Session-9: (@1hour 10 mints to 1H: 20Mints-- Shell scripting introduction)

1-10 commands

human errors

Time taking

Shell Scripting

If you keep all your commands in a single file and execute that file --> Shell Scripting

Native Linux scripting --> Linux/Shell commands

From Linux Server, if I need to fetch some info from AWS Cloud then I can choose Python for scripting

☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

Session-10: (watch till 1Hour: 18Mints)

- Git introduction(Linus Torvalds)
- Shell scripting
- Variables

Git is a Concept (below mentioned names are implementations of GIT)

GitHub

Gitlab

Bitbucket

Azure repos

AWS Code commit

Version Control System (VCS)

Need to maintain multiple versions

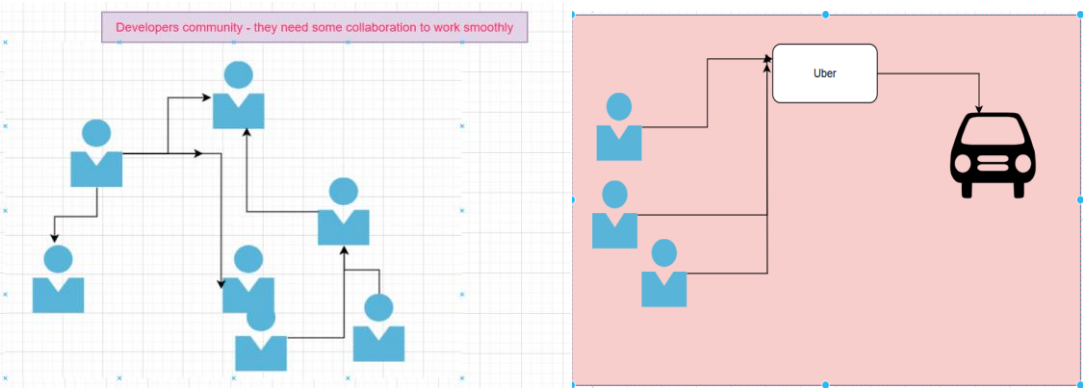
- I need to track the History of changes
- *Why I changed? When I changed? Who changed?*

20-DEC-2024 we deployed app in production

21-DEC-2024 suddenly there is issue with latest version.

19-DEC-2024 --> restore to previous version

Collaboration

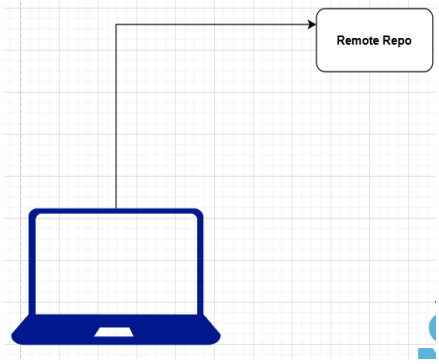


Centralised vs Decentralised

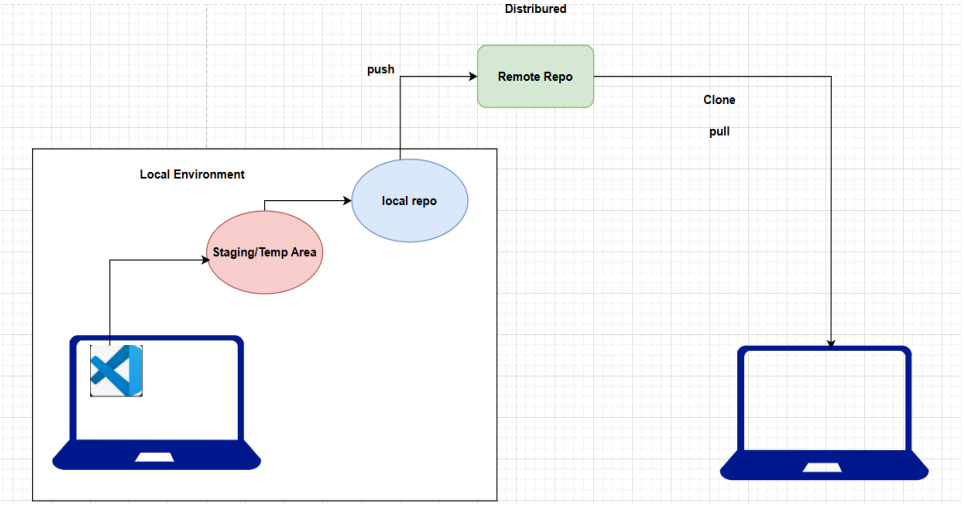
Centralised → Easy to collapse if at one place

Decentralised --> Distributed economy, if one collapse no problem to country

SVN --> Sub version control --> Centralised



Distributed version control system (DVCS)



Git repos purpose is Storing code

1. Creation of repo
2. Clone repo to our laptop

   git clone https://github.com/PEDAMALA/Shell-scripting.git

   git pull origin main →It is used to download the latest updates of cloned repository
3. We develop code
4. Select some editor.

   vscode editor. Visual studio code editor. Free editor
5. Add code to staging area

   git add <file-name>

   git add . → All files will be added to staging area
6. Commit to local repo

   git commit -m "message"
7. git push origin main

.sh --> shell script extension

C shell, K Shell, Z Shell, Shell --> Bash

#!/bin/bash → Shebang or #!/bin/sh→ Shebang(Old version)

Shebang --> It should be the first line of shell script. It is the **interpreter** to execute the commands and syntax inside shell script

which ls →this command shows actual path of particular feature.

/usr/bin/ls

For executing shell script files follow below commands on Linux servers

sh <script-name>

bash <script-name>

ssh ec2-user@18.209.70.119

password is DevOps321

sh Hello_World.sh

bash Hello_World.sh

chmod +x Hello_World.sh

./Hello_World.sh

1. Variables
2. Data types
3. Conditions
4. Functions
5. Loops

Variables

lets take x=1, y=0

derive the formual

finally submit values

A centralise place to mention the values, if you change at one place, it will reflect at all the places where it is referred

DRY == don't repeat yourself

Arguments or args --> run time variables --> no need to edit the script

sh 04-variables.sh ramesh suresh

Where user id and password need to provide, generally these credentials should not mention in the shell script files for security reasons, their Dynamic programming comes into the picture.

READ command is introduced.

```
$ Hello_World.sh ×
$ Hello_World.sh
#!/bin/bash

# This line is commented only first line with Hash is not considered as comment even if it has Hash.

# 1st line with shell script shebang.
echo -e "Hello World!\nsai_kumar kid!" #here \n is used to print in next line
echo -e "Hello World \n sai_kumar kid!"
echo "Hi Venkaiah Swami!"
echo "Hi Sai_Kumar kid!"
```

```
3.84.225.123 | 172.31.23.22 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-23-22 ~/shell-scripting-01 ]$ sh Hello_World.sh
Hello World!
sai_kumar kid!
Hello World
 sai_kumar kid!
Hi Venkaiah Swami!
Hi Sai_Kumar kid!
```

```
$ 01_variables.sh ×
Shell-scripting > $ 01_variables.sh
#!/bin/bash

echo "Sai:: Hi Sarath"
echo "Sarath:: Hello Sai"
echo "Sai:: How are you doing?"
echo "Sarath:: I am good. How are you?"
```

```
dell@SAIKUMAR MINGW64 /D/OM_NARAYANA_ADI_NARAYANA/Repository/Shell-scripting (main)
$ sh 01_variables.sh
Sai:: Hi Sarath
Sarath:: Hello Sai
Sai:: How are you doing?
Sarath:: I am good. How are you?
```

```
$ 02_Actual_Variables.sh ×
Shell-scripting > $ 02_Actual_Variables.sh
#!/bin/bash

PERSON1=Sai # no space before and after equal
PERSON2=Sarath

echo "$PERSON1:: Hi $PERSON2"
echo "${PERSON2}:: Hello $PERSON1"
echo "$PERSON1:: How are you doing?"
echo "$PERSON2:: I am good. How are you?"
```

```
dell@SAIKUMAR MINGW64 /D/OM_NARAYANA_ADI_NARAYANA/Repository/Shell-scripting (main)
$ sh 02_Actual_Variables.sh
Sai:: Hi Sarath
Sarath:: Hello Sai
Sai:: How are you doing?
Sarath:: I am good. How are you?
```

```
$ 03_args_Variables.sh M ×
Shell-scripting > $ 03_args_Variables.sh
#!/bin/bash


# we can provide values to variables while running the script, thus these items($1, $2) as run time variables.

PERSON1=$1
PERSON2=$2

echo "$PERSON1:: Hi $PERSON2"
echo "${PERSON2}:: Hello $PERSON1"
echo "$PERSON1:: How are you doing?"
echo "$PERSON2:: I am good. How are you?"
echo "$PERSON1:: I am also good. what is today's plans?"
```

```
18.209.70.119 | 172.31.31.1 | t3.micro | https://github.com/PEDAMALA/Shell-scripting.git
[ ec2-user@ip-172-31-31-1 ~/Shell-scripting ]$ sh 03_args_Variables.sh IAS IPS
IAS:: Hi IPS
IPS:: Hello IAS
IAS:: How are you doing?
IPS:: I am good. How are you?
```

Where user id and password need to provide, generally these credentials should not mention in the shell script files for security reasons, their Dynamic programming comes into the picture.

