**Bash-Scripting-02-Tasks**

**In bash scripts**

**#!//bin/bash : Shebang characters to interpret the scripts**

**### : To write the comments what we are doing and what we are executing with the script**

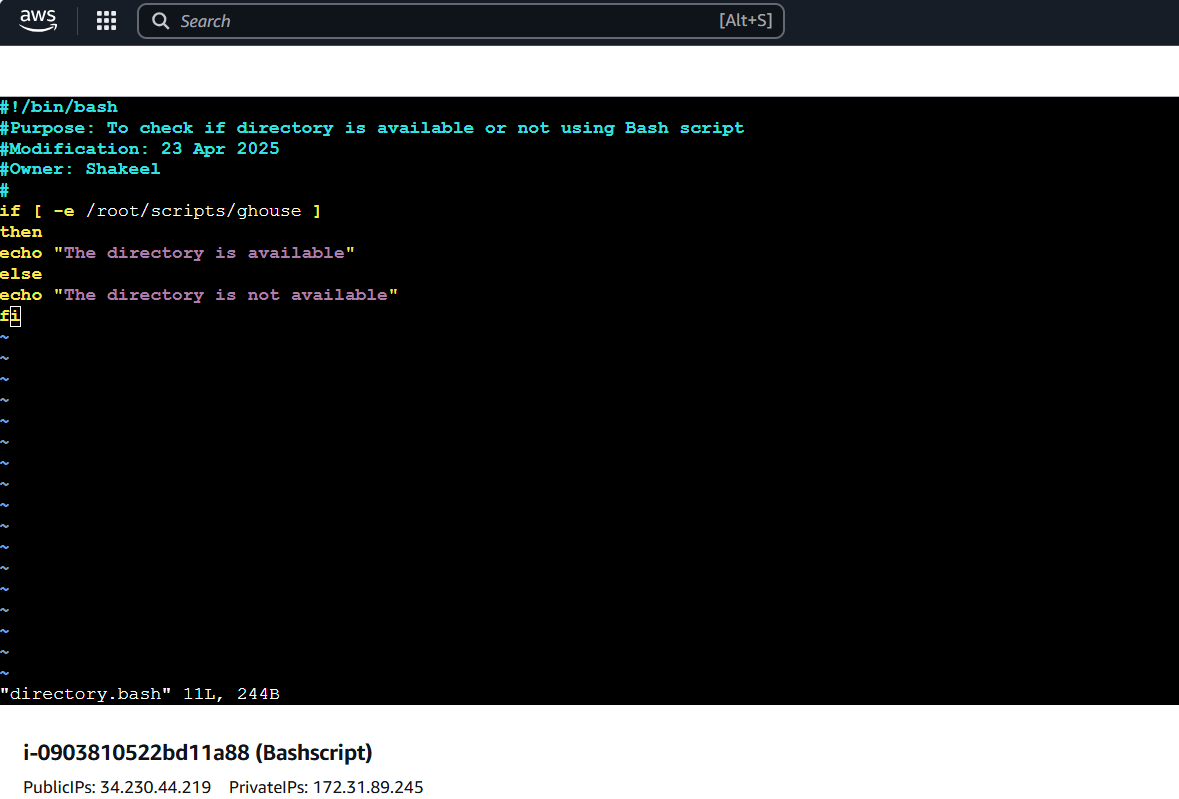
**And the file name should as per the task which we are going to do vi task\_name.bash**

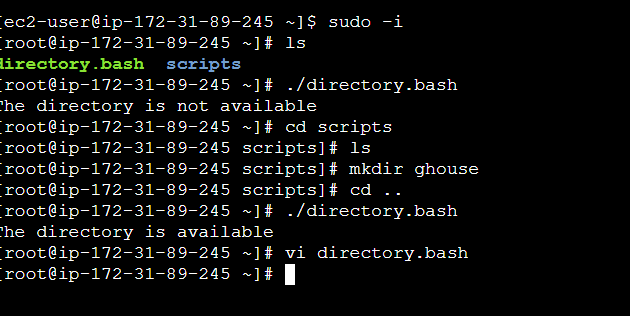
**To execute the bash the file first we need to give the permission to execute using**

