

DevOps and Build Release Engineer

- Linux
- Shell Script
- CI tool
- Jenkins
- AWS
- Bug Tracker
- Ansible

Download MobaXterm software and practice Linux Commands.

- Google Mobaxterm->get now->home edition->Installer Edition.

Linux

- Ls – List files and directories.
- Ls -l – list files and directories with long format
- Ls -lt – Long list format with time
- Ls -lrt – reverse of ls -lt.
- Pwd – present working directory
- Mkdir – creating a new directory
- Touch – used to create new empty file
- Cd – change directory
- Ls -a – list the hidden files
- Cd .. –to come out of present directory
- Cd ../.. – if you want to come out of 2 or more directories.
- Cd - : acts as a recall between last 2 commands
- Cd /temp1/temp2/ - changes to one directory to last directory
- Cat – used to display the content of a file on console
- Vi – used to edit a file on vi editor
 - Esc I – insert mode in vi editor.
 - Esc :wq! – save and quit the file. W-write,q-quit,!-forcefully.
 - Esc q! – quit without saving the content.
 - Esc :set number/ esc :setnu – which sets the number for a file.
 - Esc :setnonu – used to remove the line numbers for a file.
 - Replace a string
 - Esc :%s/string1/string2/g
 - G is not mandatory, g means globally.
 - Esc :%s/linux/docs/g
 - S is substitute
 - Esc :1 s/linux/docs/g – changes in 1st line only
 - Esc :1,5 s/linux/docs/g –changes 1 to 5th line.

- Esc :7,\$ s/linux/docs/g – changes 7th to end of line.
- Esc :1 s/linux/docs/1 – changes 1st line 1st occurrence only.
- Search a pattern/string
Esc:/pattern – it will move to 1st occurrence then keep pressing n to 2nd and so on
- Delete the whole line
Esc:dd
- Cut and paste in a file
Cut -d(delimiter) -f(colname) filename
Example: cut -d- -f2 f1.txt → it removes – and prints 2nd column in a file.
cut -d- -f2,3 f1.txt
cut -d- -f2-5 f1.txt
echo “HELLO” | cut -c 3
paste-displays lines of multiple files line by line
paste f1.txt f2.txt
paste -d: f1.txt f2.txt
paste -s f1.txt f2.txt

• Commands

- Cp command
Is used to copy content of a file from one to another file
If dest file doesnt exists, it will create and copy it.if already exists then it will override the content.
Cp src dest
Cp file1 file2
Cp -r temp1 temp2-whole temp1 info ll copied to temp2 recursively.
- Mv command
The source will get deleted and moved to destination. Or can be called as rename.
Mv file1 file2
Mv file dir/
Mv *.txt dir/
- Chmod command
Used to change permission of a file or directory
-rwxr--r--
-=file or directory or link, rwx=user,r--=group,r--=others
R=read,w=write,x=execute
It will be in binary format
Chmod 777 filename – all permission to a file
Chmod -R 777 dir – all permission to a dir
Remove or add permission
Chmod u+w file –user and write permission to a file
Group and other full permission

Chmod g+rxw,o+rxw file

Chmod g-w,o-w file

Chmod a+rxw file

- Chown command

Change ownership of a file or dir

Chown newowner filename

Chown -R newown dirname

- Chgrp command

chgrp hope file.txt

- Shells

bash rc, kshell, cshell

to change the shell use,

chsh

used to show current shell, echo \$SHELL.

- Wc command

Used to count number of lines, words, characters in a file

Wc filename

Wc -l filename, wc -w filename, wc -c filename

- Du and df command

Du-disk usage or size of a file, df-disk free / size of drive

Du -sh filename

Du -sh *

Df -h .

Df -h

- Echo command

Print statement

Echo "PRADEEP"

Echo -e "HI \n HELLO"

- Redirect and append command

Redirect (>)-used to write output of a command to a file, if not exists it will create new and write it . if already present it will override.

Echo "hi how r u" > text

Cat text

Ls -lrt > text

Cat text

Du -sh * > size

Cat size

Append (>>) – used to attach the content or output of a command to end of a file. It will not override the existing content of a file.

Echo "PRADEEP" >> log

Cat log

Echo "DEEPU" >> log

Cat log

- Grep command

Is used to search pattern/string in a file.

Grep "pattern" filename

Grep -w "pattern" filename – used to search only word

Grep "pattern" *

Grep -l "pattern" * - prints only file names which pattern is present

Grep -w -l "pattern" *

Grep -R -l "pattern" * or rgrep

Grep -c "file" * - used to count number of lines.

Grep -e "pattern1" -e "pattern2" ---so on- used to search multiple patterns

Also u can use egrep

Grep "File" * case sensitive

Grep -I "file" * case insensitive

Grep "^d" filename –used to print all lines start with d

Grep "\$" filename – used to search all files which ends with pattern 2

Grep -v "docs" filename- print all lines except docs

- Pipe filters (|)

Pipe is used to give output of a one command as input to next command

Ls | grep "^t" | wc -l

- Head and tail command

Head -Is used to display 1st part of a file, by head filename -> by default it display 10 lines.

Head -10 filename

Tail – used to display last portions of a file

Tail -2 filename

Recently modified files

Ls -lrt | tail -1 or ls -lt | head -1

Head -99 filename | tail -1 – used to print 99th line of a file

Head -5 file | tail -3 – used to print 2nd to 5th line of file

Head -1 filename;tail -1 filename – used to club commands but commands runs independent, to display 1st line and last line of a file.

Head -99 filename | tail -1 | wc -w –count number of words in 99th line of a file

- More and less command

More filename – used to display content of a file page wise, can scroll down but cant up

Less filename-used to display content page wise, we can scroll up and down.

- Sed command

Is used to edit file without opening it, need to replace string in a file.

Sed 's/file/docs/g' filename

Need to replace in a 1st line

Sed '1 s/file/dcs/g' filename

Need to replace string from 3 to 5 line

Sed '3,5/fil/dcs/g' filename

Need to replace string 7 to end of line

Sed '7,\$ s/file/dcs/g' filename

If u need to change in original files,then

Sed -I 's/fil/dcs/g' filename

-i=insert to a file/modify to file, if you don't use it wont affect in original file.

Print only 5 or 99th line

Sed -n '5 p' filename

Sed -n '99 p' filename

Print 4th to 9th line

Sed -n '4,9 p' filename

Print 5th to end of line

Sed -n '5,\$ p' filename

Delete a line from a file using a sed

Sed '4 d' filename

Sed '99 d' filename

Sed -I '4 d' filename

- Awk command

Awk -F " " '{print \$2}' filename

Is used to cut content of a file column wise and row wise.

Ls -lrt | awk -F " " '{print \$2,\$3}' -2nd and 3rd col

Ls -lrt | awk -F " " '{print \$NF}' -last col

Ls -lrt | awk -F " " '{print \$(NF-1)}' -2nd last col

- Find command

Is used to find the location of a file/dir. Find is a automatic recursive, it searches in sub-directories automatically.

Find. -name "filename"

Find . -name "log" - . is pwd

Find /home/ -name "log"

Find -iname "log"

Find /home/mobaxterm -iname "log"

Find . -iname -type f "log"

Find . -type f -iname "log"

Find . -iname -type d "log"

Find . -type d -iname "log"

-mtime -is last modified time

Find . -type f -mtime +90

Find . -type d -mtime +90

Find . -mtime +90

Find . -type f -mtime -90

Find . -type d -mtime -90

Find . -mtime -90

-mmin -is last modified time

Find . -type f -mmin -9
Find . -type f -mmin +90
Find . -type d -mmin -90
Find . -type d -mmin +90
Empty and non-empty
Find . -type f -empty
Find . -type d -empty
Find . -empty
Find . -type h -empty :h or L is a link
Find the permission of a files
Find . -type f -perm 0777
Find . -type d -perm 0644
Find . -perm 0754
Nonempty
Find . -type f -not -empty
Find . -type f ! -empty

- Maxdepth command

How do you restrict recursive search for the dir search in find command

Find . -iname "test" -maxdepth 2

- Rm command

Used to remove a file-> rm filename

Rm -rf -> used to remove dir

- Xargs

Is used to pass as a args to a next command

Output of one command will be passing as args to next command.

how to find files which are modified 90 days back and del that.

Find . -type f -mtime +90 | xargs rm

- Exec command

Find . -type f -mtime +3 -exec

Is used to replace the current shell with the command without spawning a new process and used to assign file descriptor to file.

- Soft link and hard link

Is a shortcut to a file, if you make any changes in original file it gets reflected in the link, if I delete original file soft link wont work.

Ln -s filename softlinkname

U can create a file or dir

Hard link

Ln filename hardlinkname

Is a shortcut to a file, if you make any changes in original file it gets reflected in the link, and if I delete the original file, hard link wil work because it points to inode of a file.

- Umask

Is used to set default permission on a system. It is reverse of chmod.

Umask 000 –who creates the file on a system will have all permissions.

Umask 777 means- no permission to a file

- Ssh

It is used to connect to remote server

Ssh username@server2

Password:

Hostname

Ssh uses port 22

- Scp

Is used to copy file or dir from server to another server.

Scp file [username@server2:/home/../../](#)

Password:

Scp -r dir1 [username@server2:/home/../../](#)

Scp -p file1 [username@server2:/home/../../](#)

P is preserved permission as same

Scp -rp dir [username@server2:/home/../../](#)

-rp for dir permission as same

- Ping

Is used to check whether the server is up and down

Ping ip or hostname

- telnet

it uses port 23

is used to authenticate account credentials on remote server

used to break firewall

telnet ip or hostname

- rsync

is used to copy files from one server to another server and also within the server, while copying the data if copy is stopped due to network issues in between, if I use 'scp' it will start copying from begin

if I use rsync it will start copying where it is stopped.

