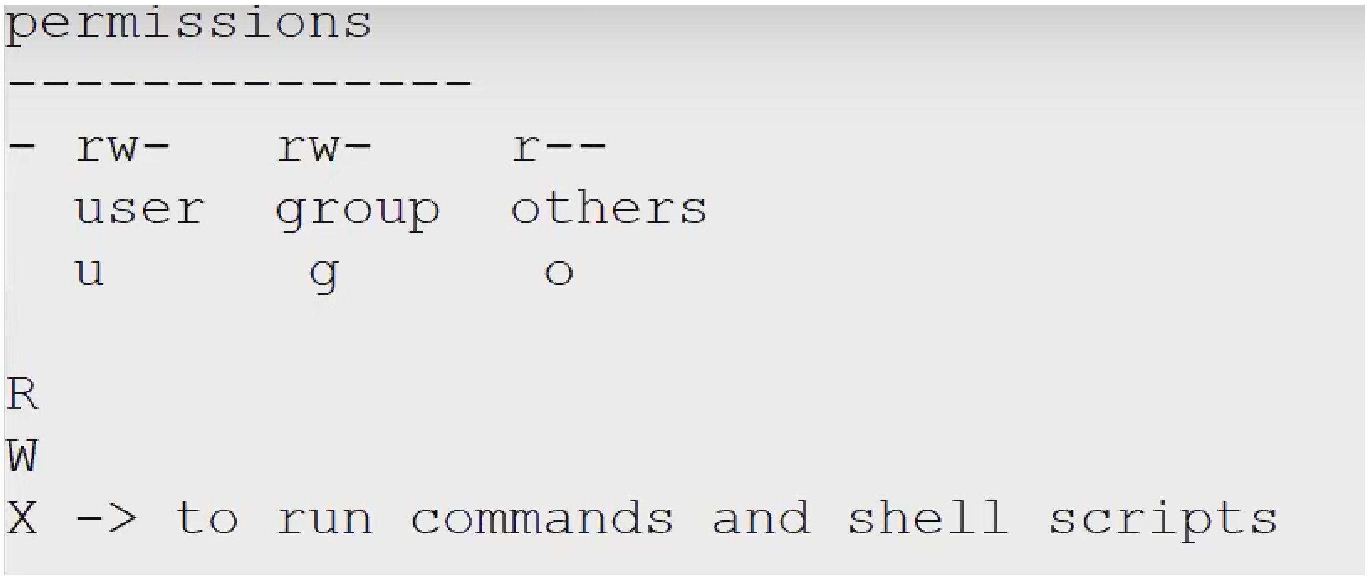
Session - 03

User management

when a user is created , a group with same name is created

permissions

```
drwxr-xr-x 1 pramo 197609        0 Mar 31  2022  Autodesk/
lrwxrwxrwx 1 pramo 197609       30 Oct 29  2022 'Application Data' ->
/c/Users/pramo/AppData/Roaming/
```



```
permissions
---------------
- rw-   rw-    r--
  user  group  others
  u      g      o

R
W
X -> to run commands and shell scripts
```

examples

1. `chmod g-rw sample` - Revokes read and write permissions for the group associated with the file "sample".

2. `chmod o-r sample` - Revokes read permissions for all other users (non-owner, non-group) for the file "sample".
3. `chmod ugo+r sample` - Grants read permissions to the owner, group, and all other users for the file "sample".
4. `chmod go+x sample` - Grants execute permissions to the group and all other users for the file "sample".

recursive access modification inside files

chmod g-w -R devops

This command recursively removes write permissions for the group from all files and directories within the "devops" directory:

```
chmod g-w -R devops
```

IQ

requirement

```
set permissions as follows
u-rwx
g-r
o-0
how to set it
since
r = 4
w = 2
x = 1

chmod ugo(values added) <file or dir>

chmod 740 devops
or
chmod u=rwx,g=r,o= devops
```

In the ugo format, the permissions "740" for the "devops" file would be represented as follows:

- **Owner**: has read, write, and execute permissions (7)
- **Group**: has read permissions (4)
- **Others**: has no permissions (0)

Therefore, in ugo format, "740" would be represented as:

```
chmod u=rwx,g=r,o= devops
```

User managemet

1. **Adding Users**: The `useradd` command is used to add new users to the system. For example:

   ```
   sudo useradd username
   ```

   This creates a new user with the specified username.

