RHEL9

Session 11 – 3rd Dec 2022 Summary

- On a single directory or a file, the different users can be given different access and through this access we can control their power this is called as Access Control List (ACL).
- The command to create a directory "/mydir" is "**mkdir /mydir**"

```
[root@localhost ~]# mkdir  /mydir
[root@localhost ~]# ls -ld /mydir
drwxr-xr-x. 2 root root 6 Dec  3 14:12 /mydir
[root@localhost ~]#
```

- The command to check ACL permission set "**getfacl /mydir**"

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

- To implement ACL on the directory **"/mydir**", basically change the mode **(-m)** on the user (**u**) "**user1**" and set "**rwx**" permissions. The command used is "**setfacl –m u:user1:rwx /mydir**/"

```
[root@localhost ~]# setfacl  -m  u:user1:rwx  /mydir/
[root@localhost ~]#
```

- The command to check ACL permission is set "**getfacl /mydir**"

```
[root@localhost ~]# getfacl /mydir/
getfacl: Removing leading '/' from absolute path names
# file: mydir/
# owner: root
# group: root
user::rwx
user:user1:rwx
group::r-x
mask::rwx
other::r-x
```

- The command to perform long list on the directory is "**ls –ld /mydir/**". The "+" sign indicates ACL is set. The "**user1**" is part of ACL and no longer remain others, we see that **user1** permissions is set to "**rwx**" and **others** to "**rx**".

```
[root@localhost ~]# ls -ld /mydir/
drwxrwxr-x+ 2 root root 6 Dec  3 14:12 /mydir/
[root@localhost ~]#
```

```
[user1@localhost mydir]$ cd /mydir/
[user1@localhost mydir]$ touch hi
[user1@localhost mydir]$ ls -l
total 0
-rw-r--r--. 1 user1 user1 0 Dec  3 14:28 hi
[user1@localhost mydir]$
[user1@localhost mydir]$
[user1@localhost mydir]$
[user1@localhost mydir]$ rm hi
```

- Similarly individual users can be given different permissions

```
[root@localhost ~]# setfacl   -m  u:user2:rx  /mydir/
[root@localhost ~]# setfacl   -m  u:user3:1  /mydir/
[root@localhost ~]# setfacl   -m  u:user4:0  /mydir/
[root@localhost ~]# getfacl  /mydir/
getfacl: Removing leading '/' from absolute path names
# file: mydir/
# owner: root
# group: root
user::rwx
user:user1:rwx
user:user2:r-x
user:user3:--x
user:user4:---
group::r-x
mask::rwx
other::r-x
```

```
[root@localhost ~]# su - user3
[user3@localhost ~]$ cd /mydir/
[user3@localhost mydir]$ ls
ls: cannot open directory '.': Permission denied
[user3@localhost mydir]$
```

- The command to set **no permissions** to **other** users on the directory "/mydir/ "**setfacl –m o:--- /mydir**/"

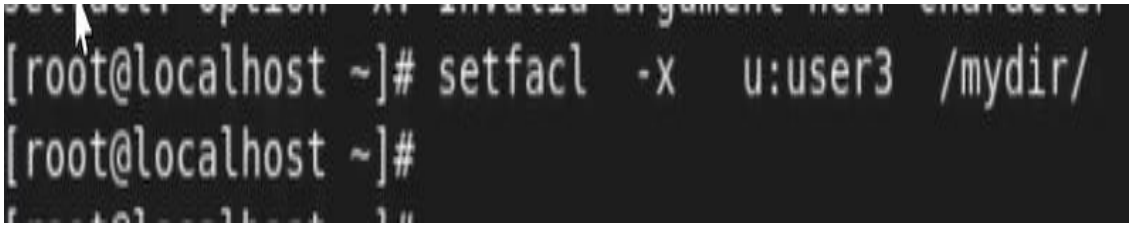
```
[root@localhost ~]# setfacl  -m  o:---  /mydir/
[root@localhost ~]# ls -ld /mydir/
drwxrwx---+ 2 root root 6 Dec  3 14:29 /mydir/
[root@localhost ~]# ls -ld /mydir/
drwxrwx---+ 2 root root 6 Dec  3 14:29 /mydir/
```

```
[root@localhost ~]# su - vimal
[vimal@localhost ~]$ cd /mydir/
-bash: cd: /mydir/: Permission denied
```

