

Linux Assignment

1) Password less SSH automate using shell script.

- To Perform this task, we need to below steps.
 - 2 EC2 Instances, one will act as source from where the keys will be generate and script will be run and other will be destination, where the keys will be copied.
 - Creating a “test” user on both EC2 instances with their home directory is also created.

Source

useradd -p abc123 -m test (Command for creating test with unencrypted password using **-p** and creating home directory using **-m**)

```
root@ip-172-31-29-95:/home/ubuntu# useradd -p abc123 -m test
root@ip-172-31-29-95:/home/ubuntu#
```

We need to reset the password once for the test user to give it an encrypted password. Since, we have setup and unencrypted password using the below command.

passwd test (command to reset the password for the user)

```
root@ip-172-31-29-95:/home/ubuntu# passwd test
New password:
Retype new password:
passwd: password updated successfully
root@ip-172-31-29-95:/home/ubuntu#
```

Now we will also verify home directory is created for the test user or not by using the below command.

cd /home/test (Command to change directory)

```
root@ip-172-31-29-95:/home/ubuntu# cd /home/test
root@ip-172-31-29-95:/home/test#
```

Home directory created and verified

Destination

useradd -p abc123 -m test (Command for creating test with unencrypted password using **-p** and creating home directory using **-m**)

```
root@ip-172-31-20-184:/home/ubuntu# useradd -p abc123 -m test
root@ip-172-31-20-184:/home/ubuntu#
```

We need to reset the password once for the test user to give it an encrypted password. Since, we have setup and unencrypted password using the below command.

passwd test (command to reset the password for the user)

```
root@ip-172-31-20-184:/home/ubuntu# passwd test
New password:
Retype new password:
passwd: password updated successfully
```

Now we will also verify home directory is created for the test user or not by using the below command.

cd /home/test (Command to change directory)

```
root@ip-172-31-20-184:/home/ubuntu# cd /home/test
root@ip-172-31-20-184:/home/test#
root@ip-172-31-20-184:/home/test#
```

Home directory created and verified

- **Providing “test” user root level access with a script which we need to run on both EC2 instances.**

Source

Create a shell script with any name, in my case I created "rootlevel.sh". Edit the file and enter the below contents in that file.

vi rootlevel.sh (Edit the file)

```
root@ip-172-31-29-95:/home/test# vi rootlevel.sh
root@ip-172-31-29-95:/home/test#
```

Contents of the script

```
#!/bin/bash
```

```
# Uncomment the file to allow password authentication
```

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

```
# Add below paramaters in the cloud sshd file
```

```
echo "PasswordAuthentication yes" > /etc/ssh/sshd_config.d/60-
cloudimg-settings.conf
```

```
echo -e "PermitRootLogin yes" >> /etc/ssh/sshd_config.d/60-
cloudimg-settings.conf
```

```
# Giving same permissions as root to test
```

```
echo "test    ALL=(ALL:ALL) ALL" > a.txt
```

```
sed -i '47r a.txt' /etc/sudoers
```

SSH Service Restart

systemctl restart ssh

```
#!/bin/bash
# Uncomment the file to allow password authentication
sed -i '/PasswordAuthentication yes/s/^#//g' /etc/ssh/sshd_config
# Add below paramaters in the cloud sshd file
echo "PasswordAuthentication yes" > /etc/ssh/sshd_config.d/60-cloudimg-settings.conf
echo -e "PermitRootLogin yes" >> /etc/ssh/sshd_config.d/60-cloudimg-settings.conf
# Giving same permissions as root to test
echo "test    ALL=(ALL:ALL) ALL" > a.txt
sed -i '47r a.txt' /etc/sudoers
# SSH Service Restart
systemctl restart ssh
```

Run the script as bash and test user will have same level of access as root

bash rootlevel.sh

```
root@ip-172-31-29-95:/home/test# bash rootlevel.sh
root@ip-172-31-29-95:/home/test#
```

Same steps needed to done on other Destination

Destination

Create a shell script with any name, in my case I created “rootlevel.sh”. Edit the file and enter the below contents in that file.

vi rootlevel.sh (Edit the file)

```
root@ip-172-31-20-184:/home/test# vi rootlevel.sh
```

Contents of the script

```
#!/bin/bash
```

```
# Uncomment the file to allow password authentication
```