**Chmod +x(755) and filename.bash**

**After this we can execute this file using ./filename.bash or bash filename.bash**

1) Create on Bash script to check if a directory is available or not





#!/bin/bash-- shebang characters to interprete

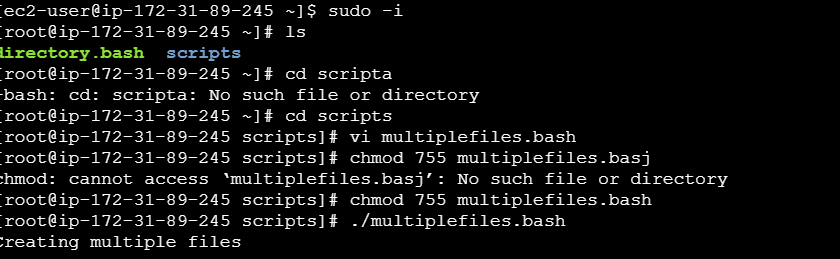
To check the directory exist or not we are using If else condition

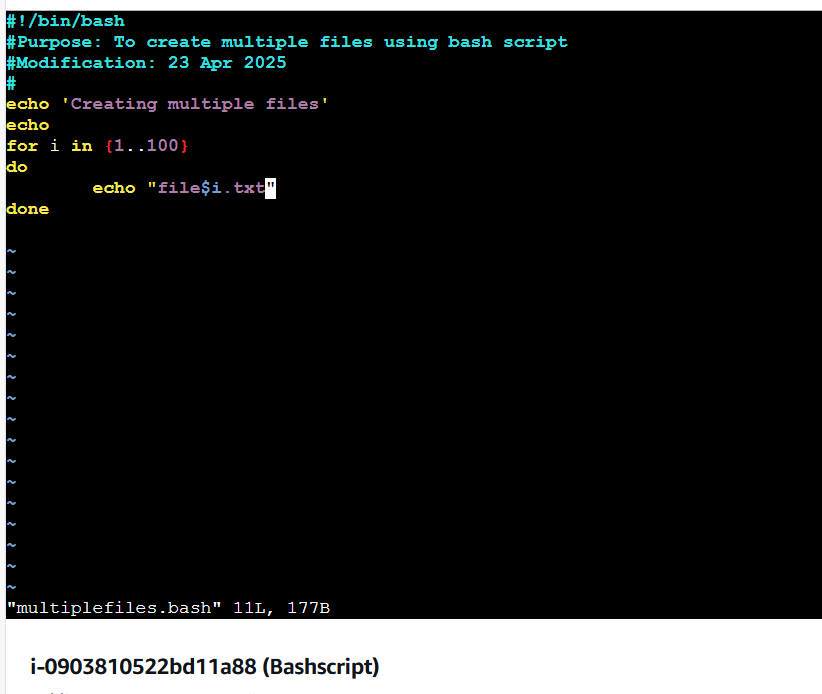
In condition we are giving direct path of directory

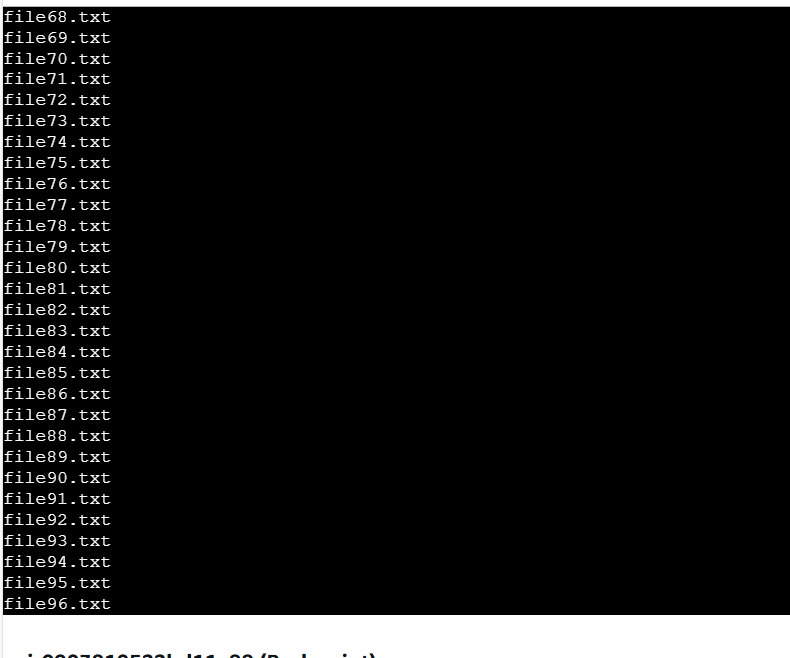
If it exisits then “echo “ will print the output as directory exist