READ command is introduced.

```
$ 04_input_variables.sh M ×
Shell-scripting > $ 04_input_variables.sh
#!/bin/bash
echo "Please enter First username::" # text entered here will be added as value to variable
    read  USERNAME_1
echo "USERNAME entered for USERNAME_1 is:: $USERNAME_1"
echo "Please enter password for $USERNAME_1:"
    read  PASSWORD1

# echo "PASSWORD1 entered for USERNAME_1 is:: $PASSWORD1"



echo "Please enter Second username::"
    read -s USERNAME_2 # Here USERNAME is also variable name but -s option is used to hide the input text while providing input
echo "USERNAME entered for USERNAME_2 is:: $USERNAME_2"
echo "Please enter password for $USERNAME_2::"
    read -s PASSWORD2 # -s option is used to hide the input text while providing input

# echo "PASSWORD2 entered for USERNAME_1 is:: $PASSWORD2"
```

```
dell@SAIKUMAR MINGW64 /D/OM_NARAYANA_ADI_NARAYANA/Repository/Shell-scripting (main)
$ sh 04_input_variables.sh
Please enter First username::
SAI
USERNAME entered for USERNAME_1 is:: SAI
Please enter password for SAI:
SAI
Please enter Second username::
USERNAME entered for USERNAME_2 is:: KUMAR
Please enter password for KUMAR::
```

Git hub commands

First create GitHub account and then git bash should be installed in PC then

For code pushing (From my laptop)

go to dedicated(local machine) path with the help of GitBash

cd /D/OM_NARAYANA_ADI_NARAYANA/Repository

git clone https://github.com/PEDAMALA/Shell-scripting.git

git status → displays any changes in main or sub branch in our machine

git add <file-name or Folder Name> → code will be pushed to staging Area

git add Hello_World.sh

git add . → all files will be added to staging area

git commit -m " <any meaning full message> " → here it will ask for identity we have to verify our identity with git hub account credentials, all these to track the history of records for future use.

git log

git push origin main

git pull origin main → used to pull the latest updates, it can be used after clone only

git add . ; git commit -m "<any meaning full message>" ; git push origin main

☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

Session-11: (Watch till 1H: 20Mints)

- Data types
- Arrays
- Conditions
- Special variables
- Exit status

int i=0

var i=0 → this data type (Var) we can see it in javascript

boolean

float

String name = "devops"

integer, float, boolean, string, array, arraylist, map, etc..

integer --> number

float --> decimal number

boolean --> true/false

string --> text

array --> (devops, aws, docker)

arraylist --> [devops, aws, docker]

map --> name: devops, duration: 120hrs

Addition of 2 numbers:-

User must give number1 and number2

Add them, print the sum

How do you run a command inside shell script and get the output

Answer: $(date)

05_Data_types_addition.sh

```
#!/bin/bash

NUMBER1=$1
NUMBER2=$2

TIMESTAMP=$(date) # keep more focus here
echo "Script executed at: $TIMESTAMP"
SUM=$(($NUMBER1+$NUMBER2))
echo "SUM of $NUMBER1 and $NUMBER2 is: $SUM"


echo "Addition completed successfully."
echo "Script execution finished at: $(date)"
```

```
3.84.225.123 | 172.31.23.22 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-23-22 ~/shell-scripting-01 ]$ sh 05_Data_types_addition.sh 100 200
Script executed at: Mon Jul 28 06:12:18 UTC 2025
SUM of 100 and 200 is: 300
Addition completed successfully.
Script execution finished at: Mon Jul 28 06:12:18 UTC 2025
```

Array:-

List of values

MOVIES=("pushpa" "rrr" "devara")

0 1 2

Size is 3...

```
$ 06_Arrays.sh ×
$ 06_Arrays.sh
1  #!/bin/bash
2
3  MOVIES=("pushpa" "rrr" "devara" "kgf")
4  # index starts from 0, size is 3
5
6  echo "First movie: ${MOVIES[0]}"
7  echo "Second movie: ${MOVIES[1]}"
8  echo "Third movie: ${MOVIES[2]}"
9  echo "Third movie: ${MOVIES[3]}"
10
11 echo "All movies are: ${MOVIES[@]}"
```

```
3.84.225.123 | 172.31.23.22 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-23-22 ~/shell-scripting-01 ]$ sh 06_Arrays.sh
First movie: pushpa
Second movie: rrr
Third movie: devara
Third movie: kgf
All movies are: pushpa rrr devara kgf
```

Special variables:-

$1, $2, $3

All variables passed: $@

Number of variables: $#

Script name: $0

Present working directory: $PWD

Home directory of current user: $HOME

Which user is running this script: $USER

Process id of current script: $$

Process id of last command in background of the current script: $!