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

```
# Add below paramaters in the cloud sshd file
```

```
echo "PasswordAuthentication yes" >  
/etc/ssh/sshd_config.d/60-cloudimg-settings.conf
```

```
echo -e "PermitRootLogin yes" >> /etc/ssh/sshd_config.d/60-  
cloudimg-settings.conf
```

```
# Giving same permissions as root to test
```

```
echo "test  ALL=(ALL:ALL) ALL" > a.txt
```

```
sed -i '47r a.txt' /etc/sudoers
```

```
# SSH Service Restart
```

systemctl restart ssh

```
#!/bin/bash
# Uncomment the file to allow password authentication
sed -i '/PasswordAuthentication yes/s/^#//g' /etc/ssh/sshd_config
# Add below parameters in the cloud sshd file
echo "PasswordAuthentication yes" > /etc/ssh/sshd_config.d/60-cloudimg-settings.conf
echo -e "PermitRootLogin yes" >> /etc/ssh/sshd_config.d/60-cloudimg-settings.conf
# Giving same permissions as root to test
echo "test    ALL=(ALL:ALL) ALL" > a.txt
sed -i '47r a.txt' /etc/sudoers
# SSH Service Restart
systemctl restart ssh
```

Run the script as bash and test user will have same level of access as root

bash rootlevel.sh

```
root@ip-172-31-20-184:/home/test# bash rootlevel.sh
```

Now setup password less ssh between master and worker by following below steps.

- **Creating a temporary file “tempUserName.txt” for passing “test” user details on source EC2 instance.**

echo "test" > tempUserName.txt (Command creates tempUserName.txt and puts test as text inside)

```
root@ip-172-31-29-95:/home/test# echo "test" > tempUserName.txt
root@ip-172-31-29-95:/home/test#
```

- **Creating a file by name “ipaddr” which will contain the Public ip address of destination on source EC2 instance.**

`echo "34.234.63.151" > ipaddr` (Command creates ipaddr file and puts the ip address as text inside)

```
root@ip-172-31-29-95:/home/test# echo "34.234.63.151" > ipaddr
root@ip-172-31-29-95:/home/test#
```

- **Setting up password less authentication between master and worker nodes from the master instance using script.**

Source

Create a shell script with any name, in my case I created “passwordless_ssh.sh”. Edit the file and enter the below contents in that file.

`vi passwordless_ssh.sh` (Edit the file)

```
root@ip-172-31-29-95:/home/test# vi passwordless_ssh.sh
```

Contents of the script

`#!/bin/bash`

`# Defining variable for ipaddr file`

`ipAddFile="./ipaddr"`

Command runs SSH-Keygen without prompting anything

```
echo -e "\n" | ssh-keygen -N "" &> /dev/null
```

Now the Script checks the IP address file and concat the username and IP address to copy the ssh pub key

```
echo "$ipAddFile"
```

```
for IP in `cat $ipAddFile`; do
```

```
    if [[ $IP == *"[ "* ]]; then
```

```
        echo "$IP"|cut -d "[" -f2 | cut -d "]" -f1>tempUserName.txt
```

```
    else
```

```
        user=$(cat tempUserName.txt)
```

```
        ssh-copy-id $user@$IP
```

```
        echo "Key copied to $IP"
```

```
    fi
```

```
done
```

Remove tempUserName.txt file

```
rm -rf tempUserName.txt
```



```
#!/bin/bash

# Defining variable for ipaddr file
ipAddrFile="./ipaddr"

# Command runs SSH-Keygen without prompting anything
echo -e "\n" | ssh-keygen -N "" &> /dev/null

# Now the Script checks the IP address file and concat the username and IP address to copy the ssh pub key
echo "$ipAddrFile"

for IP in `cat $ipAddrFile`; do
    if [[ $IP == *[""] ]]; then
        echo "$IP"|cut -d "[" -f2 | cut -d "]" -f1>tempUserName.txt
    else
        user=$(cat tempUserName.txt)
        ssh-copy-id $user@$IP
        echo "Key copied to $IP"
    fi
done

# Remove tempUserName.txt file
rm -rf tempUserName.txt
```

Run the script as bash and passwordless SSH will be setup between source and destination docker.

bash passwordless_ssh.sh

```
root@ip-172-31-29-95:/home/test# bash passwordless_ssh.sh
./ipaddr
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_ed25519.pub"
The authenticity of host '34.234.63.151 (34.234.63.151)' can't be established.
ED25519 key fingerprint is SHA256:95gSZj/eZC38TY0FNDjgRg/s5NDgUa70MtE2zKWSnig.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
test@34.234.63.151's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'test@34.234.63.151'"
and check to make sure that only the key(s) you wanted were added.

Key copied to 34.234.63.151
root@ip-172-31-29-95:/home/test#
```

We need to supply password for test user to get the key copied (Above screenshot shows key copied to the IP address in ipaddr file)

NOTE: Key will be copied to /home/test/.ssh/authorized_keys

- 2) If we put any number, the output should give 100 numbers after that using shell script. Script should ask the number and then when number is entered then it should 100 numbers following that number.

