Table of contents

- Table of contents
- Linux basic operations
  - User craetion
  - Enabling password based authentication for EC2
  - Granting sudo privileges to user
  - Scaling the storage capacity of the instance
  - Cronjob

Linux basic operations

- Agenda for today would be understanding and performing some of the basic linux operations that will help our understanding

User craetion

- Everyt operating system has a default user that comes with it . For ex . Amazon linux 1 and 2 has "ec2-user" as the default user . Ubuntu operating system has "ubuntu" and so on
- We can create other users as well based on the requirement .
- These users are generally created for project teams which use the instance for various developement purposes

```
sudo adduser test1
```

- Above command will create a user named test1
- Now that we have created a user , in order to use it , that user need to have a password

```
sudo passwd test1
```

- Once you use above command , it will prompt for entering a password .
- To test the above , login to the machine as ec2-user and use

```
su test1
```

- su stands for switch user . Above command will ask for a password , once it is entered , you will be able to log in as test1 user
- Super users do not require a password while using su command. So from root you can switch to any user that you wish without having to worry about entering the password

Enabling password based authentication for EC2

- Since the pem key allows access to ec2-user which has sudo access , ideally we shoudnt be sharing the key with the project users
- We should be creating seperate users for each user
- By Default , ec2 instances only have key based authentication . We need to enable password based authentication in order for users to access the instance with just username and password

```
cd /etc/ssh/
sudo vi sshd_config
```

- Once the file is opened , change the PasswordAuthentication as yes

```
# Authentication:

#LoginGraceTime 2m
#PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2
# but this is overridden so installations will only check .ssh/authorized_keys
AuthorizedKeysFile .ssh/authorized_keys

#AuthorizedPrincipalsFile none


# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no
PasswordAuthentication yes

# Change to no to disable s/key passwords
#ChallengeResponseAuthentication yes
ChallengeResponseAuthentication no

# Kerberos options
#KerberosAuthentication no
-- INSERT --
```

- Once the changes are made save and exit

```
sudo service sshd restart
```

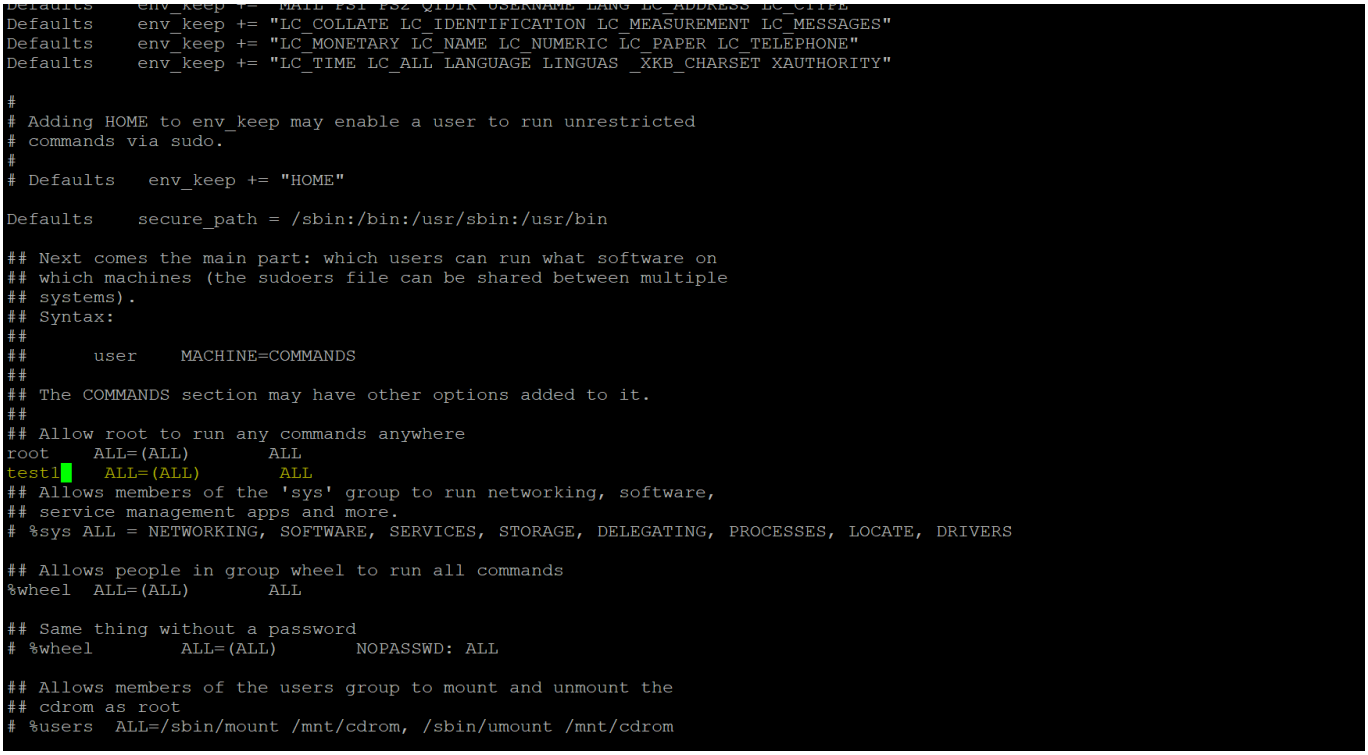
- Above command will restart the ssh daemon which reloads the configuration changes if any
- You can now try logging in using just the username (for ex- test1) and password