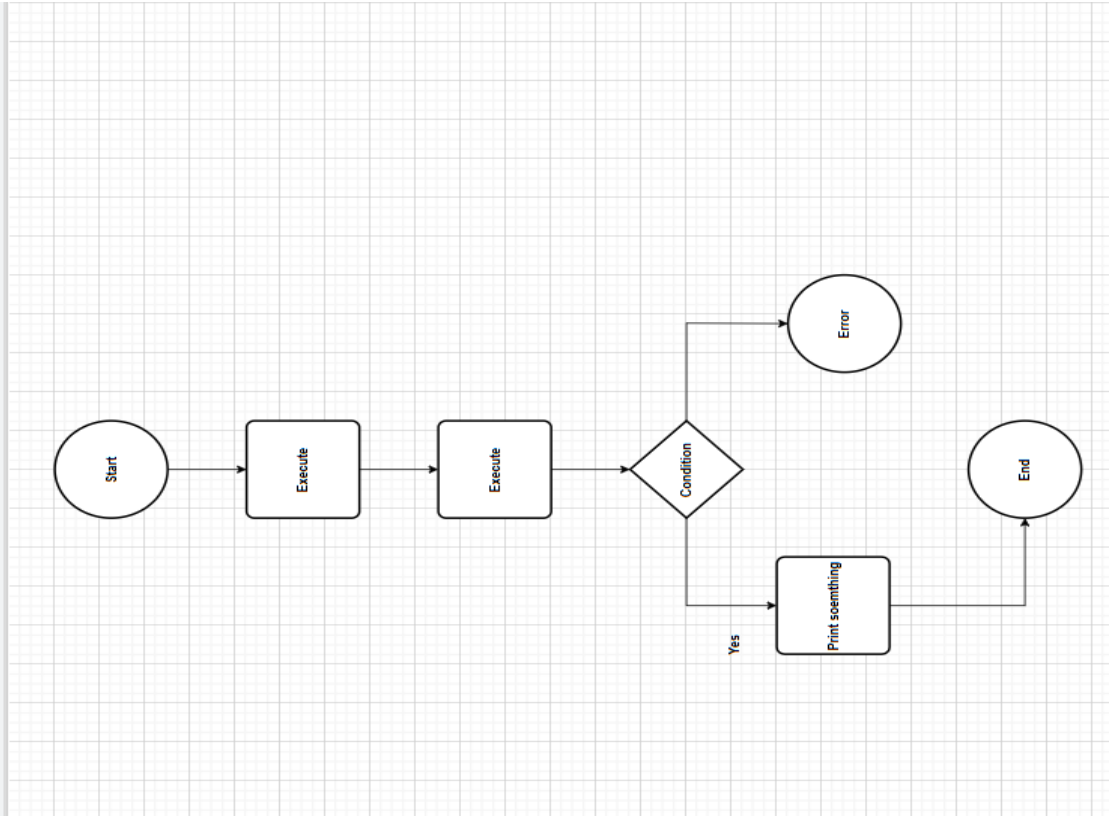
```
$ 07_Special_variables.sh ×
$ 07_Special_variables.sh
#!/bin/bash

echo "All variables passed: $@"
echo "Number of variables: $#"
echo "Script name: $0"
echo "Present working directory: $PWD"
echo "Home directory of current user: $HOME"
echo "Which user is running this script: $USER"
echo "Process id of current script: $$" #it is the process id of the current one i.e sh 07_special_variables.sh
#sleep 60 &
echo "Process id of last command in background: $!"
```

```
3.84.225.123 | 172.31.23.22 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-23-22 ~/shell-scripting-01 ]$ sleep 75 &
[1] 2363

3.84.225.123 | 172.31.23.22 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-23-22 ~/shell-scripting-01 ]$ ps
    PID TTY          TIME CMD
   1306 pts/0    00:00:00 bash
   2363 pts/0    00:00:00 sleep
   2367 pts/0    00:00:00 ps

3.84.225.123 | 172.31.23.22 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-23-22 ~/shell-scripting-01 ]$ sh 07_Special_variables.sh
All variables passed:
Number of variables: 0
Script name: 07_Special_variables.sh
Present working directory: /home/ec2-user/shell-scripting-01
Home directory of current user: /home/ec2-user
Which user is running this script: ec2-user
Process id of current script: 2373
Process id of last command in background:

3.84.225.123 | 172.31.23.22 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-23-22 ~/shell-scripting-01 ]$ ps
    PID TTY          TIME CMD
   1306 pts/0    00:00:00 bash
   2363 pts/0    00:00:00 sleep
   2377 pts/0    00:00:00 ps

3.84.225.123 | 172.31.23.22 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-23-22 ~/shell-scripting-01 ]$
```

```
$ 07_Special_variables.sh ×
$ 07_Special_variables.sh
#!/bin/bash

echo "All variables passed: $@"
echo "Number of variables: $#"
echo "Script name: $0"
echo "Present working directory: $PWD"
echo "Home directory of current user: $HOME"
echo "Which user is running this script: $USER"
echo "Process id of current script: $$" #it is the process id of the current one i.e sh 07_special_variables.sh
sleep 60 &
sleep 200 &
echo "Process id of last command in background in the current script only: $!"
```

```
3.84.225.123 | 172.31.23.22 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-23-22 ~/shell-scripting-01 ]$ sh 07_Special_variables.sh
All variables passed:
Number of variables: 0
Script name: 07_Special_variables.sh
Present working directory: /home/ec2-user/shell-scripting-01
Home directory of current user: /home/ec2-user
Which user is running this script: ec2-user
Process id of current script: 2418
Process id of last command in background in the current script only: 2420

3.84.225.123 | 172.31.23.22 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-23-22 ~/shell-scripting-01 ]$ ps
    PID TTY          TIME CMD
   1306 pts/0    00:00:00 bash
   2419 pts/0    00:00:00 sleep
   2420 pts/0    00:00:00 sleep
   2424 pts/0    00:00:00 ps
```

Conditions




```
if (expression) {
        execute this if expression is true
}
```

```
if (expression){
        execute this if expression is true
}
else {
        execute this if expression is false
}
```

```
if [ expression ]
then
        statements
else
        statements
fi
```

Print holiday or not

1. I need to find what is today
2. If today is not Sunday, I have to go school
3. Otherwise today is holiday

```
$ 101-Practice.sh ×
$ 101-Practice.sh
#!/bin/bash

# Print holiday or not
# 1. Get the current day of the week (e.g., Monday, Tuesday, ...)
# 2. If today is not Sunday, I have to go school
# 3. Otherwise today is holiday


day=$(date +%A)

echo "Today is $day"

# Check if today is Sunday
if [ "$day" = "Sunday" ]
then
    echo "It's a holiday!"
else
    echo "It's a working day."
fi


#-eq is for numeric comparison, not strings.
# To compare strings inside [ ], you should use =
```

```
3.84.225.123 | 172.31.23.22 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-23-22 ~/shell-scripting-01 ]$ sh 101-Practice.sh
Today is Monday
It's a working day.
```

Print a number is greater than 100 or not

1. Get the input number
2. Check it is more than 100 or not
3. If more than 100, print more than 100
4. Otherwise print less than or equal to 100

```
$ 08_Conditions.sh ×
Shell-scripting > $ 08_Conditions.sh
#!/bin/bash

NUMBER=$1

# -gt, -lt, -eq, -ge, -le

if [ $NUMBER -gt 100 ]
then
    echo "Given number is greater than 100"
else
    echo "Given number is less than or equal to 100"
fi
```

```
dell@SAIKUMAR MINGW64 /D/OM_NARAYANA_ADI_NARAYANA/Repository/Shell-scripting (main)
$ sh 08_Conditions.sh 200
Given number is greater than 100
```

-gt → greater than

-lt → less than

-eq → equal

-ne → not equal to

-ge → greater than or equal

-le → Less than or equal

Install MySQL through shell script:-

dnf install mysql -y

check if the user is running the script as a root user or not

if root user

allow him

else

show the error properly and exit the script

Run install command

Check installation is success

If success, our task is done

If not success, throw the error message

Exit status:-

How can you check previous command is success or not in shell script?

By checking the exit status, if exit status is 0 it is success, otherwise it is failure

$?

Example: - ls –l

echo $? → It will print zero on screen

While filling any form if 20 things needs to fill and after filling the form when we try to submit then that website throws messages of term and conditions of all fields. That one is not good user friendly.

Instead while filling each filed, it gives popup to fill correctly then it csn be considered as good user friendly.

```
$ 10_Install_script_mysql U ×
Shell-scripting > $ 10_Install_script_mysql
#!/bin/bash

USERID=$(id -u)

if [ $USERID -ne 0 ]
then
    echo "ERROR:: You must have sudo access to execute this script" # here :: is used to separate the error message from the command
    exit 1 # Here in place of 1 we can provide any value other than 0
fi

dnf list installed mysql
if [ $? -ne 0 ]
then # not installed
    dnf install mysql -y
    if [ $? -ne 0 ]
    then
        echo "Installing MySQL ... FAILURE"
        exit 1
    else
        echo "Installing MySQL ... SUCCESS"
    fi
else
    echo "MySQL is already ... INSTALLED"
fi

dnf list installed git
if [ $? -ne 0 ]
then
    dnf install git -y
    if [ $? -ne 0 ]
    then
        echo "Installing Git ... FAILURE"
        exit 1
    else
        echo "Installing Git ... SUCCESS"
    fi
else
    echo "Git is already ... INSTALLED"
fi
```

```
[ ec2-user@ip-172-31-23-22 ~/shell-scripting-01 ]$ sudo sh 10_Install_script_mysql.sh
Error: No matching Packages to list
Last metadata expiration check: 1:29:03 ago on Mon Jul 28 05:51:51 2025.
Dependencies resolved.
================================================================================================================================
 Package                                  Architecture            Version                     Repository                     Size
================================================================================================================================
Installing:
 mysql                                    x86_64                  8.0.41-2.el9_5              rhel-9-appstream-rhui-rpms     2.8 M
Installing dependencies:
 mariadb-connector-c-config               noarch                  3.2.6-1.el9_0               rhel-9-appstream-rhui-rpms      11 k
 mysql-common                             x86_64                  8.0.41-2.el9_5              rhel-9-appstream-rhui-rpms      77 k

Transaction Summary
================================================================================================================================
Install  3 Packages

Total download size: 2.9 M
Installed size: 60 M
Downloading Packages:
(1/3): mysql-common-8.0.41-2.el9_5.x86_64.rpm                                                   1.1 MB/s |  77 kB     00:00
(2/3): mariadb-connector-c-config-3.2.6-1.el9_0.noarch.rpm                                      157 kB/s |  11 kB     00:00
(3/3): mysql-8.0.41-2.el9_5.x86_64.rpm                                                           17 MB/s | 2.8 MB     00:00
--------------------------------------------------------------------------------------------------------------------------------
Total                                                                                            13 MB/s | 2.9 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                                        1/1
  Installing       : mariadb-connector-c-config-3.2.6-1.el9_0.noarch                                                        1/3
  Installing       : mysql-common-8.0.41-2.el9_5.x86_64                                                                     2/3
  Installing       : mysql-8.0.41-2.el9_5.x86_64                                                                            3/3
  Running scriptlet: mysql-8.0.41-2.el9_5.x86_64                                                                            3/3
  Verifying        : mariadb-connector-c-config-3.2.6-1.el9_0.noarch                                                        1/3
  Verifying        : mysql-8.0.41-2.el9_5.x86_64                                                                            2/3
  Verifying        : mysql-common-8.0.41-2.el9_5.x86_64                                                                     3/3

Installed:
  mariadb-connector-c-config-3.2.6-1.el9_0.noarch          mysql-8.0.41-2.el9_5.x86_64          mysql-common-8.0.41-2.el9_5.x86_64

Complete!
Installing MySQL ... SUCCESS
Installed Packages
git.x86_64                                          2.43.5-1.el9_4                                      @rhel-9-appstream-rhui-rpms
Git is already ... INSTALLED

3.84.225.123 | 172.31.23.22 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-23-22 ~/shell-scripting-01 ]$
```