vi 100num.sh

```
root@ip-172-31-84-27:/home/ubuntu# vi 100num.sh
root@ip-172-31-84-27:/home/ubuntu#
```

Below are script contents

#!/bin/bash

read -p "Enter the number: " num

echo \$num

max=\$((\$num + 100))

echo \$max

echo "Next 100 numbers are:"

for ((i="\$num"+1; i<="\$max"; i++))

do

 echo "\$i"

done

```
#!/bin/bash

read -p "Enter the number: " num

echo $num
max=$(( $num + 100 ))
echo $max
echo "Next 100 numbers are:"

for (( i=$num+1; i<=$max; i++ ))
do
    echo "$i"
done
```

Run the script as bash

bash 100num.sh

```
root@ip-172-31-84-27:/home/ubuntu# bash 100num.sh
Enter the number: 20
20
120
Next 100 numbers are:
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
```

```
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86 105
87 106
88 107
89 108
90 109
91 110
92 111
93 112
94 113
95 114
96 115
97 116
98 117
99 118
100 119
101 120
102
103
104
```

3) If we put any number, it should print the table of that number using shell script.

vi table.sh

```
root@ip-172-31-84-27:/home/ubuntu# vi table.sh
root@ip-172-31-84-27:/home/ubuntu#
```

Below are script contents

#!/bin/bash

```
read -p "Enter the number: " num
```

```
echo $num
```

```
echo "Table is as follows:"
```

```
for (( i=1; i<=10; i++ ))
```

```
do
```

```
    val=$(( num * i ))
```

```
    echo "$num * $i = " $val
```

```
done
```

```
#!/bin/bash

read -p "Enter the number: " num

echo $num
echo "Table is as follows:"

for (( i=1; i<=10; i++ ))
do
    val=$(( num * i ))
    echo "$num * $i = " $val
done
```

Run the script as bash

```
bash table.sh
```

```
root@ip-172-31-84-27:/home/ubuntu# bash table.sh
Enter the number: 5
5
Table is as follows:
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
root@ip-172-31-84-27:/home/ubuntu#
```

- 4) If I have a file with contents, the script should print the number of lines, words and characters in that file using shell script.

For this task, we will create file name test.txt with random content in it.

File is created under the path /home/ubuntu

vi test.txt

```
root@ip-172-31-84-27:/home/ubuntu# vi test.txt
root@ip-172-31-84-27:/home/ubuntu#
root@ip-172-31-84-27:/home/ubuntu#
```

Contents inside the test.txt file

```
kbgekjdngNFSbvkjznvdknb
;nbvzbv zdbzxbvb
jzVbkzdvdvkvkz
kjBvkzvbzkbvzkb
kBVkzvbzkbkdb
kBCKZBVkzvbzdkv
jVCKVbzkbkZbLV
kBCKKVbKVLV
KCBkbKVLs
kJGABdAHVjVFkBFsFBF1S
jDkVFDKFvKHJFVKF
```

Now we will create the script and define the file path for whom we need the output

vi lines.sh

```
root@ip-172-31-84-27:/home/ubuntu# vi lines.sh
root@ip-172-31-84-27:/home/ubuntu#
```

Below are script contents

#!/bin/bash

Count the number of lines in the file "test.txt"

```
line_count=$(wc -l < test.txt)
```

```
echo "Number of lines in test.txt: $line_count"
```

```
# Count the number of words in the file "test.txt"
```

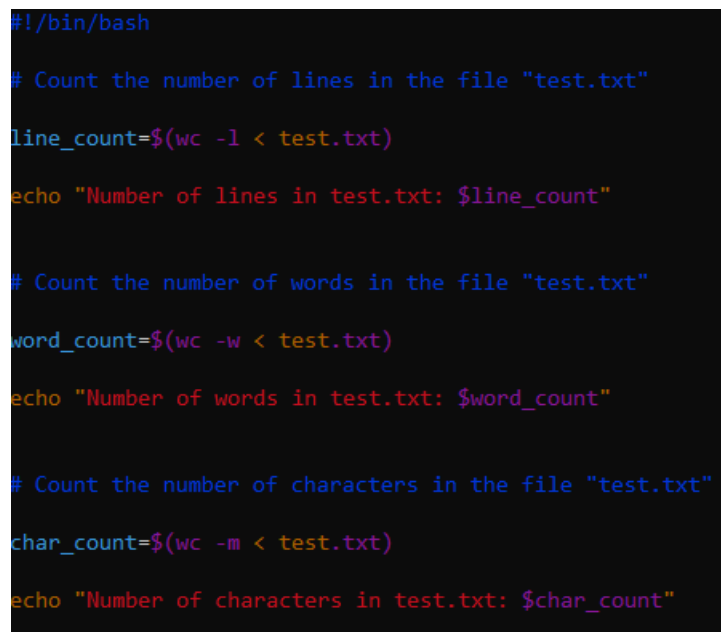
```
word_count=$(wc -w < test.txt)
```

```
echo "Number of words in test.txt: $word_count"
```

```
# Count the number of characters in the file "test.txt"
```

```
char_count=$(wc -m < test.txt)
```

```
echo "Number of characters in test.txt: $char_count"
```