2. **Setting Passwords**: After adding a user, you need to set a password for them using the `passwd` command:

3.

```
sudo passwd username
```

This will prompt you to enter and confirm the new password for the user.

to check if user is created check file /etc/passwd file

Certainly! Here's the shortened version:

1. `id username` - Get information about a specific user.
2. `getent group` - List all groups on the system and their members.

to enable password authentication enabling ssh access for new users using password

---

navigate to `/etc/ssh/sshd_config` set value of PasswordAuthentication to yes

restart sshd by systemctl restart sshd

checking file config sshd -t == this checks for file config errors

4. **Modifying User Attributes**: You can modify user account attributes using the `usermod` command. For example, to change a user's home directory:

   ```
   sudo usermod -d /path/to/new/home username
   ```

5. **Deleting Users**: To delete a user account, you can use the `userdel` command:

   ```
   sudo userdel username
   ```

   This removes the user account but doesn't delete the user's home directory by default.

6. **Managing Groups**: Users can be members of one or more groups, which can be useful for granting permissions. The `groupadd`, `groupmod`, and `groupdel` commands are used to manage groups.
7. **Listing Users and Groups**: You can list all users on the system using the `cat /etc/passwd` command or `getent passwd`. Similarly, you can list groups with `cat /etc/group` or `getent group`.
8. **Switching Users**: You can switch to another user account using the `su` command. For example, to switch to the root user:

   ```
   su -
   ```

   You'll be prompted to enter the root password.
9. **Sudo Access**: Users can be granted sudo (superuser) privileges to perform administrative tasks. This is typically configured in the `/etc/sudoers` file.
10. **File Permissions**: Linux file permissions and ownership also play a role in user management. The `chmod`, `chown`, and `chgrp` commands are used to change file permissions, ownership, and group ownership respectively.
11. **Authentication and Authorization**: Linux uses various authentication mechanisms like PAM (Pluggable Authentication Modules) and authorization policies configured in files like `/etc/passwd`, `/etc/shadow`, `/etc/group`, and `/etc/sudoers`.

Remember to use these commands with caution, especially when dealing with administrative tasks. Improper user management can compromise system security.

To enable password authentication in the SSH daemon configuration file (`sshd_config`), you need to locate the file and edit it. The SSH daemon configuration file is usually located at `/etc/ssh/sshd_config`.

Once you locate the file, look for the line that says `#PasswordAuthentication yes`. If it's commented out with a `#` at the beginning, remove the `#` to uncomment it. If the line doesn't exist, add it to the file.

Here's how you can do it in one line using a command-line text editor like `sed`:

```
sudo sed -i 's/#PasswordAuthentication yes/PasswordAuthentication yes/'
/etc/ssh/sshd_config
```

This command searches for the line containing `#PasswordAuthentication yes` and replaces it with `PasswordAuthentication yes` in the `/etc/ssh/sshd_config` file.

After making the change, remember to restart the SSH service for the changes to take effect:

```
sudo service ssh restart
```

Now, password authentication should be enabled in SSH.

Group 20

creating a group

```
groupadd <group-name>
```

check created group by

```
getent group
```

this will list your group name

- every user will have a primary group and secondary group

```
usermod -g devops ramesh
```

why linux user has primary and secondary groups

In Linux, a user can belong to multiple groups. The primary group is the group that is associated with the user's account by default. Every user has one primary group, which is specified in the `/etc/passwd` file. This primary group is typically used as the default group ownership for files created by the user.

