**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.**

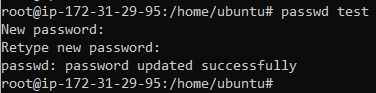
**Source**

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



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)



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)



Home directory created and verified

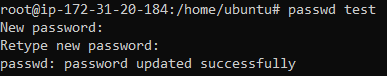
**Destination**

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



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)



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)



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)



**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

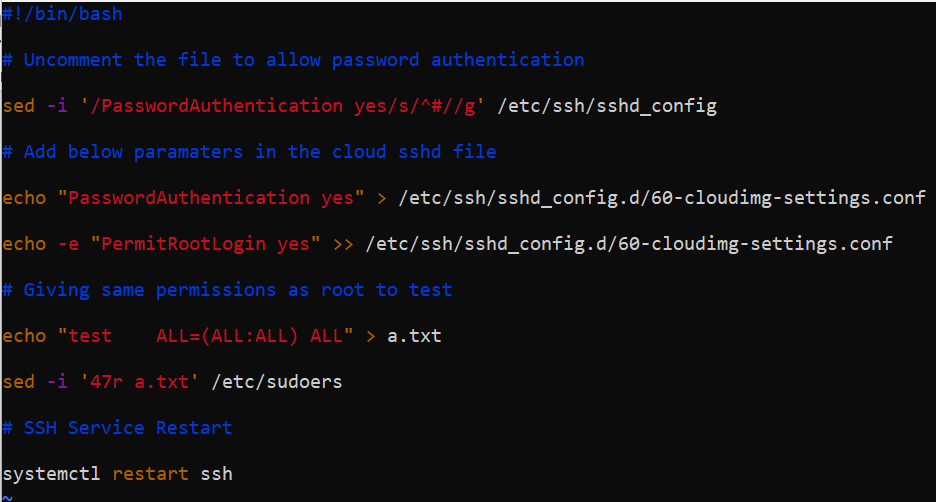
# Giving same permissions as root to test

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

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

# SSH Service Restart

systemctl restart ssh



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

bash rootlevel.sh



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)



**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

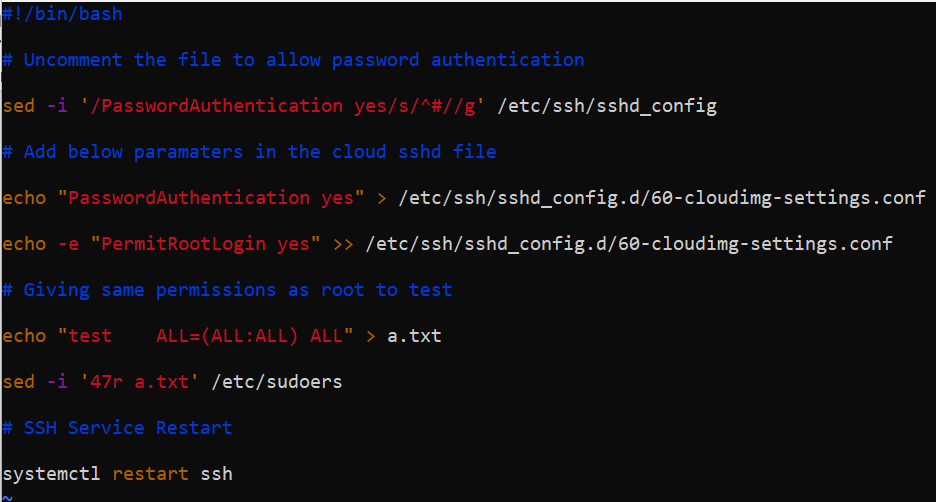
# Giving same permissions as root to test

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

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

# SSH Service Restart

systemctl restart ssh



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

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)



* **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)



* **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)