A screenshot of a terminal window with a dark background and light-colored text. The text is the same script shown in the previous blocks, including comments and commands for counting lines, words, and characters in 'test.txt'.

```
#!/bin/bash

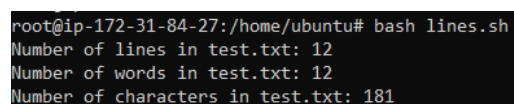
# Count the number of lines in the file "test.txt"
line_count=$(wc -l < test.txt)
echo "Number of lines in test.txt: $line_count"

# Count the number of words in the file "test.txt"
word_count=$(wc -w < test.txt)
echo "Number of words in test.txt: $word_count"

# Count the number of characters in the file "test.txt"
char_count=$(wc -m < test.txt)
echo "Number of characters in test.txt: $char_count"
```

Run the script as bash

```
bash lines.sh
```

A screenshot of a terminal window showing the output of the script. The prompt is 'root@ip-172-31-84-27:/home/ubuntu#'. The output shows three lines: 'Number of lines in test.txt: 12', 'Number of words in test.txt: 12', and 'Number of characters in test.txt: 181'.

```
root@ip-172-31-84-27:/home/ubuntu# bash lines.sh
Number of lines in test.txt: 12
Number of words in test.txt: 12
Number of characters in test.txt: 181
```

- 5) Every evening 5'o clock script should run automatically all the files that are placed in one folder (source) should be copied to another folder (destination). after 30 mins all the files and folder should be deleted from the destination folder.

This script will be divided into two parts

- First part will automatically copy the files from a source folder to destination folder on a given time.
- Second part will delete files and folder on the other give time.

Performing the steps for different parts.

- First part will automatically copy the files from a source folder to destination folder on a given time.

We will first create the source folder by the name (**srcfolder**) on the path “**/home/ubuntu/scripts**” and copy contents in it

mkdir scripts

```
root@ip-172-31-84-27:/home/ubuntu# mkdir scripts
root@ip-172-31-84-27:/home/ubuntu#
```

cd scripts/

```
root@ip-172-31-84-27:/home/ubuntu# cd scripts/
root@ip-172-31-84-27:/home/ubuntu/scripts#
```

mkdir srcfolder


```
root@ip-172-31-84-27:/home/ubuntu/scripts# mkdir srcfolder
root@ip-172-31-84-27:/home/ubuntu/scripts#
```

Copying files into the source folder

`cp 100num.sh lines.sh table.sh test.txt
/home/ubuntu/scripts/srcfolder/`

```
root@ip-172-31-84-27:/home/ubuntu# cp 100num.sh lines.sh table.sh test.txt /home/ubuntu/scripts/srcfolder/
root@ip-172-31-84-27:/home/ubuntu#
```

```
root@ip-172-31-84-27:/home/ubuntu/scripts/srcfolder# ls -ltr
total 16
-rw-r--r-- 1 root root 181 Sep 12 09:41 test.txt
-rw-r--r-- 1 root root 197 Sep 12 09:41 table.sh
-rw-r--r-- 1 root root 424 Sep 12 09:41 lines.sh
-rw-r--r-- 1 root root 182 Sep 12 09:41 100num.sh
root@ip-172-31-84-27:/home/ubuntu/scripts/srcfolder#
```

Now we will create the target folder by the name (**tgtfolder**) on the path **“/home/ubuntu/scripts”** and in which contents will be automatically at a given time.