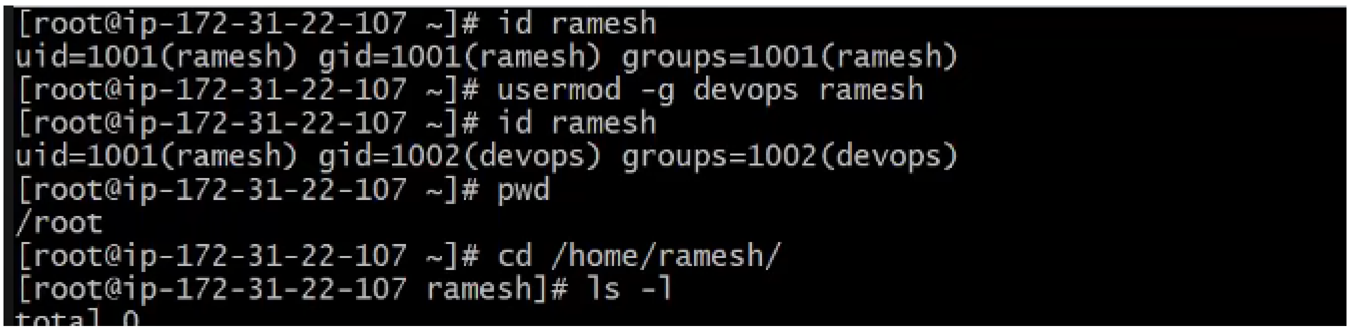
However, a user can also belong to additional groups, known as secondary groups. Secondary groups allow users to have access to resources and permissions associated with those groups in addition to their primary group. Secondary groups are specified in the `/etc/group` file.

Having both primary and secondary groups allows for more flexible access control and resource sharing in a Linux system. For example, a user can collaborate with members of different groups on projects by being a member of multiple groups simultaneously. Additionally, administrators can grant specific permissions to users based on their secondary group memberships.

Overall, primary and secondary groups provide a way to manage access to resources and facilitate collaboration among users in a Linux environment.

25:08

create a file named example in home directory for user Ramesh



```
[root@ip-172-31-22-107 ~]# id ramesh
uid=1001(ramesh) gid=1001(ramesh) groups=1001(ramesh)
[root@ip-172-31-22-107 ~]# usermod -g devops ramesh
[root@ip-172-31-22-107 ~]# id ramesh
uid=1001(ramesh) gid=1002(devops) groups=1002(devops)
[root@ip-172-31-22-107 ~]# pwd
/root
[root@ip-172-31-22-107 ~]# cd /home/ramesh/
[root@ip-172-31-22-107 ramesh]# ls -l
total 0
```

modify primary group of user to group devops

create a new user2

modify secondary group of the user2 to devops

user2 will be able to access the file created by user Ramesh because both have a group common

primary group change for user ramesh

```
usrmod -g devops ramesh
```

secondary group addition to a user

```
usermod -aG testing ramesh
```

above command adds group named testing to ramesh as secondary group

removing group

gpasswd -d ramesh testing where group name is testing

user is leaving org or complete removal of user

userdel ==> user is deleted from linux system

before doing this user should be removed from the group

1. remove user from group
2. remove user from system --> userdel
3. check user is removed --> id username

37:52 process management

---

if you follow good process you will get good results Everything in linux is a process

we can check using command `ps`

The `ps` command is used to display information about running processes on a Linux system. By default, it shows a snapshot of processes running under the current user's session.

Here's a basic usage of the `ps` command:

```
ps
```

This command displays a list of processes running under the current user's session, showing the process ID (PID), terminal associated with the process (TTY), the cumulative CPU time used (TIME), and the command associated with the process (COMMAND).

You can use various options with the `ps` command to customize the output. For example:

- To display information about all processes running on the system, you can use the `aux` options:

  ```
  ps aux
  ```

- To display a hierarchical view of processes, showing parent-child relationships, you can use the `f` option:

  ```
  ps af
  ```

- To display process information continuously, updating the output at regular intervals, you can use the `aux` options with the `--forest` option:

  ```
  ps aux --forest
  ```

These are just a few examples of how you can use the `ps` command to view information about processes on a Linux system.

`ps -ef` returns all process

how to check if jenkins is running

```
ps -ef | grep jenkins
```

This returns if jenkins is running or not

```
sleep 10 ==> blocks terminal for 10 sec
```

50:10 foreground vs background process

Foreground process blocks terminal

Background process will continue in terminal background

to make a foreground process background use `&` at the end of the command

```
for example `sleep 20 &` makes the process background
```

process will hung up from time to time killing the process

```
# get process id
kill PID
or
pkill process_name

# if kill doesn't work (when kill command is stuck)
kill -9
or
pkill -9 process_name
```

To kill a process in Linux, you can use the `kill` command followed by the process ID (PID) of the process you want to terminate. Here's the basic syntax:

```
kill PID
```

Replace `PID` with the actual process ID of the process you want to kill.