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

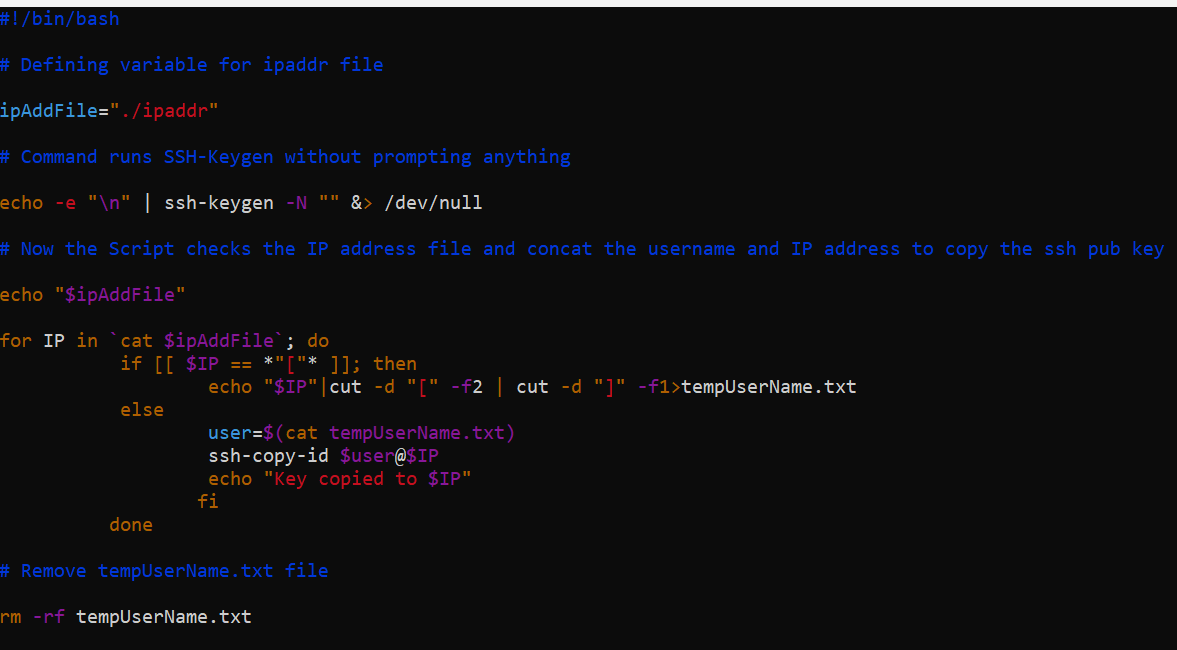
echo "Key copied to $IP"

fi

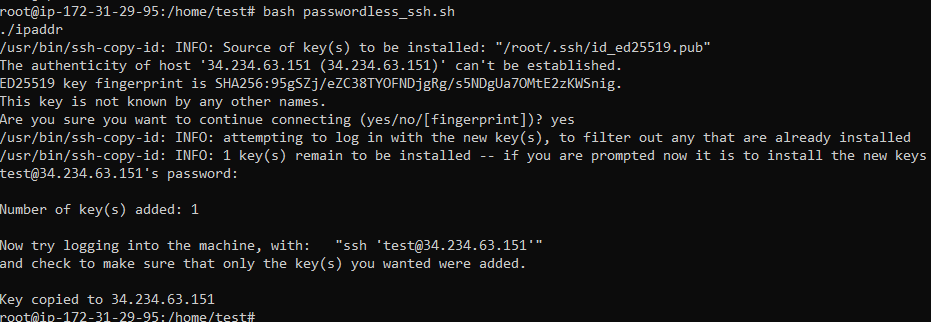
done

# Remove tempUserName.txt file

rm -rf tempUserName.txt



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

bash passwordless\_ssh.sh 

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**

1. **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



Below are script contents

#!/bin/bash

read -p "Enter the number: " num

echo $num

max=$(($num + 100))

echo $max

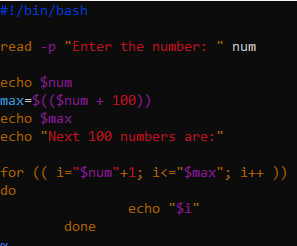
echo "Next 100 numbers are:"

