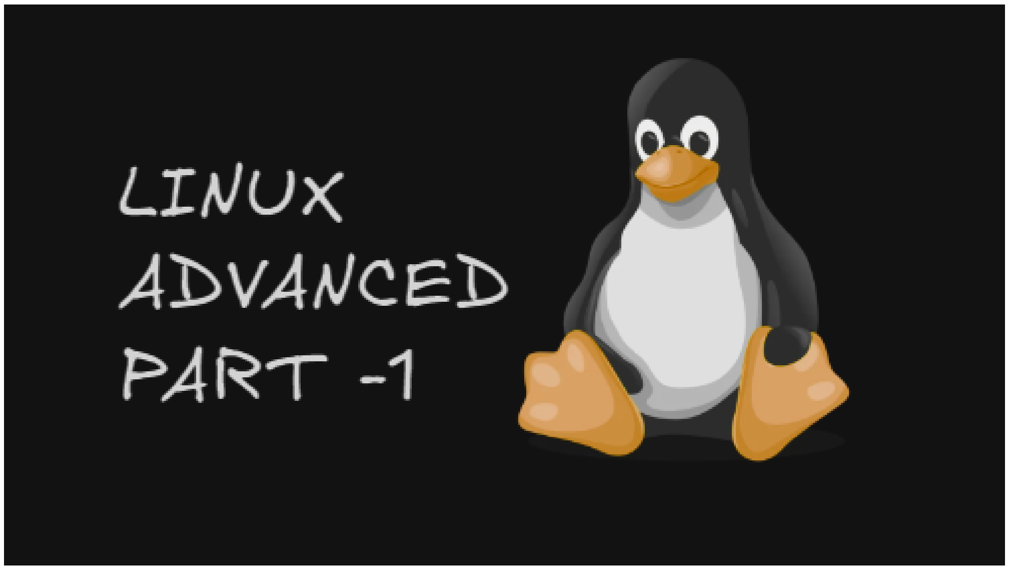
DAY 3 – LINUX ADVANCED



I] TOPICS COVERED

1. User and Groups
2. File Permissions
3. SSH, SCP, System Ctl
4. Grep, awk, Sed , Find
5. Volume Mounting

When we create any file in linux and list its detailed information we get

`-rw-r--r-- 1 username groupname 1234 Jan 01 12:34 file.txt`

1] The first character indicates the type of file:
`-` : Regular file
`d` : Directory
`l` : Symbolic link

2] The next nine characters represent the file permissions, divided into three sets:
`rw-` (read and write permissions for the owner)
`r--` (read-only permissions for the group)
`r--` (read-only permissions for other users)
`x` – (execute permission)
3] Number of Hard Links (`1`)
4] Owner `username`
5] Group `groupname`
6] File Size in Bytes
7] Last Modification Date and Time
8] Name of the File

1] Users and Groups

- In Linux, users and groups are essential for managing system security and resource allocation. Every user has a unique identity, and users can be grouped together to simplify permission management.

→ User Accounts

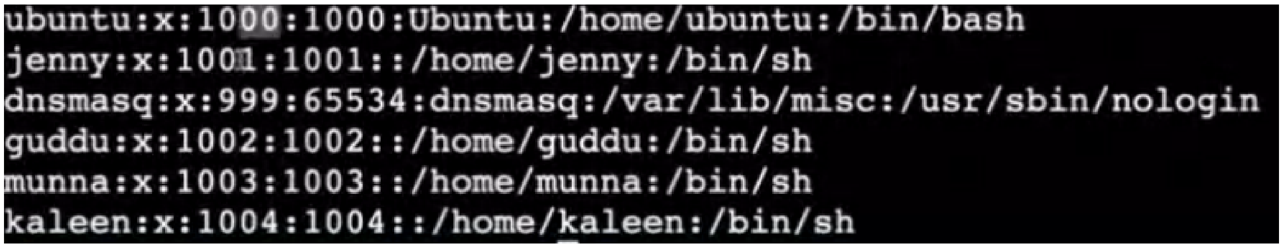
- Each user in Linux is identified by a unique username and user ID (UID). User accounts can be classified into three types:
    - **Root User**: The superuser with unrestricted access to the system. The root user can perform any task.
    - **System Users**: Used by system processes and services. These accounts typically do not have login privileges.
    - **Regular Users**: Created for human users, with varying levels of access and permissions.

→ User Management

- Managing users involves creating, modifying, and deleting user accounts. Let's look at the commands and files involved in these tasks.

1. Creating and Deleting Users → The `useradd` command is used to create new user accounts: `sudo useradd username`
2. `sudo passwd username` → The `passwd` command sets the password for the new user.
3. To delete a user, we use the `userdel` command: `sudo userdel username`
4. To Modify User accounts we use → `usermod` command
    1. The `usermod` command is used to modify existing user accounts. For example, to change a user's home directory: `sudo usermod -d /new/home/directory username`
    2. You can also use `usermod` to add a user to a new group: `sudo usermod -aG groupname username`
    3. To change a user's primary group: `sudo usermod -g newgroup username`

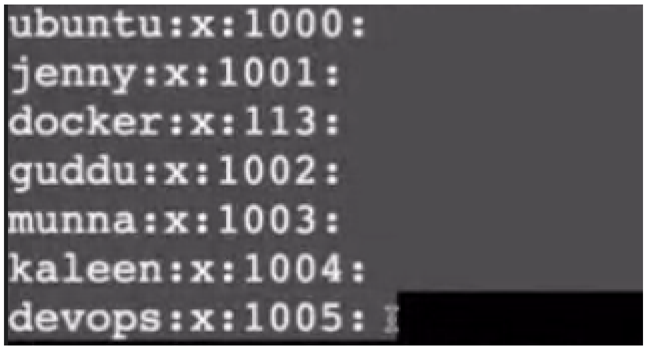
- **cat/etc/shadow**: Stores encrypted user passwords.
- When you finish adding users and now want to see the users which you have added we can use `cat /etc/passwd` command . This file is crucial as it contains information about all the user accounts on the system. Each line in this file represents a single user account and follows a specific format, with fields separated by colons (`:`).\
- By default when we open Ubuntu the default user uses **shell** → **bash**
- Whereas the other user that we create uses **shell** → **sh**
- Example →



```
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
jenny:x:1001:1001::/home/jenny:/bin/sh
dnsmasq:x:999:65534:dnsmasq:/var/lib/misc:/usr/sbin/nologin
guddu:x:1002:1002::/home/guddu:/bin/sh
munna:x:1003:1003::/home/munna:/bin/sh
kaleen:x:1004:1004::/home/kaleen:/bin/sh
```

→ Group Management

- Groups are collections of users. Each group is identified by a unique group name and group ID (GID). Groups help manage permissions for multiple users efficiently. For example, you can assign file access permissions to a group rather than to each user individually.
- Whenever we create a user an user group is also created.

- The `groupadd` command creates new groups:
    - `sudo groupadd groupname`
- To delete a group, use the `groupdel` command:
    - `sudo groupdel groupname`
- When you run the `cat /etc/group` command in a Linux terminal, it displays the contents of the `/etc/group` file. This file contains information about the groups on the system. Each line in this file represents a single group and follows a specific format, with fields separated by colons (`:`).


```
ubuntu:x:1000:
jenny:x:1001:
docker:x:113:
guddu:x:1002:
munna:x:1003:
kaleen:x:1004:
devops:x:1005:
```

- Here's an example entry from the `/etc/group`
- Example → `developers:x:1002:alice,bob,charlie`
- **Group Name**: `developers`
- **Password Placeholder**: `x`
- **Group ID (GID)**: `1002`
- **Group Members**: `alice`, `bob`, `charlie`
  In this example, `alice`, `bob`, and `charlie` are members of the `developers` group.
- Adding a User to the Group
    - To add a user to a group, you can use the `gpasswd -a` command followed by the username and the group name.
    - `sudo usermod -aG groupname username`
    - `sudo gpasswd -a username groupname` → This adds a single user to group
    - To add multiple user to a single group we can use: `sudo gpasswd -M user1,user2,user3 groupname`
- Removing the User from the Group
    - `sudo gpasswd -d username groupname`
- To Change the group of file we use:
    - `sudo chgrp groupname filename.txt`

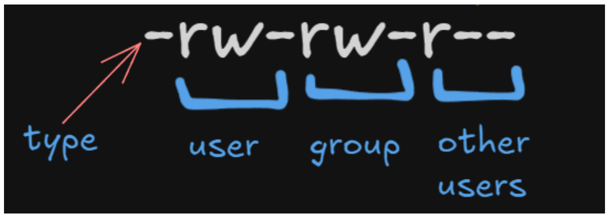
2] File Permissions

1.read(r)