If not else condition will run and gives the output of “Directory doesn’t exist”

2) Create a bash script which will create multiple files

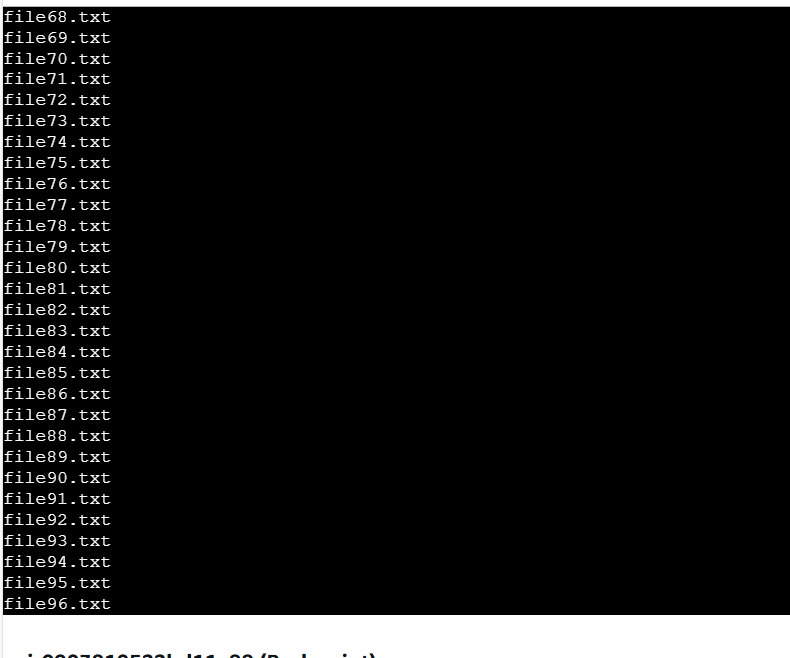




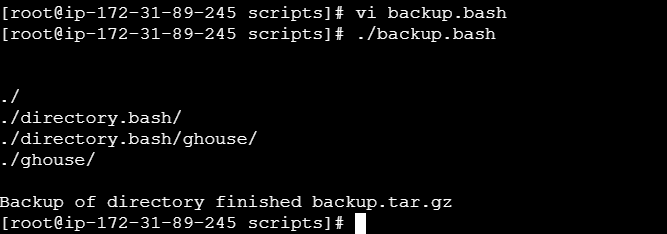


To create multiple files here I am using for loop

In loop i have given the number from 1to100 that means loop will run for 100 times and create the 100 files in the format of file$i.txt means file1.txt and file2.txt and ....file100.txt

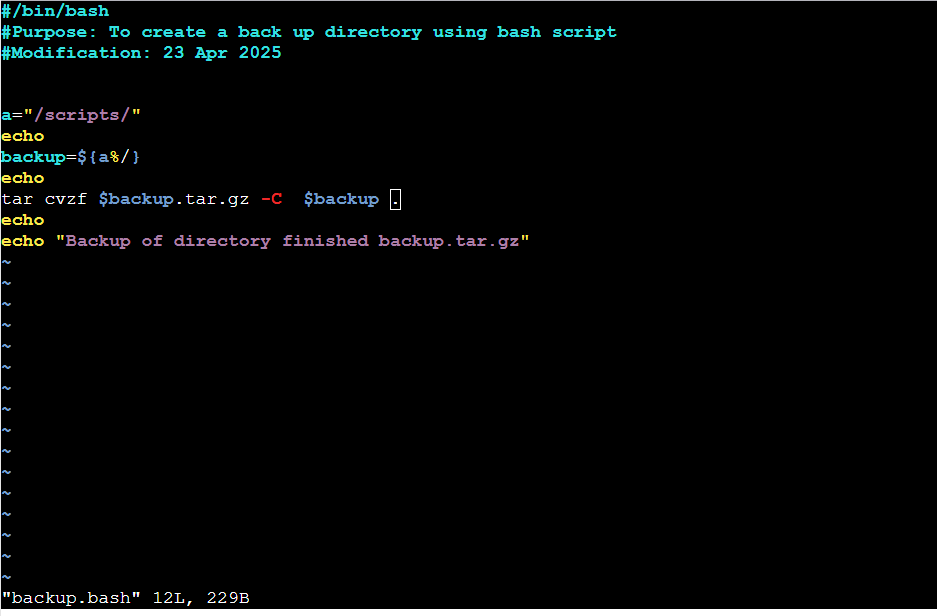


3. Create a bash script to take backup of a directory

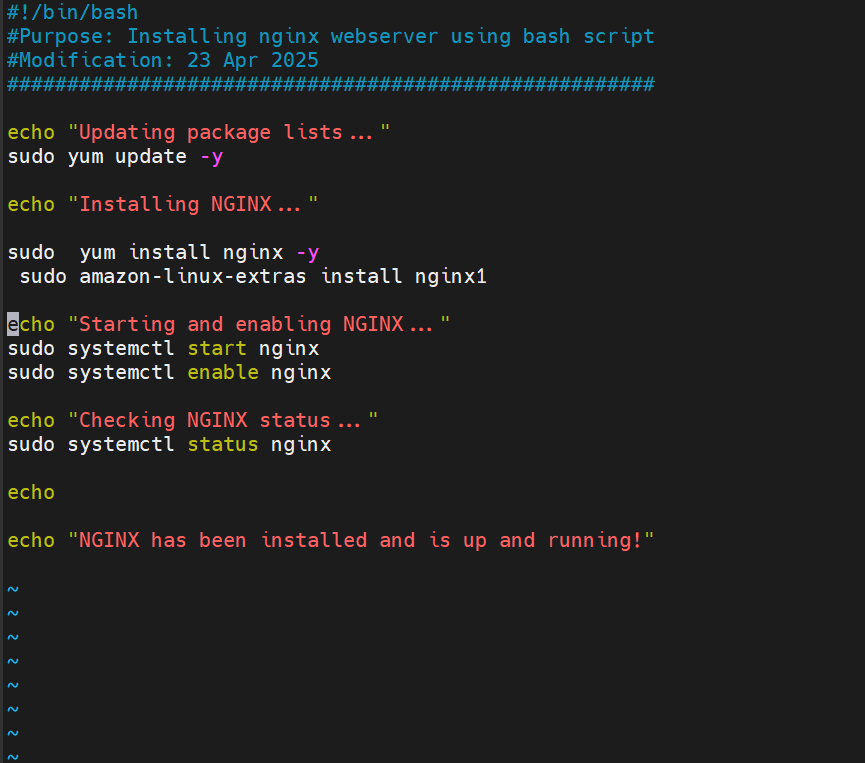


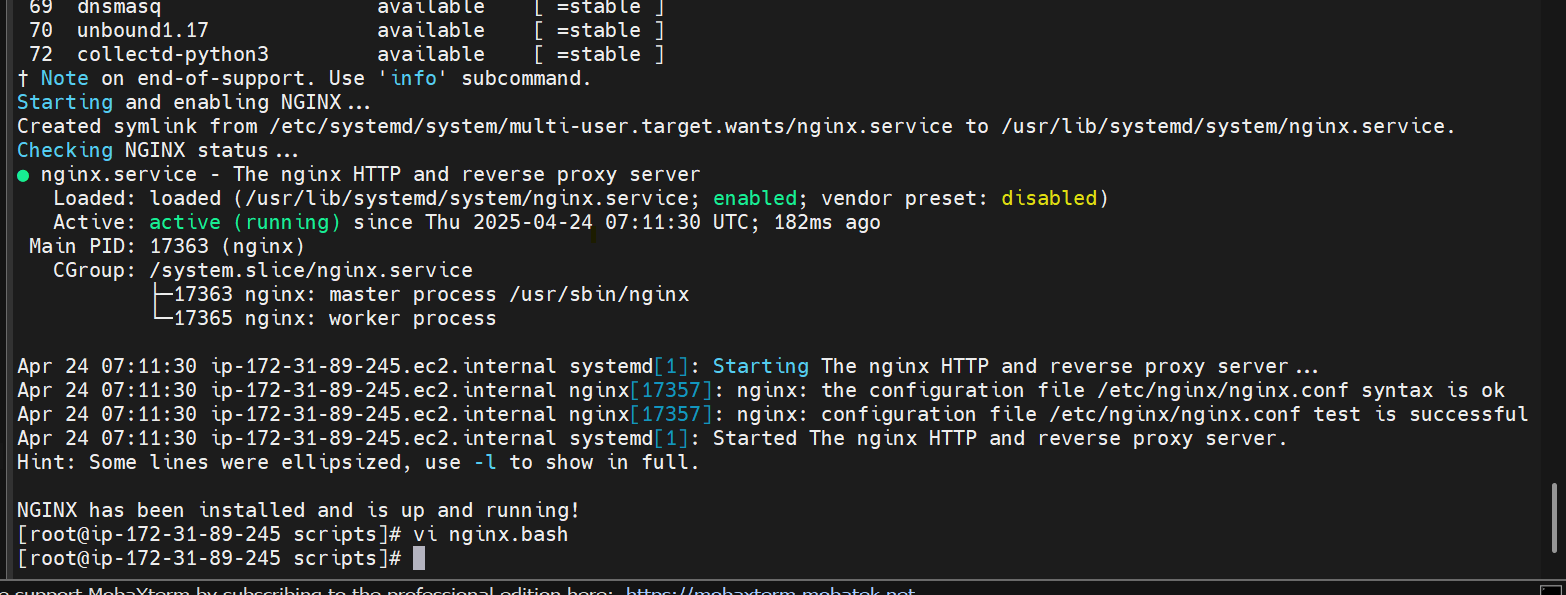
To create a back up directory here i have used tar command in the format of

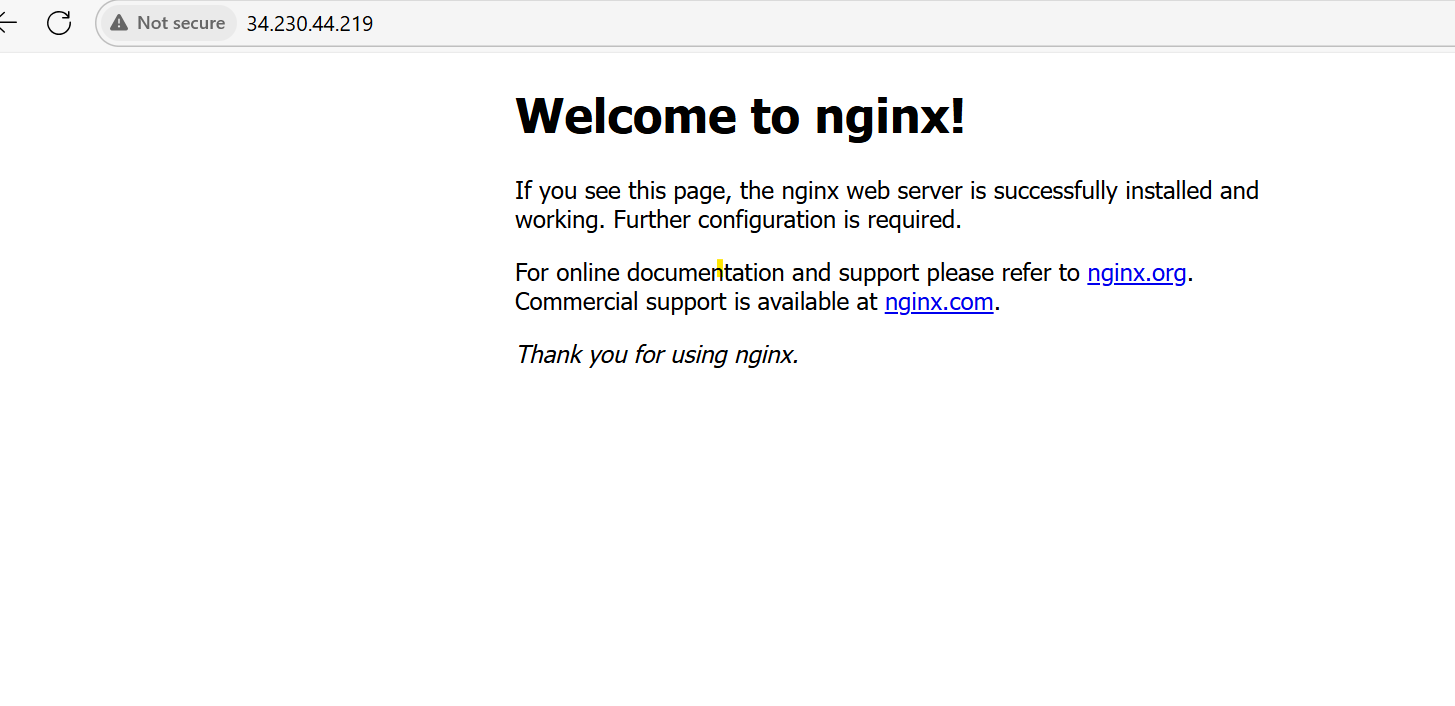
Tar cvzf directory name.tar.gaz a directory name