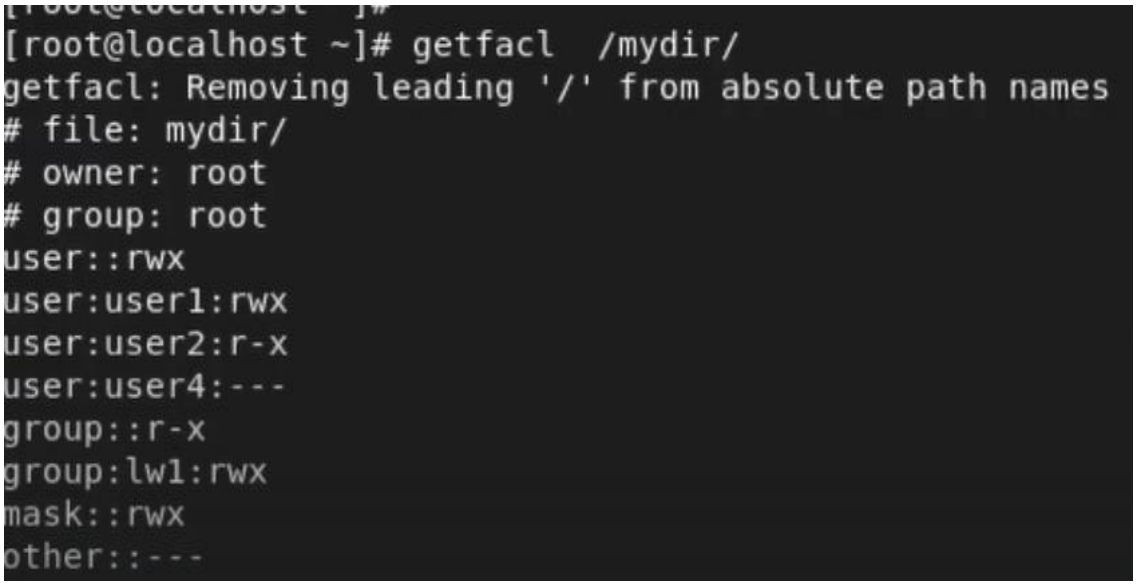
- The command to see the help of "setfacl" command is "**setfacl –h**", to see some of the important keywords

```
[root@localhost ~]# setfacl  -h
setfacl 2.3.1 -- set file access control lists
Usage: setfacl [-bkndRLP] { -m|-M|-x|-X ... } file ...
  -m, --modify=acl        modify the current ACL(s) of file(s)
  -M, --modify-file=file  read ACL entries to modify from file
  -x, --remove=acl        remove entries from the ACL(s) of file(s)
  -X, --remove-file=file  read ACL entries to remove from file
  -b, --remove-all        remove all extended ACL entries
  -k, --remove-default    remove the default ACL
      --set=acl           set the ACL of file(s), replacing the current ACL
      --set-file=file     read ACL entries to set from file
      --mask              do recalculate the effective rights mask
  -n, --no-mask           don't recalculate the effective rights mask
  -d, --default           operations apply to the default ACL
  -R, --recursive         recurse into subdirectories
  -L, --logical           logical walk, follow symbolic links
  -P, --physical          physical walk, do not follow symbolic links
      --restore=file      restore ACLs (inverse of `getfacl -R')
      --test              test mode (ACLs are not modified)
  -v, --version           print version and exit
  -h, --help              this help text
[root@localhost ~]#
```