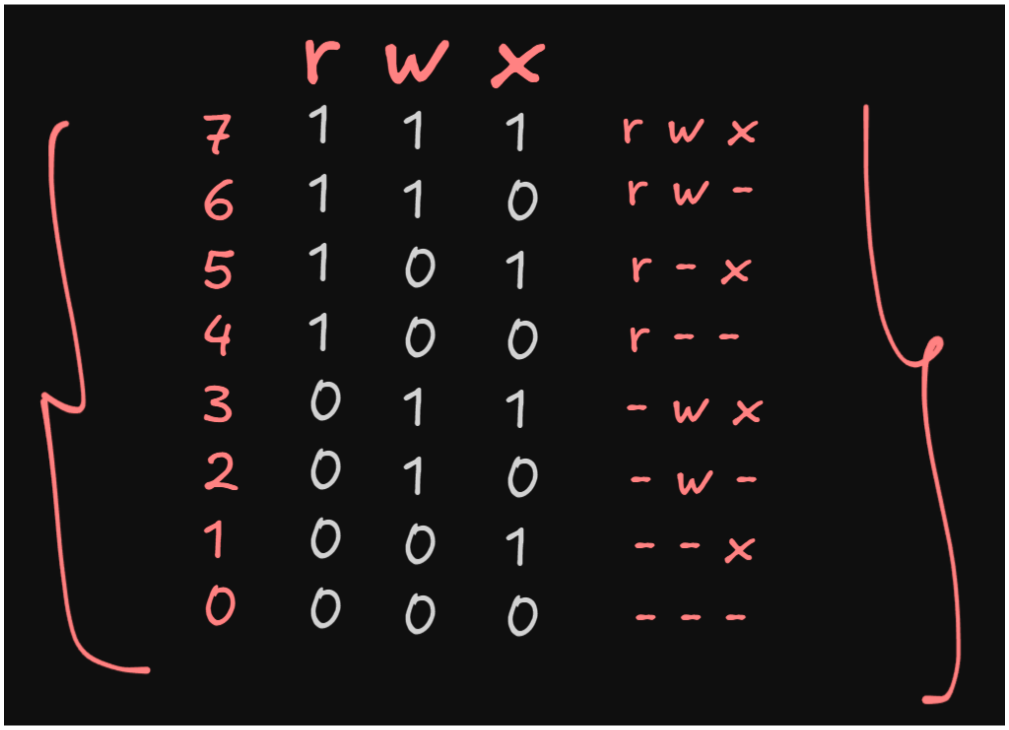
2.write(w)

3.execute(x)

- Now the default permission when we create a file or directory is `-rw-rw-r--`

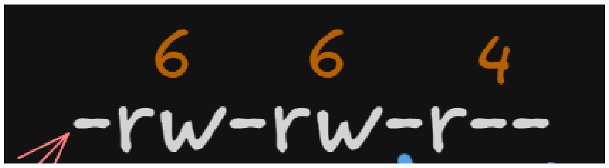


- List of all possible Permissions.



	r	w	x	
7	1	1	1	r w x
6	1	1	0	r w -
5	1	0	1	r - x
4	1	0	0	r - -
3	0	1	1	- w x
2	0	1	0	- w -
1	0	0	1	- - x
0	0	0	0	- - -

- Now giving numbers to the default permission we get

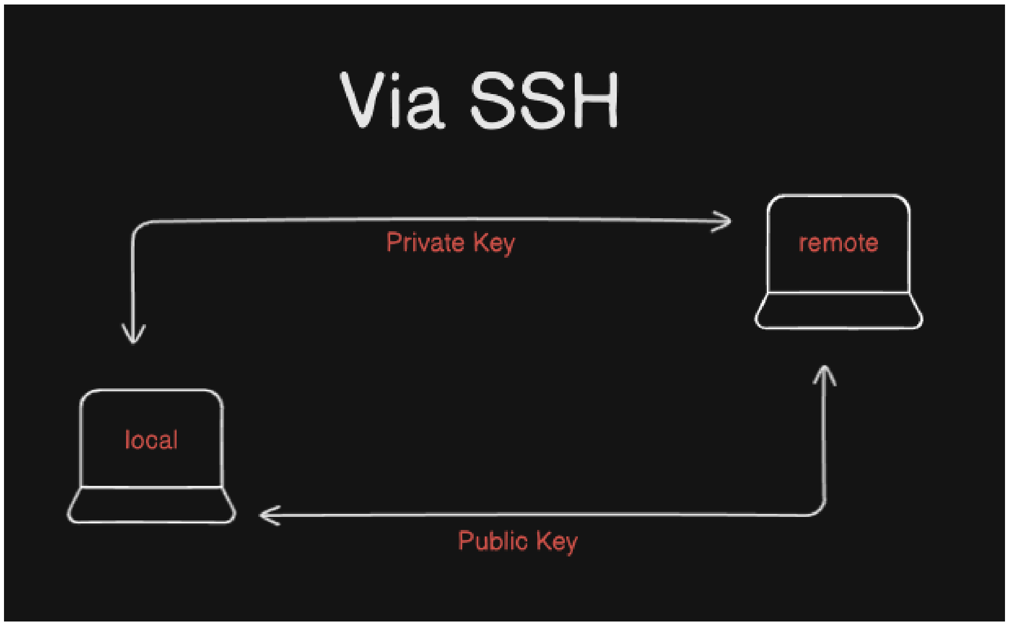


- Now if we want all users to have read, write and execute permissions (rwx) then we can use :
  - `chmod 777 filename.txt`
  - here chmod command is used to change access and permissions.
  - 777 indicates code to grant 7(r,w,x) to all users.
- The table given above can also be linked with the binary numbers like

- Binary of 7 is 111 → this means (rwx)
- Binary of 6 is 110 → this means (rw-)
- Binary of 2 is 010 → this means (-w-)

3] SSH, SCP, System Ctl

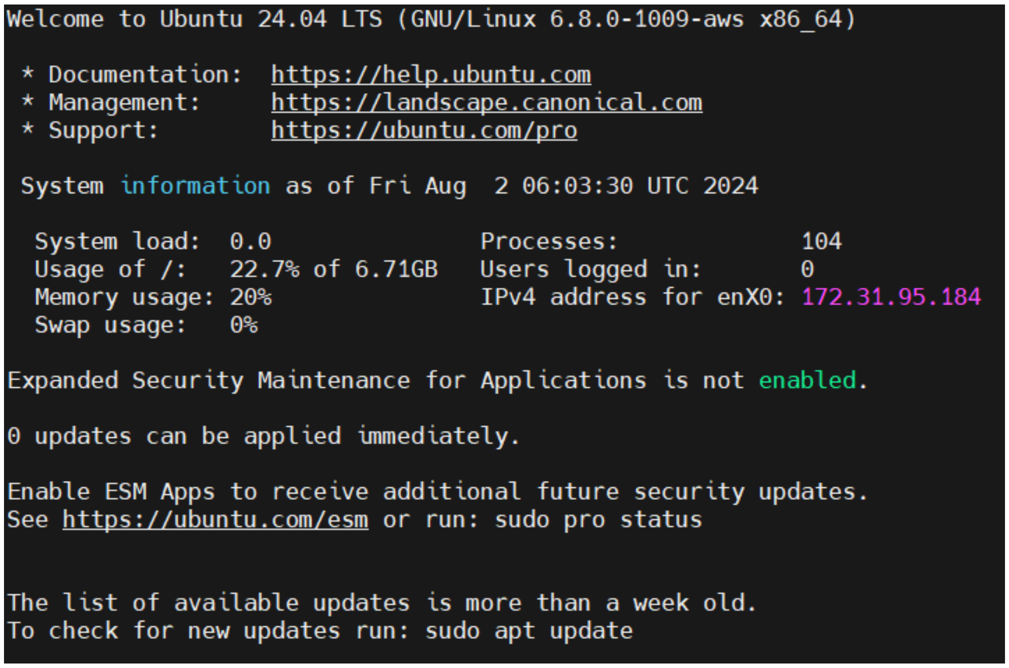
- ### SSH →
  - It stands for Secure Shell.
  - It is a protocol that provides a way to access a remote computer over an insecure network.
  - It also encrypts the connection, ensuring that data sent over the network is protected.
  - Its basic use is to connect to a Remote server.



  - For a SSH connection to establish we need keys that are Private and Public Key.
  - For a local machine (A) to connect to remote machine (B) , machine (A) should have Private Key and machine(B) Should have Public key.
  - When both these keys match (authentication) connection is established.
  - example → ram → hanuman(with pvt key) → sita (public key)
- ### Connecting To a Remote User via SSH from Local Machine (Using MobaXterm)
  1. Create a AWS EC2 instance

2. After the instance starts running, click on instance and click connect.
3. Go to SSH client.
4. Open an SSH client.
5. Locate your private key file. The key used to launch this instance is devops-prac.pem
6. Run this command to change the file permission such that only user can read → `chmod 400 "devops-prac.pem"`
7. Connect to your instance using its Public DNS:→ `ec2-44-204-50-8.compute-1.amazonaws.com`
8. Final Command which you will enter in the SSH client is :

- → `ssh -i "devops-prac.pem" ubuntu@ec2-44-204-50-8.compute-1.amazonaws.com`
- where `-i` stands for path and ubuntu is user.
- Finally Connection is Established!!

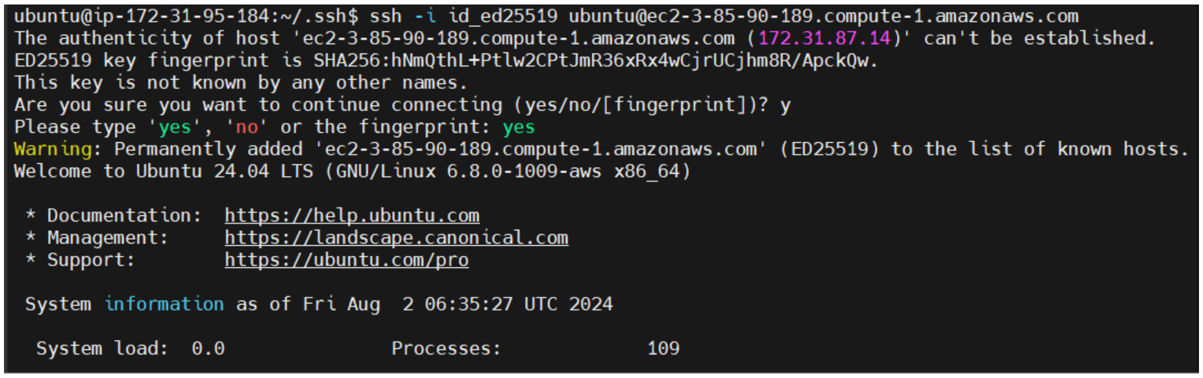


```
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1009-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Fri Aug  2 06:03:30 UTC 2024

  System load:  0.0               Processes:             104
  Usage of /:   22.7% of 6.71GB   Users logged in:       0
  Memory usage: 20%               IPv4 address for enX0: 172.31.95.184
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
```

- ## To connect via SSH from one machine to other.