Reduce number of lines, get the same productivity

< → less than symbol means inputs

mysql -h mysql.psk135.tech -uroot -pExpenseApp@1 < /app/schema/backend.sql

☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

Session-12: (Watch till 1H: 20Mints)

- Functions
- Colors
- Logs and Redirections
- Loops
- Quiz is started @1H:10M

Functions: -

Takes some input and do something

DRY --> don't repeat yourself

Repeated code we can keep in function, give it a name. Whenever you want you can call that function

```
FUNC_NAME () {

        code related to function
}


FUNC_NAME # calling function
```

args --> sh script-name.sh **arg1 arg2** --> $1=arg1 $2=arg2

```
FUNC_NAME input1 input2

FUNC_NAME () {
        $1=input1
        $2=input2
        code related to function
}
```

What it knows and what it does?

$ 11-functions.sh

```
#!/bin/bash

USERID=$(id -u)

VALIDATE(){
    if [ $1 -ne 0 ]
    then
        echo "$2 ... FAILURE"
        exit 1
    else
        echo "$2 ... SUCCESS"
    fi
}

if [ $USERID -ne 0 ]
then
    echo "ERROR:: You must have sudo access to execute this script"
    exit 1 #other than 0
fi

dnf list installed mysql

if [ $? -ne 0 ]
then # not installed
    dnf install mysql -y
    VALIDATE $? "Installing MySQL"
else
    echo "MySQL is already ... INSTALLED"
fi


dnf list installed git

if [ $? -ne 0 ]
then
    dnf install git -y
    VALIDATE $? "Installing Git"
```

```
dnf list installed git

if [ $? -ne 0 ]
then
    dnf install git -y
    VALIDATE $? "Installing Git"
else
    echo "Git is already ... INSTALLED"
fi

```

Colors --> success (green), failure (red), already installed (yellow)

R --> 31
G --> 32
Y --> 33

\e[31m

```
$ 12-colors.sh ×
$ 12-colors.sh
#!/bin/bash

USERID=$(id -u)
R="\e[31m"
G="\e[32m"
Y="\e[33m"
N="\e[0m"

VALIDATE(){
    if [ $1 -ne 0 ]
    then
        echo -e "$2 ... $R FAILURE$N"
        exit 1
    else
        echo -e "$2 ... $G SUCCESS$N"
    fi
}

if [ $USERID -ne 0 ]
then
    echo "ERROR:: You must have sudo access to execute this script"
    exit 1 #other than 0
fi

dnf list installed mysql

if [ $? -ne 0 ]
then # not installed
    dnf install mysql -y
    VALIDATE $? "Installing MySQL"
else
    echo -e "MySQL is already ... $Y INSTALLED$N"
fi


dnf list installed git

dnf list installed git

if [ $? -ne 0 ]
then
    dnf install git -y
    VALIDATE $? "Installing Git"
else
    echo -e "Git is already ... $Y INSTALLED$N"
fi

```

```
3.84.225.123 | 172.31.23.22 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-23-22 ~/shell-scripting-01 ]$ sudo sh 12-colors.sh
Installed Packages
mysql.x86_64                                        8.0.41-2.el9_5                                  @rhel-9-appstream-rhui-rpms
MySQL is already ...  INSTALLED
Installed Packages
git.x86_64                                          2.47.3-1.el9_6                                  @rhel-9-appstream-rhui-rpms
Git is already ...  INSTALLED

3.84.225.123 | 172.31.23.22 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-23-22 ~/shell-scripting-01 ]$
```

logs → logging the result to some file

redirectors

< → Less than symbol means input

> → Greater than symbol means output

1 --> success

2 --> failure

& --> both success and failure

/var/logs/shellscirpt-logs/13-logs.sh.log

script-name.log

13-logs.sh --> 13-logs

13-logs-01-01-2025.log

```
$ 13-logs.sh ×
$ 13-logs.sh
Find

#!/bin/bash

USERID=$(id -u)
R="\e[31m"
G="\e[32m"
Y="\e[33m"
N="\e[0m"

LOGS_FOLDER="/var/log/shellscript-logs"
mkdir -p $LOGS_FOLDER # be careful here, if it is not mentioned , the script will not work as expected
                      # and here -P means pass or skip the process if the folder already exists
LOG_FILE=$(echo $0 | cut -d "." -f1 )
TIMESTAMP=$(date +%Y-%m-%d-%H-%M-%S)
LOG_FILE_NAME="$LOGS_FOLDER/$LOG_FILE-$TIMESTAMP.log"


VALIDATE(){
    if [ $1 -ne 0 ]
    then
        echo -e "$2 ... $R FAILURE $N"
        exit 1
    else
        echo -e "$2 ... $G SUCCESS $N"
    fi
}

echo "Script started executing at: $TIMESTAMP"
echo "Script started executing at: $TIMESTAMP" &>>$LOG_FILE_NAME

if [ $USERID -ne 0 ]
then
    echo "ERROR:: You must have sudo access to execute this script"
    exit 1 #other than 0
fi

dnf list installed mysql &>>$LOG_FILE_NAME
```

```
dnf list installed mysql &>>$LOG_FILE_NAME

if [ $? -ne 0 ]
then # not installed
    dnf install mysql -y &>>$LOG_FILE_NAME
    VALIDATE $? "Installing MySQL"
else
    echo -e "MySQL is already ... $Y INSTALLED $N"
fi


dnf list installed git &>>$LOG_FILE_NAME

if [ $? -ne 0 ]
then
    dnf install git -y &>>$LOG_FILE_NAME
    VALIDATE $? "Installing Git"
else
    echo -e "Git is already ... $Y INSTALLED $N"
fi

echo "Script completed execution at: $TIMESTAMP" &>>$LOG_FILE_NAME
echo "Script completed execution at: $TIMESTAMP"
```

```
5. 54.91.71.90

54.91.71.90 | 172.31.17.239 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-17-239 ~/shell-scripting-01 ]$ sudo sh 13-logs.sh
Script started executing at: 2025-07-29-05-52-42
MySQL is already ...  INSTALLED
Git is already ...  INSTALLED
Script completed execution at: 2025-07-29-05-52-42

54.91.71.90 | 172.31.17.239 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-17-239 ~/shell-scripting-01 ]$
```

```
54.91.71.90 | 172.31.17.239 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-17-239 ~/shell-scripting-01/expense-project-shell ]$ ls -l /var/log/shellscript-logs/
total 48
-rw-r--r-- 1 root root 2749 Jul 29 05:12 13-logs-2025-07-29-05-08-39.log
-rw-r--r-- 1 root root  300 Jul 29 05:13 13-logs-2025-07-29-05-13-50.log
-rw-r--r-- 1 root root  300 Jul 29 05:33 13-logs-2025-07-29-05-33-55.log
-rw-r--r-- 1 root root  300 Jul 29 05:34 13-logs-2025-07-29-05-34-06.log
-rw-r--r-- 1 root root  300 Jul 29 05:49 13-logs-2025-07-29-05-49-16.log
-rw-r--r-- 1 root root  300 Jul 29 05:52 13-logs-2025-07-29-05-52-42.log
-rw-r--r-- 1 root root   49 Jul 29 06:37 15-apps-install-with-loops-2025-07-29-06-37-24.log
-rw-r--r-- 1 root root 3022 Jul 29 06:37 15-apps-install-with-loops-2025-07-29-06-37-48.log
-rw-r--r-- 1 root root 7088 Jul 29 06:44 15-apps-install-with-loops-2025-07-29-06-44-36.log
-rw-r--r-- 1 root root   49 Jul 29 06:30 15-loops-2025-07-29-06-30-57.log
-rw-r--r-- 1 root root  249 Jul 29 06:33 15-loops-2025-07-29-06-33-43.log
```