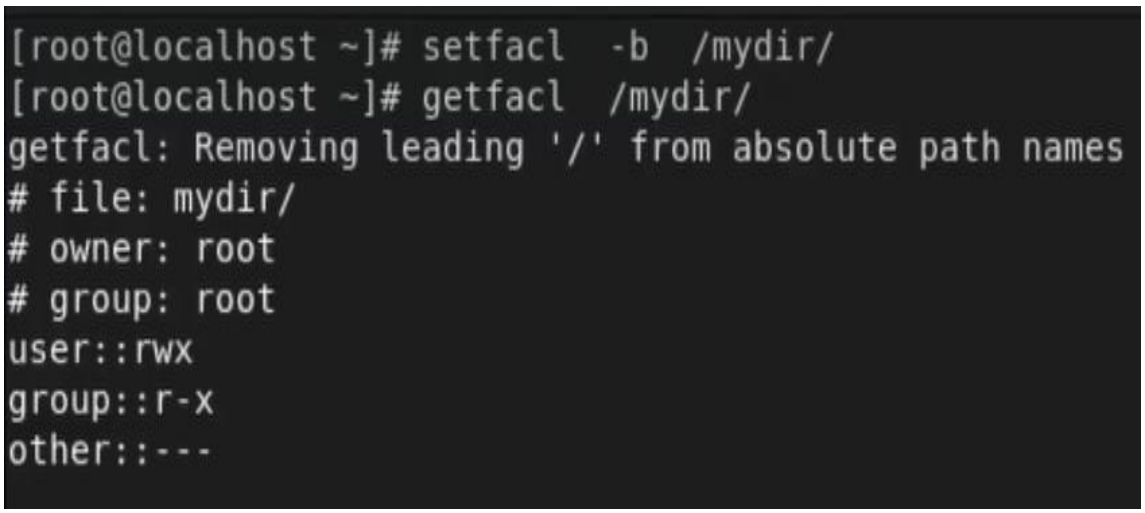
- The command to remove the ACL permission "**setfacl –x u:user3 /mydir**/"



- The command to check ACL permission removed "**getfacl /mydir**"



- The command to remove all the ACL permissions "**setfacl –b /mydir**/"



- The command to set default ACL "**setfacl –m d: u: user1: rwx /mydir**", if any user creates a file or directory inside a directory, the ACL is set by default.

```
[root@localhost ~]# setfacl  -m  d:u:user1:rwx  /mydir/
[root@localhost ~]# getfacl  /mydir/
getfacl: Removing leading '/' from absolute path names
# file: mydir/
# owner: root
# group: root
user::rwx
group::r-x
other::---
default:user::rwx
default:user:user1:rwx
default:group::r-x
default:mask::rwx
default:other::---
```

```
[root@localhost ~]# cd /mydir/
[root@localhost mydir]# touch hi
[root@localhost mydir]# mkdir ddd
[root@localhost mydir]# ls -l
total 0
drwxrwx---+ 2 root root 6 Dec  3 14:35 ddd
drwxr-xr-x. 2 root root 6 Dec  3 14:34 dira
-rw-rw----+ 1 root root 0 Dec  3 14:35 hi
[root@localhost mydir]# getfacl  ddd
# file: ddd
# owner: root
# group: root
user::rwx
user:user1:rwx
group::r-x
mask::rwx
other::---
default:user::rwx
default:user:user1:rwx
default:group::r-x
default:mask::rwx
default:other::---
```

- The command to check the manual of "setfacl" command is "**man setfacl**"

```
[root@localhost ~]#
[root@localhost ~]# man setfacl
```

```
--mask
    Do recalculate the effective rights mask, even if an ACL mask entry
    was explicitly given. (See the -n option.)
```

- The command to create a directory "**mkdir /lwdir1**"

```
[root@localhost ~]# mkdir /lwdir1
[root@localhost ~]# ls -ld /lwdir1
drwxr-xr-x. 2 root root 6 Dec  3 14:38 /lwdir1
[root@localhost ~]#
```

- The command to change the user owner of the directory "/lwdir1" is "**chown vimal /lwdir1**"

```
[root@localhost ~]# chown  vimal /lwdir1
[root@localhost ~]# ls -ld /lwdir1
drwxr-xr-x. 2 vimal root 6 Dec  3 14:38 /lwdir1
```

- The command to set ACL permission "rx" on user "vimal" on the directory "/lwdir1"

```
[root@localhost ~]# setfacl  -m  u:vimal:rx  /lwdir1/
```