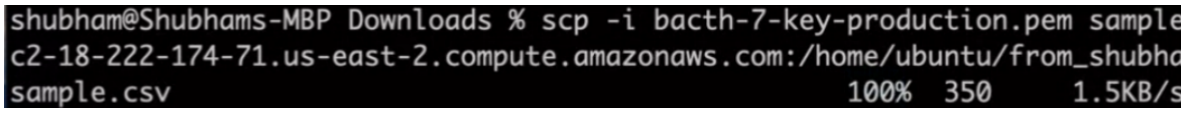
1. First create two AWS EC2 instances. example- devops1 and devops2.
2. We are trying to connect to devops2 from devops1, therefore we need private key for devops1 and public key for devops2.
3. Connect to devops1 in SSH client (mobaXterm).
4. Generating ssh keys with command in devops1 machine : `ssh-keygen` → this command generates ssh keys (both private and public)

5. ssh folder is hidden by default in ubuntu so use the command : `ls -a` to list all folders or we can directly use `cd .ssh` command to go to ssh folder.
6. After going inside ssh folder we see that the keys are generated (public and private key)
7. Display the public key with help of command `cat publickey`
8. Copy the public key.
9. Connect to devops2 machine with help of EC2 connect in AWS.
10. After connecting open hidden ssh folder → It contains authorized key file → open that with the help of `vim authorized key` and paste the public key that you copied earlier from devops1 machine.
11. Now the devops2 machine has public key.
12. Go to devops1 machine and type command : `ssh -i privatekeyofdevops1 ubuntu@dns of devops2 machine`
13. Finally Connection is established.



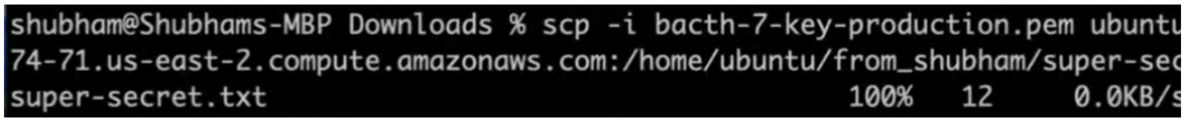
```
ubuntu@ip-172-31-95-184:~/.ssh$ ssh -i id_ed25519 ubuntu@ec2-3-85-90-189.compute-1.amazonaws.com
The authenticity of host 'ec2-3-85-90-189.compute-1.amazonaws.com (172.31.87.14)' can't be established.
ED25519 key fingerprint is SHA256:hNmQthL+Ptlw2CPtJmR36xRx4wCjrUCjhm8R/ApckQw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'ec2-3-85-90-189.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1009-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Fri Aug  2 06:35:27 UTC 2024

  System load:  0.0               Processes:             109
```

- ## SCP (SECURE COPY PROTOCOL)
  - SCP (Secure Copy Protocol) is used for transferring files between a local and a remote host or between two remote hosts. It uses SSH for data transfer, ensuring that the files are encrypted during the transfer.
  - If we want to transfer files from one machine to other machine via ssh we use SCP protocol
  - Example1 : If we are transferring a file named `test1.csv` from local machine to remote machine → `scp -i privatekey.pem test1.csv ubuntu@dns_of_remote_machine:/home/ubuntu/ashu`
    - `test.csv` is the local file (Source)
    - `:/home/ubuntu/ashu` → is the path where the file must be transferred on remote machine.(DESTINATION)



```
shubham@Shubhams-MBP Downloads % scp -i bacth-7-key-production.pem sample
c2-18-222-174-71.us-east-2.compute.amazonaws.com:/home/ubuntu/from_shubha
sample.csv                                    100%  350     1.5KB/s
```

- Example 2 : If we have a file named (super-special.txt) on a remote server and we want it to be transferred on local machine we will use : → `scp -i privatekey.pem ubuntu@dns of remote machine:/home/ubuntu/ashu/super-special.txt .`
    - where `:/home/ubuntu/ashu/super-special.txt` → is the path of file on remote machine. (SOURCE)
    - `.` → denotes that transfer the file to the current directory of local machine. (DESTINATION)



```
shubham@Shubhams-MBP Downloads % scp -i bacth-7-key-production.pem ubuntu
74-71.us-east-2.compute.amazonaws.com:/home/ubuntu/from_shubham/super-sec
super-secret.txt                              100%   12     0.0KB/s
```

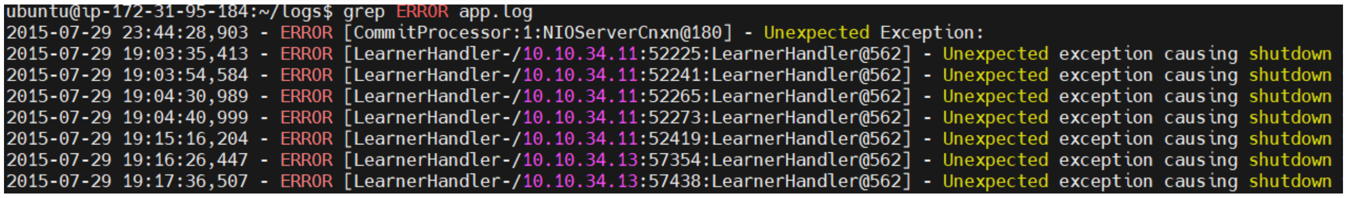
- **systemctl (System Control)**
    - `systemctl` is a command-line utility in Linux for controlling the systemd system and service manager.
    - It is basically used to manage the services on machine like to → start , stop, restart , status.
    - To start a service : `sudo systemctl start service_name`
    - Example → `sudo systemctl start docker`
    - To stop a service : `sudo systemctl stop service_name`
    - Example → `sudo systemctl stop docker`
    - To restart we use : `sudo systemctl restart service_name`
    - To enable the service to start on boot : `sudo systemctl enable service_name`
    - To disable the service to start on boot : `sudo systemctl disble service_name`
    - To check the status of the services : `systemctl status service_name`
    - ALTERNATIVELY WE CAN ALSO USE `Service` command for all
        - `service docker status`
        - `service docker start`
        - `service docker stop`
        - `service docker restart`.

- We can also use `journalctl` command to see the logs of information about system and Services.
- Example → `sudo journalctl -fu docker.service`

4] grep , awk , sed , find

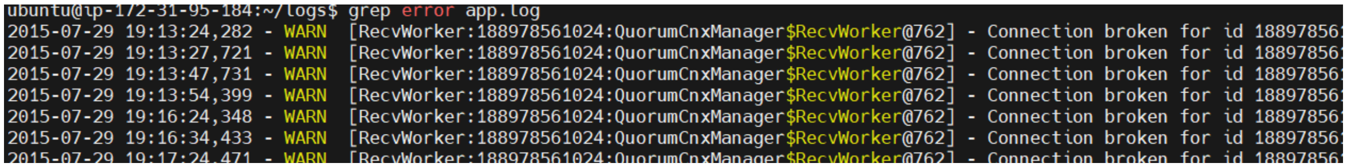
1}grep : →

- `grep` is used to search text or search the given file for lines containing a match to the specified strings or words. By default, `grep` prints the matching lines.
- It is used to find patterns and regular expressions.
- Search for a string in a file : `grep "search_string" filename`
- Example → `grep hello file.txt`



```
ubuntu@ip-172-31-95-184:~/logs$ grep ERROR app.log
2015-07-29 23:44:28,903 - ERROR [CommitProcessor:1:NIOServerCnxn@180] - Unexpected Exception:
2015-07-29 19:03:35,413 - ERROR [LearnerHandler-/10.10.34.11:52225:LearnerHandler@562] - Unexpected exception causing shutdown
2015-07-29 19:03:54,584 - ERROR [LearnerHandler-/10.10.34.11:52241:LearnerHandler@562] - Unexpected exception causing shutdown
2015-07-29 19:04:30,989 - ERROR [LearnerHandler-/10.10.34.11:52265:LearnerHandler@562] - Unexpected exception causing shutdown
2015-07-29 19:04:40,999 - ERROR [LearnerHandler-/10.10.34.11:52273:LearnerHandler@562] - Unexpected exception causing shutdown
2015-07-29 19:15:16,204 - ERROR [LearnerHandler-/10.10.34.11:52419:LearnerHandler@562] - Unexpected exception causing shutdown
2015-07-29 19:16:26,447 - ERROR [LearnerHandler-/10.10.34.13:57354:LearnerHandler@562] - Unexpected exception causing shutdown
2015-07-29 19:17:36,507 - ERROR [LearnerHandler-/10.10.34.13:57438:LearnerHandler@562] - Unexpected exception causing shutdown
```

- To search recursively in all files of directory: `grep -r "string" dir/`
- Example → `grep -r error /home/ubuntu/log/`
- It is case sensitive.



