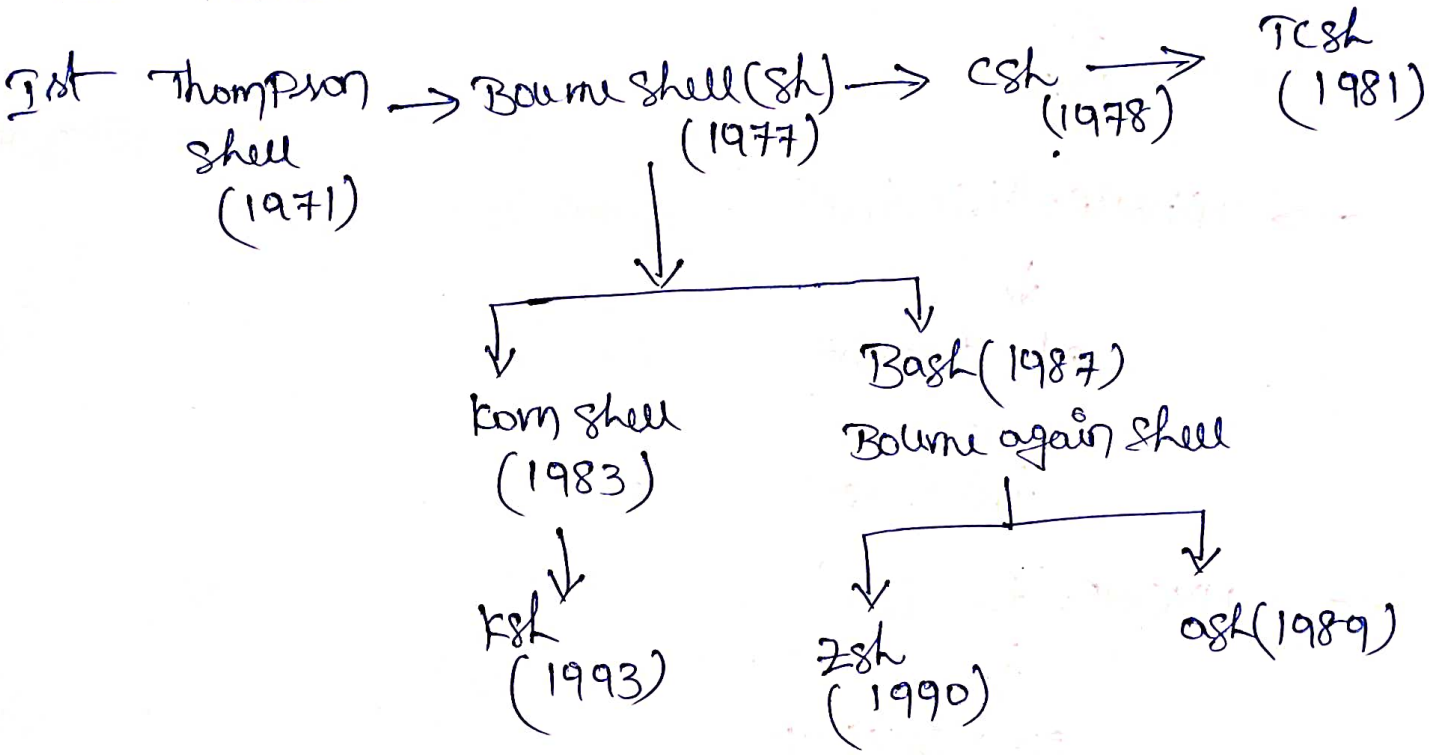
Scripting

Bash Scripting:- A Bash script is a series of commands written in a file. these are read & executed by the bash Program.

Types of shells:-



→ which shell you are in machine.

↳ echo $0 [Bash, csh, Tcsh, sh]

→ checking version of bash.

↳ bash -version

→ if csh, tcsh not installed in server then you need to install it.

↳ yum install csh & tcsh ↳ apt install tcsh & csh.

↳ bash -version

→ show path directory.

