux as client of SAMBA

MariaDB

MariaDB Server is one of the most popular database servers in the world. It's made by the original developers of MySQL and guaranteed to stay open source.

Server Profile:

- Packages: mariadb,mariadb-server,mariadb-libs
- Daemon Name: mariadb
- Service name: mysql
- Port Number: 3306
- Configuration file path: /etc/my.cnf

Keypoints:

i. MariaDB is a relational database server which stores data in Tables (rows and columns).
ii. We have to create databases to store database tables in a Database server.

INSTALLING MARIADB ON RHEL 7/CentOS 7:

⇨ # yum install mariadb* -y

```
[root@server ~]# yum install mariadb* -y
Loaded plugins: fastestmirror, langpacks
ferozbase                                                | 3.6 kB  00:00:00
Loading mirror speeds from cached hostfile
Package 1:mariadb-embedded-5.5.56-2.el7.x86_64 already installed and latest version
Package 1:mariadb-server-5.5.56-2.el7.x86_64 already installed and latest version
Package 1:mariadb-test-5.5.56-2.el7.x86_64 already installed and latest version
Package 1:mariadb-devel-5.5.56-2.el7.x86_64 already installed and latest version
Package 1:mariadb-bench-5.5.56-2.el7.x86_64 already installed and latest version
Package 1:mariadb-embedded-devel-5.5.56-2.el7.x86_64 already installed and latest versi
on
Package 1:mariadb-5.5.56-2.el7.x86_64 already installed and latest version
Package 1:mariadb-libs-5.5.56-2.el7.x86_64 already installed and latest version
Nothing to do
[root@server ~]# rpm -q mariadb
mariadb-5.5.56-2.el7.x86_64
[root@server ~]#
```

⇨ Firewalls:
⇨ # firewall-cmd - -permanent - -add-service=mysql
⇨ # firewall-cmd - -reload

Enable and Start MySQL service:

⇨ # systemctl enable mariadb.service
⇨ # systemctl start mariadb.service

Setting password for root of MariaDB server:

By default MariaDB “root” does not have a password, you have to change the root password after you install it the first time.

Now to secure mariadb, we have to set root user password for mariadb,remove anonymous user,disallow login remote,remove test database and etc.
installing the MariaDB server using below command. Hardening mariaD

⇨ # mysql_secure_installation

```
File  Edit  View  Search  Terminal  Help
[root@server ~]# ##root password##
[root@server ~]# mysql_secure_installation
```

⇨ Now connect to mariadb server with root password first time.
⇨ #mysql -u root –p
(-u for user and –p for password)

```
File  Edit  View  Search  Terminal  Help
[root@server ~]# ##login to mysql server##
[root@server ~]# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 15
Server version: 5.5.56-MariaDB MariaDB Server

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

⇨ SQL query to check available databases in database server.
⇨ > show databases;
⇨ (Note: ‘ ; ’ is compulsory at the end of any query.)

```
MariaDB [(none)]> #to see databases use 'show databases;'
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| feroz              |
| mysql              |
| new                |
| performance_schema |
+--------------------+
5 rows in set (0.11 sec)

MariaDB [(none)]>
```

Queries To Create a new database and to use that created databases:
⇨ > create database <database_name>;
Ex: create database mistdb

```
MariaDB [(none)]> #to create a database use 'create database <database-name>;'
MariaDB [(none)]> create database mistdb;
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| feroz              |
| mistdb             |
| mysql              |
| new                |
| performance_schema |
+--------------------+
6 rows in set (0.01 sec)

MariaDB [(none)]>
```

here 'mistdb' is our newly created database

⇨ Query to use a database
1. use <database_name>;
2. use mistdb;

```
MariaDB [(none)]> #to use database use 'use <database-name>;'
MariaDB [(none)]> use mistdb;
Database changed
MariaDB [mistdb]>
```

⇨ Creation of tables
⇨ > create table <tablename> followed by rows and columns along with the sizes and field names

```
MariaDB [mistdb]> #to insert data in 'mistdb' database we have to create tables
MariaDB [mistdb]> #syntax--> create table <table-name> followed by field names and data
types and their sizes as shown below
MariaDB [mistdb]> create table misttab1(course_id int(2),course_name varchar(15),durati
on varchar(15));
Query OK, 0 rows affected (0.01 sec)

MariaDB [mistdb]> #our misttab1 successfully created
MariaDB [mistdb]> #now we can insert dato into it.
```

⇨ To check description of our table 'misttab1' in mistdb database use '> desc <tablename>'

```
MariaDB [mistdb]> #to check fields and their datatypes of table
MariaDB [mistdb]> describe misttab1;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| course_id   | int(2)      | YES  |     | NULL    |       |
| course_name | varchar(15) | YES  |     | NULL    |       |
| duration    | varchar(15) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

MariaDB [mistdb]>
```

⇨ Now we can insert our data into our table misttab1
⇨ > insert into misttab1 values (01, 'linux','35 days');

```
MariaDB [mistdb]> #inserting data to the table
MariaDB [mistdb]> insert into misttab1 values(01,'linux','35 days');
Query OK, 1 row affected (0.00 sec)

MariaDB [mistdb]> insert into misttab1 values(02,'devops','35 days');
Query OK, 1 row affected (0.06 sec)