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

do

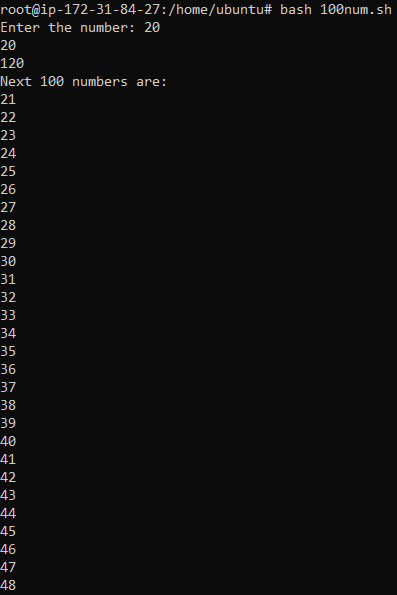
echo "$i"

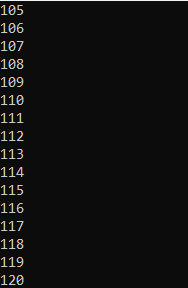
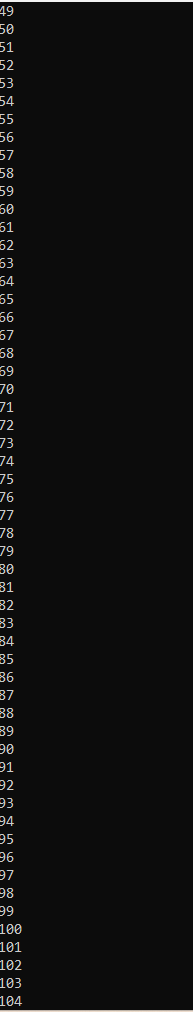
done



Run the script as bash

bash 100num.sh





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

vi table.sh



Below are script contents

#!/bin/bash

read -p "Enter the number: " num

echo $num

echo "Table is as follows:"

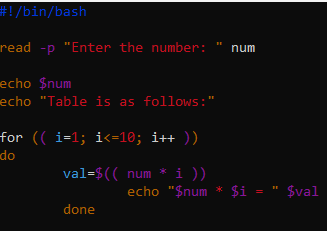
for (( i=1; i<=10; i++ ))

do

val=$(( num \* i ))

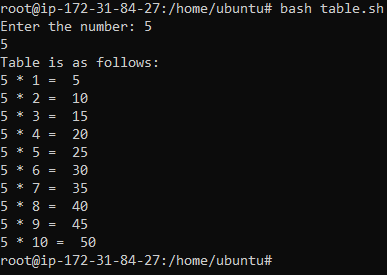
echo "$num \* $i = " $val

done

****

Run the script as bash

bash table.sh

****

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

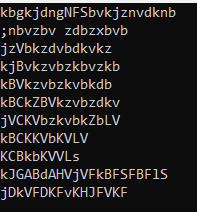
For this task, we will create file name test.txt with random content in it.

File is created under the path /home/ubuntu

vi test.txt

****

Contents inside the test.txt file

****

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

vi lines.sh



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"

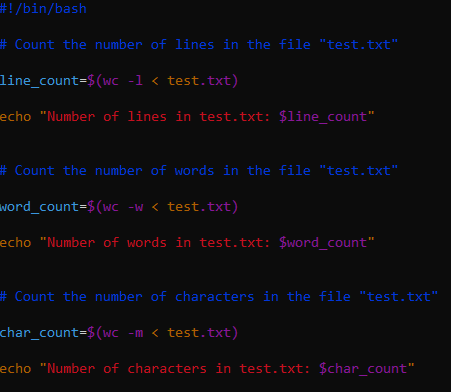
word\_count=$(wc -w < test.txt)

echo "Number of words in test.txt: $word\_count"

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

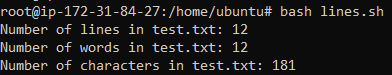
char\_count=$(wc -m < test.txt)

echo "Number of characters in test.txt: $char\_count"