4. Create a bash script to install nginx in ec2 server







Bash script to install and configure the nginx

Here we have used direct installation commands

Sudo yum update –y : To install the packages

Sudo yum install nginx –y : To install the nginx

-y : To avoid manual intervention/to run the process automatically

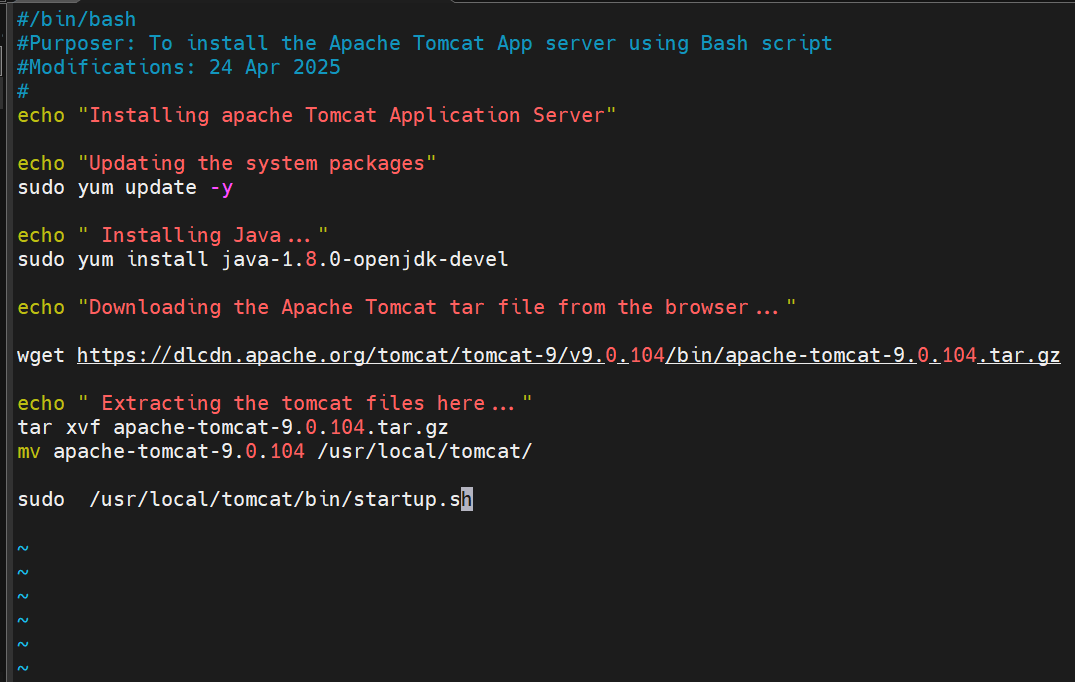
Sudo systemctl start nginx: To start the nginx