Rsync file [user@server2:/home/test/../../](#)

If you want to copy in server

Rsync src dest

- size command

find . -size +0 -shows empty file

find . -size +1 - shows nonempty file

- su

super user

su -root or su -

used to login as root user

root has highest permission

- sudo
superuser does
is used to run command with root permissions
sudo username
- who
is used to check who are all logged into system
who
who | wc -l
- whoami
is used to check, who you logged in name
whoami
- uname
to check linux version
- uname -a
complete information about linux
- uname -v
version and time of linux
- ps
used to list current running process on a system
ps -ef | grep "system32"
- kill
used to stop process
-9 is a special signal to terminate that process
Kill -9 PID
Killall -9 pname
Sudo service servername/Pname stop
Sudo service servername/Pname start
Sudo service servername/Pname restart
Pkill -9 -u "username" – stop process started by particular user
Ps -u username or ps -ef | grep "system32" – list process started by particular user
How do you run script or command in a background we need to se & at the end of the command.
How do you run process in foreground
Fg PID
- sort
is used to sort the data
sort filename
sort -r filename
- uniq
used to print only unique values (it removes duplicate)
it will work only when the contents/data are consecutive values/info/data
uniq will be always used with the sort

uniq filename

cat file | sort | uniq

- tee

used to write output of a command to a file as well as display on console/prompt

cat log | tee log1

ls -lrt | tee -a log → used to append to log

- .profile / .bashrc

Vi .bashrc

Echo "hello gm"

Pwd

Esc:wq!

It is an auto execute file which gets executed automatically as soon as you logged into the system.

- Netstat

Used to check free ports on a system.

List all listening ports tcp or udp

Netstat -a

Netstat -at

Netstat -au

Netstat -lt

Netstat -lu

- Mount

Attaches a filesystem, located on same device or other to a file tree.

Mount -t type device dir

Unmounts-detaches the specified file system from file hierarchy.

AWS

- Search AWS, signup and get registered
 - ->personal edition->basic. Pay 2rs and access for 1year.
- Download and install Putty software 64bit.
 - Is a software which used to connect linux to a windows.
- Create aws instance/server-
 - Login aws->services->EC2->launch instance->red hat linux 64bit->next->add storage->give the value and test->add tags, http https ssh and required tags give custom to anywhere for all in security->review and launch instance.
 - When you launching 1st time, give as create new key pair and download the key pair , that ll be in .pem format, just download once for all instance creation, no need of downloading twice. Then in the same window use existing key pair and ack the check box and then launch->view instance.
- Download puttygen for windows inorder to convert .pem file to .ppk file.
 - Open puttygen->load the .pem file then ->save as private key->save it in .ppk format.

Var1=123

Var2="pradi hello"

Echo "my name: \$var"

Echo "my no: \$var1"

Echo "my names: \$var2"

- Command line args

#!/bin/bash

Echo "my name is:\$1"

Echo "my city:\$2"

Echo "my filename:\$0"

Sh Ex3.sh pradeep shimoga

\$1 is 1st args,\$2 is 2nd args,\$0 is name of script, it will take \$0---\$9 after that
\${10},{11}...

- If statement (Conditional statement)
- If []

Then

Statement

Fi

- If [];then

Statement

Else

Statement

Fi

- If [];then

Statement

Elif [];then

Statement

Else

Statement

Fi

- Was(write a script) to check number is 5 or not.

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

```
If [ $1 -eq 5 ];then
```

```
Echo "$1 is five"
```

```
Else
```

```
Echo "$1 is not five"
```

```
Fi
```

```
Sh ex.sh 5, sh ex.sh 2
```

- Was to find biggest of 2 no's

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

```
If [ $1 -gt $2 ];then
```

```
Echo "$1 is big"
```

```
Else
```

```
Echo "$2 is big"
```

```
Fi
```

```
./big.sh 1 2
```

- Condition

```
-eq=equal
```

```
-gt=greater than
```

```
-ge=greater than or equal to
```

```
-lt=lesser than
```

```
-le=lesser than or equal to
```

```
-ne=not equal
```

- Biggest of 3 nos

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

```
If [ $1 -gt $2 ] && [ $1 -gt $3 ]
```

```
Then
```

```
Echo "$1 is big"
```

```
elif [ $2 -gt $2 ] && [ $2 -gt $3 ]
```

```
Then
```

```
Echo "$2 is big"
```

```
Else echo "$3 is big"
```

```
Fi
```

```
./ex.sh 1 2 3
```

- How to make a shell script file into a command

```
Cp ex.sh ex
```

```
Cp ex /usr/bin/ or cp ex /bin/
```

```
Ex 1 2 3
```

- Notations

\$#-to display total number of args

\$*-all args passed to a scripts

\$?-status of last executed command

If it is 0, success. Else failure (1,2,)

\$\$-display the present PID.

\$@-all args passed to a script are stored in array format.

\$_-PID of current running process, which went into background.

#-is used to commenting a code.

&&-AND,||-OR,!-NOT.

- Restrict to pass two args

```
If [ $1 -ne 2 ];then echo "pass only 2 args";exit 1;fi
```

- Was to check whether the args is file/dir/link.

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

```
Echo "enter the name of file/dir/link"
```

```
Read name
```

```
If [ -f $name ];then echo "$name is a file"
```

```
elif [ -d $name ];then echo "$name is a dir"
```

```
elif [ -L $name ];then echo "$name is a link"
```

```
else echo "$name doesn't exists"
```

```
fi
```

- Notations
-f=file,-d=dir,-L or -h=link,-e=exist or not,-r=read permission,-w=write,
-e=execute, readlink soft1=used to read links and display the path of the file or link.
- Was to check whether the args is file/dir/link and display the entered file content
#!/bin/bash
Echo "enter the name of file/dir/link"
Read name
If [-f \$name];then echo "\$name is a file";cat \$name
elif [-d \$name];then echo "\$name is a dir" ;cat \$name
elif [-L \$name];then echo "\$name is a link" ;cat \$name
else echo "\$name doesn't exists"
fi
- While loop
While []
Do
.....
Done
- Was to print entered number to 1
#!/bin/bash
N=\$1
While [\$n -gt 0];do
Echo "\$n"
N=`expr \$n - 1`
Done
``=backquote(below esc key)
- Was to find factorial of a given number
#!/bin/bash
Num=\$1
Fact=1
While [\$num -gt 1]
Do
Fact=`expr \$fact * \$num`
Num=`expr \$num - 1`
Done
Echo "factorial of a number is: \$fact"
- Was to add 2 numbers
#!/bin/bash
N=`expr \$1 + \$2`
Echo "\$n"
- Was to subtract from a bigger number to smaller number and print result.
#!/bin/bash

```

If [ $1 -gt $2 ];then
Sub=`expr $1 - $2`
Else
Sub=`expr $2-$1`
Fi
Echo "$sub"

```

- Was to add given number

```

#!/bin/bash
Num=$1
Add=0
While [ $num -gt 0 ]
Do
Add=`expr $add + $num`
Num=`expr $num - 1`
Done
Echo "$add"

```

- While loop to read file line by line

```

While read line
Do
Echo "$line"
Do < filename/$1
If there are 4 lines in a file, it will executed 4 times in a loop.

```

- Was to count number of words in each lone of a file

```

#!/bin/bash
Num=1
While read line
Do
words=`Echo $line | wc -w`
echo "$num lines and $words word"
num=`expr $num + 1`
done < $1 or filename

```

- Was to print names if there age is more than 40 in the data table.

U can use "AWK command as shorthand"

```
Awk -F " " '$2>40 {print $2}'
```

=====or=====

```

#!/bin/bash
Sed '1 d' $1 > temp
While read line
Do
Age=`echo "$line" | awk -F " " '{print $3}'`
If [ $age -gt 40 ]
Then
Echo $line | awk -F " " '{print $2}'

```

Fi

Done < \$1

- Was to reverse a content of file into reverse order.

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

```
File=$1
```

```
Lines=`cat $file | wc -l`
```

```
While [ $lines -gt 0 ]
```

```
Do
```

```
    Head -$lines $file | tail -1 >> reverse
```

```
Lines=`expr $lines - 1`
```

```
Done
```

```
Cat reverse
```

```
Rm reverse
```

```
=====or=====
```

```
Tac filename (reverse of cat)
```

- Was to reverse a word

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

```
Str="hello"
```

```
Len=`expr $str | wc -c`
```

```
Len=`expr $len - 1`
```

```
Rev=""
```

```
While [ $len -gt 0 ]
```

```
Do
```

```
    Rev1=`echo $str | cut -c$len`
```

```
    Rev=$rev$rev1
```

```
Len=`expr $len - 1`
```

```
Done
```

```
Echo "$rev"
```

```
-----or-----
```

```
Echo "pradi" | rev
```

- For loop

```
For I in 1 2 3 4 5
```

```
Do
```

```
    Echo "$i"
```

```
Done
```

- Was to add n numbers

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

```
Var="2 3 4 5"
```

```
Sum=0
```

```
For I in $var
```

```
Do
```

```
    Sum=`expr $sum + $i`
```

```
Done
```


Echo "\$sum"

- Was to find biggest of n numbers

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

```
Big=$1
```

```
For I in $*
```

```
Do
```

```
    If [ $big -lt $i ]
```

```
Then
```

```
    Big=$i
```

```
Fi
```

```
Done
```

```
Echo "big is $big"
```

- Set -x

Used to debug a shell script

- Was to find smallest of n numbers

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

```
Set -x
```

```
Small=$1
```

```
For I in $*
```

```
Do
```

```
    If [ $small -gt $i ]
```

```
Then
```

```
    Small=$i
```

```
Fi
```

```
Done
```

```
Echo "small is $small"
```

- Top

Whether each process is taking how much cpu usage and memory.