↳ echo $dirstack --show dir

variable Declaration:-

→ MYNAME=ttt

↓

echo $MYNAME

↓

ttt

→ MYNAME="trie tree"

↓

echo $MYNAME

↓

trie tree

→ MYCOMP='trie tree'

↓

echo $MYCOMP

↓

trie tree

→ MYCOMP="my name is: $MYNAME"

↓

echo $MYCOMP

↓

my name is: trie tree

[if you giving 'single quotation' it doesn't accept as variable it's accepts only as string]

→ for checking PPID for shells.

↳ echo $$PPID

→ for readonly variables we can't unset that means we can't remove.

readonly READONLYVAR=String

↓

echo $READONLYVAR

↓

String

→ normal variable we can unset easily.

MYNAME=ttt

↓

unset $MYNAME

↓

NoDataFound.

→ if 'export' the variable then you can use that variable in any bash (switching another bash).

export MYNAME=ttt

↓

echo $MYNAME

↓

ttt

↓

bash (switching another bash)

↓

echo $MYNAME

↓

ttt

→ for checking 'export' variables,

env

↓

shows all the system variables.

declare

↓

shows all functions.

→ for checking shell level,

echo $SHLVL

↓

number shows which shell you are.

files displaying :-

→ touch file1 file2 file3

↓

ls (file1 file2 file3)

↓

ls file[12] (or) ls file[0-2]

↓

file1 file2

↓

ls fil(a-z)(1-9)

↓

file1 file2 file3

↓

ls file?

↓

file1 file2 file3

→ dot file(.files) shows

ls -al

pipes & Redirect :-

→ echo "tre tree tech"

↓

tre tree tech

↓

echo "trie tree tech" > trie [Redirecting to particular file]

↓

Cat trie

↓

trie tree tech

grep Commands,

Cat trie | grep -c tree

↓

1 [only in one line this word]

↓

vi trie [trie tree tech
tree
trie]

↓

Cat trie | grep -c tree

↓

2 [only in 2 lines this word]

Pipe means output of 1st Command is as a input of 2nd Command.

→ ">" means overide. overide means removing the existing Content & place it new Content.

→ ">>" means vfend. vfend means place it new Content with in existing Content.

→ hostname

↓

trietreetechnologies.com

→ echo "my hostname is: $(hostname)"

↓

my hostname is: freetruetechnologies.com.

→ echo "my hostname is: `hostname`" [:Backsticks']

↓

my hostname is: freetruetechnologies.com

→ touch file1 file2

↓

echo $(echo file1 $(echo file2))

↓

file1 file2

File Permissions :-

rwx → Executive
read, write

	owner	Group	other
-	rwx	r--	r--

rwx
4 2 1

↳ chmod 700 filename
for only owner Permissions.

↳ chmod a+x filename
for giving execute permissions

↳ chmod a+w filename → to All for write Permissions to All.

Script Standards :-

→ Naming Convention for understanding script anyone

→ files Permission [chmod]

→ Script format & Execution [#!/bin/bash]

absolute Path :- An absolute path is a full path specifying the location of file or directory from root directory.

Relative Path :- its location relative to the current working directory. [file which located in current directory]

•Simple Tasks :-

1) vi Basictask.sh

↓

```
#!/bin/bash
# Author: Trio tree technology
# this Script main used for knowing
    all CPU, disk, free space & Ram ...etc
echo "welcome to devops"
echo
top |head 10   for knowing CPU
echo
df -h   for knowing disk space
echo
free -m   for knowing free & consumed space (swap)
echo
uptime   for knowing server login time & user
```

echo
Pwd for present working directory
echo
ls for list of files
echo
touch file1 file2 for file1 & file2 creating
echo
iostat for giving data to users how much

↓

:wq!

Here taking ip from usr & Converted that as a output.