****

Run the script as bash

bash lines.sh



1. **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



cd scripts/



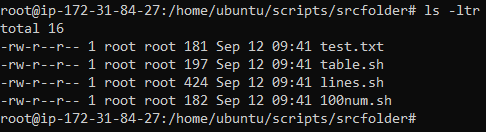
mkdir srcfolder

****

Copying files into the source folder

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



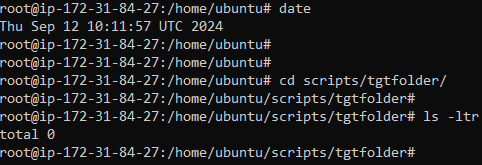
****

**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

****

**Nothing in the tgt folder as of now.**

****

**Writing the script for the first part of archiving job**

vi archive.sh

****

Contents of the script

#!/bin/bash

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

srcPath=/home/ubuntu/scripts/srcfolder

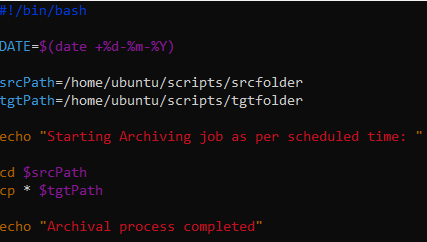
tgtPath=/home/ubuntu/scripts/tgtfolder

echo "Starting Archiving job as per scheduled time: "

cd $srcPath

cp \* $tgtPath

echo "Archival process completed"

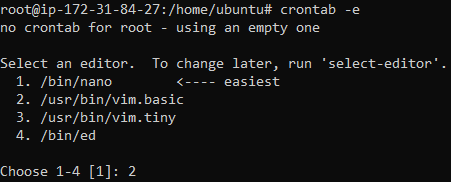
****

**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**

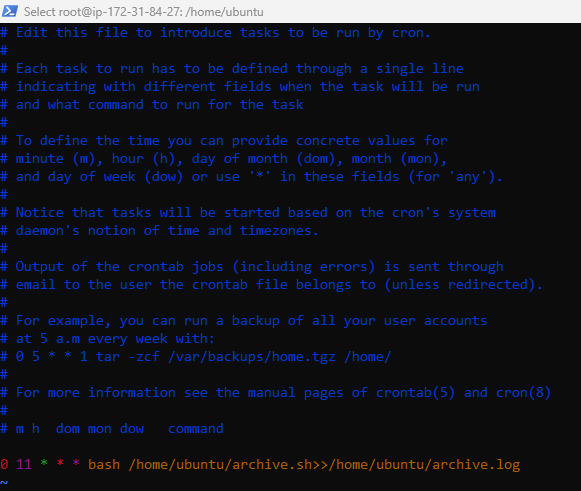
crontab -e

We choose option 2 to edit the file in Vim

****

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

****

**Output:**

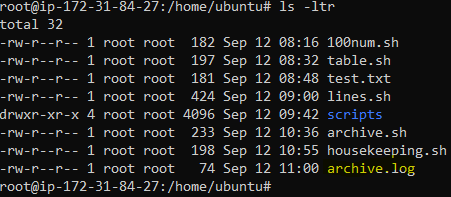
**Time is 11:00, we will verify the time and see if the log file is generated and cronjob ran successfully and also check the output.**

date

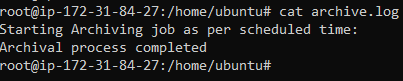
****

ls -ltr

archive.log is created

****

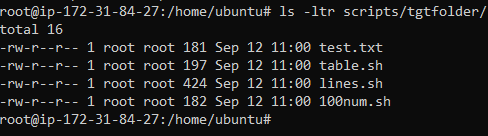
cat archive.log

****

We will verify whether the contents are copied to tgtfolder

ls -ltr scripts/tgtfolder

All the files have been copied from srcfolder to tgtfolder

****

**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



Contents of the script

#!/bin/bash

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

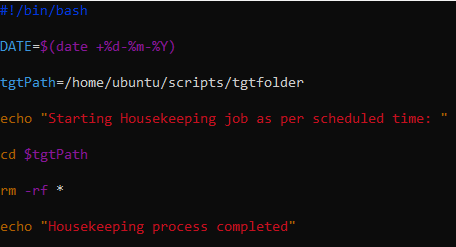
tgtPath=/home/ubuntu/scripts/tgtfolder

echo "Starting Housekeeping job as per scheduled time: "