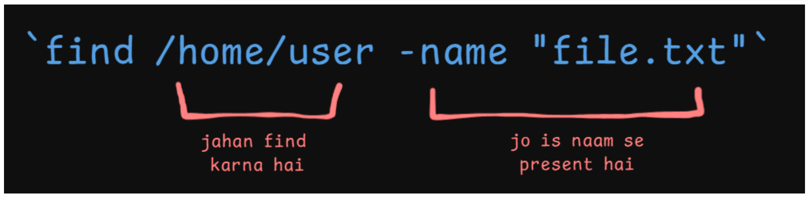
```
ubuntu@ip-172-31-95-184:~/logs$ grep error app.log
2015-07-29 19:13:24,282 - WARN  [RecvWorker:188978561024:QuorumCnxManager$RecvWorker@762] - Connection broken for id 18897856
2015-07-29 19:13:27,721 - WARN  [RecvWorker:188978561024:QuorumCnxManager$RecvWorker@762] - Connection broken for id 18897856
2015-07-29 19:13:47,731 - WARN  [RecvWorker:188978561024:QuorumCnxManager$RecvWorker@762] - Connection broken for id 18897856
2015-07-29 19:13:54,399 - WARN  [RecvWorker:188978561024:QuorumCnxManager$RecvWorker@762] - Connection broken for id 18897856
2015-07-29 19:16:24,348 - WARN  [RecvWorker:188978561024:QuorumCnxManager$RecvWorker@762] - Connection broken for id 18897856
2015-07-29 19:16:34,433 - WARN  [RecvWorker:188978561024:QuorumCnxManager$RecvWorker@762] - Connection broken for id 18897856
2015-07-29 19:17:24,471 - WARN  [RecvWorker:188978561024:QuorumCnxManager$RecvWorker@762] - Connection broken for id 18897856
```

  we get error as well as WARN.
- To make the search case insensitive we use the command: `grep -i error /home/ubuntu/log`

2} find

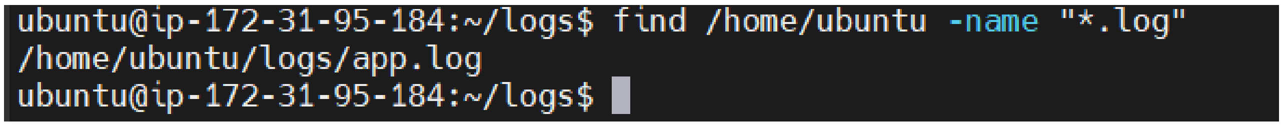
- `find` is used to search for files and directories in a directory hierarchy.
- Find files by name : `find /path/to/search -name "filename"`

- Example 1 → `find /home/user -name "file.txt"`



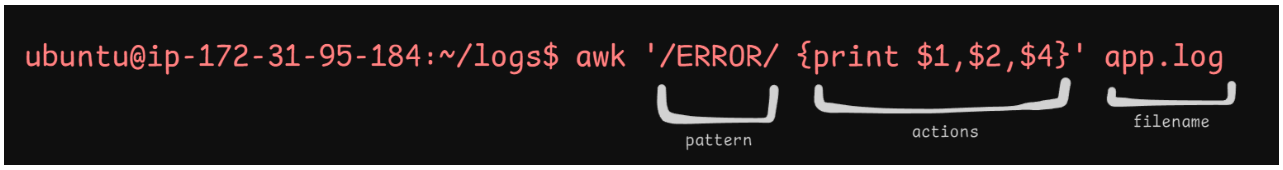
```
`find /home/user -name "file.txt"`
```

- Example 2 → it finds everything (used `*`) that ends with `.log`



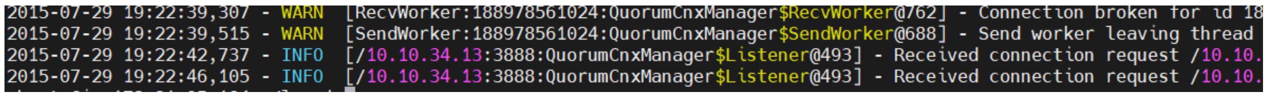
```
ubuntu@ip-172-31-95-184:~/logs$ find /home/ubuntu -name "*.log"
/home/ubuntu/logs/app.log
ubuntu@ip-172-31-95-184:~/logs$ 
```

3}awk

- `awk` is a powerful text-processing tool in Unix/Linux used for pattern scanning and processing. It can perform actions on lines that match a specified pattern.
- (awk vo hai jo aapke file me column wise iterate karta hai and jo aapko chiaye nikal ke deta hai)
- 

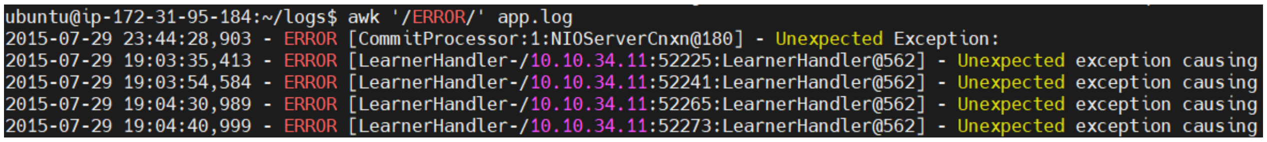
```
ubuntu@ip-172-31-95-184:~/logs$ awk '/ERROR/ {print $1,$2,$4}' app.log
```

- Syntax → `awk 'pattern { action }' input_file`
  - **pattern**: This specifies the condition that must be met for the action to be executed.
  - **action**: This is what you want to do when the pattern matches (e.g., print specific fields).
  - **input_file**: The file you are processing.
- If no pattern is specified, AWK will apply the action to all lines in the input file.
- Examples
  - Suppose we have this log file and we want to analyze this.(to tell anyone that which error occurred on which date/time/what was the error)



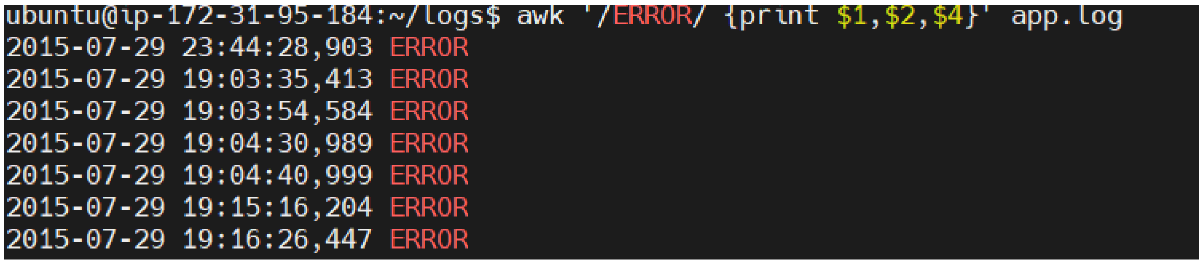
```
2015-07-29 19:22:39,307 - WARN  [RecvWorker:188978561024:QuorumCnxManager$RecvWorker@762] - Connection broken for id 18
2015-07-29 19:22:39,515 - WARN  [SendWorker:188978561024:QuorumCnxManager$SendWorker@688] - Send worker leaving thread
2015-07-29 19:22:42,737 - INFO  [/10.10.34.13:3888:QuorumCnxManager$Listener@493] - Received connection request /10.10.
2015-07-29 19:22:46,105 - INFO  [/10.10.34.13:3888:QuorumCnxManager$Listener@493] - Received connection request /10.10.
```

- We want to print all the logs with the words ERROR in them .



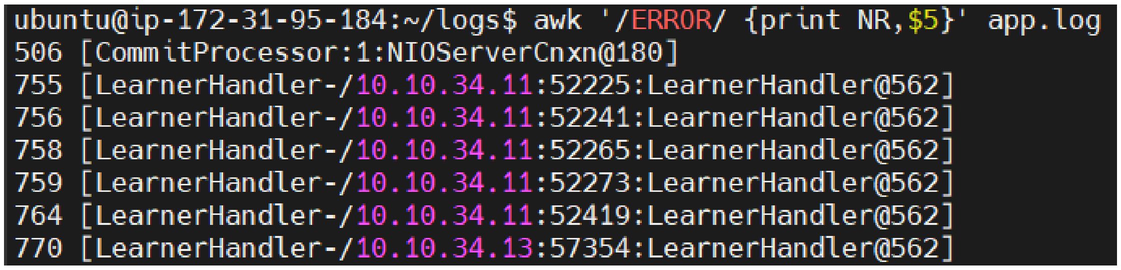
```
ubuntu@ip-172-31-95-184:~/logs$ awk '/ERROR/' app.log
2015-07-29 23:44:28,903 - ERROR [CommitProcessor:1:NIOServerCnxn@180] - Unexpected Exception:
2015-07-29 19:03:35,413 - ERROR [LearnerHandler-/10.10.34.11:52225:LearnerHandler@562] - Unexpected exception causing
2015-07-29 19:03:54,584 - ERROR [LearnerHandler-/10.10.34.11:52241:LearnerHandler@562] - Unexpected exception causing
2015-07-29 19:04:30,989 - ERROR [LearnerHandler-/10.10.34.11:52265:LearnerHandler@562] - Unexpected exception causing
2015-07-29 19:04:40,999 - ERROR [LearnerHandler-/10.10.34.11:52273:LearnerHandler@562] - Unexpected exception causing
```

- When we want to Print all the ERROR logged lines only for ($1 ,$2 ,$3 ..etc) where $1 denotes the column.



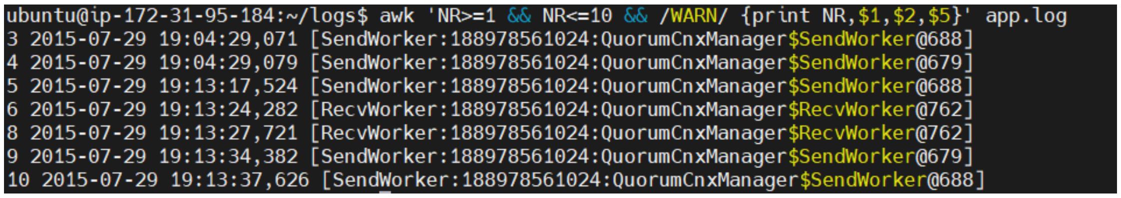
```
ubuntu@ip-172-31-95-184:~/logs$ awk '/ERROR/ {print $1,$2,$4}' app.log
2015-07-29 23:44:28,903 ERROR
2015-07-29 19:03:35,413 ERROR
2015-07-29 19:03:54,584 ERROR
2015-07-29 19:04:30,989 ERROR
2015-07-29 19:04:40,999 ERROR
2015-07-29 19:15:16,204 ERROR
2015-07-29 19:16:26,447 ERROR
```

- To print lines that contain the word "error": `awk '/error/ { print }' filename.txt`
- To print information of employees under the age of 40: `awk '$3 < 40 { print }' information.txt`
- We can also put conditions in awk like if we want only the logs before a specified date.
    - Like if we want to get the rows in which the error is present.



