=> In last session we discussed below commands

=> vi
=> sed

=> vi command we will use to edit file data (vi command will open file and we can

=> sed means steam editor. We can perform operations on data without opening file command

---

File Permissions In Linux

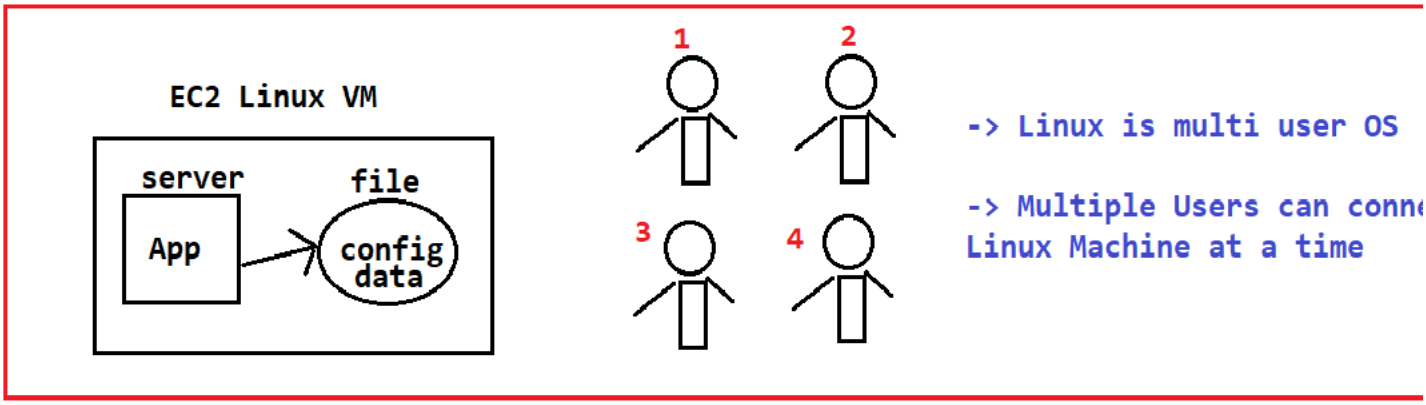
=> We will use files to store the data

=> In order to protect our data we need to secure our files

=> We can secure our files using File Permissions in linux

Usecase to understand importance of file permissions

=> We have a devops team with 4 members. 2 Members are experience and 2 members



-> Our application running inside a server which is installed in Linux VM

-> Our application reading config-data from a file
(db config, mail config, security config)

(db config, mail config, security config)

Note: We should not allow everybody to access config file because it is havi
sensitive data

As we have 4 members in devops team, we should allow only experienced members to modify the file and freshers can see the file but they shouldn't modify the file

Note: If we allow freshers to modify the file they may do some mistake in config then total project will be effected so we shouldn't take the risk hence i want to only read access for frehsers and read & write access for experienced people.

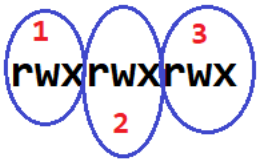
Working with File Permissions in Linux OS

=> We can execute 'ls -l' command to see file permissions information

```
ubuntu@ip-172-31-36-45:~$ ls -l
total 12
drwxrwxr-x 2 ubuntu ubuntu 4096 Mar  1 15:05 dir1
-rw-rw-r-- 1 ubuntu ubuntu    0 Feb 28 15:24 f3.txt
-rw-rw-r-- 1 ubuntu ubuntu  191 Feb 28 15:07 file1.txt
-rw-rw-r-- 1 ubuntu ubuntu  101 Feb 28 15:01 file2.txt
ubuntu@ip-172-31-36-45:~$
```

r ===> read
w ===> write
x ===> Execute
- ===> No permission

=> File Permission contains 3 segments



=> 1 section represents user permissions o
=> 2 section represents group permissions
=> 3 section represents others permission o

```
-rw-rw-r--  f1.txt
```

-> First section contains 'rw-'
that means user having only read & write permission (no execute permi

-> Second section contains 'rw-'

that means group having only read & write permission (no execute perr

-> Third section contains 'r--'

that means others having only read permission (no write & execute pe

-r-xrw---x f2.txt

-> user having read & execute (no write)
-> group having read & write (no execute)
-> other having only execute (no read & write)

-r--rw--w- f3.txt

-> user having only read permis
-> group having read & write pe
-> others having only write per

```
ubuntu@ip-172-31-36-45:~$ ls -l
drwxrwxr-x 2 ubuntu ubuntu 4096 Mar  1 15:05 dir1
-rw-rw-r-- 1 ubuntu ubuntu    0 Feb 28 15:24 f3.txt
-rw-rw-r-- 1 ubuntu ubuntu  191 Feb 28 15:07 file1.txt
-rw-rw-r-- 1 ubuntu ubuntu  101 Feb 28 15:01 file2.txt
```

=> Add write permission for others on f3.txt file

```
$ chmod  o+w f3.txt
```

o : means others
+ : means add
w : means write

=> Remove read permission for group on f3.txt file

```
$ chmod g-r f3.txt
```

g : means group
- : means remove
r : means read

=> Add execute permission for user on f3.txt file

```
$ chmod u+x f3.txt
```

u : means user

+ : means add

x : means execute

u-w : remove write permission for user

g+x : adding execute permission for group

o+r : adding read permission for others

u-x : remove execute permission for user

Also, you can use Symbolic Mode to modify permissions like the following:

Number	Permission
0	No permission
1	Execute
2	Write
3	Execute and Write
4	Read
5	Read and Execute
6	Read and Write
7	Read, Write and Execute

For example, let's give every per
for all with:

$ chmod 777 section.txt

Then the permissions will be: -rw

Let's remove the execute from the group and the write from other

$ chmod 765 section.txt

Then the permission will be : -rwxrw-r-x