- Here we see that from POSIX permissions of user owner has "rwx" permissions and new ACL permissions of user vimal has "rx" permissions, but here POSIX permission applies with user perspective (because it's at the top and ACL below – and the priority is given from top to bottom)

```
[root@localhost ~]# setfacl  -m  u:vimal:rx  /lwdir1/
[root@localhost ~]# getfacl  /lwdir1
getfacl: Removing leading '/' from absolute path names
# file: lwdir1
# owner: vimal
# group: root
user::rwx
user:vimal:r-x
group::r-x
mask::r-x
other::r-x

[root@localhost ~]#
[root@localhost ~]# ls -ld /lwdir1
drwxr-xr-x+ 2 vimal root 6 Dec  3 14:38 /lwdir1
```

```
[root@localhost ~]# su - vimal
[vimal@localhost ~]$ cd /lwdir1/
[vimal@localhost lwdir1]$ ls
[vimal@localhost lwdir1]$ touch  hi.txt
[vimal@localhost lwdir1]$ ls
hi.txt
```

- Now login as user “jack” , and we see here the user is able to go into a folder(x), and perform list (r)and when the user jack tries to create files (w), the permission is denied, because “jack” is other user with permission “r-x”

```
[root@localhost ~]# su - jack
[jack@localhost ~]$ cd /lwdir1/
[jack@localhost lwdir1]$ ls
hi.txt
[jack@localhost lwdir1]$ touch hiiii
touch: cannot touch 'hiiii': Permission denied
```

- Now if we give a dedicated permission (ACL) that is “**rwx”** to the user “jack”, even though jack is other user, now we see that when user jack login, the user is able to create files(w)

```
[root@localhost ~]# setfacl  -m u:jack:rwx /lwdir1/
[root@localhost ~]# getfacl  /lwdir1
getfacl: Removing leading '/' from absolute path names
# file: lwdir1
# owner: vimal
# group: root
user::rwx
user:vimal:r-x
user:jack:rwx
group::r-x
mask::rwx
other::r-x
```

```
[root@localhost ~]# su - jack
[jack@localhost ~]$ cd /lwdir1/
[jack@localhost lwdir1]$ touch erhetjy
[jack@localhost lwdir1]$ ls
erhetjy  hi.txt
[jack@localhost lwdir1]$
```

- The command to set a **mask** that is only “x” permission is given on the folder “/lwdir1/”, that is block “rw” and allow only “x”. Here even though the user jack has “rwx” permission, the final permission after setting mask is “x” only, this is called effective permissions.

```
[root@localhost ~]# setfacl  -m  m::x   /lwdir1/
[root@localhost ~]# getfacl   /lwdir1/
getfacl: Removing leading '/' from absolute path names
# file: lwdir1/
# owner: vimal
# group: root
user::rwx
user:vimal:r-x                  #effective:--x
user:jack:rwx                   #effective:--x
group::r-x                      #effective:--x
mask::--x
other::r-x
```

```
[root@localhost ~]# su - jack
[jack@localhost ~]$ cd /lwdir1/
[jack@localhost lwdir1]$ touch dfdfhg
touch: cannot touch 'dfdfhg': Permission denied
[jack@localhost lwdir1]$
```

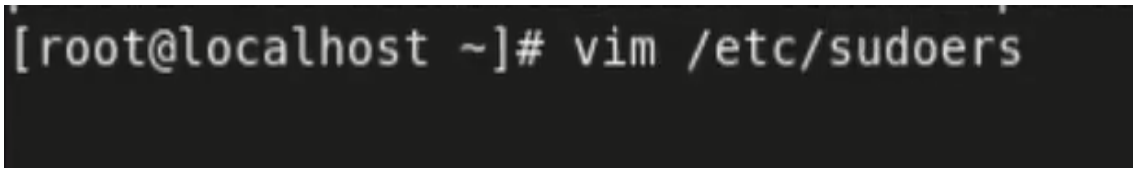
- As soon as we create directories or files, automatically some permissions are already set, the **umask** sets these default permissions, the maximum permissions in a directory is 777 and in file is 666, due to security reasons ”x” permission is not given by default on the files. [Ex:- 777-022=755 & 666-022=644]

```
[root@localhost ~]# mkdir /dd
[root@localhost ~]# ls -ld /dd
drwxr-xr-x. 2 root root 6 Dec  3 14:56 /dd
[root@localhost ~]# touch   /hh
[root@localhost ~]# ls -l /hh
-rw-r--r--. 1 root root 0 Dec  3 14:56 /hh
[root@localhost ~]#
```