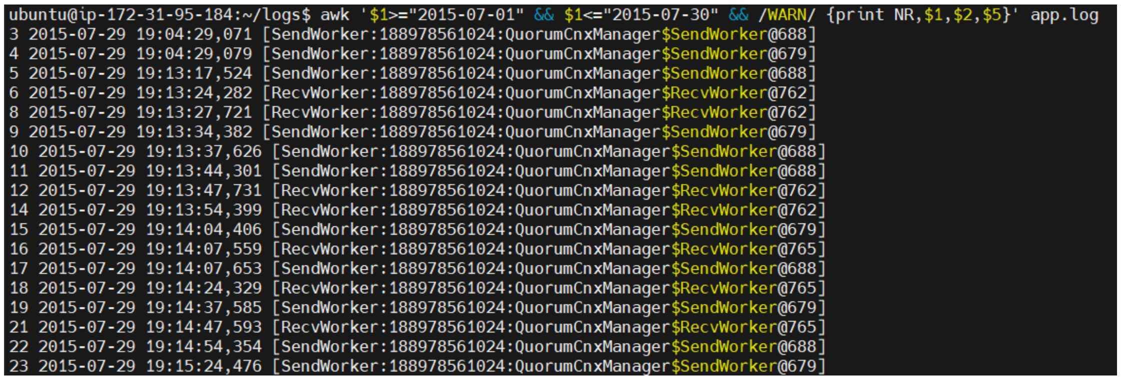
```
ubuntu@ip-172-31-95-184:~/logs$ awk '/ERROR/ {print NR,$5}' app.log
506 [CommitProcessor:1:NIOServerCnxn@180]
755 [LearnerHandler-/10.10.34.11:52225:LearnerHandler@562]
756 [LearnerHandler-/10.10.34.11:52241:LearnerHandler@562]
758 [LearnerHandler-/10.10.34.11:52265:LearnerHandler@562]
759 [LearnerHandler-/10.10.34.11:52273:LearnerHandler@562]
764 [LearnerHandler-/10.10.34.11:52419:LearnerHandler@562]
770 [LearnerHandler-/10.10.34.13:57354:LearnerHandler@562]
```

    - If we want to get the first 10 WARN logged lines with its message ,time and date.



```
ubuntu@ip-172-31-95-184:~/logs$ awk 'NR>=1 && NR<=10 && /WARN/ {print NR,$1,$2,$5}' app.log
3 2015-07-29 19:04:29,071 [SendWorker:188978561024:QuorumCnxManager$SendWorker@688]
4 2015-07-29 19:04:29,079 [SendWorker:188978561024:QuorumCnxManager$SendWorker@679]
5 2015-07-29 19:13:17,524 [SendWorker:188978561024:QuorumCnxManager$SendWorker@688]
6 2015-07-29 19:13:24,282 [RecvWorker:188978561024:QuorumCnxManager$RecvWorker@762]
8 2015-07-29 19:13:27,721 [RecvWorker:188978561024:QuorumCnxManager$RecvWorker@762]
9 2015-07-29 19:13:34,382 [SendWorker:188978561024:QuorumCnxManager$SendWorker@679]
10 2015-07-29 19:13:37,626 [SendWorker:188978561024:QuorumCnxManager$SendWorker@688]
```

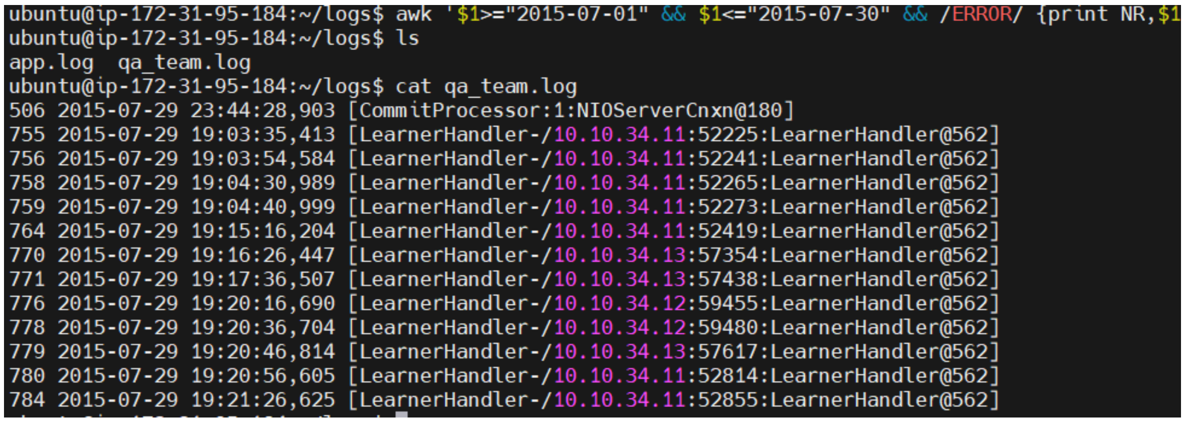
- If we want to get WARN logs between date 2015-07-01 → 2015-07-30.



```
ubuntu@ip-172-31-95-184:~/logs$ awk '$1>="2015-07-01" && $1<="2015-07-30" && /WARN/ {print NR,$1,$2,$5}' app.log
3 2015-07-29 19:04:29,071 [SendWorker:188978561024:QuorumCnxManager$SendWorker@688]
4 2015-07-29 19:04:29,079 [SendWorker:188978561024:QuorumCnxManager$SendWorker@679]
5 2015-07-29 19:13:17,524 [SendWorker:188978561024:QuorumCnxManager$SendWorker@688]
6 2015-07-29 19:13:24,282 [RecvWorker:188978561024:QuorumCnxManager$RecvWorker@762]
8 2015-07-29 19:13:27,721 [RecvWorker:188978561024:QuorumCnxManager$RecvWorker@762]
9 2015-07-29 19:13:34,382 [SendWorker:188978561024:QuorumCnxManager$SendWorker@679]
10 2015-07-29 19:13:37,626 [SendWorker:188978561024:QuorumCnxManager$SendWorker@688]
11 2015-07-29 19:13:44,301 [SendWorker:188978561024:QuorumCnxManager$SendWorker@688]
12 2015-07-29 19:13:47,731 [RecvWorker:188978561024:QuorumCnxManager$RecvWorker@762]
14 2015-07-29 19:13:54,399 [RecvWorker:188978561024:QuorumCnxManager$RecvWorker@762]
15 2015-07-29 19:14:04,406 [SendWorker:188978561024:QuorumCnxManager$SendWorker@679]
16 2015-07-29 19:14:07,559 [RecvWorker:188978561024:QuorumCnxManager$RecvWorker@765]
17 2015-07-29 19:14:07,653 [SendWorker:188978561024:QuorumCnxManager$SendWorker@688]
18 2015-07-29 19:14:24,329 [RecvWorker:188978561024:QuorumCnxManager$RecvWorker@765]
19 2015-07-29 19:14:37,585 [SendWorker:188978561024:QuorumCnxManager$SendWorker@679]
21 2015-07-29 19:14:47,593 [RecvWorker:188978561024:QuorumCnxManager$RecvWorker@765]
22 2015-07-29 19:14:54,354 [SendWorker:188978561024:QuorumCnxManager$SendWorker@688]
23 2015-07-29 19:15:24,476 [SendWorker:188978561024:QuorumCnxManager$SendWorker@679]
```

- If we want to list the ERROR logs for the same and save it inside a file.

  It lists all the ERROR logged inside the file qa_team.log



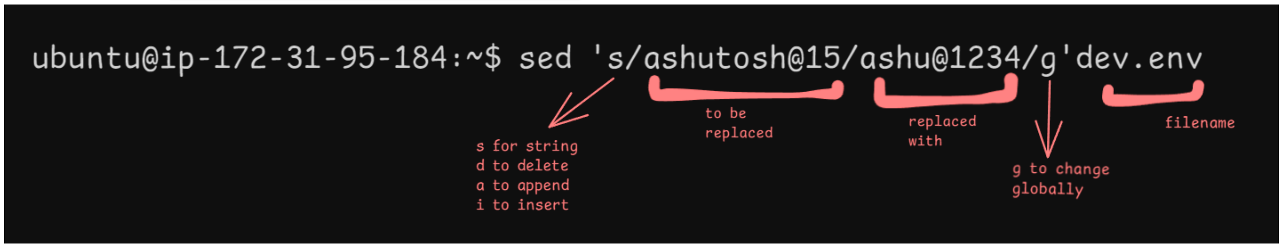
```
ubuntu@ip-172-31-95-184:~/logs$ awk '$1>="2015-07-01" && $1<="2015-07-30" && /ERROR/ {print NR,$1
ubuntu@ip-172-31-95-184:~/logs$ ls
app.log  qa_team.log
ubuntu@ip-172-31-95-184:~/logs$ cat qa_team.log
506 2015-07-29 23:44:28,903 [CommitProcessor:1:NIOServerCnxn@180]
755 2015-07-29 19:03:35,413 [LearnerHandler-/10.10.34.11:52225:LearnerHandler@562]
756 2015-07-29 19:03:54,584 [LearnerHandler-/10.10.34.11:52241:LearnerHandler@562]
758 2015-07-29 19:04:30,989 [LearnerHandler-/10.10.34.11:52265:LearnerHandler@562]
759 2015-07-29 19:04:40,999 [LearnerHandler-/10.10.34.11:52273:LearnerHandler@562]
764 2015-07-29 19:15:16,204 [LearnerHandler-/10.10.34.11:52419:LearnerHandler@562]
770 2015-07-29 19:16:26,447 [LearnerHandler-/10.10.34.13:57354:LearnerHandler@562]
771 2015-07-29 19:17:36,507 [LearnerHandler-/10.10.34.13:57438:LearnerHandler@562]
776 2015-07-29 19:20:16,690 [LearnerHandler-/10.10.34.12:59455:LearnerHandler@562]
778 2015-07-29 19:20:36,704 [LearnerHandler-/10.10.34.12:59480:LearnerHandler@562]
779 2015-07-29 19:20:46,814 [LearnerHandler-/10.10.34.13:57617:LearnerHandler@562]
780 2015-07-29 19:20:56,605 [LearnerHandler-/10.10.34.11:52814:LearnerHandler@562]
784 2015-07-29 19:21:26,625 [LearnerHandler-/10.10.34.11:52855:LearnerHandler@562]
```

- Difference b/w grep and awk → **Grep** is a simple tool for searching and displaying lines that match a pattern, while **awk** is a powerful text processing language that allows for pattern matching, data extraction, manipulation, and report generation.
- Also for a file to be analyzed by awk the file should be space separated such that it can be divided in columns and searched.