2) vi inop.sh

↓

```
#!/bin/bash
host='hostname'
l=ls
P=Pwd
wa=whoami
```
} variables representing.

```
echo "my hostname is : $host"

echo
echo "what is your name"
read name
echo "hello $name"
```
→ here reading the variable

```
echo
echo $l
```
→ prints list of files

```
echo
echo $P
```
→ shows Pwd directory

```
echo
echo $wa
```
→ who you are shows

```
echo
```

:wq!

3) 'if then' statements,

(i) vi ifthen.sh

↓

```
#!/bin/bash
count=100
if [ $count -eq 100 ]
then
echo "Count is 100"
else
echo "Count is not 100"
fi
```

if statment starts ← if

Ends ← fi

↓

:wq!

↓

Scripts executes & give results "Count is 100".

(ii) vi ifthen1.sh

↓

```
#!/bin/bash
clear
if [ -e /root/scripts/error.sh ]
then
echo "this file exists"
else
echo "this file not exists"
fi
```

→ for checking file exists or not in that particular path.

↓ :wq!

↓

script execute & give results.

if ifty exists → "this file exists"

if doesn't exists → "this file not exists"

(iii) Taken i/p from user & giving o/p through if then statements

vi inputifthen.sh

↓

```
#!/bin/bash
# Author: tri tree technology
clear
echo "Hi subscriber, what is your name?"
read name          → here reading the variable
echo "Hellow $name"
echo "would like my bash scripting training (y/n)"
read like          → reading the variable
if ["$like" == y]
then
echo "Thank you"
elif ["$like" == n]
then
echo "please send ur feedback"
echo
fi
```

↓

:wq!

↓

Script execute & give results. after giving the name,

if it is "y" → thankyou

if it is "n" → please send ur feedback

(iv) Case statements,

vi casestish

↓

```
#!/bin/bash
echo
echo "Please choose option below"
echo "a=Display system date"
echo "b= list of all files"
echo "c= who logged into this system"
echo "d= what is the uptime of this server"
read choices → reading variable
case $choices → choosing case variable
    a) date;;
    b) ls;;
    c) who;;
    d) uptime;;
    *) echo "Invalid training choice -- Thankyou"
esac
```

Case statements starts ←

Ends ←

4) for loop statements ⇒ it is repeats a set of commands for every item in a list.

↓

vi forloop.sh

↓

```
#!/bin/bash
for j in liked subd shared Commented
do
echo "Jai hay $j"
done
```

↓

:wq!

↓

script execute & give results

```
Jai hay liked
Jai hay Subd
Jai hay Shared
Jai hay Commented.
```

5) do while loop statments ⇒ It is used for perform the set of commands an unknown number of times as long as the given Condition evaluates to true.

(i) vi ~~for~~ while loop.sh

↓

```
#!/bin/bash
count=0
num=10
while [ $count -lt 10 ]   → less than
do
echo "$num seconds left to stop this $1"
```

```
sleep 1
num=`expr $num - 1`
count=`expr $count + 1`
done
echo "$1 Process is stoped !!!"
```

↓

:wq!

↓

Script execute & give results (./forloop.sh).

10 seconds left to stop this
9 seconds left to stop this
8 " " "
7 " " "
6 " " "
5 " " "
4 " " "
3 " " "
2 " " "
1 " " "

Process is stopped !!!

↓

Shows Processes running & getting ID } ps -ef | grep -i forloop.sh

↓

./forloop.sh 4(and)

↓

ps -ef | grep -i (ID)

(iv) vi ~~for~~ while loop1.sh

↓

```
#!/bin/bash
c=1
while [$c -le 5]
do
echo "welcome $c times"
((c++))
done
```

(-le → less than)

↓

:wq!

↓

Script execute & give result

welcome 1 times
" 2 "
" 3 "
" 4 "
" 5 "

Exit Codes:-

0 - ok
1 - general error
2 - misuse of shell

126 - Cannot execute
127 - no file found matching Command
128 - invalid exit value

after 128 - Process kill with signal end

↳ Command for checking exit code → echo $?

vi exist.sh

↓

```
#!/bin/bash
ls -l /root/scripts/exist.sh
if [ $? -eq 0 ]
then
 echo "file exists"
else
 echo "no file"
 fi
```

↓

:wq!