- Uptime

Is used to check from how long the system is running and also used to check load average.

- Load average

Load is a measurement work of system performing, this measurement displayed as numbeeer. A complete idle system or pc is numbered as 0. Each running process for cpu resource adds 1 to load average.

- Command used to list only files and directories

```
Ls -lrt | grep "^d" ---for directories
```

```
Ls -lrt | grep -v "^d" ---for directories
```

```
Find . -type d -ls --- dir
```

```
Find . -type f -ls ---files
```

```
Find . -maxdepth 1 -type d
```

```
Find . -maxdepth 1 -type f
```

- Special characters

Have special meaning to a shell

To escape special meaning or special characters we use backslash (\) to special characters.

Special characters like \$, \$#, \$!, \$*, \, *, ^, ` and etc

- Was to find factorial of a given any number in command line args.

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

```
For I in $*
```

```
Do
```

```
    Fact=1
```

```
    Num=$i
```

```
    While [ $num -gt 1 ];do
```

```
        Fact=`expr $fact \* $num`
```

```
        Num=`expr $num -1`
```

```
    Done
```

```
    Echo "$fact"
```

```
Done
```

- Crontab (vv imp)

Is a scheduler, which is used to schedule scripts on linux server.

Crontab -e ---is used to edit cron job

Crontab -l ---is used to list cron jobs

```
* * * * * /script path
```

Mins |hrs |date |month |day of week

- Schedule to run the script on specified date and time

- 12th june at 10 am mon

```
00 10 12 06 01 /script path
```

- 13th june at 4 pm tue

```
00 16 13 06 02 /script path
```

- Everyday at 2pm

```
00 14 * * * /script path
```

- 03.30 pm only on tue

```
30 14 * * 02 /script path
```

- Mon-fri at 4pm

```
00 16 * * 01-05 /script path
```

- Mon and Tuesday at 5.30pm

```
30 17 * * 01,02 /script path
```

- Mon every one hour

```
00 * * * 01 /script path
```

- Mon every 30 mins

```
*/30 * * * 01 /script path
```

- Mon at 4pm, script should run at every 5min

```
*/5 16 * * 01 /script path
```

- 05.15pm one the friday

```
15 17 * * 05 /script path
```

- Sat 10pm
00 22 * * 06 /script path
- Every month 1st at 11 am
00 11 01 * * /script path
- If ec2 doesn't consists crontab, then
Sudo yum instal cron
-----or-----
Sudo yum install update
And to check crontab use,
which crontab
then,
/home/...//big3.sh > log1
Crontab -e
*****/script path
Crontab -l
- Need to set a crontab every month end at 11 am
00 23 * * * [`date +%d` -eq `echo `cal` | awk '{print \$NF}'`] &&
job.sh

- Example realistic scripts!!!!!!!!!!!! Vv imp!!!!!!
- Builds(different compilation or developing files) were failing due to memory issue it was giving an error like “no space on disk”. To resolve this problem/issue, I have written a script to check disk memory and send a email notification, saying that memory is 90%, and please take appropriate action.

Vi memch.sh

#!/bin/bash

Mem=`df -h . | tail -1 | awk -F " " '{print \$4}' | sed 's%/g'`

If [\$mem -gt 40]

Mail -s “memory reached” -c pradi@gmail.com pradeep@gmail.com < filename

Fi

-----filename at the end of mail line is,

You can write a body of mail in that file can read to the mail command.

Else

U can try like,

Echo “hi memory reached” | Mail -s “memory reached” -c pradi@gmail.com pradeep@gmail.com

To run memch.sh for every min,

Crontab -e

* * * * * /memch.sh

- Mail command
Is used to email the file or information to the specified email.

Mail -s "memory reached" -c pradi@gmail.com pradeep@gmail.com < filename

Echo "hi memory reached" | Mail -s "memory reached" -c pradi@gmail.com pradeep@gmail.com

Mail -s "memory reached" "body of mail" -c pradi@gmail.com pradeep@gmail.com

- They wanted me to write a script to monitor the 'Tomcat' services on the server, if any service or process is topped accidentally by someone, we should get email notification saying that service is stopped and also script should restrict the service.-----realistic script example-----

Proch.sh

#!/bin/bash

Services="ser1 serv2 serv3 "

For service in \$services

Do

Ps -ef | grep "\$service"

If [\$? -ne 0]

Then

Mail -s "\$service is stopped and trying to restart"
pradi@gmail.com

Sudo service \$service restart

Fi

Done

Use crontab to run every min,

Crontab -e

* * * * * /proch.sh

- Case statement
Case \$var in
Value1) statement 1
;;
Value2) statement 2
;;
Value3) statement 3
;;
*) statement 4
;;
Esac
Was to check number from 1 to 3
#!/bin/bash
Case \$1 in
1)echo "this is one"

```

;;
2)echo "this is two"
;;
3)echo "this is three"
;;
*)echo "this is invalid"
;;
Esac
Was to do the following tasks,
Monday-copy the file
Tuesday-rename a file
Wed-backup
Thu-delete a file
Fri-del a file
Sat and sun-create a file
#!/bin/bash
Case $1 in
Mon)echo "copy a file"
Cp file1 file2
;;
tue)echo "rename a file"
mv file1 file2
;;
wed)echo "backup a file"
mv file1 file2
;;
Thu|fri)echo "remove a file"
rm file1
;;
Sat|sun)echo "create a file"
Vi file1
;;
*)echo "invalid input"
;;
Esac
Was to display menu based script for following,
1.search word
2.edit file
3.create softlink
4.find location of file or dir
5.checkname
6.exit
#!/bin/bash
Echo "press number to enter into menu:"

```

```

Echo -e "1.search word\n
2.edit file\n
3.create softlink\n
4.find location of file or dir\n
5.checkname\n
6.exit\n"
Echo "select option you want to perform:"
Read opt
Case $opt in
1)echo "enter word"
Read word
Grep -l "$word" * > log
If [ $? -eq 0 ] then
Echo "$word is found"
Cat log
Else
Echo "$word is not found"
Fi
;;
2)echo "enter file you want to edit"
Read file
If [ -f file ];then
Echo "$file is a file"
Vim $file
Elif [ -d $file ];then echo "$file is a dir"
Elif [ -L $file ];then echo "$file is a link"
Else echo "$file not exists"
Fi
;;
3)echo "enter filename to create a softlink"
Read file
If [ ! -f $file ];then echo "file doesn't exists"
Exit 1
Fi
Echo "enter softlink name"
Read link
If [ -L $link ];then
Echo "$link already exists"
Exit 1
Fi
Ln -s $file $link
;;
4)echo "enter file or dir name to find location"
Read name

```

```

Find . -iname "$name" > temp
If [ $? -eq 0 ];then
Echo "$name is in below loc"
Cat temp
Else
Echo "$name is not found"
Fi
;;
5) echo "check the file is a directory/file/link"
read name
if [ -f $name ];then echo "$name is a file"
elif [ -d $name ];then echo "$name is a dir"
elif [ -L $name ];then echo "$name is a link"
else echo "$name is doent exists"
fi
;;
*)exit 1
;;
Esac

```

- Environmental variables

Are variables can be exported to sub shell also, we use export command to export the variable.

Echo \$PATH

Echo \$SHELL

Export PATH="\$PATH ../../.." → ../../ path of script to added as env variables.

Then u can use .sh file as command.

GIT

- Used to track versions of files and drectories.
- Type
 - GIT,SVN,clearcase,TFS,perforce,CVS,Mercury
- Workspace
 - It's a olace where we edit modify project related files
- GIT Arch
 - Workspace → adding area→Commit area
- If I run git add files will be moved to staging area, it's a intermediate area, where we can save the changes.
- If I run git commit files will be added to git repository then we can track the file version.
- Mkdir git_repo
- Git init (used to create non-bare repo)
- Vi test1 (add some content)

- Git status (used to check whether files are workspace, staging area or in git repo)
- Git add test1 (this will move file from workspace to staging area)
- Git status (it will shows changes to be committed)
- Git commit -m " " (this will move files from staging area to git repo)
- Git status (this will show working directory clean)
- If It wont work use for 1st time to configure,
 - Git config -global user-email p@gmail.com
 - Git config -global user-name "p"
- Git commit -m " "
- Git status
- Git log (used to check the repo history)
- Git log filename (specified filename)
- Git log -2 (last 2 commits)
- Git log -2 filename (last 2 commits of the file)
- Git checkout commitId (used to switch to a previous version , used to switch to branch, also to switch to tags)
- Git checkout master (gives to the latest version)

Tags

- Tag is a name given to set of versions of files and directories. It indicates milestones of a project we can easily remember tags in the future. If I want good code in the future we tag it.
- Command to list tags is,
 - Git tag
- To create a tag,
 - Git tag tagname (it tags the latest version of code by default)

Branching

- Is a parallel development, two teams will work on same piece of code on two different branches, later they can integrate the changes by merging.
- Git branch - to list the branches
- Git branch branchname -to list branchnames
- If '*' is on the branch , when I run git branch branchname, branch will get created from the branch which has a '*'.
- To switch to a branchname
 - Git checkout branchname
- Git merge
 - Is used to integrate two branches
 - Git merge branchname
- Merging Conflicts