4}sed

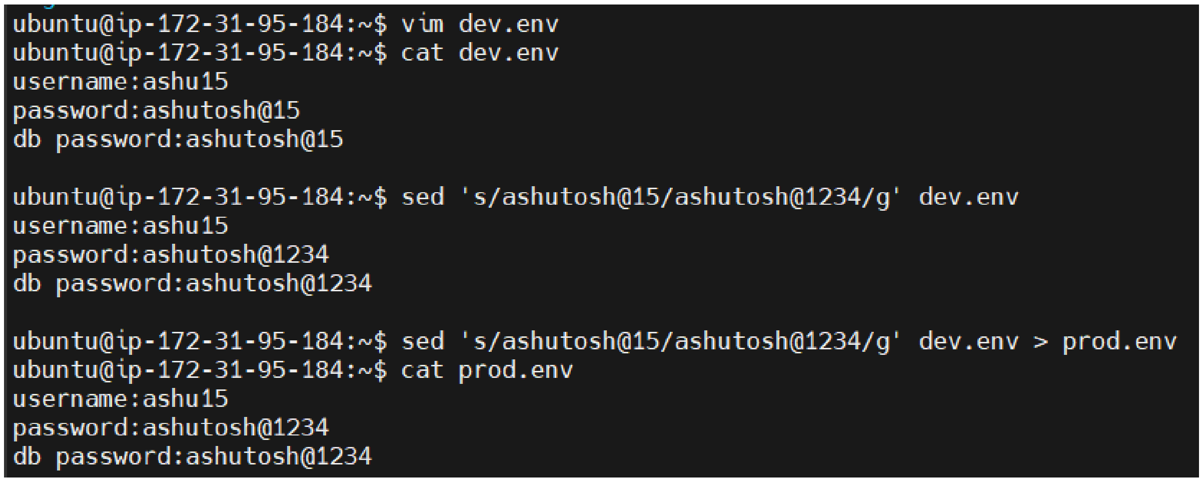
- The `sed` command, short for "stream editor," is a powerful Unix/Linux tool used for parsing and transforming text. It reads input line by line and applies a series of commands to each line, then outputs the result.

- Syntax is : `sed 'command' filename`



```
ubuntu@ip-172-31-95-184:~$ sed 's/ashutosh@15/ashu@1234/g'dev.env
```

- Examples
  1. In this example i made a file named dev.env and filled in the details . I had to do the same for Prod.env but i just had to change the passwords

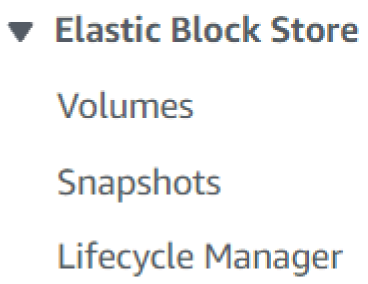


```
ubuntu@ip-172-31-95-184:~$ vim dev.env
ubuntu@ip-172-31-95-184:~$ cat dev.env
username:ashu15
password:ashutosh@15
db password:ashutosh@15

ubuntu@ip-172-31-95-184:~$ sed 's/ashutosh@15/ashutosh@1234/g' dev.env
username:ashu15
password:ashutosh@1234
db password:ashutosh@1234

ubuntu@ip-172-31-95-184:~$ sed 's/ashutosh@15/ashutosh@1234/g' dev.env > prod.env
ubuntu@ip-172-31-95-184:~$ cat prod.env
username:ashu15
password:ashutosh@1234
db password:ashutosh@1234
```

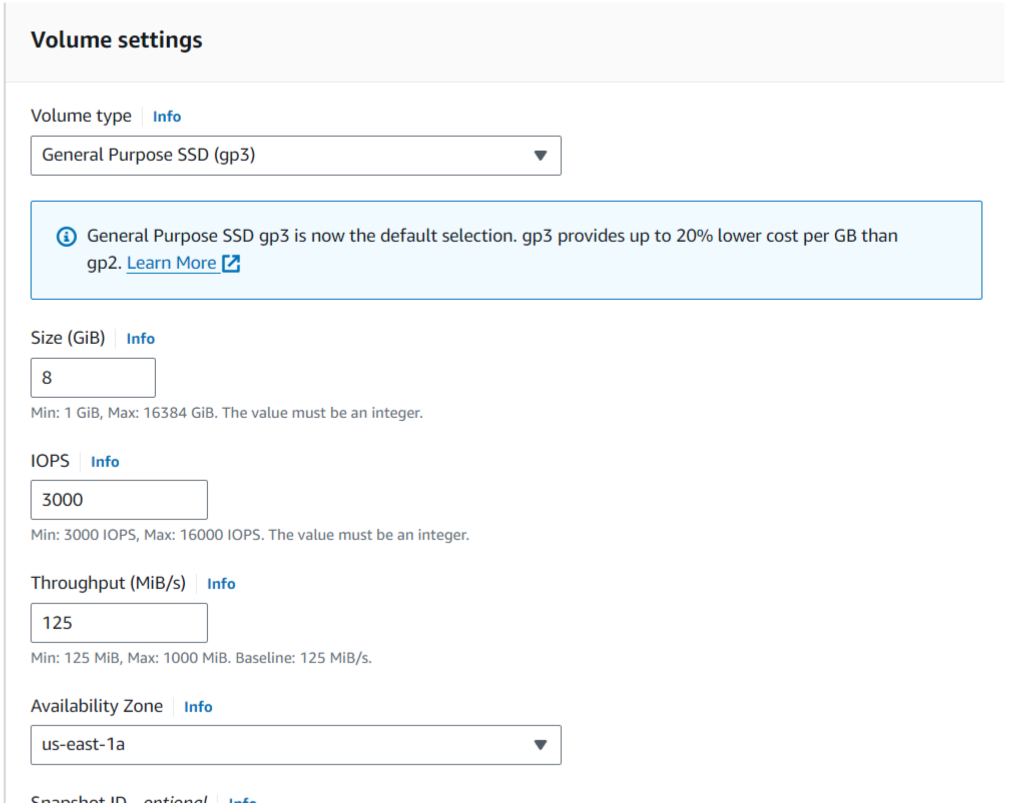
  2. To replace the word only in 1st line we use : `sed '1s/cool/fool/' file.txt`
  3. To replace the first occurrence of "old" with "new" on each line: where s denotes string , old → to be replaced and new → replaced with
     `sed 's/old/new/' filename.txt
  4. To replace all occurrences of "old" with "new" on each line: we add /g
     `sed 's/old/new/g' filename.txt`
  5. To delete lines containing a specific pattern, e.g., lines containing "delete": use /d
     `sed '/delete/d' filename.txt`
  6. To print only specific lines, e.g., line 2 to line 3:
     `sed -n '2,3p' filename.txt`
     1. This prints only line 2 and line 3
     2. `sed -n '1p' file.txt` → this prints only 1st line.
  7. To insert text before a specific line, e.g., before line 2:
     1. `sed '2i\Inserted line' filename.txt`
  8. To append text after a specific line, e.g., after line 2:
     1. `sed '2a\Appended line' filename.txt`

5] Volume Mounting

- Volume mounting in Linux involves attaching a storage device (like a hard drive or USB drive) to a specific directory so that its contents can be accessed through the filesystem. Here's how you can do it:
- If we have a instance and we want to expand storage of the server we can do this by adding additional storage via AWS console.
- Step1 → In Ec2/home go to Elastic Block Store → Volume



- Step 2 (Creating Volume) → Create Volume → Select the appropriate Volume Settings.



Volume settings

Volume type | Info
General Purpose SSD (gp3)

General Purpose SSD gp3 is now the default selection. gp3 provides up to 20% lower cost per GB than gp2. Learn More

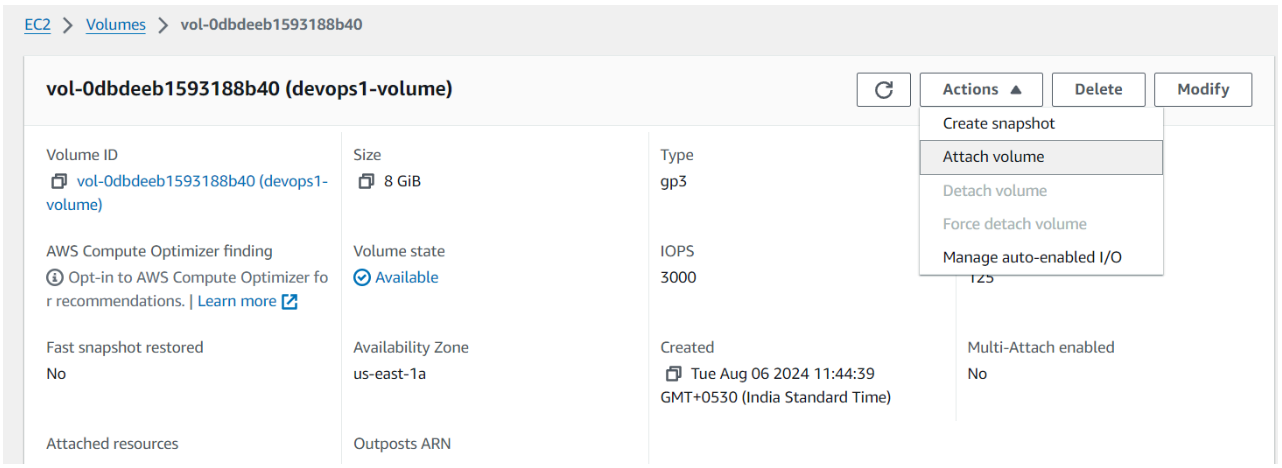
Size (GiB) | Info
8
Min: 1 GiB, Max: 16384 GiB. The value must be an integer.

IOPS | Info
3000
Min: 3000 IOPS, Max: 16000 IOPS. The value must be an integer.

Throughput (MiB/s) | Info
125
Min: 125 MiB, Max: 1000 MiB. Baseline: 125 MiB/s.

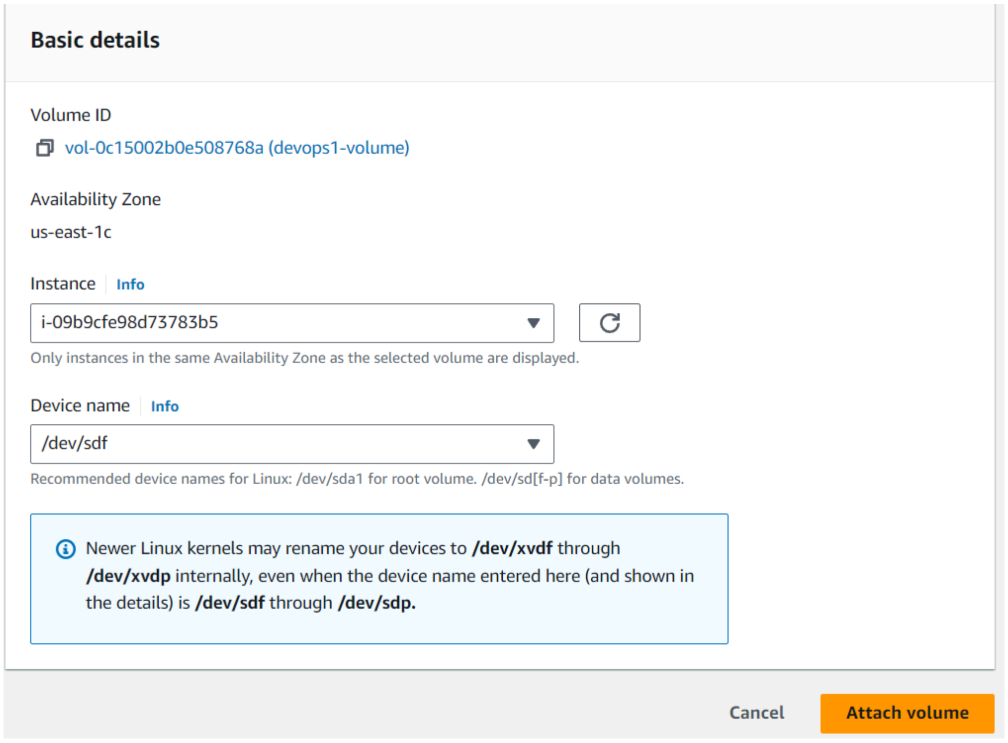
Availability Zone | Info
us-east-1a

- Step 3 (Attaching Volume to Ec2 instance) → After Successfully creating the volume. The Volume is ready/available and is ready to be attached → Click Attach in Actions menu and attach the volume by selecting the instance (to which the volume will be attached) and device name.



vol-0dbdeeb1593188b40 (devops1-volume)

Volume ID vol-0dbdeeb1593188b40 (devops1-volume)	Size 8 GiB	Type gp3	
AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more	Volume state Available	IOPS 3000	125
Fast snapshot restored No	Availability Zone us-east-1a	Created Tue Aug 06 2024 11:44:39 GMT+0530 (India Standard Time)	Multi-Attach enabled No
Attached resources	Outposts ARN		



Basic details

Volume ID
vol-0c15002b0e508768a (devops1-volume)

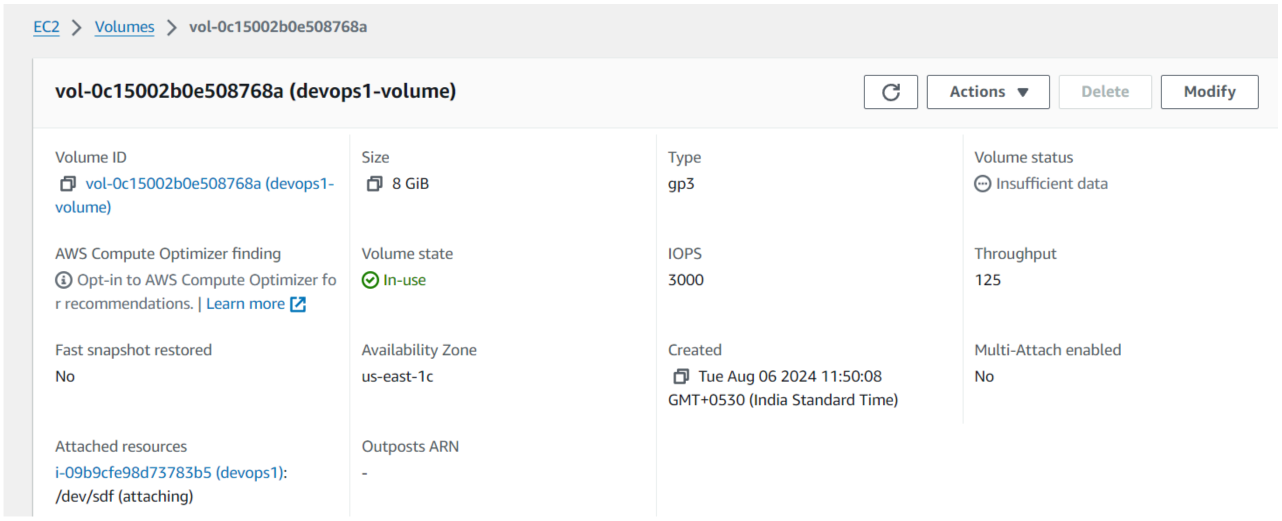
Availability Zone
us-east-1c

Instance Info
i-09b9cfe98d73783b5
Only instances in the same Availability Zone as the selected volume are displayed.

Device name Info
/dev/sdf
Recommended device names for Linux: /dev/sda1 for root volume. /dev/sd[f-p] for data volumes.

Newer Linux kernels may rename your devices to **/dev/xvdf** through **/dev/xvdp** internally, even when the device name entered here (and shown in the details) is **/dev/sdf** through **/dev/sdp.**

- Now the volume becomes in use →

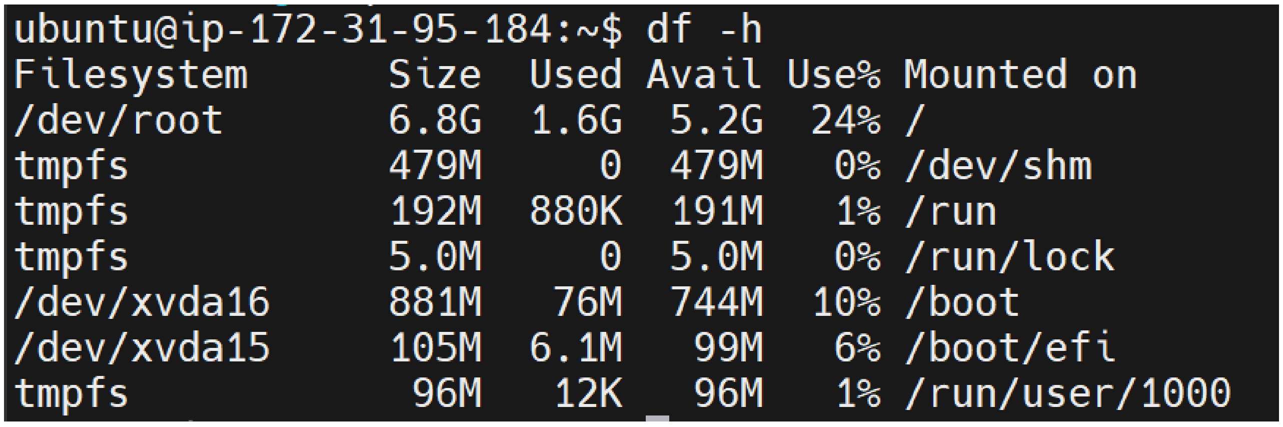


vol-0c15002b0e508768a (devops1-volume)

Volume ID vol-0c15002b0e508768a (devops1-volume)	Size 8 GiB	Type gp3	Volume status Insufficient data
AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more	Volume state In-use	IOPS 3000	Throughput 125
Fast snapshot restored No	Availability Zone us-east-1c	Created Tue Aug 06 2024 11:50:08 GMT+0530 (India Standard Time)	Multi-Attach enabled No
Attached resources i-09b9cfe98d73783b5 (devops1): /dev/sdf (attaching)	Outposts ARN -		

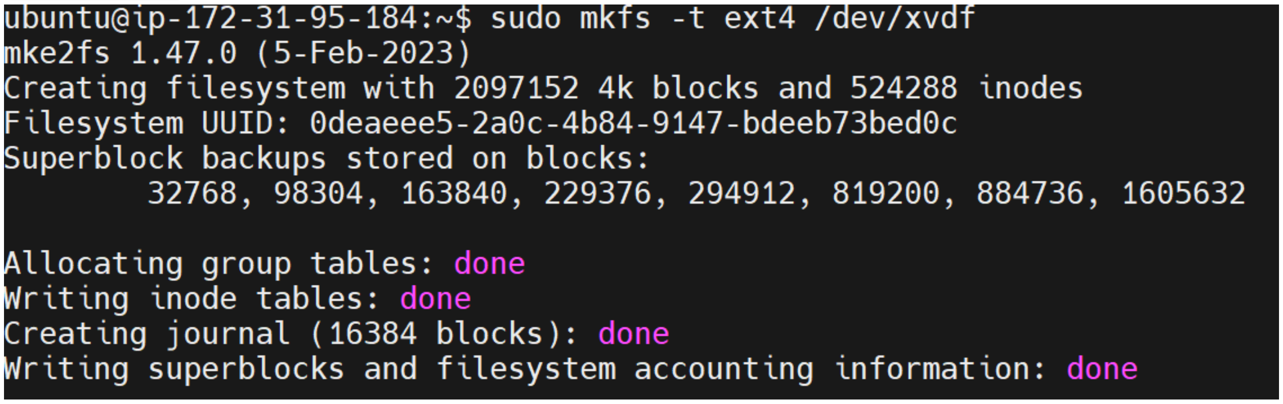
- Step 4 (Attaching volume as a Block)→ Till this step we have only created a volume which is attached to our server/instance. Now to make it of use or to store files in it we have to mount the volume into our device.

- This shows the current file system →



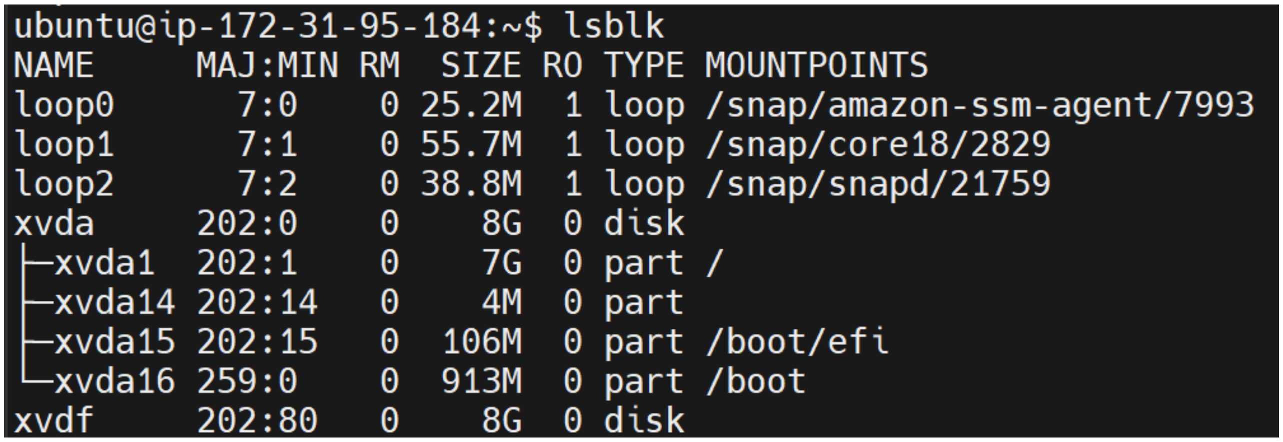
```
ubuntu@ip-172-31-95-184:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       6.8G  1.6G  5.2G  24% /
tmpfs           479M     0  479M   0% /dev/shm
tmpfs           192M  880K  191M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
/dev/xvda16     881M   76M  744M  10% /boot
/dev/xvda15     105M  6.1M   99M   6% /boot/efi
tmpfs            96M   12K   96M   1% /run/user/1000
```

- We don't have the volume that we created ,in the current file system , so we create an external filesystem for our disk file system using command : `sudo mkfs -t ext4 /dev/xvdf`



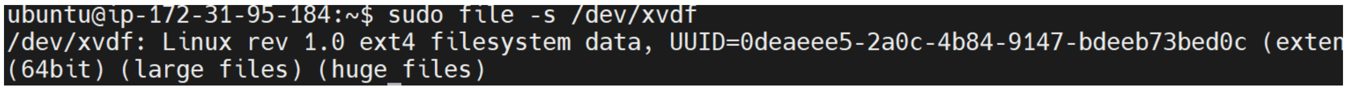
```
ubuntu@ip-172-31-95-184:~$ sudo mkfs -t ext4 /dev/xvdf
mke2fs 1.47.0 (5-Feb-2023)
Creating filesystem with 2097152 4k blocks and 524288 inodes
Filesystem UUID: 0deaeee5-2a0c-4b84-9147-bdeeb73bed0c
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

- Now to see the attached volume (as a block) to the server we use command : `lsblk` → it will list all the blocks which are attached to the server.

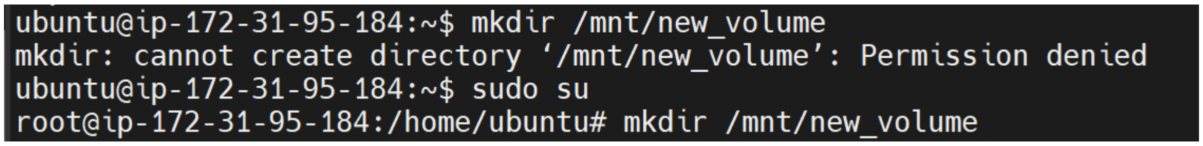


```
ubuntu@ip-172-31-95-184:~$ lsblk
NAME     MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0      7:0    0 25.2M  1 loop /snap/amazon-ssm-agent/7993
loop1      7:1    0 55.7M  1 loop /snap/core18/2829
loop2      7:2    0 38.8M  1 loop /snap/snapd/21759
xvda     202:0    0    8G  0 disk
├─xvda1  202:1    0    7G  0 part /
├─xvda14 202:14   0    4M  0 part
├─xvda15 202:15   0  106M  0 part /boot/efi
└─xvda16 259:0    0  913M  0 part /boot
xvdf     202:80   0    8G  0 disk
```

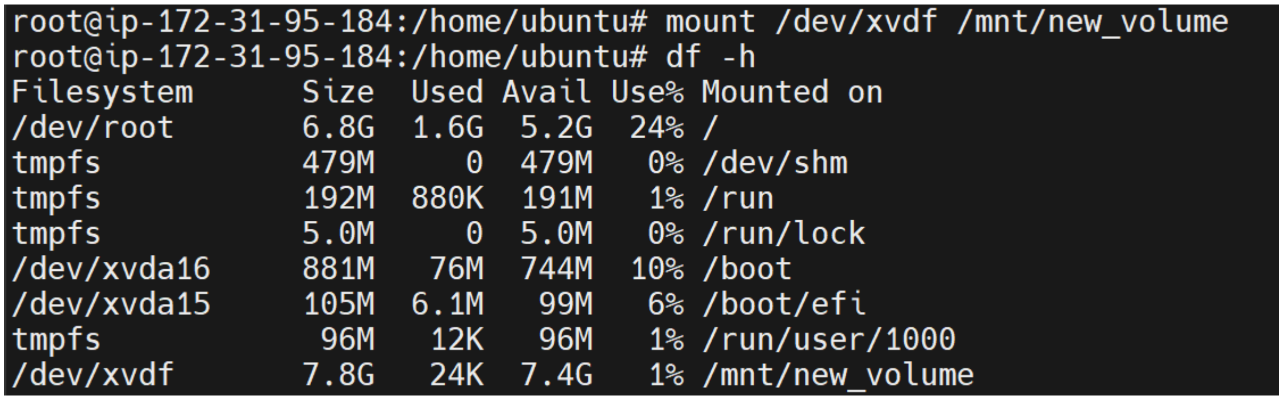
- Step 4 (Mounting the Volume) → Volume has been attached to the server as a block but in Linux every thing is either a file or a directory so we have to make ebs volume as a file system so that we can mount it on our server. To do that we use the command : `sudo file -s /dev/xvdf` → this command converts the ebs volume storage from block → file system.



```
ubuntu@ip-172-31-95-184:~$ sudo file -s /dev/xvdf
/dev/xvdf: Linux rev 1.0 ext4 filesystem data, UUID=0deaeee5-2a0c-4b84-9147-bdeeb73bed0c (exten
(64bit) (large files) (huge_files)
```

- Now the volume can be mounted but first we have to create a path on which it can be mounted. We do this simply by making a folder.
  - `sudo su` → normal user don't have permission to access mnt folder inside `/`
  - `mkdir /mnt/new_volume`
  - 

```
ubuntu@ip-172-31-95-184:~$ mkdir /mnt/new_volume
mkdir: cannot create directory ‘/mnt/new_volume’: Permission denied
ubuntu@ip-172-31-95-184:~$ sudo su
root@ip-172-31-95-184:/home/ubuntu# mkdir /mnt/new_volume
```

- Now finally mounting the ebs volume to server with command : `mount /dev/xvdf /mnt/new_volume` and then we see that the volume is successfully mounted.
- 

```
root@ip-172-31-95-184:/home/ubuntu# mount /dev/xvdf /mnt/new_volume
root@ip-172-31-95-184:/home/ubuntu# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       6.8G  1.6G  5.2G  24% /
tmpfs           479M     0  479M   0% /dev/shm
tmpfs           192M  880K  191M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
/dev/xvda16     881M   76M  744M  10% /boot
/dev/xvda15     105M  6.1M   99M   6% /boot/efi
tmpfs            96M   12K   96M   1% /run/user/1000
/dev/xvdf       7.8G   24K  7.4G   1% /mnt/new_volume
```

Summary

- Create volume.
- Attach Volume.
- Convert volume (block) to file system.
- Add path to be mounted on.
- Mount the ebs Volume.