↓

Script execute & give results if files exist & not shows.

Realtime Scripts:-

NOTE:-

Error logs messages Path → /var/log/messages

(i) → for checking error logs by using Pgrep Commands,

mkdir RealtimeScript

↓

cd RealtimeScript

↓

cp /var/log/messages /root/scripts/Realtimescripts/messages

↓

Pwd

(root/scripts/realtimeScripts/messages)

↓

vi allerrors.sh

↓

```
#!/bin/bash
#Real time Errors file
#Author: ttt
grep -i error /root/scripts/Realtimescripts/messages
grep -i warn    "          "          "
grep -i fail    "          "          "
```

↓

:wq!

↓

script Execute & give results

[error, warn, fail messages lines shows]

If you want Particular seperate file error, warn & fail messages then you can redirect to Particular file.

→ grep -i error /root/scripts/Realtime scripts/messages > /root/scripts/error.file

→ " " warn " " " " > " " /warn.file

→ " " fail " " " " > " " /fail.file

ii) vi admintasks.sh

↓

```
#!/bin/bash
date | awk '{Print $1}'      → In date only 1st column Printing
uptime | awk '{Print $1}'    → In uptime only 1st column Print
df -h | grep -i tmpfs        → In desktop whatever 'tmpfs' word is there those lines Pri
```

↓
:wq!
↓
Script execute & give results.

(iii) for checking remot server connectivity,

vi remotserver.sh
↓
```
#!/bin/bash
ping -c1 localhost &> /dev/null
if [ $? -eq 0 ]
then
echo "ping works"
else
echo "ping not works"
fi
```
↓
:wq!
↓
Script execute & give results
[ping works & not]

iv) if you have some hosts for checking those connections, first you need to create 'host file'.

vi myhosts
↓
```
localhost
192.168.0.1
192.168.0.2
```
↓ :wq!

vi pinghost.sh

↓

```
#!/bin/bash
hosts="/root/scripts/realtimeScripts/myhosts"
```

→ Give host file path.

```
for ip in $(cat $hosts)
do
    ping -c1 $ip &> /dev/null
    if [ $? -eq 0 ]
    then
    echo "$ip ping works"
    else
    echo "$ip is not working"
    fi
done
```

↓

:wq!

↓

script execute & give results

```
localhost ping works
192.168.0.1 is not working
192.168.0.2 is not working
```

v) Based on Crone Expression checking host connection.

~~vim~~ ↓

~~crone~~ crontab -e

↓

Right now time → 13:02

04 13 * * * /root/scripts/realtimescripts/pinghost.sh > /root/scripts/ping.out

min hour Everyday Every month Every year

for pinging file in that particular time

04 13 * * * /root/scripts/realtimescripts/pinghost.sh | mail -s "Connection status" ramu060895@gmail.com

Sendmail to this

↓

:wq!

↓

Check date at exactly at 13:04 pts pinging starts & check file(ping.out) in /root/scripts/ping.out.

vi) Removing old files, rename the files.

→ touch -d "MON 1 MAR 2020 12:30:00" marchfile

→ " " " " marchfile1

→ " " " " marchfile2

} create for old files

These files created at that particular date.

↳ ls -lrt for shows all old & new files

→ find /root/scripts/realtimescripts/ -mtime +90 -exec ls -l {} \;

for showing Past 90 days files

→ find /root/scripts/realtimescripts/ -mtime +90 -exec rm {} \;

for deleting Past 90days files

→ find /root/scripts/rattimescripts/ -mtime +90 -exec mv {} {}.old \;

for renaming the Past 90 days files.

→ ls -lrt

Ans :→ marchfile.old
marchfile1.old
marchfile2.old

NOTE:-

1) system log files will haves in /var directory.