Alternatively, you can use the `pkill` command to terminate a process by its name. Here's how you can do it:

```
pkill process_name
```

Replace `process_name` with the name of the process you want to terminate.

By default, both `kill` and `pkill` send a SIGTERM signal to the process, asking it to gracefully terminate. If the process doesn't respond to the SIGTERM signal, you can forcefully terminate it by sending a SIGKILL signal using the `-9` option with `kill`:

```
kill -9 PID
```

or with `pkill`:

```
pkill -9 process_name
```

Be cautious when using the SIGKILL signal (`-9`) as it forcefully terminates the process without giving it a chance to clean up or release resources properly. It should generally be used as a last resort.

54:39 package management

Package management in Linux involves the installation, removal, and maintenance of software packages on a Linux system. Different Linux distributions use different package management systems, but the two most common ones are:

1. **Debian-based systems (e.g., Ubuntu, Debian)**:
   - Package Manager: APT (Advanced Package Tool)
   - Command-Line Tools: `apt`, `apt-get`, `apt-cache`
   - Package Repository: Packages are stored in repositories, and you can use `apt` to install, remove, and update packages. For example:

     ```
     sudo apt update       # Update package lists
     sudo apt install <package_name>   # Install a package
     sudo apt remove <package_name>    # Remove a package
     sudo apt upgrade      # Upgrade installed packages
     ```

2. **Red Hat-based systems (e.g., CentOS, Fedora, RHEL)**:
   - Package Manager: RPM (Red Hat Package Manager)
   - Command-Line Tools: `rpm`, `yum`, `dnf`
   - Package Repository: Similar to APT, packages are stored in repositories, and you can use `yum` or `dnf` to manage packages. For example:

     ```
     sudo yum install <package_name>   # Install a package
     sudo yum remove <package_name>    # Remove a package
     sudo yum update       # Update installed packages
     ```

In addition to these command-line tools, most Linux distributions also provide graphical package managers (such as Synaptic, GNOME Software, or KDE Discover) for users who prefer a graphical interface.

Package management systems help ensure that software installation is streamlined, dependencies are resolved automatically, and updates are easy to manage. They also provide a centralized repository for software distribution, which enhances security and stability by enabling timely updates and patches.

Centos editing yum repo

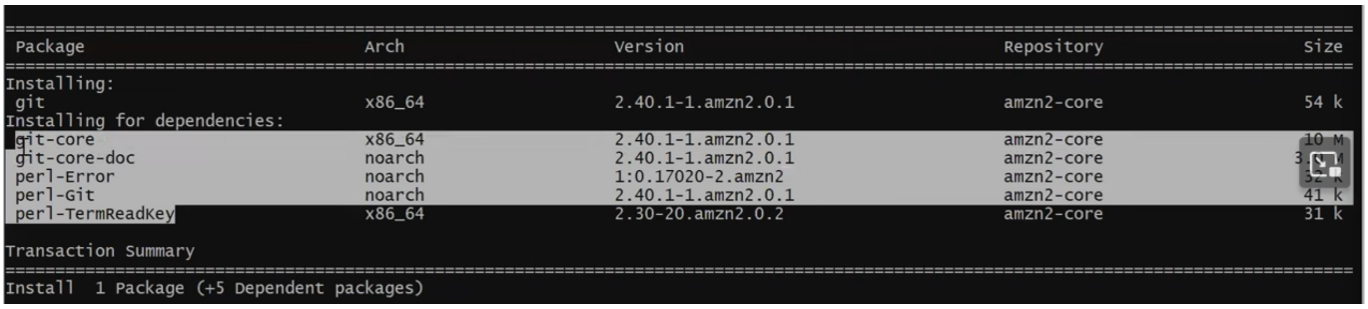
navigate to `cd /etc/yum.repos.d/`
get list by `ls -l`
cat the repo by using `cat amaz2-core.repo`
this will list repo urls from where packages are fetched