- Will occur when the same piece of code is changed on 2 different branches, when we try to merge those two branches, then merging conflict will occur,
- To resolve this issue, I don't know whose change should I take to merge, so I contact developers changes the code, person who modified code of branch1 and branch2. Then they will decide and tell us whose changes should I take into merge.
- Then I take that change and I commit it. I get to know who modified the code on branch1 and branch2 using git log command.
- to delete branch or tag
 - git branch -d branchname
 - git tag -d tagname
- used to create branch2 and automatically checkout to that branch
 - git checkout -b branch2
- how do you create a branch from tag?
 - You need to checkout to that tag and create a branch using git branch branchname
 - Git checkout tagname
 - Git branch branchname
- Git rebase branchname
 - Is nothing but a merge, it will add history also to the other branch.
 - It will not allow you to switch any other branch until unless you resolve merging conflicts.
 - Git rebase branchname
 - Git rebase --continue
 - Git rebase --skip
 - Git rebase --abort
 - Latest commit id is called as HEAD, (is a internal tag)
- Difference between merge and rebase
 - Merge it will merge latest content of 2 branches
 - Simple and easy to understand the merging concepts
 - Maintain original context as source code
 - Rebase will add the history also.
 - Unifies the lines development by rewriting changes from source branch so that they appear as children
 - Simplifies your history
 - Slightly more complex
 - Code history remain flat and readable.
- Assignment
 - Checkout to master, include 2 files big3.c and fact.c

- Then create a tag, inside tag create a branch1, inside that create palindrome.c then create a tag called release1 then in tat include reverse.c then again tag as release 2, then merge it to master with a tag name as release2.1.
- In the master again create a branch2, in tat include even.c and prime.c program and create a tag for those and merge it to master, then use tag as release 3.1 in master.
- Solution
 - Git checkout master
 - Vi big3.c then add big3.c and commit it
 - Git status
 - Vi fact.c then add and commit the file.
 - Git status
 - Git tag tag1
 - Git tag
 - Git branch branch1
 - Git checkout branch1
 - Vi pali.c
 - Add and commit using git of the file
 - Git status
 - Git tag tag2
 - Git tag
 - Vi reverse.c
 - Add and commit it
 - Git status
 - Git tag tag3
 - Git tag
 - Git checkout to master
 - Git merge brnach1
 - Git tag tag4
 - Git tag
 - Git branch branch2
 - Git checkout branch2
 - Vi even.c
 - Add and commit that
 - Vi prime.c
 - Add and commit that
 - Git status
 - Git log
 - Git tag tag5
 - Git checkout master
 - Git merge branch2
 - Git checkout brnach2
 - Git merge master

- Git tag tag6
 - Git tag
- Git revert
 - Used to undo the committed changes , history will not be removed we can track the reverted the changes in the git log
 - Git revert HEAD
 - Git revert CommitId
- Git reset
 - Used to undo the committed changes but history will be removed
 - There are 3 kinds of reset.
 - Git reset --mixed or git reset HEAD
 - Is used to move files from staging area to workspace
 - Git reset --soft commitId
 - Is used to move files from git repo to staging area it reset to specified commit id and history will be removed.
 - Git reset --hard commitId
 - Is used to remove the changes from git repo, workspace and also from staging area and commit id also removed.
 - Is as good as you are not committed the changes.
- Difference between bashrc and bashrc_profile
 - .bash_Profile
 - Executed for login shells
 - When you login via console, either sitting at machine or remote ssh, this is executed to configure your shell before the initial command prompt
 - Personal initialization file executed for login shells.
 - .bashrc
 - Executed for interactive non-login shells
 - If you already logged into system and open a new terminal then this is executed before the window command prompt
 - Individual per-interactive shell startup file.
- Git stash
 - If I am working on one branch, if I get some critical work, which needs to be fixed on the other branch. I don't want to commit changes in other branch, as I not completed the work, I will do git stash and I switch to other branch I will fix the issue and I come back to the previous branch to continue my work. I need to run git stash pop.

- Git stash will save files somewhere in temporary area. It will not store in working space, staging area or git repo.
 - Touch f1 f2 f3
 - Git add *
 - Git commit -m "files added"
 - Switch to other branch and check.
 - Git stash – then come back to other branch and try
 - Git status pop
 - Git reset HEAD
- Git cherry-pick
 - Git cherry-pick commitId
 - Used to merge specific commit to a current branch.
 - If I want to commit 3 commits use, git cherry-pick commitId1 commitId2 commitId3
- Git conflict
 - Is same as merge conflict
 - Sometime you get merge conflicts when merging or pulling from a branch.
 - Git will then tell you something like conflicts(content) merge conflict in fakefile.
 - It also tells you to fix conflicts and commit the changes.
- Git bisect
 - Use binary search to find the commit that introduced a bug.
 - Git bisect <subcommand> <options>
 - This command uses binary search algorithm to find which commits in your projects history introduced a bug.
 - Can be used to find the commit that changed any property of your project.
 - Git bisect start
 - Git bisect bad
 - Git bisect good
 - Git bisect reset
- There are 2 kinds of repository in git
 - Bare repo
 - Non-bare repo
- Bare repo
 - It acts as a central-repo, we can only push and pull the changes to the repository.
 - "In bare repository, you can't run any git operation on bare repo.
- Non bare repo
 - It's a local repo we can modify the files and push to a central repo we can run all the git commands. Command to create a non bare repo is git init

- Command to create a bare repo is
 - Git init --bare
- Git clone
 - Bringing the remote repo to local workspace for the first time called as git clone.
 - Git clone user@servername: /home/path/central_repo/
- Git push
 - Moving the changes from workspace to remote repo or central repo
 - Git push user@servername: /path../central_repo
- Git pull
 - Bring the changes from remote repository and merges to local repo automatically.
 - Git pull user@servername: /home../path/central_repo
- Git fetch
 - Bring the changes from remote repo and stores it on a separate branch, you can review the changes and merge normally if it is required.
 - Git fetch [user@servername:/home../central_repo/](#)
 - Git pull = git fetch + git merge
- Any source code execution
 - .c
 - Preprocessing
 - Compilation (.s)
 - Assembly conversion (.o)
 - Linking and loading
 - Executable files (.exe)
- Try this
 - Mkdir central_repo
 - Mkdir workspace1 workspace2
 - Cd central_repo
 - Git init --bare
 - Cd ..
 - Cd workspace1
 - Git clone ../central_repo/
 - Cd ../..
 - Vi big3.c
 - Git add and commit
 - Git push ../central_repo/
 - Cd ../..
 - Cd workspace2
 - Git clone ../central_repo/
 - Cd workspace2/central_repo

- Vi rev.c
- Git add and commit
- Git push ../../central_repo/
- Cd ../../
- Cd workspace1/central_repo
- Git pull ../../central_repo/
- Vi pali.c
- Git add and commit
- Git push ../../central_repo/
- Cd workspace2/central_repo/
- Git fetch ../../central_repo/
- Git checkout FETCH_HEAD
- Git checkout master
- Git merge FETCH_HEAD

Build Tools

- Makefile → c and c++
- ANT or Maven → java
- c/c++ → make → executable format (.exe)
- java → ANT or Maven → jar/war/ear
- compilers we use
 - gcc-c
 - g++-c++
 - javac-java
- Makefile or makefile
 - Is used to generate executable from c and c++ source code.
 - Syntax
 - Target:dep1 dep2 dep3
 - <TAB>system commands
 - Suppose there are 3 .c files you need to build .exe for those
 - Abc.exe:f1.o f2.o f3.o
 - <TAB>gcc -o abc.exe f1.o f2.o f3.o
 - F1.o:f1.c
 - <TAB>gcc -c f1.c
 - F2.o:f2.c
 - <TAB>gcc -c f2.c
 - F3.o:f3.c
 - <TAB>gcc -c f3.c
- Makefile
 - Abc.exe:big.o facr.o rev.o
 - <TAB>gcc -o abc.exe big.o facr.o rev.o
 - big.o:big.c
 - <TAB>gcc -c big.c

- `fact.o:fact.c`
- `<TAB>gcc -c fact.c`
- `rev.o:rev.c`
- `<TAB>gcc -c rev.c`
- **Build**
 - Is an executable or binary file, but not yet released to a testing team.
 - Is an untested build.
 - We deliver build to a testing team, not to a customer.
- **Release**
 - It is a tested build, which is ready to release to a customer.
- **ANT**
 - When I run ANT, it looks `build.xml`
- **Maven**
 - When I run maven, it looks `pom.xml`
- **Hooks**
 - `.git` → hooks → 2 sub topics. 1) pre commit 2) post commit
 - You need to trigger some task or action before commit or after commit, we go for hooks.
- **Difference between git and cvs (other version control tool)**
 - **Git**
 - Is a distributed version control system
 - It means whole repository will be there in the local workspace
 - If I want to go previous version of file, I can go directly in local workspace itself.
 - Git has many advanced features like rebase, fetch, stash, merge etc.
 - These advance features are not there in CVS.
 - **CVS**
 - Is a centralized repo and also SVN also.
 - If I want to goto previous version of a file, I need to checkout from centralised_repo because initially it will have only one version of a file in local repo.

Makefile

- Works on time stamp basis, if target time is less than the dependencies time it will re-generate the target, that means makefile will compare target time with its dependencies time.
 - If target time is latest than its dependencies it will not re-generate the target.
 - If there are thousand files, if I change 5 files, only 5 files will get compiled and incorporated at the build.
- Patch build
- It is a critical fix which needs to be deliver to a customer within few hours. Developers will change only required files, make will compile only changed files and changes will get incorporated to the build so patch build will take less time.

Load build

- We compile source code from the scratch, before we start this build we delete all intermediate files (.o files), so that all files will get compile from scratch. So it take more time.
 - Which make
 - Which gcc
 - Mkdir srccode
 - Vi big3.c
 - Vi fact.c
 - Vi main.c
 - Main () { fact(); big3(); }
 - Vi makefile
 - Syntax
 - Example code
 - Abc:main.o fact.o big3.o
 - Gcc -o abc main.o fact.o big3.o
 - Main.o:main.c
 - Gcc -c main.c
 - Fact.o:fact.c
 - Gcc -c fact.c
 - Big3.o:big3.c
 - Gcc -c big3.c
 - make
 - The target file abc.exe will get created
 - You can executed ./abc to get output of all files.
 - You can add the files and related syntax in makefile.
 - Even you can give your own makefile name as pradeepmakefile, but while making, you should use,
 - Make -f pradeepmakefile and while creating makefile use vi pradeepmakefile.
 - If you want to remove .o and .exe
 - Rm -rf *.o *.exe
 - Vi makefile
 - Syntax
 - Example code
 - Abc:main.o fact.o big3.o
 - Gcc -o abc main.o fact.o big3.o
 - Main.o:main.c
 - Gcc -c main.c
 - Fact.o:fact.c
 - Gcc -c fact.c
 - Big3.o:big3.c
 - Gcc -c big3.c

- Clean:
 - Rm -rf *.o
- While making, try
 - Make clean
 - It will remove all .o files.

○ Build.sh

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

```
Git pull ../../../../
```

```
If [ $? -ne 0 ];then
```

```
Git clone ../../../../
```

```
Fi
```

```
Cd /central_repo
```

```
Make targetname | tee build.log
```

```
If [ $? -eq 0 ];then
```

```
Echo " " | mail -s "build success" emailed
```

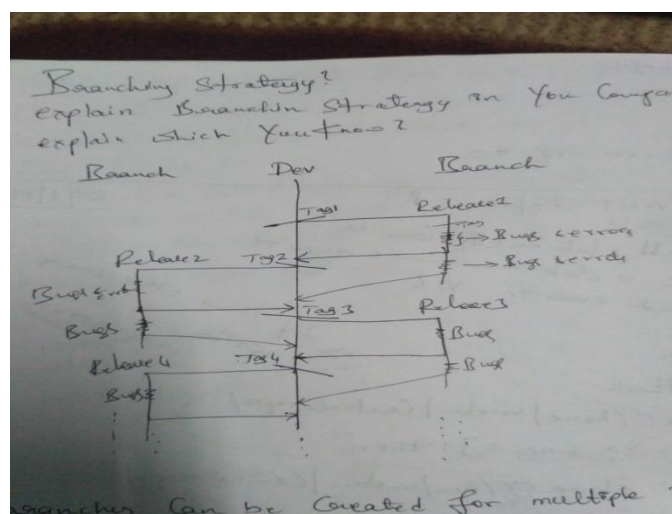
```
Scp targetname user:server:../../../../
```

```
Else
```

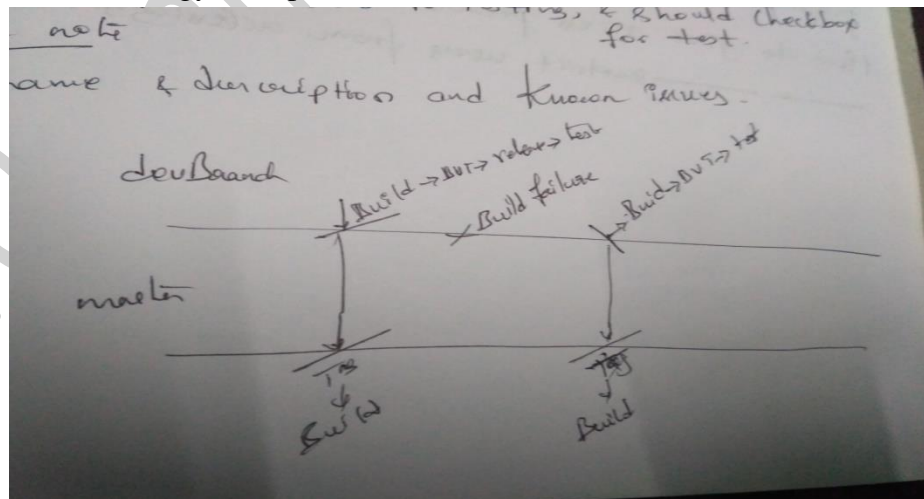
```
Mail -s "build failure" emailed
```

```
Fi
```

- Branching Strategy or explain branching strategy in your company, explain which u know?



- Branches can be created for multiple reasons, here we create branches for releases.
- Development team will be going on development branch, once the code is ready for the first release, we create a release1 branch from dev branch and we make a release (we do build and release it) from this release1 branch.
- Whatever the issues specific to this release will be foxed on release1 brnach only. It will act as a maintenance branch for release1.
- Simultaneously development will be going on dev branch for 2nd release. Once the code is ready for 2nd releasebefore we create a branch for 2nd release, we merge, release 1 branch to dev branch and then we create release2 branch for dev branch for 2nd release. So what ever the issues we have seen in 1sr release should not be visible in the 2nd release and so on.
 - Build→build success→BVT or sanity test→sanity report→release note→test team
 - Build→build fail→dev team
- BVT(build verification test) or sanity test.
- Is a basic functionality of a build which should never break.
- Sanity report.
 - Is a report related to testing and should checkbox for test.
- Release note
 - Tag name and description and known issues.
- Branch strategy example2



We have 2brnches, one is dev branch another one is master or production branch, developers are allowed to commit changes on dev branch. Once developers commits the code on dev branch. We build it, we do sanity or BVT and we release to a testing team. We merge this code to production branch. If the build is failed after developer commits the changes, we don't merge the code from dev branch to production branch. We work with development team to fix the issue. So we always merge good code to productive brnahc so that

production branch will have clean working branch always. If developer needs latest good code they can pull it from production branch, but they can't push it to production branch, because git push is restricted.

- How to give access permission to a git?
- How to give restrict users from accessing git?
- Maven and ANT difference?

Ant	Maven
Ant doesn't has formal conventions , so we need to provide information of the project structure in build.xml file.	Maven has a convention to place source code, compiled code etc. So we don't need to provide information about the project structure in pom.xml file.
Ant is procedural , you need to provide information about what to do and when to do through code. You need to provide order.	Maven is declarative , everything you define in the pom.xml file.
There is no life cycle in Ant.	There is life cycle in Maven.
It is a tool box.	It is a framework .
It is mainly a build tool .	It is mainly a project management tool .
The ant scripts are not reusable .	The maven plugins are reusable .
It is less preferred than Maven.	It is more preferred than Ant.

- Maven and ANT?
 - Maven is a project management and comprehension tool. Maven provides developers a complete build lifecycle framework. Development team can automate the project's build infrastructure in almost no time as Maven uses a standard directory layout and a default build lifecycle.
 - Apache Ant is a Java based build tool from Apache Software Foundation. Apache Ant's build files are written in XML and they take advantage of being open standard, portable and easy to understand.
- Git squash ?
 - This method only allows you to squash the last X consecutive commits into a single commit. Also, if you have merged master into your branch along the way, you will have to manually merge your new (squashed) commit into master and resolve the merge conflicts.
 - Use this method if you have not merged master into your branch, you plan to combine all commits into one, and you only changed one feature of the project; or, regardless of those conditions, you must use this method if you intend to open a pull request to merge your code.
 - **Combining the commits**
 - To squash the last 3 commits into one:
 - `git reset --soft HEAD~3`
 - `git commit -m "New message for the combined commit"`

- How to find errors in vi?

Deployment

- Web applications
 - Apache tomcat
 - Websphere
 - Jboss
- Executed/binary code→
 - Dev environment
 - QA or SIT (system integration testing)→UAT or pre-production testing(user acceptance testing)→production testing.
- Once the build is generated (war file). We deploy war file to QA and then to UAT and finally we deploy to production, on each environment tomcat will be running, we deploy to Webapps folder under tomcat server.
- Deployment status
 - Stop services→move logs→take backup of existing build→copy new build (war file)→start services.
 - If new build is not working, I will rollback the last build and use that and fix the issues in new build this process is called Rollback.
- Deployment script //(realistic script)

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

```
Services="ser1 ser2 ser3"
```

```
For I in $services
```

```
Do
```

```
Sudo service $i STOP
```

```
Done
```

```
Mv *.log /temp
```

```
Mv current.war current_date.war
```

```
Scp filewar user@server:../../
```

```
For I in $services
```

```
Do
```

```
Sudo service $i START
```

```
Done
```

- Above code is used to stop and start the services while deploying.

- At the time of deployment if there is any error will try to debug and fix it within the deployment window.
- If I am not able to resolve the issue I will roll back it.
- Deployment used in shell scripts as → used to automate the process of running process or anything automatically.

Introduction about yourself.

- *I am working as a DevOps and build engineer. I worked on linux, shell scripts and build tools like makefile and ANT, bug tracking tools like bugzilla and zerra, CI (continuous integration) like Jenkins. Version tool like GIT and CVS and AWS is the cloud environment, ansible is the server configuration management tool. I worked on apache tomcat server.*
- *I take care of writing scripts for manual and repetitive tasks whenever automation is necessary, we write scripts.*
- *I take care of Jenkins, create jobs in Jenkins, install plugins, schedule nightly and weekly builds in Jenkins.*
- *I take care of setting master-slave*
- *I work on deployments, deploying the build tool QA, UAT and production environment. This is also automated through Jenkins.*
- *I worked on release activities, we have releases once in a month that is..4 weeks sprint. At the time of release, we need to create a branch, merge the branches and resolve merge conflicts with the help of development team. So I am a beginner in AWS and ANSIBLE. I installed ansible on EC2-Instance, I even installed Jenkins on it, I am just practicing ansible, we are not yet completely migrated to AWS and Ansible.*
- *I take care of builds, if the build is failed need to debug identify why the build is broken. If there is any compilation issue, we need to work with development team and fix it.*
- *If there is any build environment issue, we need to debug our team and fix it. Build environment issue can be memory issue, network issue. If cpu usage is high on the server or it may be network issue.*
- **What is configure file for Tomcat?**
 - In order to configure a Context within Tomcat a *Context Descriptor* is required. A Context Descriptor is simply an XML file that contains Tomcat related configuration for a Context, e.g naming resources or session manager configuration. In earlier versions of Tomcat the content of a Context Descriptor configuration was often stored within Tomcat's primary configuration file *server.xml*

XML Configuration Files

The two most important configuration files to get Tomcat up and running are called `server.xml` and `web.xml`. By default, these files are located at `TOMCAT-HOME/conf/server.xml` and `TOMCAT-HOME/conf/web.xml`

The `server.xml` file is Tomcat's main configuration file, and is responsible for specifying Tomcat's initial configuration on startup as well as defining the way and order in which Tomcat boots and builds.

The `web.xml` file is derived from the [Servlet](#) specification, and contains information used to [deploy](#) and configure the components of your web applications.

- Interview que for Tomcat?
- CATALINA.log?
 - `catalina.log` is where the Tomcat engine writes log messages pertaining to Tomcat itself.
- Difference between web server and application server?
 - Application server are more heavy than **web server** in terms of resource utilization.
 - **Application Server** supports **distributed transaction and EJB**. While Web Server only supports Servlets and JSP.
 - A Web server exclusively handles HTTP requests, whereas an application server serves business logic to application programs through any number of protocols.
- Install tomcat and type on browser as publicIP:80 and install Jenkins and type on browser as publicIP:8080

```
jenkins.io----->>>>download--->>>>>>> Download
Jenkins 2.60.1 for: redhat/fedora/centos----->>>>then u
ll get commands
```


To use this repository, run the following command:

```
sudo wget -O /etc/yum.repos.d/jenkins.repo
https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-
stable/jenkins.io.key
```

If you've previously imported the key from Jenkins, the "rpm --import" will fail because you already have a key. Please ignore that and move on.

With that set up, the Jenkins package can be installed with:

```
yum install Jenkins
or
```

```
sudo wget https://pkg.jenkins.io/redhat-stable/jenkins-2.7.4-1.1.noarch.rpm
```

```
sudo yum install jenkins-2.7.4-1.1.noarch.rpm  
Whereis jenkins
```

```
-----  
Sudo yum install tomcat  
Whereis tomcat
```

```
ip:8080 --->jenkins
```

```
ip:80 --->tomcat
```

dev→build→test→dev→build→test

open→assign→progress→fixed→testing→solved

bugtracking tool like bugzilla and jira....

- Severity of bug also mentioned in bug tracking tool.
 - Critical, high and normal 3stages
- If a single commit is committed, then you should build a war file automatically after and before commit, use→
 - Git hooks
 - Pre commit and post commit.

Jenkins

- Jenkins is a CI tool and automated framework is used to integrate development changes automatically without manual intervention.
 - Goto Jenkins man page
 - Click on new item
 - Give job name
 - Select job type(free style)
 - Ok
 - Configure page
 - Description of job
 - SCM(source code management)
 - Select git
 - Path of git central repo, add branches, credentials
- Jenkins will create its own workspace and pulls latest code to it.

Build Triggers

- Is used to trigger the job based on the time,based on commit or where other jobs complete/success.
- Build periodically:we need to mention time like a crontab,it will trigger the job based on the time., and also it is a scheduler crontab.

- Poll scm: will trigger the job based on commits within the specified time job will get triggered based on commit.
- If there is a commit in between a hour, job ll get triggered else simply without commits it won't trigger.

Build step

- This section is used to compile and make a target file, and also compile and generate a binary. We can execute shell or invoke aNT.
- If I use execute shell, I can use shell script which internally calls makefile or I can use invoke ANT to compile java source code.

Post build action

- This is used to post build activities after build a project, like deploying build, running test cases or send a email notification or copying the build to shared path.

Master slave

- Master: server on which Jenkins has installed is called master.
- Slave: server on which we run the jobs from the master is called slave.
- Why master-slave or slave machines required?
 - To distribute the load to a different servers from the master we go for master-slave.
 - We need to run specific job on specific environment. for example, java project/java job we need to compile it on server which has java environment we go for master slave/slave machines.
- How to create slave machines/nodes?
 - Jenkins
 - Manage nodes
 - New node
 - Give node name and description
 - Number of executors
 - Label name
 - Root directory
 - Launch method (SSH), HOST, credentials.
 - Number of executors means, we can run specific number of jobs simultaneously on that server depending on capacity of server we specify number of executors.
 - If number of executors is '2' means, we can run 2 jobs from Jenkins simultaneously.
- How to run this slave on the new job or already created job or on the server
 - Got to configure page of job
 - Configure

- Restrict this project can run
- Mention the node name/slave name
- We go for labels to run job on available server, we need to add all server names to a same label. We need to mention the label name in the job instead of node name.
- This will work in round robin fashion.
- How to add ec2 instance on Jenkins?
 - First, go to [EC2](#) and sign up for the service. Once you've installed the plugin, you navigate to the main "Manage Jenkins" > "Configure System" page, and scroll down near the bottom to the "Cloud" section. There, you click the "Add a new cloud" button, and select the "Amazon EC2" option. This will display the UI for configuring the EC2 plugin. Then enter the Access Key and Secret Access Key which act like a username/password (see IAM section). Because of the way EC2 works, you also need to have an RSA private key that the cloud has the other half for, to permit sshing into the instances that are started. If you have already been using EC2 and have your own key, you can paste it here. Otherwise, you can have Jenkins generate one. If you let Jenkins generate one, you should save this private key in your file system as well, as you'll need this to interactively logon to EC2 instances.
 - Once you have put in your Access Key and Secret Access Key, select a region for the cloud (not shown in screenshot). You may define only one cloud for each region, and the regions offered in the UI will show only the regions that you don't already have clouds defined for them.
 - Use "Test Connection" button to verify that Jenkins can successfully talk to EC2. If you are using UEC you need to click on Advanced and fill out the endpoint details for your cluster.
 - http://www.bogotobogo.com/DevOps/Jenkins/Jenkins_on_EC2_setting_up_master_slaves.php
- Http port 80
- https port 443

how to install plugins

Manage Jenkins → manage plugins → update available installed advanced → in advanced just upload a plugin file .hpi format.

File will be get download from Jenkins updater center.

Parameterized plugins

- is used to give or pass parameters or input to a job at the run time. We have deployment job, we run that job based on request from dev team or development team on test team, at the running the deployment job we need to select build number and server name, these 2 parameters will be taken as input to a job.

Build pipeline (upstream and downstream)

- used to trigger one job after the other and we can pass parameter from one job to another using upstream and downstream plgins.
- In the build now page
- Configure page for build job
 - Upstream-_____
 - Downstream-deploy job
- Configure page for deploy job
 - Upstream-build job
 - Downstream-test job
- Configure page for build job
 - Upstream-deploy job
 - Downstream-_____
- Build job will get triggered by commit, once the build job is success it will automatically trigger deployment job.
- If deployment job is success it will automatically trigger testing jobs, we can trigger job serially using upstream and downstream plugins.

Gearman plugins

- It is a high availability plugin, if jenkins master goes down, jenkins will move go down if I install this plugins.
- German plugin will allow you to configure other server details, if the master goes down other server will act as a master. Both the server will be in a sink frequently.
- Other server will allow have complete Jenkins backup. So jenkins will never go down.

What is the most challenging task you done it?

- When I installed Jenkins for 1st time on the server, after a month server got crashed due to some hardware issue, server didn't come up at all. I lost the Jenkins builds were stopped from the day, we couldn't recover Jenkins. I had to resetup Jenkins on the other server.
- Now I searched for the high availability plugin, I got to know about GEARMAN PLUGIN. I installed gearman plugin in Jenkins and I configured other server details in it. If the master goes down, other server will act as a master. So Jenkins will never go down now.

How do you take Jenkins backup?

/var/lib/Jenkins

- We create separate git_repo for Jenkins and can push Jenkins configuration files to git_repo. Then we can clone it, whenever we need it. ((challenge)).

Jenkins safe restart

- It will allow to current running jobs to complete but it will not allow new jobs to be trigger, and it will restart the Jenkins, once the current running jobs are completed.
 - <http://localhost:8080/saferestart>

Jenkins security

- How to give security permissions to access Jenkins?
- How do you add new user in Jenkins?
 - We use matrix based security, it will allow to give required permission for a user by clicking the checkboxes.
 - Manage Jenkins → configure global security properties → click on enable security (these Jenkins own users db) → allow users to signup → in authorization section → matrix based security will be checked → you can add users and give permission security → then save it.
- How do you add environment variables?
 - Goto manage Jenkins → configure system → global env → add environment variables.
 - Name: var
 - Value: 1
 - We use configure system to configure smtp server allows.
- Git diff current_cmtid previous_cmtid (what it modified)
- Git show commitid (list all specified modified files in specified commits)

CI

- Is a continuous integration, integrating the development changes continuously without manual intervention.
- As soon as development team commits the code to a git, build job get triggered and build will get generated automatically.

CD

- Is a continuous deployment, deploying the build to testing environment automatically or less manual effort.
- Each change from development team is build and deployed to test environment automatically.

CD

- Is a continuous delivery, releasing deploying tested build to a production environment as quickly as possible.

Best practices of SCM (source code management)

- Keep workspace clean, mean delete temporary or unwanted files from workspace.
- Go for branching, if it is necessary.
- Tag the source code, whenever if it is necessary or the good code.

- Document the tasks which you do.

AWS

- Cloud
- Different types
 - Iaas saas paas
- Different cloud
 - Private public hybrid
- AWS service
 - Ec2
 - S3
 - DB
 - IAM
 - Cloud
 - VPC
- Diff instance type
 - T2.micro
 - T2.nano
 - T2.small
 - T2.large
- Spot instance n reserved instance
- Tenancy (shared and dedicated)
- Diff type sof storage
- Snapshots and backup of instance
- Auto scaling
- RDP remote desktop protocol
- How www.google.com work?
- Edges in AWS
- Diff platform in aws
 - Redhat
 - Ubuntu
 - Amazon
- IAM
 - Identity access management
 - IAM→group→create new grp→name it→attach policy→ec2-full→next step.
 - Roles-high level authentication.
- S3
 - Used to store web applications and all.
 - Static info
 - Dynamic info
 - Create bucket
- Glacier

- To store huge data or big data

Installing tomcat n Jenkins and mainly configuring user n ports

- 1) Sudo /usr/share/tomcat/conf/tomcat.conf
 - a. Tomcat-user="tomcat"→"root"
 - i. Conf/server.xml
 - ii. 8080—80
 - iii. 8443-443
 - b. Netstat -ntl
 - i. Running ports for tomcat
 - 2) Sudo yum install tomcat
 - a. Sudo vi /usr/share/tomcat/conf/tomcat.conf
 - i. Change java_opts path
 - b. Sudo yum install tomcat-webapps tomcat-admin-webapps
 - c. Sudo vi .usr/share/tomcat/conf/tomcat-users.xml
 - i. <tomcat-users>
 1. <user username="admin" password="passwd" roles="manager-gui,admin-gui"/>
 - ii. </tomcat-users>
 - d. Sudo service tomcat start
 - e. Sudo service tomcat restart
 - f. Chkconfig tomcat on
 - g. Service start
 - h. IP:8080
 - i. Apache home page
 - 3) Install Jenkins and config it
 - a. Sudo vi /etc/sysconf/Jenkins
 - i. Change jenkinsuser=root
 - ii. Then jenkinsport=8081
 1. Bcoz both tomcat n jenkins will take same ports that is 8080.
 - iii. Sudo service Jenkins start
- Auto scaling and Load balancer in AWS
 - Gerrit (code review tool)
 - Is a code review tool, is used to review the source code.
 - Once developer commits the code when he pushes the code it will create a review in gerrit. Reviewer has to vote it, if reviewer approves it, code will be pushed to central _repo.
 - Deploying to UAT and production env is based on the request.
 - Deploying to QA
 - QA and UAT deployments will be approved by managers.
 - Production deployment is approved by CAB (change advisory Board meeting) meeting.
 - Change management tool.

- Bugzilla
- JIRA
- We deploy the build sat early mrg and sat nights.
- BAT (build automation tool)
 - We were using it before as of now we were migrated to Jenkins.
- Docker
 - Is a container tool, is used to run build in virtual env. All env will be there in this container.
- Virtualization and hypervisor.
- Install docker and check for interview ques.
- Install nagios(monitors tool) and check the interview ques.
- Build steps
 - SCM
 - Build
 - Code coverage(check the code quality) -> install it.
 - Tools
 - Sonarqube—install>>>>>>
 - Code collaborator
 - Deploy
- Groovy
 - I know , but I have not used,
 - We have some automation tools to use for builds, so we not used groovy.
- Jenkins version- 2.7 (used about 3yrs)
- tomcat- 8
- issues – memory issues, tomcat issues, build issues due to compilation n some small errors.
- VPC (virtual private cloud)
 - Is a network
 - 10.0.0.0, 1st two are network next 2 are subnet.
 - 10.0.0.0/32
- Private network- used in internal world/env/company.
- Public network-used in external network
- Subnet (is a sub network, is used in segregation)

How do you setup passwordless connection between 2 servers?

Server1 -----> server2

Ssh-keygen

I will be getting private and public key

From server1 copy the public key of server1 to server2 in .ssh/authorized_keys

Then u can access server2 from server1 passwordless connection.

We can use winscp plugin or software to connect to server from password less connection.

We need to run ssh keygen on server1 it generates the key we need to copy public key to authorized keys files under .ssh folder. So we can connect s2 from s1 without password.

DOCKER

Is a tool that allows developers, sys-admin etc to easily deploy their applications on sandbox(container) to run on host OS ie linux.

Container is to use virtual machine to run software apps, VMs run apps inside a guest OS.

Why?

Containers easy to use and creating a way for the community to collaborate around libraries of containers.

Sudo yum repolist disabled

Sudo yum-config-manager --enable

Sudo yum search docker

Sudo yum -y install docker

Sudo systemctl enable docker

Sudo systemctl start docker

Sudo systemctl status docker

Sudo groupadd docker

Sudo usermod -aG docker ec2-user

Grep ^docker /etc/group

Docker pull rhel7

Sudo docker images

Sudo docker run -ti rhel7 /bin/bash

Cat /etc/redhat-release

[root@-----]df -h

Sudo chkconfig docker on

Sudo service docker start

Sudo chkconfig docker on

Sudo docker run hello-world

Sudo docker run busybox

Sudo docker images

Sudo docker run busybox

Sudo docker run busybox echo "pradeep"

Sudo docker ps

Sudo docker ps -a

Sudo docker run -it busybox sh

```
#ls
#date
#exit
Sudo docker run prakhar1989/static-site
```

----->prakhar public static repository!! Imported and tested in docker engine

```
Sudo docker run -d -P --name static-site prakhar1989/static-site
Sudo docker port static-site
Static-site is a file or hub or folder
443/tcp->0.0.0.0->32768
80/tcp->0.0.0.0->32769
Sudo docker run -p 8888:80 prakhar1989/static-site
```

Localhost:IP
Docker static site page will get displayed.

Ansible

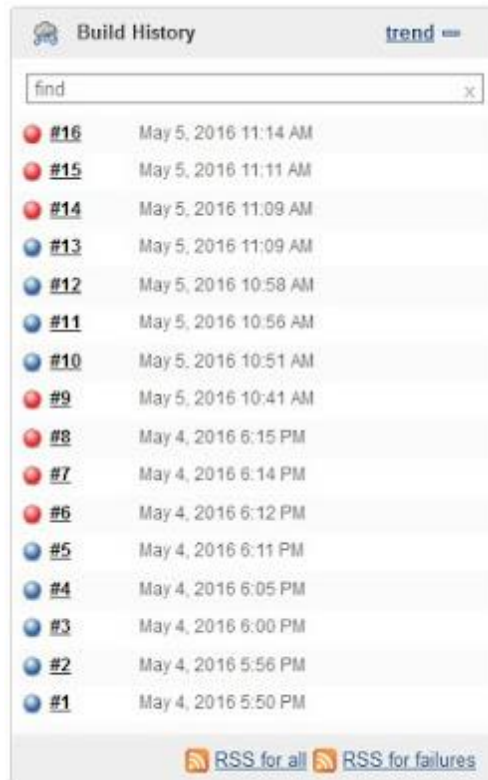
Server configuration management tool

- Playbook is a collection of scripts in YAML
- YAML->is a script with .yaml extension
- Ansible tower->is a UI page for ansible.
- Yaml is an extension of playbook.
- Passwordless within server of different users.
- Passwordless within two servers.
- Ansible roles
- Ansible modules.
- Vi /etc/ansible/hosts
- Different between ansible and puppet.
- S3(storage)
 - If instance gets terminated, the data will be lost. So I create a bucket and attach to instance. So even if the instance gets terminated, I can get data of that server, bcz it is linked in S3.

[How to clean and reset Jenkins build history](#)

Posted on Wednesday, May 04, 2016 by nayana

When you are working with Jenkins while building your solution, definitely you have to run your project multiple times. It will result to a long Build History list that looks ugly and unnecessary for your production environment.



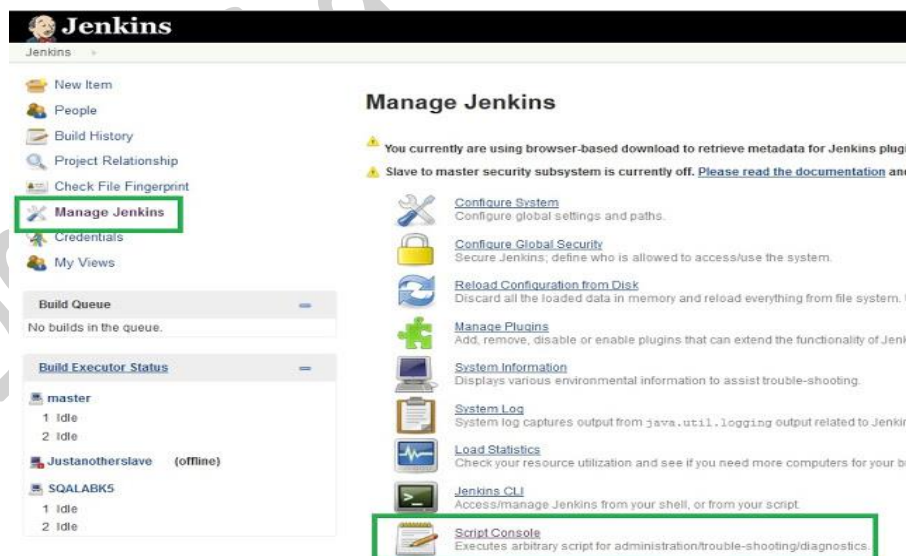
The screenshot shows the Jenkins Build History page. It features a search bar at the top with the text 'find'. Below it is a list of builds, each with a status icon (red for failed, blue for successful), a build number, and a timestamp. The builds are numbered #1 through #16. At the bottom, there are two RSS feed links: 'RSS for all' and 'RSS for failures'.

Build Number	Timestamp
#16	May 5, 2016 11:14 AM
#15	May 5, 2016 11:11 AM
#14	May 5, 2016 11:09 AM
#13	May 5, 2016 11:09 AM
#12	May 5, 2016 10:58 AM
#11	May 5, 2016 10:56 AM
#10	May 5, 2016 10:51 AM
#9	May 5, 2016 10:41 AM
#8	May 4, 2016 6:15 PM
#7	May 4, 2016 6:14 PM
#6	May 4, 2016 6:12 PM
#5	May 4, 2016 6:11 PM
#4	May 4, 2016 6:05 PM
#3	May 4, 2016 6:00 PM
#2	May 4, 2016 5:56 PM
#1	May 4, 2016 5:50 PM

If you need to clean this Build History, and reset the build number, you can run a simple script in Jenkins Script Console. Here are the steps to do so.

1. Go to `Jenkins Script Console`

Go to your Jenkins home page > Manage Jenkins > Script Console



2. Run the script to clean and reset

Copy and paste this script to your Console Script text area and change the "copy_folder" to project name that you need to clean the history. Then click the "Run button".

```
def jobName = "copy_folder"
def job = Jenkins.instance.getItem(jobName)
job.getBuilds().each { it.delete() }
job.nextBuildNumber = 1
job.save()
```

Now check your Build History for the particular project, Wow!, it's clean now!

Interview Questions Links

<https://career.guru99.com/top-16-tomcat-interview-questions/>

<http://www.javatechnologycenter.com/question/tomcat/>

<https://career.guru99.com/top-12-jenkin-interview-questions/>

<https://www.edureka.co/blog/interview-questions/jenkins-interview-questions/>

<http://www.javainterview.in/p/continuous-integration-interview.html>

<https://career.guru99.com/top-40-interview-questions-on-git/>

<https://www.edureka.co/blog/interview-questions/git-interview-questions/>

<https://www.linuxtechi.com/20-linux-commands-interview-questions-answers/>

<https://career.guru99.com/top-50-linux-interview-questions/>

<https://career.guru99.com/shell-scripting-interview-questions/>

<http://www.geekinterview.com/Interview-Questions/Operating-System/Shell-Scripting>

<https://www.edureka.co/blog/interview-questions/top-devops-interview-questions-2016/>

<https://career.guru99.com/shell-scripting-interview-questions/>

<https://linuxide.com/linux-shell-script/shell-scripting-interview-questions-answers/>

<https://github.com/Leo-G/DevopsWiki>

<https://gitenterprise.me/2015/04/17/gerrit-ver-2-11-is-now-available-for-download/>

<http://docs.aws.amazon.com/AmazonECS/latest/developerguide/docker-basics.html>

<https://www.tecmint.com/install-nagios-in-linux/>

<https://support.nagios.com/kb/article/nagios-core-installing-nagios-core-from-source.html#RHEL>

https://community.spiceworks.com/how_to/93152-installing-nagios-core-on-amazon-linux-instance

http://www.bogotobogo.com/DevOps/Jenkins/Jenkins_Adding_Code_Coverage_and_Metrics.php

<https://devopscube.com/setup-and-configure-sonarqube-on-linux/>

<https://www.youtube.com/watch?v=DXNS-EP9sXM>

<https://www.youtube.com/watch?v=mt32JEAxrA4>

Video File Links

- 1) <https://drive.google.com/drive/folders/0BzQsvgxHwSqNOWx3T1hwNnYtUmM?usp=sharing>

Above link is for creating AWS instance and installing Jenkins, tomcat, git, java and running on browser and changing Jenkins and tomcat port numbers.

- 2) <https://drive.google.com/drive/folders/0BzQsvgxHwSqNTkYzdURkMWhESWM?usp=sharing>

Creating a git repository and uploading the .c files and creating a build file and deploying to tomcat or copying to webapps folder.

Interview Questions

- 1) linux commands
- 2) shell scripts examples (relax)
- 3) jenkins flow
- 4) aws basics
- 5) ansible chef puppet explanation
- 6) diff between devops and build eng
- 7) what is ur role ?
- 8) issues?
- 9) roles and introduction
- 10) how u del commented lines in shell script
- 11) diff ant and maven
- 12) makefile
- 13) process of ur work
- 14) day to day activities
- 15) if there is so much of diff files how u decide whether u should use ant or maven?
- 16) y devops and value?
- 17) where u deploy war file?
- 18) full flow for jenkins!!
- 19) makefile
- 20) explain current project
- 21) master-slave concept
- 22) plugin installation and uninstallation and explain some plugins
- 23) jenkins backup procedure
- 24) build trigger types and description
- 25) security mechanism used in jenkins
- 26) pre-requisites in jenkins
- 27) continuous integration
- 28) property of ant
- 29) waterfall model
- 30) make files
- 31) how do u deploy and where do you deploy?

- 32) Write some realistic scripts, day today activities, issues, how u resolve it.
- 33) Ansible, yaml scripts, bug tracker, how u use jira?
- 34) Write flowchart of ur work and explain in deep.
- 35) Syntax for makefile, ant, maven.
- 36) Difference between ant,makefile,maven.
- 37) Maven life cycle and repositories
- 38) Y Devops ??
- 39) What is Jenkins?
- 40) What are the benefits of using Jenkins?
- 41) What are the pre-requisites for using Jenkins?
- 42) Mention some of the useful plugins in Jenkins?
- 43) Which SCM tools Jenkins supports?
- 44) Mention what are the commands you can use to start Jenkins manually?
- 45) Explain how you can set up Jenkins job?
- 46) Explain how to create a backup and copy files in Jenkins?
- 47) How will you secure Jenkins?
- 48) Explain how you can deploy a custom build of a core plugin?
- 49) What is the relation between Hudson and Jenkins?
- 50) What you do when you see a broken build for your project in Jenkins?
- 51) Explain how you can move or copy Jenkins from one server to another?
- 52) What are the various ways in which build can be scheduled in Jenkins ?
- 53) What is the difference between Maven, Ant and Jenkins?
- 54) What is another option for merging in git?
- 55) What is the syntax for "RebaWhat is the difference between 'git remote' and 'git clone'?"
- 56) What is GIT version control?
- 57) Mention some of the best graphical GIT client for LINUX?
- 58) What is Subgit? Why to use Subgit?
- 59) What is the function of 'git diff' in git?
- 60) What is 'git status' is used for?
- 61) What is the difference between the 'git diff' and 'git status'?
- 62) What is the function of 'git checkout' in git?
- 63) What is the function of 'git rm'?
- 64) What is the function of 'git stash apply'?
- 65) What is the use of 'git log'?
- 66) What is 'git add' is used for?
- 67) What is the function of 'git reset'?
- 68) What is git Is-tree?
- 69) How git instaweb is used?What does 'hooks' consist of in git?
- 70) Explain what is commit message? How can you fix a broken commit?
- 71) Why is it advisable to create an additional commit rather than amending an existing commit?
- 72) What is 'bare repository' in GIT?
- 73) Name a few Git repository hosting services?

- 74) What is GIT?
- 75) What is a repository in GIT?
- 76) What is the command you can use to write a commit message?
- 77) What is the difference between GIT and SVN?
- 78) What are the advantages of using GIT?
- 79) What language is used in GIT?
- 80) What is the function of 'GIT PUSH' in GIT?
- 81) Why GIT better than Subversion?
- 82) What is "Staging Area" or "Index" in GIT?
- 83) What is GIT stash?
- 84) What is GIT stash drop?
- 85) How will you know in GIT if a branch has been already merged into master?
- 86) What is the function of git clone?
- 87) What is the function of 'git config'?
- 88) What does commit object contain?
- 89) How can you create a repository in Git?
- 90) What is 'head' in git and how many heads can be created in a repository?
- 91) What is the purpose of branching in GIT?
- 92) What is the common branching pattern in GIT?
- 93) How can you bring a new feature in the main branch?
- 94) What is a 'conflict' in git? How can conflict in git resolved?
- 95) To delete a branch what is the command that is used?
- 96) What is Linux?
- 97) What does a nameless (empty) directory represent?
- 98) What is the pwd command? What is grep command?
- 99) Write a command that will look for files with an extension "c", and has the occurrence of the string "apple" in it.
- 100) Write a command that will display all .txt files, including its individual permission.
- 101) What is the command to calculate the size of a folder?
- 102) How can you find status of a process? How can you check the memory status?
- 103) Explain how to color the Git console?
- 104) How can you append one file to another in Linux?
- 105) Explain how you can find a file using Terminal?
- 106) What is a shell? What is the difference between soft and hard links?
- 107) How will you pass and access arguments to a script in Linux?
- 108) What is the significance of \$#?
- 109) What is the difference between \$* and \$@?
- 110) How will you find the 99th line of a file using only tail and head command?
- 111) Print the 10th line without using tail and head command. Explain about "s" permission bit in a file?
- 112) How can you find out how long the system has been running?
- 113) What is the difference between \$\$ and \$!?
- 114) What is the difference between grep and egrep?

- 115) What is the significance of \$? ?
- 116) How do we delete all blank lines in a file?
- 117) How will I insert a line “ABCDEF” at every 100th line of a file?
- 118) Write a command sequence to find all the files modified in less than 2 days and print the record count of each.
- 119) How can we find the process name from its process id?
- 120) What is the use of a shebang line?

DevOps and Build Engg,