`mkdir tgtfolder`

```
root@ip-172-31-84-27:/home/ubuntu/scripts# mkdir tgtfolder
root@ip-172-31-84-27:/home/ubuntu/scripts#
```

Nothing in the tgt folder as of now.

```
root@ip-172-31-84-27:/home/ubuntu# date
Thu Sep 12 10:11:57 UTC 2024
root@ip-172-31-84-27:/home/ubuntu#
root@ip-172-31-84-27:/home/ubuntu#
root@ip-172-31-84-27:/home/ubuntu# cd scripts/tgtfolder/
root@ip-172-31-84-27:/home/ubuntu/scripts/tgtfolder#
root@ip-172-31-84-27:/home/ubuntu/scripts/tgtfolder# ls -ltr
total 0
root@ip-172-31-84-27:/home/ubuntu/scripts/tgtfolder#
```

Writing the script for the first part of archiving job

vi archive.sh

```
root@ip-172-31-84-27:/home/ubuntu#  
root@ip-172-31-84-27:/home/ubuntu# vi archive.sh  
root@ip-172-31-84-27:/home/ubuntu#
```

Contents of the script

#!/bin/bash

DATE=\$(date +%d-%m-%Y)

srcPath=/home/ubuntu/scripts/srcfolder

tgtPath=/home/ubuntu/scripts/tgtfolder

echo "Starting Archiving job as per scheduled time: "

cd \$srcPath

cp * \$tgtPath

echo "Archival process completed"

```
#!/bin/bash  
  
DATE=$(date +%d-%m-%Y)  
  
srcPath=/home/ubuntu/scripts/srcfolder  
tgtPath=/home/ubuntu/scripts/tgtfolder  
  
echo "Starting Archiving job as per scheduled time: "  
  
cd $srcPath  
cp * $tgtPath  
  
echo "Archival process completed"
```

NOTE: To show the output while creating the script, I have taken the time 11:00 AM UTC time to start the cron job.

Now we will setup the script as cron job

crontab -e

We choose option 2 to edit the file in Vim

```
root@ip-172-31-84-27:/home/ubuntu# crontab -e
no crontab for root - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano      <---- easiest
 2. /usr/bin/vim.basic
 3. /usr/bin/vim.tiny
 4. /bin/ed
Choose 1-4 [1]: 2
```

We have added our script as cron job in the below file to run the script at 11:00 server time and also pointed it to generate in a log file so that we can see the output.

0 11 * * * bash

/home/ubuntu/archive.sh>>/home/ubuntu/archive.log

```
Select root@ip-172-31-84-27: /home/ubuntu
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
0 11 * * * bash /home/ubuntu/archive.sh>>/home/ubuntu/archive.log
```

Output:

Time is 11:00, we will verify the time and see if the log file is generated and cronjob ran successfully and also check the output.

date

```
root@ip-172-31-84-27:/home/ubuntu# date
Thu Sep 12 11:00:35 UTC 2024
root@ip-172-31-84-27:/home/ubuntu#
```

ls -ltr

archive.log is created

```
root@ip-172-31-84-27:/home/ubuntu# ls -ltr
total 32
-rw-r--r-- 1 root root 182 Sep 12 08:16 100num.sh
-rw-r--r-- 1 root root 197 Sep 12 08:32 table.sh
-rw-r--r-- 1 root root 181 Sep 12 08:48 test.txt
-rw-r--r-- 1 root root 424 Sep 12 09:00 lines.sh
drwxr-xr-x 4 root root 4096 Sep 12 09:42 scripts
-rw-r--r-- 1 root root 233 Sep 12 10:36 archive.sh
-rw-r--r-- 1 root root 198 Sep 12 10:55 housekeeping.sh
-rw-r--r-- 1 root root 74 Sep 12 11:00 archive.log
root@ip-172-31-84-27:/home/ubuntu#
```

cat archive.log

```
root@ip-172-31-84-27:/home/ubuntu# cat archive.log
Starting Archiving job as per scheduled time:
Archival process completed
root@ip-172-31-84-27:/home/ubuntu#
```

We will verify whether the contents are copied to tgtfolder

ls -ltr scripts/tgtfolder

All the files have been copied from srcfolder to tgtfolder

```
root@ip-172-31-84-27:/home/ubuntu# ls -ltr scripts/tgtfolder/
total 16
-rw-r--r-- 1 root root 181 Sep 12 11:00 test.txt
-rw-r--r-- 1 root root 197 Sep 12 11:00 table.sh
-rw-r--r-- 1 root root 424 Sep 12 11:00 lines.sh
-rw-r--r-- 1 root root 182 Sep 12 11:00 100num.sh
root@ip-172-31-84-27:/home/ubuntu#
```

First part of script completed successfully.

- Second part will delete files and folder on the other give time.