ex try `yum install git`

this will fetch the git package from repo

yum => package manager in linux RHEL, centos, fedora, aws linux 2
amazon linux will have `amazon-linux-extras` to install few extra packages from amazon-linux-extras repo

for example when you hit yum install git it will fetch dependencies from repository



```
================================================================================================================
 Package                  Arch              Version                        Repository                      Size
================================================================================================================
Installing:
 git                      x86_64            2.40.1-1.amzn2.0.1             amzn2-core                      54 k
Installing for dependencies:
 git-core                 x86_64            2.40.1-1.amzn2.0.1             amzn2-core                      10 M
 git-core-doc             noarch            2.40.1-1.amzn2.0.1             amzn2-core                     3.0 M
 perl-Error               noarch            1:0.17020-2.amzn2              amzn2-core                      32 k
 perl-Git                 noarch            2.40.1-1.amzn2.0.1             amzn2-core                      41 k
 perl-TermReadKey         x86_64            2.30-20.amzn2.0.2              amzn2-core                      31 k

Transaction Summary
================================================================================================================
Install  1 Package (+5 Dependent packages)
```

package management commands

`yum list installed` to get list of packages
`yum list installed | grep git`we can grep the list by
`yum list available` lists packages that can be installed from repos
`yum list available | wc -l` lists word count
`yum list available | grep mysql` lists the mysql packages from repos
`yum search mysql` searches for mysql packages in repos
`yum remove git -y` removes package git from repo `yum install nginx -y` installs nginx this is available through amazon-linux-extras in amazon linux
`amazon-linux-extras install nginx1` to install nginx in amazon linux

start a package `systemctl start nginx` starts nginx `ps -ef | grep nginx

user management missed topic - enabling ssh access for new users using key

Raheem joined the project, how will you enable ssh access through private key

- create user
- get user's public key (through email) (assuming user already created a key by `ssh-keygen -f raheem` && `cat raheem.pub`)
- create dir `.ssh` and set permissions as 700
-

`useradd raheem` create user
get user public key
navigate to created user's home directory for example `cd /home/raheem/`
create a dir named .ssh in home dir `cd /home/raheem/` by `mkdir .ssh`
`chown 700 .ssh` because only user must have full access and no one except raheem is allowed to access
`ls -la` check permissions
create file in `.ssh/authorized_keys` in raheem/new User's name and paste public key of raheem's/new user to this file
modify permission of `authorized_keys` to 400 or 600 because it doesn't need execute permission by `chmod 400 athorized_keys`
`ls -la` check permissions
change ownwership of ssh by `chown -R raheem:raheem .ssh` because raheem needs to be owner of his .ssh dir
user raheem connects using `ssh -i privatekey username@hostIP`

chmod vs chown

`chmod` and `chown` are two different commands used in Linux for managing file permissions and ownership, respectively.

1. **chmod**:
   - Stands for "change mode."
   - Used to change the permissions (mode) of a file or directory.
   - Allows you to grant or revoke read, write, and execute permissions for the owner, group, and others.
   - Examples:
     - `chmod u+x file`: Adds execute permission for the owner of the file.
     - `chmod go-w file`: Removes write permission for the group and others.
     - `chmod 755 file`: Sets read, write, and execute permissions for the owner, and read and execute permissions for the group and others.
2. **chown**:
   - Stands for "change owner."
   - Used to change the ownership of a file or directory.
   - Allows you to change the user and/or group ownership of a file or directory.
   - Examples:
     - `chown user:group file`: Changes both the user and group ownership of the file.
     - `chown user file`: Changes only the user ownership of the file.
     - `chown :group file`: Changes only the group ownership of the file.