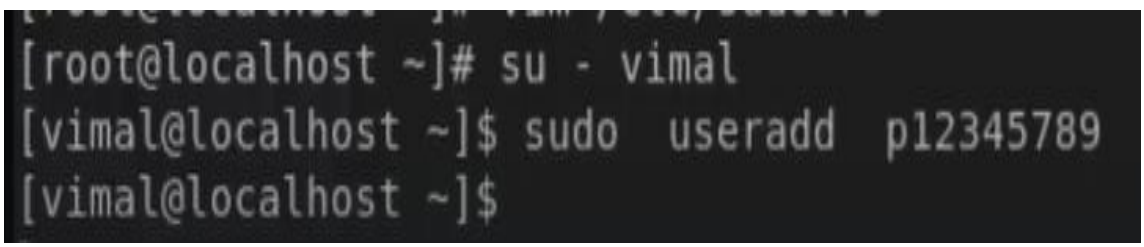
```
[root@localhost ~]# umask
0022
[root@localhost ~]#
```

- The “root” with UID “0” has unlimited power and can run all the admin level command but general user has limited power and cannot run the admin level command, for this some limited users can be given superuser power to run some commands. This is called “**sudo**” power, sudo is the one that gives the general user some extra power to run admin level command

- For this first we have to login as “**root**” user and open the configuration file **“/etc/sudoers**”, here we have to specify the **user, machine, become** and **the location of the command**. The keyword “**become**” specifies the user can become “root” user to run commands with sudo power. The “**machine**” keyword represents from which system the sudo command can be used.



- Now login as “vimal” user and the user is able to create other users with **sudo** power, that is the user is able to run the command with the power of root



- Create, set password and give power for the user “amit” to run “useradd” command only

```
[root@localhost ~]# vim /etc/group
[root@localhost ~]#
[root@localhost ~]# useradd  amit
[root@localhost ~]# passwd amit
Changing password for user amit.
New password:
BAD PASSWORD: The password is a palindrome
Retype new password:
passwd: all authentication tokens updated successfully.
[root@localhost ~]# vim /etc/sudoers
```

```
amit        ALL=(ALL)   /usr/sbin/useradd
```

- Login as user “amit” and the command to check whether a user has sudo power is “**sudo –l**”.

```
[amit@localhost ~]$ sudo -l

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

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

[sudo] password for amit:
Matching Defaults entries for amit on localhost:
    !visiblepw, always_set_home, match_group_by_gid, always_query_group_plugin,
    env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR LS_COLORS",
    env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE",
    env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES",
    env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE",
    env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY",
    secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User amit may run the following commands on localhost:
    (ALL) /usr/sbin/useradd
```

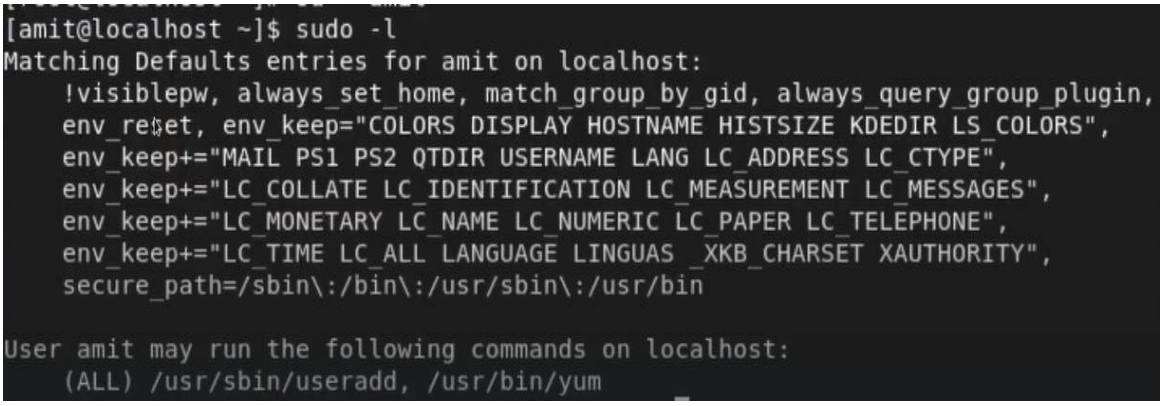
- The user “amit” with sudo power is **able to create users**, but **not able to install the software** using “yum” command, because the user has not given the sudo power to install the software.

```
[amit@localhost ~]$
[amit@localhost ~]$ sudo  useradd   aa12
[amit@localhost ~]$ sudo  useradd   aa123
```