Variables, data types, conditions, functions, loops → mostly (80%) all Programing languages need this topics only

loops

In C language

```
for(int i=0; i<100; i++){
        print $i
}
```

In shell scripting

```
for i in {0..1000}
do
        echo $i
done
```

```
$ 14-loops.sh ×
$ 14-loops.sh
1   #!/bin/bash
2
3   for i in {0..5}
4   do
5       echo $i
6   done
7
8   # echo 1
9   # echo 2
10  # echo 3
11  # echo 4
12  # echo 5
```

```
54.91.71.90 | 172.31.17.239 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-17-239 ~/shell-scripting-01 ]$ sh 14-loops.sh
0
1
2
3
4
5

54.91.71.90 | 172.31.17.239 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-17-239 ~/shell-scripting-01 ]$
```

sh install-script git mysql gcc nginx

package=git
package=mysql

```
$ 15-apps-install-with-loops.sh ×
$ 15-apps-install-with-loops.sh
#!/bin/bash

USERID=$(id -u)
R="\e[31m"
G="\e[32m"
Y="\e[33m"
N="\e[0m"


LOGS_FOLDER="/var/log/shellscript-logs"
mkdir -p $LOGS_FOLDER
LOG_FILE=$(echo $0 | cut -d "." -f1 )
TIMESTAMP=$(date +%Y-%m-%d-%H-%M-%S)
LOG_FILE_NAME="$LOGS_FOLDER/$LOG_FILE-$TIMESTAMP.log"

VALIDATE(){
    if [ $1 -ne 0 ]
    then
        echo -e "$2 ... $R FAILURE $N"
        exit 1
    else
        echo -e "$2 ... $G SUCCESS $N"
    fi
}

CHECK_ROOT(){
    if [ $USERID -ne 0 ]
    then
        echo "ERROR:: You must have sudo access to execute this script"
        exit 1 #other than 0
    fi
}

echo "Script started executing at: $TIMESTAMP" &>>$LOG_FILE_NAME

CHECK_ROOT

echo "Script started executing at: $TIMESTAMP"
CHECK_ROOT

for package in $@ #it is a list of packages passed as arguments, it changed the entire script to accept multiple packages
do
    dnf list installed $package &>>$LOG_FILE_NAME
    if [ $? -ne 0 ]
    then
        dnf install $package -y &>>$LOG_FILE_NAME
        VALIDATE $? "Installing $package"
    else
        echo -e "$package is already $Y ... INSTALLED $N"
    fi
done

echo "Script completed executing at: $TIMESTAMP"
```

```
54.91.71.90 | 172.31.17.239 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-17-239 ~/shell-scripting-01 ]$ sudo sh 15-apps-install-with-loops.sh mysql-server nodejs nginx
Script started executing at: 2025-07-29-06-44-36
Installing mysql-server ...  SUCCESS
Installing nodejs ...  SUCCESS
nginx is already  ... INSTALLED
Script completed executing at: 2025-07-29-06-44-36

54.91.71.90 | 172.31.17.239 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-17-239 ~/shell-scripting-01 ]$
```

☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

Session-13:

- Expense project using shell
  - MySQL
  - Backend
  - Frontend
- Idempotency
- Delete old logs in a folder

Variables, data types, conditions, functions, loops → mostly (80%) all Programing languages need this topics only, now by using them we will automate app installing process on Linux server with script.

Plain server → app runtime (nodejs), create user, create app folder, download the code, install dependencies, create systemctl services, and start the application

Check user has root access or not
Store logs
Try to use colors

Install mysql-server
Enable it
Start it
Set the root password

$ mysql.sh

expense-project-shell > $ mysql.sh

We can treat it as common code for all scripts

```bash
#!/bin/bash

USERID=$(id -u)
R="\e[31m"
G="\e[32m"
Y="\e[33m"
N="\e[0m"

LOGS_FOLDER="/var/log/expense-logs"
mkdir -p $LOGS_FOLDER # here -P means pass or skip the process if the folder already exists
LOG_FILE=$(echo $0 | cut -d "." -f1 )
TIMESTAMP=$(date +%Y-%m-%d-%H-%M-%S)
LOG_FILE_NAME="$LOGS_FOLDER/$LOG_FILE-$TIMESTAMP.log"

VALIDATE(){
    if [ $1 -ne 0 ]
    then
        echo -e "$2 ... $R FAILURE $N"
        exit 1
    else
        echo -e "$2 ... $G SUCCESS $N"
    fi
}

CHECK_ROOT(){
    if [ $USERID -ne 0 ]
    then
        echo "ERROR:: You must have sudo access to execute this script"
        exit 1 #other than 0
    fi
}

echo "Script started executing at: $TIMESTAMP"
echo "Script started executing at: $TIMESTAMP" &>>$LOG_FILE_NAME
```

$ mysql.sh

expense-project-shell > $ mysql.sh

```bash
CHECK_ROOT

dnf install mysql-server -y &>>$LOG_FILE_NAME
VALIDATE $? "Installing MySQL Server"

systemctl enable mysqld &>>$LOG_FILE_NAME
VALIDATE $? "Enabling MySQL Server"

systemctl start mysqld &>>$LOG_FILE_NAME
VALIDATE $? "Starting MySQL Server"

mysql -h mysql.daws82s.online -u root -pExpenseApp@1 -e 'show databases;' &>>$LOG_FILE_NAME

if [ $? -ne 0 ]
then
    echo "MySQL Root password not setup" &>>$LOG_FILE_NAME
    mysql_secure_installation --set-root-pass ExpenseApp@1
    VALIDATE $? "Setting Root Password"
else
    echo -e "MySQL Root password already setup ... $Y SKIPPING $N"
fi

TIMESTAMP1=$(date +%Y-%m-%d-%H-%M-%S)
echo "Script completed executing at: $TIMESTAMP1"
```

```
54.91.130.52 | 172.31.80.170 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-80-170 ~/shell-scripting-01/expense-project-shell ]$ sudo sh mysql.sh
Script started executing at: 2025-07-30-06-24-33
Installing MySQL Server ...  SUCCESS
Enabling MySQL Server ...  SUCCESS
Starting MySQL Server ...  SUCCESS
Setting Root Password ...  SUCCESS
Script completed executing at: 2025-07-30-06-24-33

54.91.130.52 | 172.31.80.170 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-80-170 ~/shell-scripting-01/expense-project-shell ]$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 4 (delta 3), reused 4 (delta 3), pack-reused 0 (from 0)
Unpacking objects: 100% (4/4), 344 bytes | 172.00 KiB/s, done.
From https://github.com/PEDAMALA/shell-scripting-01
   1114976..a343ff3  main       -> origin/main
Updating 1114976..a343ff3
Fast-forward
 expense-project-shell/mysql.sh | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)

54.91.130.52 | 172.31.80.170 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-80-170 ~/shell-scripting-01/expense-project-shell ]$ sudo sh mysql.sh
Script started executing at: 2025-07-30-06-31-08
Installing MySQL Server ...  SUCCESS
Enabling MySQL Server ...  SUCCESS
Starting MySQL Server ...  SUCCESS
Password already set, You cannot reset the password with mysql_secure_installation
Setting Root Password ...  SUCCESS
Script completed executing at: 2025-07-30-06-33-28

54.91.130.52 | 172.31.80.170 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-80-170 ~/shell-scripting-01/expense-project-shell ]$
```

Idempotency --> even you run any number of times, it should not change the result

HTTP GET --> idempotent
HTTP POST --> chance of duplicates or errors, we need to handle this in programming
HTTP PUT --> no problem, but we can say it is already updated
HTTP DELETE --> chance of error, resource not found. Handle this in scripting/programming

Deployment --> updating new version

Remove old code
Download new code
Install dependencies
Restart the server --> stop and start