Granting sudo privileges to user

- Whenever any new user is created , it only has permissions to operate on its own home folder i.e. /home/test1
- If there is a scenario where the user needs permissions to install or make some changes in other folder , user will need sudo privileges
- In order to grant the permissions

```
sudo visudo
```

- This will open a file , make the changes as below



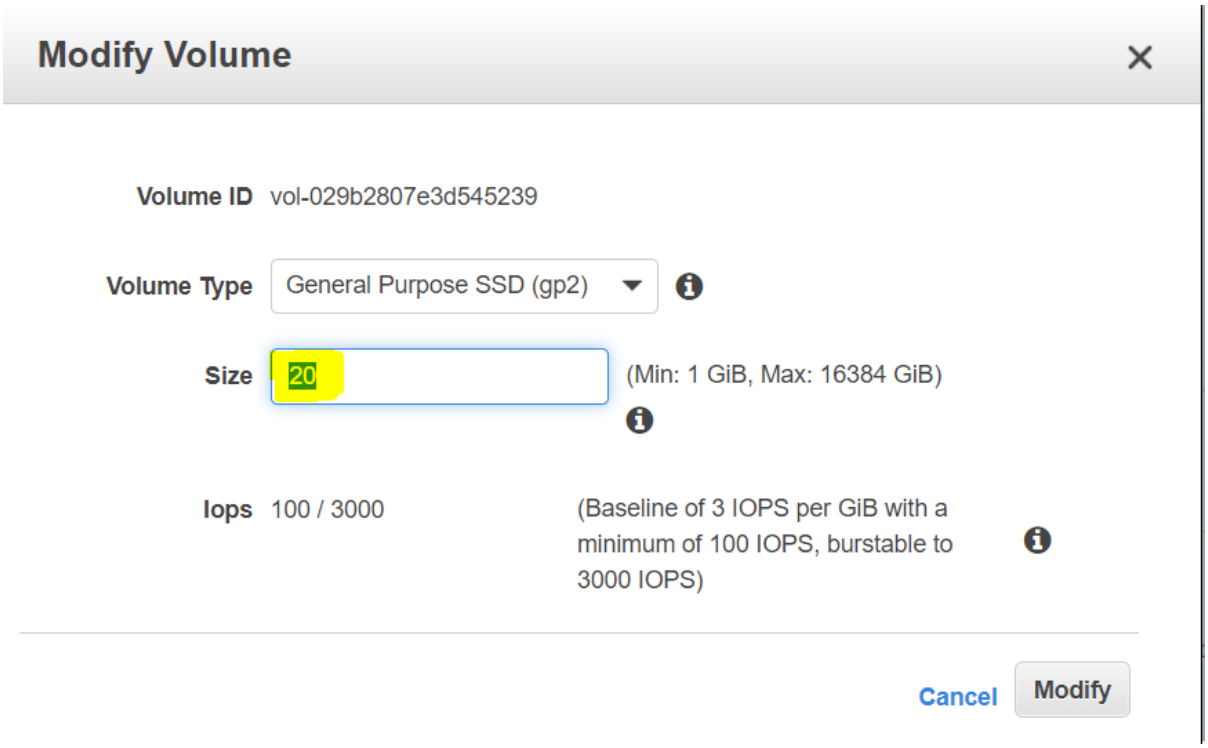
- Save the file and exit . Post which user will be able to run sudo commands

Scaling the storage capacity of the instance

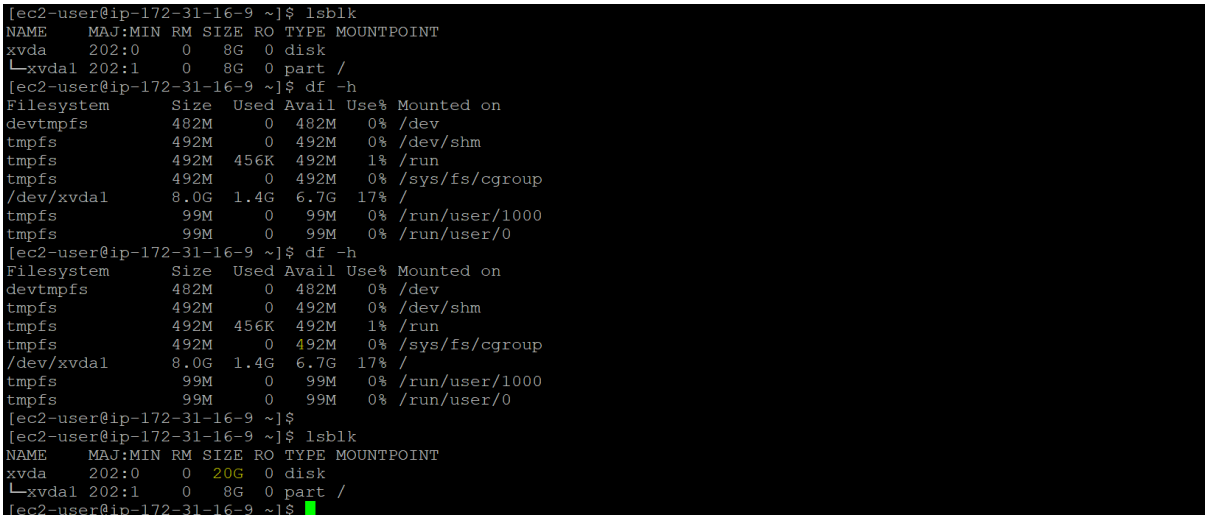
- We know that we can change the instance type if we run out of computig capacity . In order to scale the storage capacity there are couple of approaches we can take
    - Scale the existing ebs volume
        - The general purpose ebs volume has upto 16tb of capacity . If a disc runs out of capacity , we can scale the existing volume

```
[ec2-user@ip-172-31-16-9 ~]$ lsblk
NAME    MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0   8G  0 disk
└─xvda1 202:1    0   8G  0 part /
[ec2-user@ip-172-31-16-9 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        482M     0  482M   0% /dev
tmpfs           492M     0  492M   0% /dev/shm
tmpfs           492M  456K  492M   1% /run
tmpfs           492M     0  492M   0% /sys/fs/cgroup
/dev/xvda1      8.0G  1.4G  6.7G  17% /
tmpfs            99M     0   99M   0% /run/user/1000
tmpfs            99M     0   99M   0% /run/user/0
[ec2-user@ip-172-31-16-9 ~]$
```

- As seen above , the root volume is of 8 gb . It is currently 17% used . We will now scale it
- Navigate to the aws console
- Go to the instance , and select the ebs volume attached to it . It navigates to the volumes screen . Click on actions and select modify volume
- Remember that we can only increase ebs volume .
- Add the new capacity and click on modify

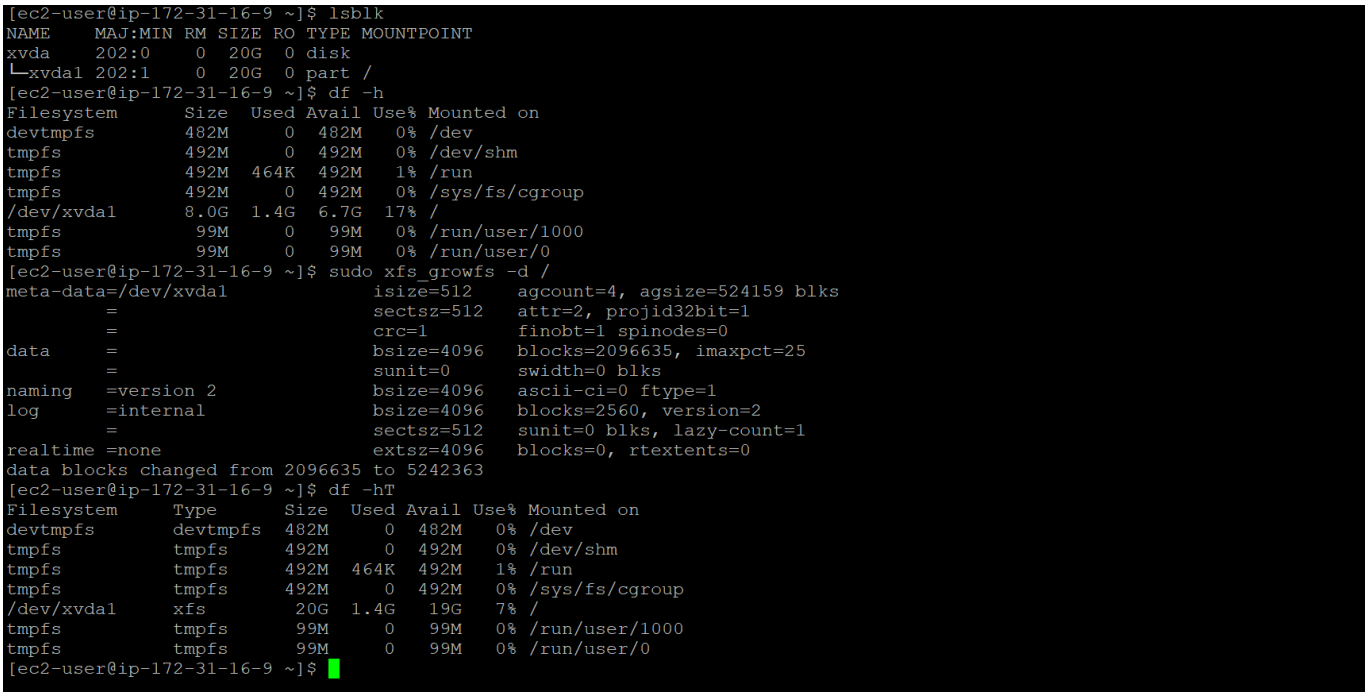


- Once the volume is modified , come back to the terminal



- Notice that the volume is increased , but it wont be usable until we extend the filesystem

```
# based on the block id use
sudo growpart /dev/xvda 1
#above command will increase the partition to accomodate the new size
lsblk
#observe the changes from previous output
df -hT
# if you dont see any changes reflected in above command , use
sudo xfs_growfs -d /
df -hT
```



- This way we can scale an existing volume
- Since the general purpose ebs has 16tb limit , some teams will run out of storage .
- In that scenario we can create new ebs volume and attach it to the instance
- In order to do that , first we need to create a volume
- Navigate to the volumes screen in ec2 service
- Click on create new volume
- Make sure this volume is created in the exact same AZ as the instance
- Once this volume is created attach it the instance
- Notice the changes by using "lsblk" command

```
[ec2-user@ip-172-31-16-9 ~]$ lsblk
NAME    MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0  20G  0 disk
└─xvda1 202:1    0  20G  0 part /
[ec2-user@ip-172-31-16-9 ~]$ lsblk
NAME    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
xvda    202:0    0   20G  0 disk
└─xvda1 202:1    0   20G  0 part /
xvdf    202:80   0  100G  0 disk
[ec2-user@ip-172-31-16-9 ~]$
```

- Notice that the mountpoint section infront of newly added block device is blank . Which means it is not usable yet
- First we will have to create a file system on the volume . Remember , without file system we cannot store data on any disc

```
# first let us check if the volume has a file system
sudo file -s /dev/xvdf
# If output of above command comes as "data", then it means it does not have a
file system and we can go ahead and create it .
# However if the output is x86 or xfs , it means it already has a filesystem and
we shoudnt create it . If you create a filesystem on this it will erase all the
existing data . SO make sure to check before you proceed
 sudo mkfs -t xfs /dev/xvdf
 # Above command will create a file system on the disc
 #now we need a directory where the disc needs to be mounted. It is basically the
address where the disc will be utilized
 sudo mkdir /data
 sudo mount /dev/xvdf /data
```

```
[ec2-user@ip-172-31-16-9 ~]$ sudo file -s /dev/xvdf
/dev/xvdf: data
```

```
[ec2-user@ip-172-31-16-9 ~]$ sudo mkfs -t x86 /dev/xvdf
mkfs: failed to execute mkfs.x86: No such file or directory
[ec2-user@ip-172-31-16-9 ~]$ sudo mkfs -t xfs /dev/xvdf
meta-data=/dev/xvdf              isize=512    agcount=4, agsize=6553600 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=0
data     =                       bsize=4096   blocks=26214400, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal log           bsize=4096   blocks=12800, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
[ec2-user@ip-172-31-16-9 ~]$
```

```
[ec2-user@ip-172-31-16-9 ~]$ sudo mkdir /data
[ec2-user@ip-172-31-16-9 ~]$ sudo mount /dev/xvdf /data
[ec2-user@ip-172-31-16-9 ~]$ lsblk
NAME    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
xvda    202:0    0   20G  0 disk
└─xvda1 202:1    0   20G  0 part /
xvdf    202:80   0  100G  0 disk /data
[ec2-user@ip-172-31-16-9 ~]$
```

- Now the directory and the new volume are usable
- This mounted directory however will be reset if the instance goes through a reboot
- In order to avoid that , we need to enter these new changes in fstab file
- The entry requires a UUID , which is globally unique

```
sudo blkid
# above command will give you UUID of all the block devices attached to the
machine. Copy for the newly created volume
# Now we will make the changes the fstab file. fstab file is one of the most
cruicial files in linux, if there are any mistakes in the entry , then after the
next reboot , you will not be able to login on the instance . So make the changes
carefully , you can also take backup of the current config by creating a copy of
it
sudo vi /etc/fstab
#save and exit
# in order to confirm if all the entries in fstab are correct try below command
sudo mount -a
# If above command runs successfully , then the entry syntax is correct . You can
try rebooting the instance and verify the same
```

```
[ec2-user@ip-172-31-16-9 ~]$ sudo cat /etc/fstab
#
UUID=15c7809d-e6e3-4062-a5eb-afeb1939fc6e     /           xfs    defaults,noatime  1   1
[ec2-user@ip-172-31-16-9 ~]$ sudo blkid
/dev/xvda1: LABEL="/" UUID="15c7809d-e6e3-4062-a5eb-afeb1939fc6e" TYPE="xfs" PARTLABEL="Linux" PARTUUID="cfb2e895-ecec-41c5-afc2-40ffe2
e4384a"
/dev/xvdf: UUID="434aa9a1-f68c-4254-af5a-1bc8df9f35f3" TYPE="xfs"
[ec2-user@ip-172-31-16-9 ~]$ sudo vi /etc/fstab
[ec2-user@ip-172-31-16-9 ~]$
```

```
#
UUID=15c7809d-e6e3-4062-a5eb-afeb1939fc6e     /           xfs    defaults,noatime  1   1
UUID=434aa9a1-f68c-4254-af5a-1bc8df9f35f3     /data       xfs    defaults,noatime  1   1
~
~
~
~
~
~
~
~
~
```

Cronjob

- Cronjobs are a great way of scheduling linux level tasks or scripts
- We have briefly seen cloudwatch rules which used cron expression
- For ex we want to run "uptime " command which shows duration since the last reboot
- uptime >> test.txt will stora the output in the test.txt file

```
[ec2-user@ip-172-31-16-9 ~]$ uptime
 20:22:21 up  1:37,  1 user,  load average: 0.00, 0.00, 0.00
[ec2-user@ip-172-31-16-9 ~]$ uptime >> test.txt
[ec2-user@ip-172-31-16-9 ~]$ cat test.txt
 20:22:32 up  1:37,  1 user,  load average: 0.00, 0.00, 0.00
[ec2-user@ip-172-31-16-9 ~]$
```

- If we wish to schedule above command to run at a certain schedule , we can use cronjob

```
crontab -e
# this opens up the cronjob editor which is exactly like vi editor
# For testing purposes we will make entry "* * * * * uptime >> test.txt"
# Save and exit

crontab -l

# use above to list the cronjobs. Remember cronjobs are user specific , so while
scheduling dont use sudo or else the jobs will be scheduled from root user
```

```
* * * * * uptime >> test.txt
~
~
~
~
~
~
```

```
[ec2-user@ip-172-31-16-9 ~]$ cat test.txt
 20:22:32 up  1:37,  1 user,  load average: 0.00, 0.00, 0.00
 20:27:01 up  1:42,  1 user,  load average: 0.00, 0.00, 0.00
 20:28:01 up  1:43,  1 user,  load average: 0.00, 0.00, 0.00
[ec2-user@ip-172-31-16-9 ~]$
```

- This can even be used to schedule shell/python scripts to be executed at a certain schedule