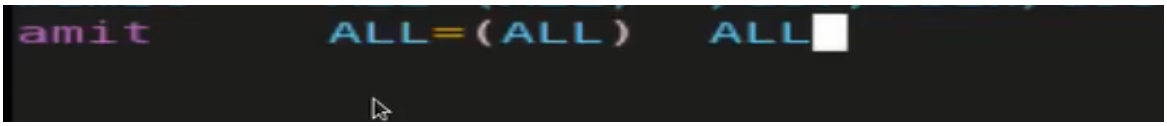
```
[amit@localhost ~]$ sudo  yum  install  httpd
Sorry, user amit is not allowed to execute '/bin/yum install httpd' as root on l
ocalhost.localdomain.
```

- For the user “amit” to get sudo power to install software using “yum” command, in the configuration file “/etc/sudoers” we have to specify the location of “yum” command

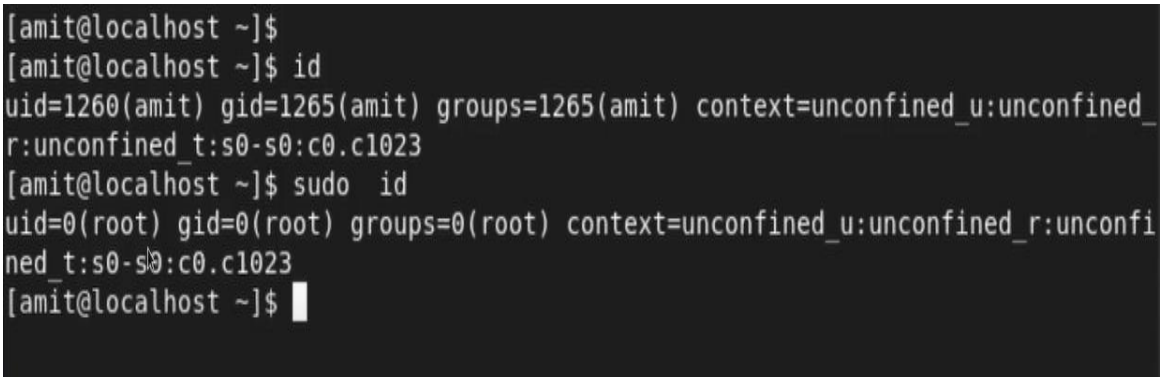
- Login as user “amit” and the command to check whether a user has sudo power is “**sudo –l**”, now here we see that the user has power to run “useradd” command and install software using “yum” command



- For the user “amit” to get sudo power to run all admin level commands, in the configuration file “/etc/sudoers” we have to specify “ALL” keyword



- Any command a user runs with sudo power, internally the command is run by the power of root. Here we see the user “amit” runs the “id” command with sudo power



- Create a group “lwgrp” and add users to this group, instead of giving sudo power to individual users, it can be given to group. To specify it’s a group name we have to use “%” symbol.

```
%lwgrp   ALL=(ALL)     ALL
```

- If we want any user to give all the power then we can add the user in **wheel group,** for this we have to create users in the wheel group

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

- The command to add user “tom12333” in “wheel” group is “**useradd –G wheel tom12333**”

```
[root@localhost ~]# useradd    -G wheel  tom12333
[root@localhost ~]# su - tom12333
[tom12333@localhost ~]$ groups
tom12333 wheel
```

- Any user of this group when they run commands with sudo, they do not need a password

```
%lwgrp   ALL=(ALL)     NOPASSWD: ALL
```

- Instead of using the main configuration **“/etc/sudoers**”, it a good practice to make changes in the secondary configuration file **“/etc/sudoers.d”**
- The command to directly open the main configuration file “/etc/sudoers” is **“visudo”**

```
[root@localhost ~]# visudo
```