Delete old logs in Linux server

14 days log files --> will be in server
Archive and move to storage servers

Delete files older than 14 days from now
only delete **.log** files

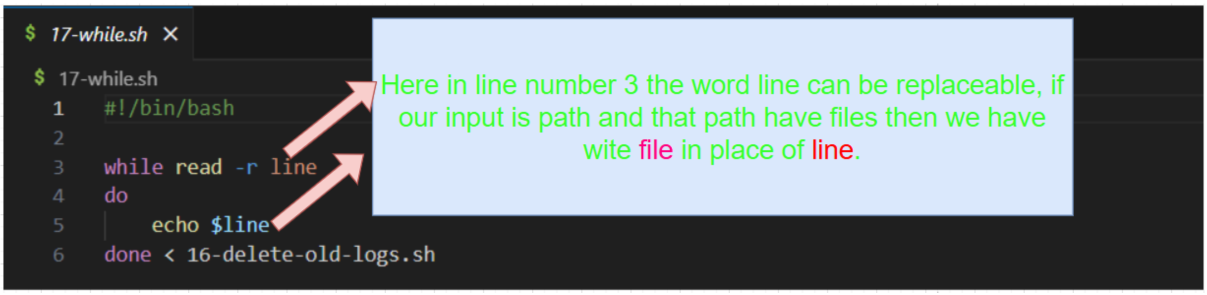
touch –d YYYYMMDD Filename → here this command creates file with whatever date we needed.

touch –d 20240101 shippping.log
touch –d "365 days ago" ship.txt

```
44.208.164.149 | 172.31.85.94 | t2.micro | null
[ ec2-user@ip-172-31-85-94 ~ ]$ find . -name "*.log"
./expense.log
./shippping.log
./ship.log
./sai.log

44.208.164.149 | 172.31.85.94 | t2.micro | null
[ ec2-user@ip-172-31-85-94 ~ ]$ find . -name "*.log" -mtime +14
./ship.log
./sai.log

44.208.164.149 | 172.31.85.94 | t2.micro | null
[ ec2-user@ip-172-31-85-94 ~ ]$ ls -l
total 0
-rw-r--r-- 1 ec2-user ec2-user 0 Jul 31 03:06  20240101
-rw-r--r-- 1 ec2-user ec2-user 0 Jul 31 03:06  20240201
-rw-r--r-- 1 ec2-user ec2-user 0 Jul 31 02:59 '65 days ago'
-rw-r--r-- 1 ec2-user ec2-user 0 Jul 31 02:59  expense.log
-rw-r--r-- 1 ec2-user ec2-user 0 Jul  1 03:00  oldfile.txt
-rw-r--r-- 1 ec2-user ec2-user 0 Jul  1 03:18  sai.log
-rw-r--r-- 1 ec2-user ec2-user 0 Feb  2  2024  sheeip.text
-rw-r--r-- 1 ec2-user ec2-user 0 Jan  1  2024  ship.log
-rw-r--r-- 1 ec2-user ec2-user 0 Jan  2  2024  ship.text
-rw-r--r-- 1 ec2-user ec2-user 0 Jul 31 03:06  shippping.log
-rw-r--r-- 1 ec2-user ec2-user 0 Jul 31 03:06  –d
```

cd /var/log
chown ec2-user:ec2-user -R shellscript-logs

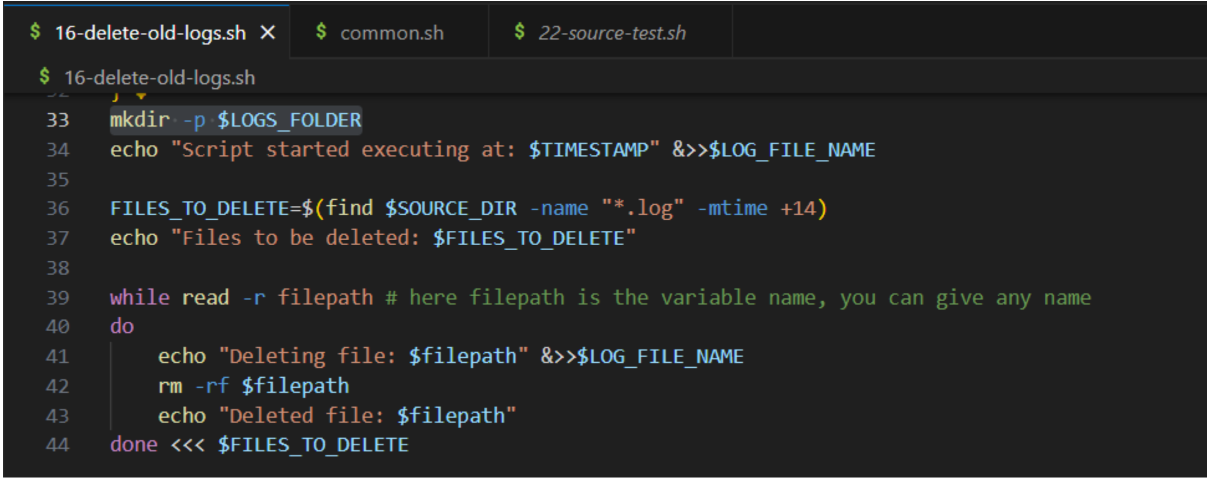
While loop:-



```
1  #!/bin/bash
2
3  while read -r line
4  do
5      echo $line
6  done < 16-delete-old-logs.sh
```

Here in line number 3 the word line can be replaceable, if our input is path and that path have files then we have wite file in place of line.

```
#!/bin/bash

USERID=$(id -u)
R="\e[31m"
G="\e[32m"
Y="\e[33m"
N="\e[0m"

SOURCE_DIR="/home/ec2-user/app-logs"

LOGS_FOLDER="/var/log/shellscript-logs"
LOG_FILE=$(echo $0 | cut -d "." -f1 )
TIMESTAMP=$(date +%Y-%m-%d-%H-%M-%S)
LOG_FILE_NAME="$LOGS_FOLDER/$LOG_FILE-$TIMESTAMP.log"

VALIDATE(){
    if [ $1 -ne 0 ]
    then
        echo -e "$2 ... $R FAILURE $N"
        exit 1
    else
        echo -e "$2 ... $G SUCCESS $N"
    fi
}

CHECK_ROOT(){
    if [ $USERID -ne 0 ]
    then
        echo "ERROR:: You must have sudo access to execute this script"
        exit 1 #other than 0
    fi
}
mkdir -p $LOGS_FOLDER
echo "Script started executing at: $TIMESTAMP" &>>$LOG_FILE_NAME

FILES_TO_DELETE=$(find $SOURCE_DIR -name "*.log" -mtime +14)
echo "Files to be deleted: $FILES_TO_DELETE"
```



```
mkdir -p $LOGS_FOLDER
echo "Script started executing at: $TIMESTAMP" &>>$LOG_FILE_NAME

FILES_TO_DELETE=$(find $SOURCE_DIR -name "*.log" -mtime +14)
echo "Files to be deleted: $FILES_TO_DELETE"

while read -r filepath # here filepath is the variable name, you can give any name
do
    echo "Deleting file: $filepath" &>>$LOG_FILE_NAME
    rm -rf $filepath
    echo "Deleted file: $filepath"
done <<< $FILES_TO_DELETE
```

Interview scope of question.

- find top 5 repetitive words in a file?
  - Read the file,
  - count the number of words,
  - find top 5 repetitive words

☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

Session-14:

- Backup script
- Cronjob
- Script as native command

app --> app logs
daily schedule few jobs, they run in particular time every, archieve the logs and move it to seperate folder

source directory
zip the files
destination directory
how many days old logs --> optional. If user provides number of days we take them. otherwise we take 14 days by default

1. user may forget to provide source and dest directory. throw the error with proper usage
2. user may forget one of these 2 parameters. throw the error with proper usage
3. user may give both. but they may not exist. throw the error with proper usage
4. find the files
5. if files are there zip it
6. if zip success, then remove the files

$# --> number of parameters

find <DIR> -name "*.log" –mtime +<days>

If there are files, I can zip.
    If there are no files. I can't zip

app-logs-$TIMESTAMP.zip

I should check zip is success or not, if success then I should delete the files. if failure I should throw the error