Script for deleting the contents or housekeeping job

vi housekeeping.sh

```
root@ip-172-31-84-27:/home/ubuntu# vi housekeeping.sh
root@ip-172-31-84-27:/home/ubuntu#
root@ip-172-31-84-27:/home/ubuntu#
```

Contents of the script

#!/bin/bash

DATE=\$(date +%d-%m-%Y)

tgtPath=/home/ubuntu/scripts/tgtfolder

echo "Starting Housekeeping job as per scheduled time: "

cd \$tgtPath

rm -rf *

echo "Housekeeping process completed"

```
#!/bin/bash

DATE=$(date +%d-%m-%Y)

tgtPath=/home/ubuntu/scripts/tgtfolder

echo "Starting Housekeeping job as per scheduled time: "

cd $tgtPath

rm -rf *

echo "Housekeeping process completed"
```

Now we will setup the script as cron job

crontab -e

We choose option 2 to edit the file in Vim

```
root@ip-172-31-84-27:/home/ubuntu# crontab -e
no crontab for root - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano      <---- easiest
 2. /usr/bin/vim.basic
 3. /usr/bin/vim.tiny
 4. /bin/ed

Choose 1-4 [1]: 2
```

We have added our script as cron job in the below file and also pointed it to generate in a log file so that we can see the output.

30 * * * * bash

/home/ubuntu/housekeeping.sh>>/home/ubuntu/housekeeping.log

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
0 11 * * * bash /home/ubuntu/archive.sh>>/home/ubuntu/archive.log
30 * * * * bash /home/ubuntu/housekeeping.sh>>/home/ubuntu/housekeeping.log
```

Output

Time is 11:30, we will verify the time and see if the log file is generated and cronjob ran successfully and also check the output.

date

```
root@ip-172-31-84-27:/home/ubuntu# date
Thu Sep 12 11:30:53 UTC 2024
root@ip-172-31-84-27:/home/ubuntu#
```

ls -ltr

housekeeping.log is created

```
root@ip-172-31-84-27:/home/ubuntu# ls -ltr
total 36
-rw-r--r-- 2 root root 197 Sep 12 08:32 table.sh
-rw-r--r-- 2 root root 197 Sep 12 08:32 new.sh
-rw-r--r-- 1 root root 181 Sep 12 08:48 test.txt
-rw-r--r-- 1 root root 424 Sep 12 09:00 lines.sh
drwxr-xr-x 4 root root 4096 Sep 12 09:42 scripts
-rw-r--r-- 1 root root 233 Sep 12 10:36 archive.sh
-rw-r--r-- 1 root root 198 Sep 12 10:55 housekeeping.sh
-rw-r--r-- 1 root root 74 Sep 12 11:00 archive.log
lrwxrwxrwx 1 root root 9 Sep 12 11:17 number.sh -> 100num.sh
-rw-r--r-- 1 root root 81 Sep 12 11:30 housekeeping.log
root@ip-172-31-84-27:/home/ubuntu#
```

cat housekeeping.log

```
root@ip-172-31-84-27:/home/ubuntu# cat housekeeping.log
Starting Housekeeping job as per scheduled time:
Housekeeping process completed
root@ip-172-31-84-27:/home/ubuntu#
```

We will verify whether the contents are copied to tgtfolder

ls -ltr scripts/tgtfolder

All the files that had been archived to tgtfolder are deleted successfully.

```
root@ip-172-31-84-27:/home/ubuntu# ls -ltr scripts/tgtfolder/
total 0
root@ip-172-31-84-27:/home/ubuntu#
```

Second part of script is also completed successfully.

Task is completed successfully.

6) If we mount 15 GB, Will i see all storage or some storage and why?

Linux reserves around 5% of storage for root user and system services. If disk space is full and no space left on drive, then no one will be able to login. Hence for smooth functioning of drive, space is reserved.

7) Soft link and Hard Link

Soft Link: A [soft link](#) (also known as a Symbolic link) acts as a pointer or a reference to the file name. It does not access the data available in the original file. If the earlier file is deleted, the soft link will be pointing to a file that does not exist anymore.

Example of soft link

ln -s 100num.sh number.sh

```
root@ip-172-31-84-27:/home/ubuntu# ln -s 100num.sh number.sh
root@ip-172-31-84-27:/home/ubuntu#
```

ls -ltr

soft link created

```
root@ip-172-31-84-27:/home/ubuntu# ls -ltr
total 32
-rw-r--r-- 1 root root 182 Sep 12 08:16 100num.sh
-rw-r--r-- 1 root root 197 Sep 12 08:32 table.sh
-rw-r--r-- 1 root root 181 Sep 12 08:48 test.txt
-rw-r--r-- 1 root root 424 Sep 12 09:00 lines.sh
drwxr-xr-x 4 root root 4096 Sep 12 09:42 scripts
-rw-r--r-- 1 root root 233 Sep 12 10:36 archive.sh
-rw-r--r-- 1 root root 198 Sep 12 10:55 housekeeping.sh
-rw-r--r-- 1 root root 74 Sep 12 11:00 archive.log
lrwxrwxrwx 1 root root 9 Sep 12 11:17 number.sh -> 100num.sh
root@ip-172-31-84-27:/home/ubuntu#
```

cat 100num.sh