cd $tgtPath

rm -rf \*

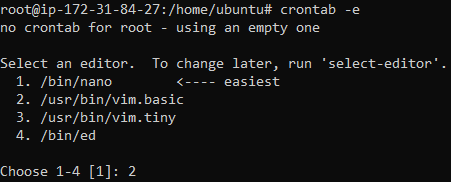
echo "Housekeeping process completed"



**Now we will setup the script as cron job**

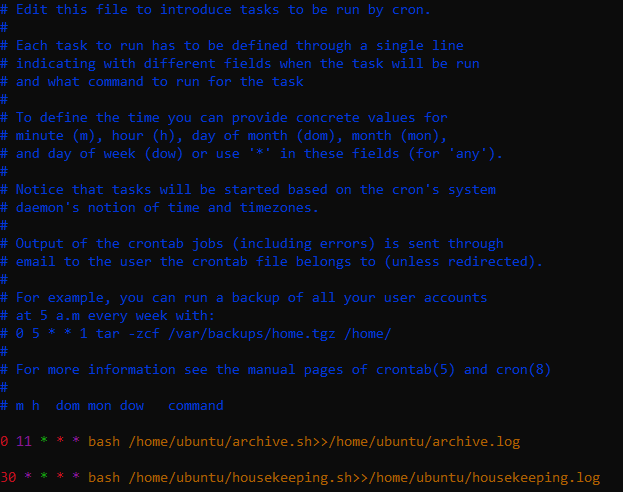
crontab -e

We choose option 2 to edit the file in Vim

****

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

****

**Output**

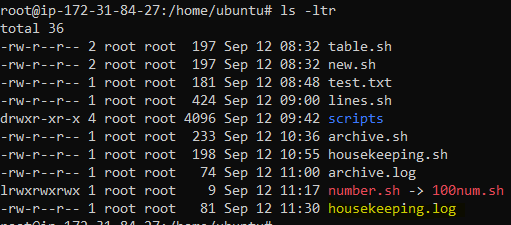
**Time is 11:30, we will verify the time and see if the log file is generated and cronjob ran successfully and also check the output.**

date

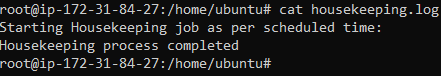
****

ls -ltr

housekeeping.log is created

****

cat housekeeping.log

****

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.

****

**Second part of script is also completed successfully.**

Task is completed successfully.

1. **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.

1. **Soft link and Hard Link**

**Soft Link:** A [soft link](https://www.geeksforgeeks.org/soft-hard-links-unixlinux/) (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.