2) application Config. files will haves in /etc directory.

vi) Backup the data from /etc /var to temp & moving to any server.

vi backup.sh

↓

#!/bin/bash

tar -cvf /temp/backup.tar /etc /var

↓ zip the file in the form backup.tar from /etc /var data

gzip /temp/backup.tar → Compress the file

find /temp/backup.tar.gz -mtime -1 -type f -print > /dev/null

If [$? -eq 0]

then

echo "backup was created"

echo

echo "archieving backup"

scp /temp/backup.tar.gz root@:IP:/Path → for moving data to another Server

else

echo "backup failed"

fi

↓

:wq!

↓

Script execute & give results.

[backup was created
cd /temp
↓
ls -lrt
(backup.tar.gz)]

vii)

vi count.sh

↓

#!/bin/bash

for i in {1..10}

do

echo $i

sleep 1 → sleep 1 second

done

↓

:wq!

↓ chmod 744 count.sh

Script execut & give results

[1
2
3
4
.
10]

viii) Create files.

vi createfile

↓

```
#!/bin/bash
for i in {1..10}
do
touch jai.$i
done
```

↓

:wq!

↓

script execute & give results

jai.1
jai.2
:
jai.10

ix) user interactive by using for loop you can create files.

vi createusr-input.sh

↓

```
#!/bin/bash
echo "how many files need to be created?"
read t
echo
echo "please enter starting name of file"
read n
for j in $(seq 1 $t)
do
touch $n.$j
done
```

↓ :wq!

↓

Script execute & give results

Give how many files = 3

enter starting name - trie

[trie.1
trie.2
trie.3]

x) Permissions giving to files by using script.

1) vi permission.sh

↓

```
#!/bin/bash
for j in trie.*
do
echo "Assign Permissions to $j"
chmod a+x $j    for giving execute Permissions to trie1,2,3
sleep 1
done
```

↓

:wq!

↓

Script execute & give results

rwx	r-x	r-x
0	G	0

for trie*

2) Based on time we are giving Permissions to files.

9.

vi permissionone.sh

↓

```
#!/bin/bash
total='ls -l trie* | wc -l'        → total file name files shows "3"
echo "it will take $total seconds, to complete"
echo
for i in trie*
do
echo "Assign write permission to $i"
chmod a+w $i
sleep 1        → sleep 1 second
done
```

↓

:wq!

↓

script execute & give results

[Assign write permission to trie.1
trie.2
" " " trie.3
" " "
"]

[rwx rwx rwx] for trie*

3) Rename the files also.

vi renamefile.sh

↓

```
#!/bin/bash
for filename in *.sh
do
mv $filename ${filename%.sh}.bash
done
```

↳ replacing files names ending with .sh to .bash

↓

:wq!

↓

Script execute & give results

[.bash]

4) In /home directory we having some users & directories(mkdir). we are checking user assigned or not.

vi userassignornot.sh

↓

```
#!/bin/bash
cd /home
for DIR in *          → checking all directories
do                    ↓ variable
CHK=$(grep -c "/home/$DIR" /etc/passwd)
if [ $CHK -ge 1 ]
                      ↓ greaterthen
then
echo "$DIR is assigned to a user"
else
echo "$DIR not assigned"
fi
done
```

↓

Script execute & give results.

↓

In /home directory whatever users are there those "are assigned to users" & whatever directories are there those "are not assigned".

5) who are ~~logged~~ logged in this system checking script.

→ last → used for who are logged in recently shows

→ ~~grep~~ last | grep "Mon Aug 22" for showing users only on particular date

vi today logged in.sh

↓

```
#! /bin/bash
today='date | awk '{print $1, $2, $3}'
last | grep "$today" | awk '{print $1}'
```

↓

:wq!

↓

script execute & give results

```
root pts/0 124.123.16.12 Mon Aug 17 22 still log in)
root
```

6) user loggedin script checking.

vi userlogin.sh

↓

```
#! bin/bash
echo "please enter day (e.g Mon)"
read day
echo "please enter Month (e.g Aug)"
read month
echo "please enter date (e.g 21)"
read date
 echo
last | grep "$day $month $date"
```

↓ :wq!

when you login to the machine you are Jost like a "ec2-user". that means you are normal user & you don't have access to create any files.. etc. (Permission denied). if you enter any command then it will search in these Paths.

↳ echo ${PATH}

Any /usr/local/bin: /bin: /usr/bin: /usr/local/sbin: /usr/sbin: /sbin: /opt/aws/bin: /home/ec2-user/.local/bin: /home/ec2-user/bin:

if you are become a superuser [sudo su -].

↳ echo ${PATH}

/usr/local/sbin: /usr/local/bin: /sbin: /bin: /usr/sbin: /usr/bin: /opt/aws/bin: /root/bin

↳ man hier for and for detailed information these Paths.

→ if your scripts keeps in /usr/local/bin this path then that scripts access to all users & root.

if your scripts keeps in /usr/local/sbin this path then that scripts access to only root user.

→ If you creating any script on root user then 'root' user will able to execute the script. then 'ec2-user' not having a permission for executing this script.

→ if you want execute 'ec2-usr' for this script then Copy that script file to usr/local/bin path & give permissions for that script.

↳ sudo chmod 707 scriptfile

[then ec2-user having 'sudo' permission]

→ But if you having a another user (testusr1) will not able to access the through sudo permission to script file.

↳ sudo useradd -m testusr1 for creating user

↳ Passwd testusr1 for creating password

↳ chmod 777 scriptfile.

if you want access & execute the scriptfile through testusr1 then you need to add this user in 'visudo' file, (sudo access),

↳ visudo

↓

Allow root to run any commands anywhere

root ALL=(ALL) ALL

testusr1 ALL=(ALL) ALL

↓

:wq!

↓

↳ su - testusr1 for switching to user

↓

sudo su (become root of testusr1)

↓

then execute the script here.

(or)

↳ create 'testuser' & add this user in wheel group of 'sudo' file. [usermod -aG wheel testuser]

su - testuser for switching to user

↓

sudo su - (become root of testuser)

↓

then execute the script here.

→ 'hash' command will shows all ~~execute~~ whatever commands you are use Previously those all commands will save in Particular path

↳ hash

↳ type -a cat (where that command), uptime, ifconfig

these are all Inbuild Commands

variables:-

→ if you want to see Particular PublicIPV4 & Security group of instance through the variables.

↳ MYPIP='curl -sL http://169.254.169.254/latest/meta-data/Public-IPV4

↓

echo $MYPIP (or) echo ${MYPIP} (or) echo "${MYPIP}"

↓

Public IPV4 shows

↳ MYSG=$(curl -sL http://169.254.169.254/latest/meta-data/Security-group)

↓

echo $MYSG → Security group shows.

1) vi test.sh

↓

#! /bin/bash

set -x → debugging Purpose for seeing out in server

NUMONE=100
NUMTWO=200
echo ${NUMONE}
echo ${NUMTWO}
useradd -m testusr1
useradd -m testusr2
useradd -m testusr1
useradd -m testusr3 → for creating users in /home path

{ set -xe
↓
debugging
↓
error messages stop at that line.

↓
:wq!
↓
Script execute & give results

```
+NUMONE=100
+NUMTWO=200
+echo 100
 100
+echo 200
 200
+useradd -m testusr1
+useradd -m testusr2
useradd: user 'testusr1' already exists
+useradd -m testusr3
```

↳ Here if you get error also then moves further Command execution.

↳ if you keep 'set -xe' then if you get error message then STOP at that Particular line not moves to further Command execution. {

2) Creating user & Password with the help script.

vi createuser.sh

↓

#!/bin/bash

Set -xe

read -p "Please enter username:" USER_NAME

read -s -p "Please enter Password:" USER_PASS

Secret mode will not shows.

echo ${USER_NAME}

↓

:wq!

↓

Script execut & give results

[read variables, assign to username & Password]

NOTE:- By default Password Authentication is not working in AWS. So you need enabling the passwd Authentication in sshd_Config file.

↳ cd /etc/ssh/sshd_Config

↓

Passwd Authentication yes

↓

:wq!

↓

Servile sshd restart.

3) Creating user & Random Password generator through script.

vi createuser1.sh

↓

```
#!/bin/bash
#set -x -c ##for hide the all content showing
read -p "please enter the username:" USER_NAME
PASSWORD=$(curl -sL https://hellobcm.com/api/random/?n=14)
useradd -m ${USER_NAME}
echo ${PASSWORD} | Passwd --stdin ${USER_NAME}
echo "the username is ${USER_NAME} & Password is ${PASSWORD}"
```

[Google: random password generator]

↓

14 letter Password Creating

:wq!

↓

Script execute & give results.

give username & Password automatically takes from remot URL. Take that username (testuser@PublicIPAdd) & Password Connected to Putty that means it's work.

4) with Remote Random Password generator, you making own Password.

i) → X=${RANDOM} for creating Random number

Any echo ${X}

30562

ii) → date +%s for giving seconds from 1970-01-01 to till now.

Any 1553047943

→ date +%s%N for giving nano seconds " "

Any 155304795369306794 2

["man date" for checking these s,N & letters values]

→ NEWPASS=${RANDOM}$(date +%s%N)

Any 11915155304798767314950 6

iii) → SPEC='!@#$%^&*()_' (&) ALPHAS='a..zA..Z'

↳ echo ${SPEC} | fold -w1 | shuf | head -1 for these

Any #

characters will shows one by one with the Suffeling finally only one character shows.

→ SPECCHAR=$(echo ${SPEC} | fold -w1 | shuf | head -1)

~~$~~ echo ${SPECCHAR}

Any (&) & # & !

vi createuser2.sh

↓

```
#!/bin/bash
Set -xe
read -P "enter the username:" USER_NAME
SPEC='!@#$%^&*()_'
SPECCHAR=$(echo ${SPEC} | fold -w1 | shuf | head -1)
PASSWORD=$(RANDOM)$(date +%s%N) ${SPECCHAR}
Useradd -m ${USER_NAME}
echo ${PASSWORD} | Passwd --stdin ${USER_NAME}
echo "succefully created user ${USER_NAME} with
      Password is ${PASSWORD}"
```

↓

:wq!

↓

script execute & give results

give username & Password was generated based on your charctors, take those Credentials & Connect to Putty.

NOTE:- ↳ Cat etc/Passwd | grep -i EC2-user → will shows EC2-user existing or not (shows capital & small letters in Cat etc/Passwd.

↳ man useradd → shows more information

↳ man passwd → " "

5) Before Creating do user checking wheather user exists or not.
(If Condition) vi createuser3.sh

↓

```
#!/bin/bash
#this script used for chek the user before creating it.
#Read username from keyboard
read -p "please enter username:" USER_NAME
#Create Complex Password
 SPEC='!@#$%^&*()_'
SPECCHAR=$(echo ${SPEC} | fold -w1 | shuf | head -1
PASSWORD=India@${RANDOM}${SPECCHAR}
#chek if the user exists if exists throw an error.
EXUSER=$(cat /etc/passwd | grep -i ${USER_NAME} | cut -d":" -f1)
                                                  ↓
                                              delimitor

echo "The existing username is ${EXUSER}."
if [[ ${EXUSER} == ${USER_NAME} ]]
then
echo "user already exists. Please use different user..!!"
else
echo "Create new user"
 Sleep 3s
Useradd -m ${USER_NAME}
echo ${PASSWORD} | Passwd --stdin ${USER_NAME}
Passwd -e ${USER_NAME} → for Empire Password
```

#Print the username & Password

echo "Username is ${USER_NAME} Password is ${PASSWORD}"

fi

↓

:wq!

↓ Script execute & give results.

Based on username & password Connect to Putty. here create new password.

NOTE:- for checking shell script in online "shell script validate" (shellcheck.net)