In summary, `chmod` is used to change file permissions, while `chown` is used to change file ownership. They are complementary commands used for different purposes in managing file access and control in Linux.

meaning by changing ownership vs changing permissions in linux

Changing ownership and changing permissions are both ways to control access to files and directories in Linux, but they serve different purposes:

1. **Changing Ownership**:
   - **Definition**: Changing ownership means modifying the user and/or group associated with a file or directory.
   - **Command**: This is done using the `chown` command.
   - **Purpose**: Changing ownership determines which user and group have control over a file or directory. The owner of a file or directory typically has full control over it, including the ability to read from, write to, and execute it. Changing ownership allows you to delegate ownership to specific users or groups, which can be useful for collaboration or delegation of responsibilities.
   - **Example**: `chown user:group file` changes both the user and group ownership of the file to the specified user and group.
2. **Changing Permissions**:

- **Definition**: Changing permissions means modifying the access rights granted to the owner, group, and others for a file or directory.
- **Command**: This is done using the `chmod` command.
- **Purpose**: Changing permissions allows you to control who can read, write, or execute a file or directory. Permissions can be set separately for the owner, group, and others, and can be adjusted to restrict or grant access as needed. This is particularly important for security and access control, as it allows you to limit access to sensitive files and directories.
- **Example**: `chmod u+x file` adds execute permission for the owner of the file.

In summary, changing ownership determines who has control over a file or directory, while changing permissions determines what actions (read, write, execute) can be performed on the file or directory by different users and groups. Both ownership and permissions work together to manage access and security in a Linux system.

explanation of command chown -R rajesh:rajesh .ssh

The command `chown -R rajesh:rajesh .ssh` changes the ownership recursively (`-R` option) of the `.ssh` directory and all its contents to the user `rajesh` and the group `rajesh`.

Here's what each part of the command means:

- **`chown`**: Stands for "change owner." It's used to change the ownership of files and directories.
- **`-R`**: This option stands for "recursive" and is used to apply the ownership change to all files and directories within the specified directory, including subdirectories and their contents.
- **`rajesh:rajesh`**: Specifies both the user and group ownership. In this case, `rajesh` is both the user and group to which ownership will be changed.
- **`.ssh`**: This is the directory whose ownership is being changed. The dot (`.`) denotes the current directory, and `.ssh` is the name of the directory within the current directory.

So, `chown -R rajesh:rajesh .ssh` means recursively change the ownership of the `.ssh` directory and all its contents to the user `rajesh` and the group `rajesh`. This command is often used to ensure that the `.ssh` directory and its contents are owned by the user who will be using SSH on the system, which helps maintain proper permissions and security.

1:11:12 questions

End of Session - 03

`apt-get` used in ubuntu multiple keys needs to be included in `.ssh/authorized_keys` when multiple users login needs to be setup
Linux process killing signals In Linux, signals are software interrupts that are used to communicate between processes or between the kernel and processes. Signals can be generated by a variety of events, such as user actions, hardware exceptions, or system events.

Here are some common signals in Linux:

1. **SIGINT (Signal Interrupt)**:

   - Generated by pressing `Ctrl+C` in the terminal.
   - Typically used to request that a process terminate gracefully.

2. **SIGKILL (Signal Kill)**:
   - Cannot be caught or ignored by the process.
   - Immediately terminates the process without giving it a chance to clean up resources.
   - Often used as a last resort to forcefully terminate a process that is unresponsive.
3. **SIGTERM (Signal Terminate)**:
   - Allows a process to perform cleanup operations before termination.
   - Sent by the `kill` command without specifying a signal number.
4. **SIGHUP (Signal Hangup)**:
   - Originally intended to notify processes of a terminal disconnect.
   - Often used to instruct daemons or background processes to reload configuration files.
5. **SIGUSR1 and SIGUSR2 (User-defined Signals)**:
   - Can be used by applications to implement custom signal handling.
6. **SIGSTOP and SIGCONT (Stop and Continue Signals)**:
   - SIGSTOP is used to suspend a process temporarily.
   - SIGCONT is used to resume a suspended process.

These are just a few examples of the signals available in Linux. Each signal has a specific purpose and behavior, and understanding them is important for managing processes effectively on a Linux system.

1:17:50
user mask for permission setting

It seems like you might be referring to the command `umask` rather than "umusk."

`umask` is a command and a concept in Unix-like operating systems, including Linux. It stands for "user file creation mask" and is used to control the default file permissions assigned to newly created files and directories.

When a file or directory is created, its permissions are typically set based on the default permissions specified by the umask value, which is a set of octal digits. The umask value is subtracted from the maximum permissions (typically 666 for files and 777 for directories) to determine the actual permissions.