```

root@ip-172-31-84-27:/home/ubuntu# cat 100num.sh
#!/bin/bash

read -p "Enter the number: " num

echo $num
max=$(( $num + 100 ))
echo $max
echo "Next 100 numbers are:"

for (( i=$num+1; i<=$max; i++ ))
do
    echo "$i"
done

```

cat number.sh

```

root@ip-172-31-84-27:/home/ubuntu# cat number.sh
#!/bin/bash

read -p "Enter the number: " num

echo $num
max=$(( $num + 100 ))
echo $max
echo "Next 100 numbers are:"

for (( i=$num+1; i<=$max; i++ ))
do
    echo "$i"
done
root@ip-172-31-84-27:/home/ubuntu#

```

If we remove, the original file, the soft link is also gone

rm 100num.sh

```

root@ip-172-31-84-27:/home/ubuntu# rm 100num.sh
root@ip-172-31-84-27:/home/ubuntu#

```

ls-ltr

```

root@ip-172-31-84-27:/home/ubuntu# ls -ltr
total 28
-rw-r--r-- 1 root root 197 Sep 12 08:32 table.sh
-rw-r--r-- 1 root root 181 Sep 12 08:48 test.txt
-rw-r--r-- 1 root root 424 Sep 12 09:00 lines.sh
drwxr-xr-x 4 root root 4096 Sep 12 09:42 scripts
-rw-r--r-- 1 root root 233 Sep 12 10:36 archive.sh
-rw-r--r-- 1 root root 198 Sep 12 10:55 housekeeping.sh
-rw-r--r-- 1 root root 74 Sep 12 11:00 archive.log
lrwxrwxrwx 1 root root 9 Sep 12 11:17 number.sh -> 100num.sh
root@ip-172-31-84-27:/home/ubuntu#
root@ip-172-31-84-27:/home/ubuntu# cat number.sh
cat: number.sh: No such file or directory
root@ip-172-31-84-27:/home/ubuntu#

```

cat number.sh

```
root@ip-172-31-84-27:/home/ubuntu# cat number.sh
cat: number.sh: No such file or directory
root@ip-172-31-84-27:/home/ubuntu#
```

Hard link: A Hard link acts as a copy (mirrored) of the selected file. It accesses the data available in the original file. If the earlier selected file is deleted, the [hard link](#) to the file will still contain the data of that file.

Example of hard link

In table.sh new.sh

```
root@ip-172-31-84-27:/home/ubuntu# ln table.sh new.sh
root@ip-172-31-84-27:/home/ubuntu#
```

ls -ltr

Hard link created

```
root@ip-172-31-84-27:/home/ubuntu# ls -ltr
total 32
-rw-r--r-- 2 root root 197 Sep 12 08:32 table.sh
-rw-r--r-- 2 root root 197 Sep 12 08:32 new.sh
```

cat table.sh

```
root@ip-172-31-84-27:/home/ubuntu# cat table.sh
#!/bin/bash

read -p "Enter the number: " num

echo $num
echo "Table is as follows:"

for (( i=1; i<=10; i++ ))
do
    val=$(( num * i ))
    echo "$num * $i = " $val
done

root@ip-172-31-84-27:/home/ubuntu#
```

cat new.sh

```

root@ip-172-31-84-27:/home/ubuntu# cat new.sh
#!/bin/bash

read -p "Enter the number: " num

echo $num
echo "Table is as follows:"

for (( i=1; i<=10; i++ ))
do
    val=$(( num * i ))
    echo "$num * $i = " $val
done

root@ip-172-31-84-27:/home/ubuntu#

```

If we remove, the original file, the hard link will still be there and content will also remain.

rm table.sh

```

root@ip-172-31-84-27:/home/ubuntu# rm table.sh
root@ip-172-31-84-27:/home/ubuntu#

```

ls -ltr

```

root@ip-172-31-84-27:/home/ubuntu# ls -ltr
total 32
-rw-r--r-- 1 root root 197 Sep 12 08:32 new.sh

```

cat new.sh