18-backup.sh

```bash
#!/bin/bash

R="\e[31m"
G="\e[32m"
Y="\e[33m"
N="\e[0m"

SOURCE_DIR=$1 # /home/ec2-user/app-logs
DEST_DIR=$2 # /home/ec2-user/archieve
DAYS=${3:-14} # if user is not providing number of days, we are taking 14 as default

LOGS_FOLDER="/home/ec2-user/shellscript-logs"
LOG_FILE=$(echo $0 | awk -F "/" '{print $NF}' | cut -d "." -f1 )
TIMESTAMP=$(date +%Y-%m-%d-%H-%M-%S)
LOG_FILE_NAME="$LOGS_FOLDER/$LOG_FILE-$TIMESTAMP.log"

USAGE(){
    #echo -e "$R USAGE:: $N sh 18-backup.sh <SOURCE_DIR> <DEST_DIR> <DAYS(Optional)>"
    echo -e "$R USAGE:: $N backup <SOURCE_DIR> <DEST_DIR> <DAYS(Optional)>"
    exit 1
}

mkdir -p /home/ec2-user/shellscript-logs

if [ $# -lt 2 ]
then
    USAGE
fi

if [ ! -d "$SOURCE_DIR" ]
then
    echo -e "$SOURCE_DIR Does not exist...Please check"
    exit 1
fi

if [ ! -d "$DEST_DIR" ]
then
    echo -e "$DEST_DIR Does not exist...Please check"
    exit 1
fi
```


```bash
if [ ! -d "$DEST_DIR" ]
then
    echo -e "$DEST_DIR Does not exist...Please check"
    exit 1
fi
```


```bash
echo "Script started executing at: $TIMESTAMP" &>>$LOG_FILE_NAME

FILES=$(find $SOURCE_DIR -name "*.log" -mtime +$DAYS)

if [ -n "$FILES" ] # true if there are files to zip and Here double quotes are important
then
    echo "Files are: $FILES"
    ZIP_FILE="$DEST_DIR/app-logs-$TIMESTAMP.zip"
    find $SOURCE_DIR -name "*.log" -mtime +$DAYS | zip -@ "$ZIP_FILE"

    if [ -f "$ZIP_FILE" ]
    then
        echo -e "Successfully created zip file for files older than $DAYS"

        while read -r filepath # here filepath is the variable name, you can give any name
        do
            echo "Deleting file: $filepath" &>>$LOG_FILE_NAME
            rm -rf $filepath
            echo "Deleted file: $filepath"
        done <<< $FILES

    else
        echo -e "$R Error:: $N Failed to create ZIP file "
        exit 1
    fi

else
    echo "No files found older than $DAYS"
fi
```

Before running above script make sure that Zip must installed in your server

```
44.208.164.149 | 172.31.85.94 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-85-94 ~/shell-scripting-01 ]$ sh 18-backup.sh /home/ec2-user/app-logs/ /home/ec2-user/archive/
Files are: /home/ec2-user/app-logs/batabase.log
/home/ec2-user/app-logs/frontend.log
/home/ec2-user/app-logs/backend.log
  adding: home/ec2-user/app-logs/batabase.log (stored 0%)
  adding: home/ec2-user/app-logs/frontend.log (stored 0%)
  adding: home/ec2-user/app-logs/backend.log (stored 0%)
Successfully created zip file for files older than 14
Deleted file: /home/ec2-user/app-logs/batabase.log
Deleted file: /home/ec2-user/app-logs/frontend.log
Deleted file: /home/ec2-user/app-logs/backend.log

44.208.164.149 | 172.31.85.94 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-85-94 ~/shell-scripting-01 ]$ sh 18-backup.sh /home/ec2-user/app-logs/ /home/ec2-user/archive/
No files found older than 14

44.208.164.149 | 172.31.85.94 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-85-94 ~/shell-scripting-01 ]$ sh 18-backup.sh /home/ec2-user/app-logs/ /home/ec2-user/archive/ 100
No files found older than 100

44.208.164.149 | 172.31.85.94 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-85-94 ~/shell-scripting-01 ]$
```

crontab → To schedule the scripts as per your timeline

Crontab –e → command to open crontab

```
3. ShellScript   2. ShellScript   5. ShellScript   7. ShellScript
* * * * * sh /home/ec2-user/shell-scripting-01/18-backup.sh /home/ec2-user/app-logs /home/ec2-user/archieve
~
~
~
```

Here cronjob is scheduled for every mint to run

Crontab –l → for testing of crontab

sudo tail -f /var/log/cron → command to open cron logs

```
Aug  2 05:47:01 ip-172-31-94-26 crond[1256]: (ec2-user) RELOAD (/var/spool/cron/ec2-user)
Aug  2 05:47:01 ip-172-31-94-26 CROND[14319]: (ec2-user) CMD (sh /home/ec2-user/shell-scripting-01/18-backup.sh /home/ec2-user/app-logs /home/ec2-user/archiev
e)
Aug  2 05:47:01 ip-172-31-94-26 CROND[14318]: (ec2-user) CMDOUT (Files are: /home/ec2-user/app-logs/fronte.log)
Aug  2 05:47:01 ip-172-31-94-26 CROND[14318]: (ec2-user) CMDOUT (  adding: home/ec2-user/app-logs/fronte.log (stored 0%))
Aug  2 05:47:01 ip-172-31-94-26 CROND[14318]: (ec2-user) CMDOUT (Successfully created zip file for files older than 14)
Aug  2 05:47:01 ip-172-31-94-26 CROND[14318]: (ec2-user) CMDOUT (Deleted file: /home/ec2-user/app-logs/fronte.log)
Aug  2 05:47:01 ip-172-31-94-26 CROND[14318]: (ec2-user) CMDEND (sh /home/ec2-user/shell-scripting-01/18-backup.sh /home/ec2-user/app-logs /home/ec2-user/arch
ieve)
Aug  2 05:48:01 ip-172-31-94-26 CROND[14331]: (ec2-user) CMD (sh /home/ec2-user/shell-scripting-01/18-backup.sh /home/ec2-user/app-logs /home/ec2-user/archieve)
Aug  2 05:48:01 ip-172-31-94-26 CROND[14330]: (ec2-user) CMDOUT (No files found older than 14)
Aug  2 05:48:01 ip-172-31-94-26 CROND[14330]: (ec2-user) CMDEND (sh /home/ec2-user/shell-scripting-01/18-backup.sh /home/ec2-user/app-logs /home/ec2-user/archieve)
```

https://crontab.guru/ → for reference and it is all about 5 stars

* * * * *

home/ec2-user/shellscript-logs//home/ec2-user/shell-script/18-backup-2025-01-06-02-54-01.log

18-backup.sh
awk

It is about "How can we use our shell script as NATIVE LINUX COMMAND"

ls --> C language

Sudo cp 18-backup.sh /usr/local/bin/backup

Sudo chmod +x /usr/local/bin/backup → in this path we have limitation of Crontab to overcome it…..use below command it is not a permanent fix…

Sudo cp /usr/local/bin/backup /bin/backup

```
34.227.161.51 | 172.31.94.26 | t2.micro | null
[ ec2-user@ip-172-31-94-26 ~ ]$ crontab -l
* * * * * backup /home/ec2-user/app-logs /home/ec2-user/archieve
```

☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

Session-15:

- Disk usage
- Sending email from shell
- Running other scripts from current shell script
- Quiz

I recently developed a backup script for our Linux servers. I installed the script in /usr/local/bin and tested in the server. It worked well. So I configured it into crontab. But next day morning when I come to office I checked it is failed..

```
34.227.161.51 | 172.31.94.26 | t2.micro | null
[ ec2-user@ip-172-31-94-26 ~ ]$ $PATH
-bash: /home/ec2-user/.local/bin:/home/ec2-user/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin: No such file or directory
```

/home/ec2-user/.local/bin:

/home/ec2-user/bin:

/usr/local/bin: --> Customised commands

/usr/bin: --> System commands for normal user

/usr/local/sbin: --> customised super user commands

/usr/sbin --> System super user commands

```
[ ec2-user@ip-172-31-94-26 ~/shell-scripting-01 ]$ ls -l /
total 20
dr-xr-xr-x.   2 root root    6 Aug  9  2021 afs
lrwxrwxrwx.   1 root root    7 Aug  9  2021 bin -> usr/bin
dr-xr-xr-x.   5 root root 4096 Feb 22  2024 boot
drwxr-xr-x   19 root root 3220 Aug  2 04:54 dev
drwxr-xr-x    2 root root    0 Aug  2 04:54 efi
drwxr-xr-x.  89 root root 8192 Aug  2 05:34 etc
drwxr-xr-x.   4 root root   39 Aug 31  2024 home
lrwxrwxrwx.   1 root root    7 Aug  9  2021 lib -> usr/lib
lrwxrwxrwx.   1 root root    9 Aug  9  2021 lib64 -> usr/lib64
drwxr-xr-x.   2 root root    6 Aug  9  2021 media
drwxr-xr-x.   2 root root    6 Aug  9  2021 mnt
drwxr-xr-x.   3 root root   17 Feb 22  2024 opt
dr-xr-xr-x  225 root root    0 Aug  2 04:54 proc
dr-xr-x---.   4 root root  151 Aug 31  2024 root
drwxr-xr-x   28 root root  840 Aug  2 04:54 run
lrwxrwxrwx.   1 root root    8 Aug  9  2021 sbin -> usr/sbin
drwxr-xr-x.   2 root root    6 Aug  9  2021 srv
drwxr-xr-x.   2 root root    6 Feb 22  2024 swap
dr-xr-xr-x   13 root root    0 Aug  2 04:54 sys
drwxrwxrwt.   9 root root 4096 Aug  2 06:22 tmp
drwxr-xr-x.  12 root root  144 Feb 22  2024 usr
drwxr-xr-x.  19 root root  264 Aug 31  2024 var
```