For example:

- If the umask value is 022, the default permissions for new files will be 644 (666 - 022 = 644), and for directories, it will be 755 (777 - 022 = 755).
- If the umask value is 077, the default permissions for new files will be 600 (666 - 077 = 600), and for directories, it will be 700 (777 - 077 = 700).

You can view and set the umask value using the `umask` command:

- To view the current umask value:

```
umask
```

- To set the umask value:

  ```
  umask <octal_value>
  ```

For example:

```
umask 022
```

This sets the umask value to 022, meaning that newly created files will have default permissions of 644, and directories will have default permissions of 755.

la -l returned total 0 but has one directory why is that

When the `ls -l` command returns "total 0" but there is one directory present, it usually means that the directory is empty. The "total 0" line in the output of `ls -l` **represents the total size of all files (in blocks) listed in the directory**.

Directories themselves do not occupy space on disk, so even if there is a directory present, if it doesn't contain any files or subdirectories, the total size will be 0. The "total 0" line simply indicates that there are no files listed in the directory, hence the total size is 0.

If you want to verify that the directory is indeed empty, you can use the `ls` command without the `-l` option to list only the names of files and directories:

```
ls
```

This will show you the contents of the directory without additional details like file sizes and permissions. **If the directory is empty, it won't display anything.**

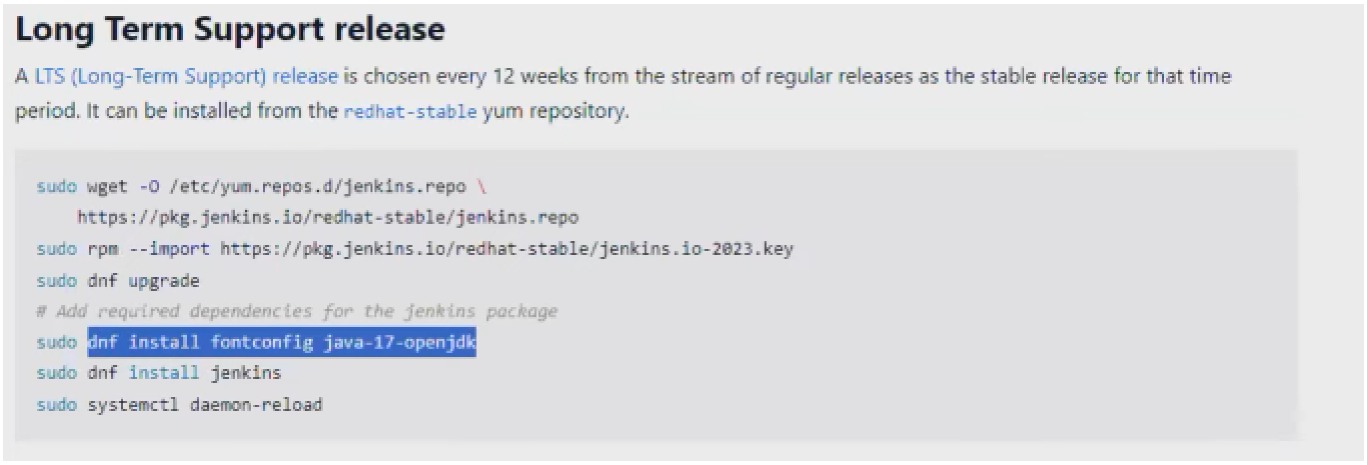
Others means the users not listed in group or the user
`chmod o+r <file-name>` can only be executed by the user or the root user

###aws session manager

1:23:22 user can add repos if needed

installing jenkins though jenkins repo

by default jenkins releases through its own repository



Long Term Support release

A LTS (Long-Term Support) release is chosen every 12 weeks from the stream of regular releases as the stable release for that time period. It can be installed from the `redhat-stable` yum repository.

```
sudo wget -O /etc/yum.repos.d/jenkins.repo \
    https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
sudo dnf upgrade
# Add required dependencies for the jenkins package
sudo dnf install fontconfig java-17-openjdk
sudo dnf install jenkins
sudo systemctl daemon-reload
```