```

root@ip-172-31-84-27:/home/ubuntu# cat new.sh
#!/bin/bash

read -p "Enter the number: " num

echo $num
echo "Table is as follows:"

for (( i=1; i<=10; i++ ))
do
    val=$(( num * i ))
    echo "$num * $i = " $val
done

root@ip-172-31-84-27:/home/ubuntu#

```

8) Iptables: In linux, firewall is implemented by Netfilter(a kernel module which regulates the internet traffic) IPTables are interface to Netfilter.

Iptables is the primary firewall utility program developed for Linux systems. The program enables system administrators to define rules and policies for filtering network traffic.

iptables is a command-line utility for configuring the built-in Linux kernel firewall. It enables administrators to define chained rules that control incoming and outgoing network traffic.

The rules provide a robust security mechanism, defining which network packets can pass through and which should be blocked. iptables protects Linux systems from data breaches, unauthorized access, and other network security threats.

Administrators use iptables to enforce network security policies and protect a Linux system from various network-based attacks.

9) Swap Memory: Swap Memory is the area where inused or inactive data of RAM is placed. System still functions if RAM is full and does not crash.

Swap memory, often referred to as swap space, is an extension of a computer's physical RAM residing on the hard drive or Solid-State Drive (SSD). When the OS exhausts its available RAM, it swaps data between RAM and the swap space. This mechanism, known as swapping, enhances memory management efficiency.

Swap memory is important for systems with restricted RAM or those executing memory-intensive tasks. Without swap memory, these systems are susceptible to crashing when RAM capacity is exceeded.

Types of Swap Memory

There are two types of swap memory:

- Swap partition. Temporary storage space used when physical memory becomes fully utilized.

- Swap file. Physical disk storage used to expand the swap space of available memory.

10) Cron Job: The cron utility is used for running scripts and commands at regular intervals, and at specific times and dates. It's built into most Linux distros, and provides a very useful way to schedule tasks on your server.

cron is an automation tool, so anything that you run on a regular basis can likely be switched over to a cron job. If you wanted to make regular daily backups, or restart a service once a week, cron can do that.

11) How storage is mounted in Unix.

Mounting can be a temporary or permanent operation, and it's typically performed by an administrator, either by logging in as the root user or by using the sudo command.

Syntax of mount Command

```
mount -t type device dir
```

Other forms:

```
mount [-l|-h|-V]
```

```
mount -a [-fFnrsvw] [-t fstype] [-O optlist]
```

```
mount [-fnrsvw] [-o options] device|dir
```

```
mount [-fnrsvw] [-t fstype] [-o options] device dir
```

These commands tell the Kernel to attach the filesystem found at device to the dir.

Note:

- If you leave the dir part of syntax it looks for a mount point in /etc/fstab.
- You can use --source or --target to avoid ambivalent interpretation.
- mount --target /mountpoint

- /etc/fstab usually contains information about which device is need to be mounted where.
- Most of the devices are indicated by files like /dev/sda4, etc. But it can be different for certain filesystems.

Displays information about file systems mounted

```
vivek@vivek-X556UQK:~$ sudo mount -l -t ext4
/dev/sda3 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
/dev/sda4 on /media/vivek type ext4 (rw,relatime,data=ordered)
/dev/sda5 on /media/vivek type ext4 (rw,relatime,data=ordered)
vivek@vivek-X556UQK:~$ sudo mount -l -t fuseblk
/dev/sda6 on /media/vivek type fuseblk (rw,relatime,user_id=0,group_id=0,allow_o
ther,blksize=4096)
vivek@vivek-X556UQK:~$
```

Mounts file systems

```
vivek@vivek-X556UQK:~$ sudo mount /dev/sda4 /media/vivek
vivek@vivek-X556UQK:~$ sudo mount /dev/sda5 /media/vivek
vivek@vivek-X556UQK:~$ sudo mount /dev/sda6 /media/vivek
vivek@vivek-X556UQK:~$ sudo mount -l -t ext4
/dev/sda3 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
/dev/sda4 on /media/vivek type ext4 (rw,relatime,data=ordered)
/dev/sda5 on /media/vivek type ext4 (rw,relatime,data=ordered)
vivek@vivek-X556UQK:~$ sudo mount -l -t fuseblk
/dev/sda6 on /media/vivek type fuseblk (rw,relatime,user_id=0,group_id=0,allow_o
ther,blksize=4096)
vivek@vivek-X556UQK:~$
```

Displays version information

```
vivek@vivek-X556UQK:~$ sudo mount -V
mount from util-linux 2.27.1 (libmount 2.27.0: selinux, assert, debug)
vivek@vivek-X556UQK:~$
```

Unmounts file systems

```
vivek@vivek-X556UQK:~$ sudo umount /dev/sda6
vivek@vivek-X556UQK:~$ sudo umount /dev/sda5
vivek@vivek-X556UQK:~$ sudo umount /dev/sda4
vivek@vivek-X556UQK:~$ sudo mount -l -t fuseblk
vivek@vivek-X556UQK:~$ sudo mount -l -t ext4
/dev/sda3 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
vivek@vivek-X556UQK:~$
```