/bin --> softlink to /usr/bin.. So keep the commands in /usr/bin

hash -r --> reload the path cache

Hash –r → command to reload cash memory in server

```
34.227.161.51 | 172.31.94.26 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-94-26 ~/shell-scripting-01 ]$ hash -r
```

```
34.227.161.51 | 172.31.94.26 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-94-26 ~/shell-scripting-01 ]$ which backup
/usr/local/bin/backup
```

```
34.227.161.51 | 172.31.94.26 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-94-26 ~/shell-scripting-01 ]$ crontab -l
* * * * * echo "$PATH"
* * * * * backup /home/ec2-user/app-logs /home/ec2-user/archieve
```

```
34.227.161.51 | 172.31.94.26 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-94-26 ~/shell-scripting-01 ]$ sudo tail -f /var/log/cron
Aug  2 07:24:01 ip-172-31-94-26 CROND[15472]: (ec2-user) CMDEND (echo "$PATH")
Aug  2 07:24:01 ip-172-31-94-26 CROND[15474]: (ec2-user) CMD (backup /home/ec2-user/app-logs /home/ec2-user/archieve)
Aug  2 07:24:01 ip-172-31-94-26 CROND[15471]: (ec2-user) CMDOUT (No files found older than 14)
Aug  2 07:24:01 ip-172-31-94-26 CROND[15471]: (ec2-user) CMDEND (backup /home/ec2-user/app-logs /home/ec2-user/archieve)
```

/usr/bin:/bin →before change crontab considers only this path

crontab environment is minimal, it is not the same environment as when I run manual

PATH=/usr/bin:/bin:/usr/local/bin

```
3. ShellScript    2. ShellScript    5. ShellScr
PATH=/usr/bin:/bin:/usr/local/bin
* * * * * echo "$PATH"
* * * * * backup /home/ec2-user/app-logs /home/ec2-user/archieve
~
~
~
~
~
~
```

```
34.227.161.51 | 172.31.94.26 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-94-26 ~/shell-scripting-01 ]$ crontab -e
crontab: installing new crontab
```

/usr/bin:/bin:/usr/local/bin → After change crontab considers these paths

Monitor Linux servers disk usage, send an email if any disk is using more than 80%

```
34.227.161.51 | 172.31.94.26 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-94-26 ~/shell-scripting-01 ]$ df -hT
Filesystem                  Type      Size  Used Avail Use% Mounted on
devtmpfs                    devtmpfs  4.0M     0  4.0M   0% /dev
tmpfs                       tmpfs     377M     0  377M   0% /dev/shm
tmpfs                       tmpfs     151M  2.5M  149M   2% /run
/dev/mapper/RootVG-rootVol  xfs       6.0G  1.8G  4.2G  30% /
/dev/mapper/RootVG-homeVol  xfs       960M   41M  920M   5% /home
/dev/mapper/RootVG-varVol   xfs       2.0G  443M  1.6G  23% /var
/dev/mapper/RootVG-logVol   xfs       2.0G   66M  1.9G   4% /var/log
/dev/mapper/RootVG-varTmpVol xfs      2.0G   47M  1.9G   3% /var/tmp
/dev/mapper/RootVG-auditVol xfs       4.4G   65M  4.3G   2% /var/log/audit
/dev/xvda3                  xfs       424M  223M  202M  53% /boot
/dev/xvda2                  vfat      122M  7.0M  115M   6% /boot/efi
tmpfs                       tmpfs      76M     0   76M   0% /run/user/1001
```

from email (pedamalasaikumar123@gmail.com),

to email (pvssai135@gmail.com)

lvbhmofihsifwyen → app password only for temporary use of only this application(to use this from mail must enable 2 step verification)

sudo su –

dnf -y install postfix cyrus-sasl-plain mutt

systemctl restart postfix

systemctl enable postfix

vim /etc/postfix/main.cf

relayhost = [smtp.gmail.com]:587

smtp_use_tls = yes

smtp_sasl_auth_enable = yes

smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd

smtp_sasl_security_options = noanonymous

smtp_sasl_tls_security_options = noanonymous

smtpd_tls_CAfile = /etc/ssl/certs/ca-certificates.crt

create app password from sender mail id

touch /etc/postfix/sasl_passwd

vim /etc/postfix/sasl_passwd

[smtp.gmail.com]:587 pedamalasaikumar123@gmail.com:xzjatdxycejaduwp

postmap /etc/postfix/sasl_passwd

vim /etc/ssl/openssl.cnf

legacy = legacy

systemctl restart postfix

dnf install mutt –y

echo "This is a test mail & Date $(date)" | mutt -s "Hi pvssai" pvssai135@gmail.com

How do you run other scripts from current shell script?

sh <script-name> → sh 21-script-1.sh

20-script-1.sh 21-script-2.sh

```
#!/bin/bash

MSG="SCRIPT1"
GREET="Hi from SCRIPT-1"


echo "Hello from: $MSG"


sh 21-script-2.sh
```

```
#!/bin/bash

MSG="SCRIPT2"

echo "Hello from: $MSG"

echo "Greeting: $GREET"
```

```
34.227.161.51 | 172.31.94.26 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-94-26 ~/shell-scripting-01 ]$ sh 20-script-1.sh
Hello from: SCRIPT1
Hello from: SCRIPT2
Greeting:
```

Here GREET it runs in seperate process, cant access variables of script1.

source ./script-name

```
#!/bin/bash

MSG="SCRIPT1"
GREET="Hi from SCRIPT-1"
source ./21-script-2.sh # This will execute 21-script-2.sh and p

echo "Hello from: $MSG"
echo "A value: $A"

#sh 21-script-2.sh
```

```
#!/bin/bash

MSG="SCRIPT2"
A="10"

echo "Hello from: $MSG"

echo "Greeting: $GREET"
```

```
34.227.161.51 | 172.31.94.26 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-94-26 ~/shell-scripting-01 ]$ sh 20-script-1.sh
Hello from: SCRIPT2
Greeting: Hi from SCRIPT-1
Hello from: SCRIPT2
A value: 10
```

2nd script executes in the process of script-1. So we can access script2 variables also……. but some variable's values are over-ridded (MSG variable value of 20-script-1 value is replaced by second script)

common.sh

```bash
#!/bin/bash

USERID=$(id -u)
R="\e[31m"
G="\e[32m"
Y="\e[33m"
N="\e[0m"

LOGS_FOLDER="/var/log/shellscript-logs"
#sudo mkdir -p $LOGS_FOLDER
LOG_FILE=$(echo $0 | cut -d "." -f1 )
TIMESTAMP=$(date +%Y-%m-%d-%H-%M-%S)
LOG_FILE_NAME="$LOGS_FOLDER/$LOG_FILE-$TIMESTAMP.log"

VALIDATE(){
    if [ $1 -ne 0 ]
    then
        echo -e "$2 ... $R FAILURE $N"
        exit 1
    else
        echo -e "$2 ... $G SUCCESS $N"
    fi
}
```

22-source-test.sh

```bash
#!/bin/bash

source ./common.sh

SOURCE_DIR="/home/ec2-user/app-logs"
echo "Script started executing at: $TIMESTAMP" &>>$LOG_FILE_NAME

FILES_TO_DELETE=$(find $SOURCE_DIR -name "*.log" -mtime +14)
echo "Files to be deleted: $FILES_TO_DELETE"

while read -r filepath # here filepath is the variable name, you can give any name
do
    echo "Deleting file: $filepath" &>>$LOG_FILE_NAME
    rm -rf $filepath
    echo "Deleted file: $filepath"
done <<< $FILES_TO_DELETE
```

3. ShellScript

```
34.227.161.51 | 172.31.94.26 | t2.micro | https://github.com/PEDAMALA/shell-scripting-01.git
[ ec2-user@ip-172-31-94-26 ~/shell-scripting-01 ]$ sudo sh 22-source-test.sh
Files to be deleted:
Deleted file:
```

😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊😊