Table of contents

Linux basic operations day 2

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

Local DNS

- DNS is domain name service which helps us define a name for an ip address or url
- linux has /etc/hosts which is a DNS file and it is the source of truth for the machine
- Any request originating from the machine will first go to this DNS for resolution , if it doesnt find it then it will go to it's ISP's DNS.
- You can see the externam dns in `/etc/resolv.conf` in "nameserver property "
- For testing /etc/hosts , we will add few entries in it to test lie below

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost6 localhost6.localdomain6
10.0.0.1    www.google.com
10.0.1.25   db1
~
~
```

- Once above entries are added to /etc/hosts , try pinging the server names

```
[root@ip-172-31-87-91 ~]# vi /etc/hosts
[root@ip-172-31-87-91 ~]# ping www.google.com
PING www.google.com (10.0.0.1) 56(84) bytes of data.
^C
--- www.google.com ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2042ms

[root@ip-172-31-87-91 ~]# ping db1
PING db1 (10.0.1.25) 56(84) bytes of data.
^C
--- db1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2041ms

[root@ip-172-31-87-91 ~]#
```

- As seen above , even though previously we were able to ping google.com, since we added the entry in local dns , it takes first precedence
- db1 name is also resolved with the ip that we have given

Linux Shell Scripts

Creating our first shell script

- Shell script is nothing but a set of linux shell commands written in an order to execute a task
- We will first use our known commands to create our first shell script
- Let us create

```
sudo su -
vi install.sh
```

- Once the file opens paste below scripts inside the file

```
#!/bin/bash
yum install httpd -y
echo "this is my first script " >> /var/www/html/index.html
service httpd start
```

```
#once we have pasted the content , run below commands
chmod 777 install.sh
#above command will give the fule executable permissions
./install.sh
#above command will execute the script , once executed observe the output
# You can even store the output of the execution in a file using ./install.sh >>
output.txt
```

-

Accepting input from the user

- We may feel the need from time and time again to get inputs from the user
- We can fetch the arguments which are passed while executing the script
- for ex ``./test.sh first secondand herefirstandsecondis an argument which is passed totest.sh`
- Similarly one can pass multiple arguments

```
#!/bin/bash
echo "the name of script is $0"
echo "first parameter is $1"
echo "second parameter is $2"
```

- While executing above script we need to pass 1 argument
- This argument will be printed in the output
- These arguments are also called as positional arguments . if there are 2 arguments which are passed we can use them with $2 ,$3 etc

```
#!/bin/bash
echo "first number you have entered is " $1
echo "second number you have entered is " $2
sum=$(($1 + $2))
echo "addition of numbers is  $sum"
```

- Above script expects 2 positional arguments to be passed while executing a script
- We can also prompt the user to give output that we need
- this output we can read in the script and operate on the same

```
#!/bin/bash
echo "Please enter your name "
read name
echo "hello $name , please enter city you are residing in "
read city
echo "$city is a good city "
```

- Above ways of getting input can be used to get away from hard coding the details
- Even though shell scripts are not as evolved as other programming languages considering these are meant for just scripting purposes , they do support looping as well
- As of today we will just see if else loop
- These are conditional loops which one can use execute simple condition based tasks

```
#!/bin/bash
clear
echo -e "What is your name : \c"
read name
echo hello $name. Welcome to Shell programming
sleep 2
clear
echo -e "Would you like to see a listing of the files in your Current Working
Directory? [y/n]: \c"
read yn
if [ $yn = y ]
then
    ls
fi
```

```
#!/bin/bash
echo "enter the number "
read num
```

```
if [ $((num%2)) -eq 0 ]
then
        echo "even number "
else
        echo "not even number "
fi
```

```
#!/bin/bash
read -p "Enter Current Year (YYYY): " input_year_val
current_year=`date +'%Y'`
if test $input_year_val == $current_year
then
    echo "You have entered the correct Year"
fi
if test $input_year_val != $current_year
then
    echo "You have not entered the Current Year. Please try again."
fi
```

```
#!/bin/bash
file=$1
echo "File path is $file"
if [ -f $file ] ; then
    echo “$file exists”
else
    echo “$file does not exist”
fi
```

`./script.sh /etc/passwd`

- as you can see above if loop begins with `if` and ends with `fi`
- things to be executed under then loop has to be properly intended
- same goes for else

```
echo "enter your access key "
read accesskey
echo "enter your secret access key "
read secret
echo "enter the bucket that you need backup in "
read bucket
echo "enter the backup folder full path "
read sourcepath
export AWS_ACCESS_KEY_ID=$accesskey
export AWS_SECRET_ACCESS_KEY=$secret
export AWS_DEFAULT_REGION=us-east-1
aws s3 cp $sourcepath s3://$bucket/backup/ --recursive
```

- above scripts puts the things we have learnt so far in perspective
- It will take path from user and then based on the inputs initiate the backup

```
#!/bin/bash
echo "enter your access key "
read accesskey
echo "enter your secret access key "
read secret
echo "enter the bucket that you need backup in "
read bucket
echo "enter the backup folder full path "
read sourcepath
export AWS_ACCESS_KEY_ID=$accesskey
export AWS_SECRET_ACCESS_KEY=$secret
export AWS_DEFAULT_REGION=us-east-1
aws s3 cp $sourcepath s3://$bucket/backup/ --recursive
if [ $? -eq 0 ]
then
        echo "backup successful"
else
        echo "backup failed , check your inputs "
fi
```

- To make the script better here , we are using `$?`operator . This stores the output of the previous command .
- If the output is 0 , previous command was successful , or else it failed
- Debugging a shell script
    - Debugging helps you troubleshoot and resolve such errors, and is one of the most important tasks a system administrator performs.

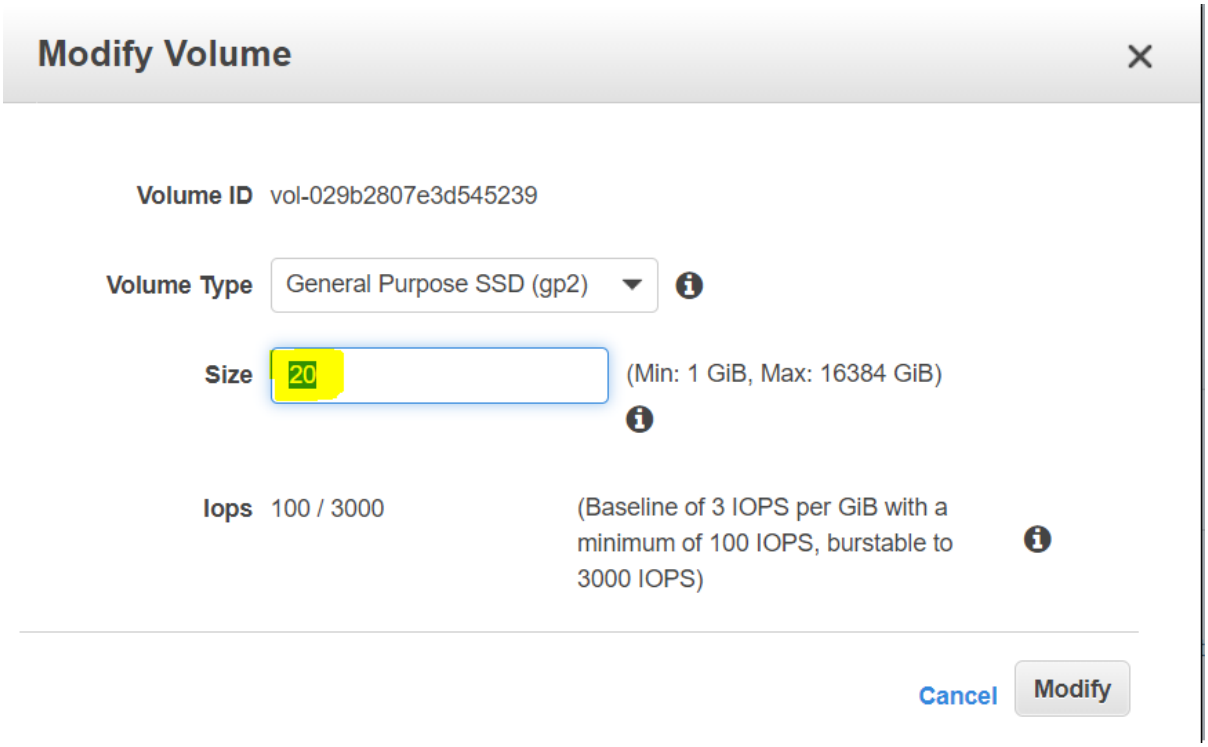
```
bash -x ./script_file
```

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

```
[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 ~]$ 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 ~]$
[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   8G  0 part /
[ec2-user@ip-172-31-16-9 ~]$
```

- 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
```

```
[ec2-user@ip-172-31-16-9 ~]$ sudo growpart /dev/xvda 1
CHANGED: partition=1 start=4096 old: size=16773087 end=16777183 new: size=41938911 end=41943007
```

```
[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 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        482M     0  482M   0% /dev
tmpfs           492M     0  492M   0% /dev/shm
tmpfs           492M  464K  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 ~]$ sudo xfs_growfs -d /
meta-data=/dev/xvda1             isize=512    agcount=4, agsize=524159 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1 spinodes=0
data     =                       bsize=4096   blocks=2096635, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal               bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 2096635 to 5242363
[ec2-user@ip-172-31-16-9 ~]$ df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
devtmpfs       devtmpfs  482M     0  482M   0% /dev
tmpfs          tmpfs     492M     0  492M   0% /dev/shm
tmpfs          tmpfs     492M  464K  492M   1% /run
tmpfs          tmpfs     492M     0  492M   0% /sys/fs/cgroup
/dev/xvda1     xfs        20G  1.4G   19G   7% /
tmpfs          tmpfs      99M     0   99M   0% /run/user/1000
tmpfs          tmpfs      99M     0   99M   0% /run/user/0
[ec2-user@ip-172-31-16-9 ~]$
```

- 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
```