MariaDB [mistdb]> insert into misttab1 values(03,'aws','30 days');
Query OK, 1 row affected (0.00 sec)

MariaDB [mistdb]> #we inserted 3 rows of data
```

⇨ To check the data from a table use “select * from <table-name>;”

```
MariaDB [mistdb]> #to check the data of a table
MariaDB [mistdb]> select * from misttab1;
+-----------+-------------+----------+
| course_id | course_name | duration |
+-----------+-------------+----------+
|         1 | linux       | 35 days  |
|         2 | devops      | 35 days  |
|         3 | aws         | 30 days  |
+-----------+-------------+----------+
3 rows in set (0.00 sec)

MariaDB [mistdb]>
```

⇨ Sql queries slicing

```
MariaDB [mistdb]> #sql queries
MariaDB [mistdb]> select course_name from misttab1 where course_id=2;
+-------------+
| course_name |
+-------------+
| devops      |
+-------------+
1 row in set (0.00 sec)

MariaDB [mistdb]> select duration from misttab1 where course_name='linux';
+----------+
| duration |
+----------+
| 35 days  |
+----------+
1 row in set (0.00 sec)

MariaDB [mistdb]>
```

⇨ <u>Creating users:</u>

```
MariaDB [(none)]> #creating a normal user
MariaDB [(none)]> create user 'red'@'localhost' identified by 'redhat';
Query OK, 0 rows affected (0.01 sec)

MariaDB [(none)]> create user 'blue'@'localhost' identified by 'redhat';
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> #here 'redhat' is password to login to database server
MariaDB [(none)]>
```

⇨ Giving privileges to the normal users that we created over the the database mistdb;

```
MariaDB [(none)]> #giving privilleges on our mistdb database to normal users
MariaDB [(none)]> grant all on mistdb.* to 'red'@'localhost';
Query OK, 0 rows affected (0.05 sec)

MariaDB [(none)]> grant select on mistdb.* to 'blue'@'localhost';
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> #here for user red we have given all privilleges but to blue we have
given only select operation over our database 'mistdb'
```

⇨ Connecting to our database with normal user "red" who have all permissions over our database.

```
# mysql  -u  red –p
```

```
[root@server ~]# mysql -u red -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 17
Server version: 5.5.56-MariaDB MariaDB Server

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use mistdb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed

MariaDB [mistdb]> show tables;
+------------------+
| Tables_in_mistdb |
+------------------+
| misttab1         |
+------------------+
1 row in set (0.00 sec)

MariaDB [mistdb]>
```

⇨ User 'red' have all privileges over mistdb database so 'red' can update new data to our table.

```
MariaDB [mistdb]> insert into misttab1 values(4,'ms-azure','30 days');
Query OK, 1 row affected (0.00 sec)

MariaDB [mistdb]> select * from misttab1;
+-----------+-------------+----------+
| course_id | course_name | duration |
+-----------+-------------+----------+
|         1 | linux       | 35 days  |
|         2 | devops      | 35 days  |
|         3 | aws         | 30 days  |
|         4 | ms-azure    | 30 days  |
+-----------+-------------+----------+
4 rows in set (0.01 sec)

MariaDB [mistdb]>
```

⇨ Now connect with another user 'blue' who have only 'select' privilege.

```
[root@server ~]# mysql -u blue -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 18
Server version: 5.5.56-MariaDB MariaDB Server

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use mistdb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [mistdb]>
```

⇨ User ‘blue’ can not insert data in the tables except reading data.

```
MariaDB [mistdb]> select * from misttab1;
+-----------+-------------+----------+
| course_id | course_name | duration |
+-----------+-------------+----------+
|         1 | linux       | 35 days  |
|         2 | devops      | 35 days  |
|         3 | aws         | 30 days  |
|         4 | ms-azure    | 30 days  |
+-----------+-------------+----------+
4 rows in set (0.00 sec)

MariaDB [mistdb]> insert into misttab1 values(5,'google cloud','40 days');
ERROR 1142 (42000): INSERT command denied to user 'blue'@'localhost' for table 'misttab
1'
MariaDB [mistdb]>
```

⇨ Creating a Remote mariadb user.

```
File  Edit  View  Search  Terminal  Help
MariaDB [(none)]> #creating a remote user
MariaDB [(none)]> create user 'feroz'@'172.25.1.%' identified by 'redhat';
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> grant all on mistdb.* to 'feroz'@'172.25.1.%';
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]>
```

⇨ On server2

3. Install mariadb package and run following command to connect to mariadb server over the network.
4. # mysql -u <username> -h <host ip or hostname> -p
5. #mysql -u feroz –h 172.25.1.10 -p
   Enter password to connect to remote database server.

```
File  Edit  View  Search  Terminal  Help
[root@client ~]# rpm -q mariadb
mariadb-5.5.56-2.el7.x86_64
[root@client ~]# mysql -u feroz -h 172.25.1.10 -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 5.5.56-MariaDB MariaDB Server

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use mistdb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [mistdb]> show tables;
+------------------+
| Tables_in_mistdb |
+------------------+
| misttab1         |
+------------------+
1 row in set (0.00 sec)

MariaDB [mistdb]>
```

==========