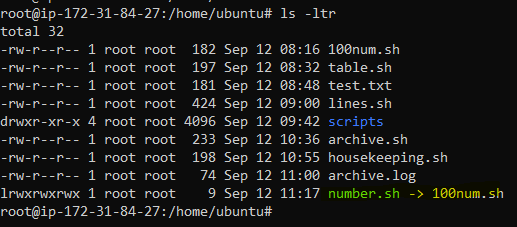
**Example of soft link**

ln -s 100num.sh number.sh

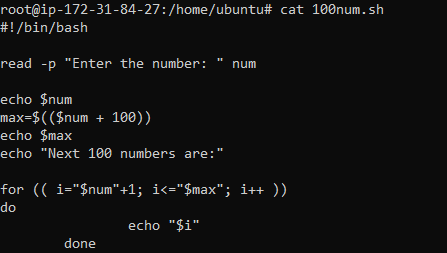
****

ls-ltr

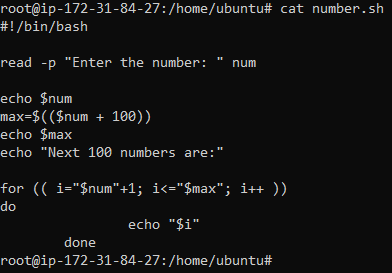
soft link created



cat 100num.sh



cat number.sh

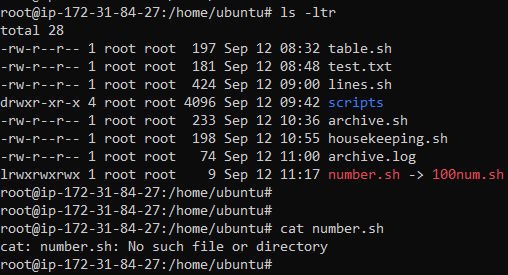
****

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

rm 100num.sh

****

ls-ltr



cat number.sh

****

**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](https://www.geeksforgeeks.org/soft-hard-links-unixlinux/)to the file will still contain the data of that file.

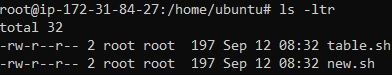
**Example of hard link**

ln table.sh new.sh

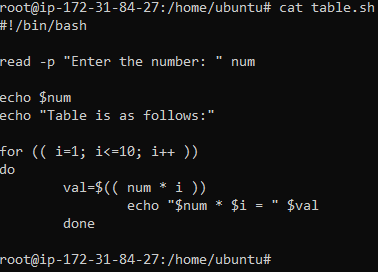
****

ls -ltr

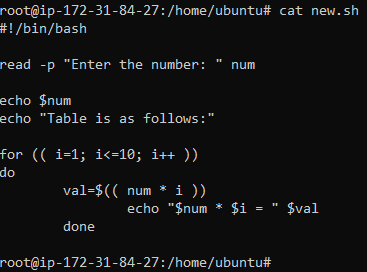
Hard link created

****

cat table.sh

****

cat new.sh

****

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

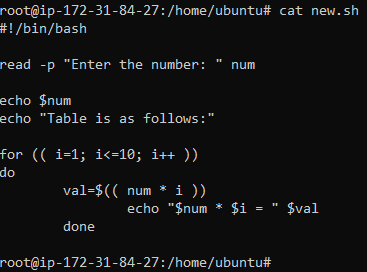
rm table.sh

****

ls -ltr

****

cat new.sh

****

1. **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.

1. **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.

1. **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.

1. **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

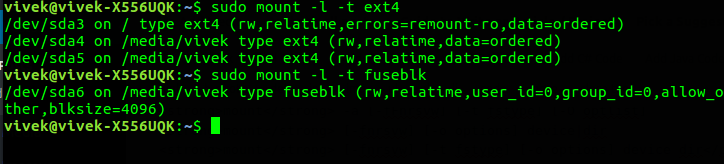
mount [-fnrsvw] [-t fstype] [-o options] device dir

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

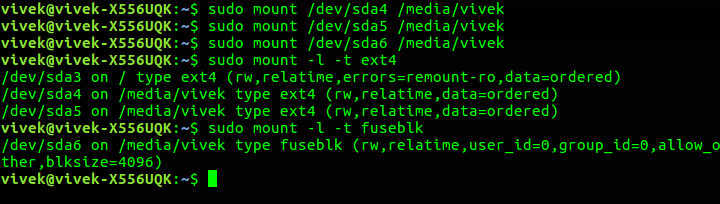
**Note:**

* 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.

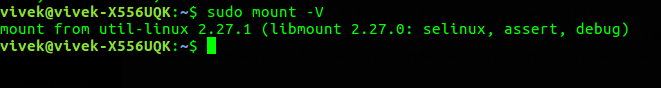
**Displays information about file systems mounted**



**Mounts file systems**



**Displays version information**



**Unmounts file systems**