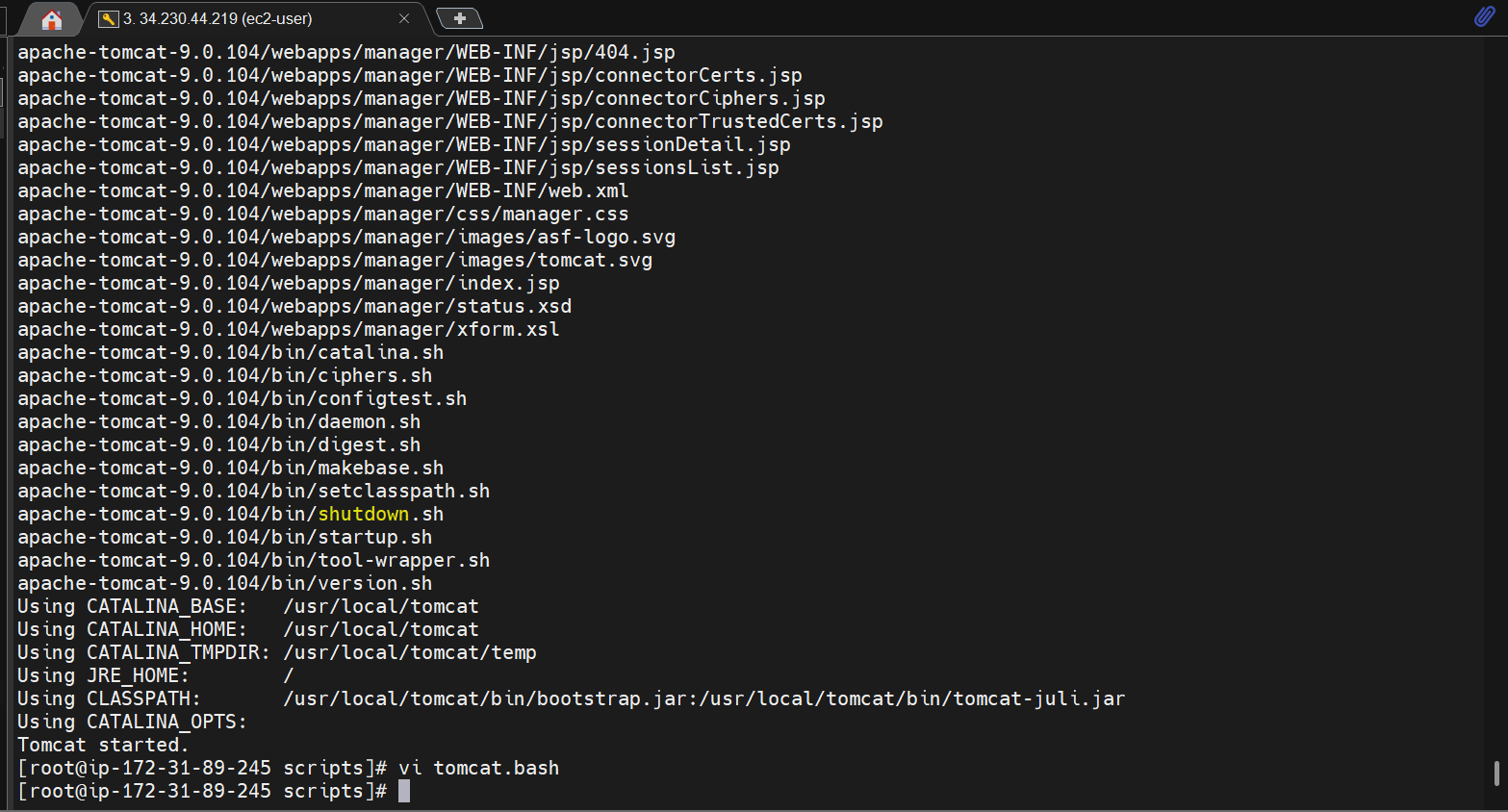
Sudo systemctl status nginx: To check the nginx status whether it is working or not

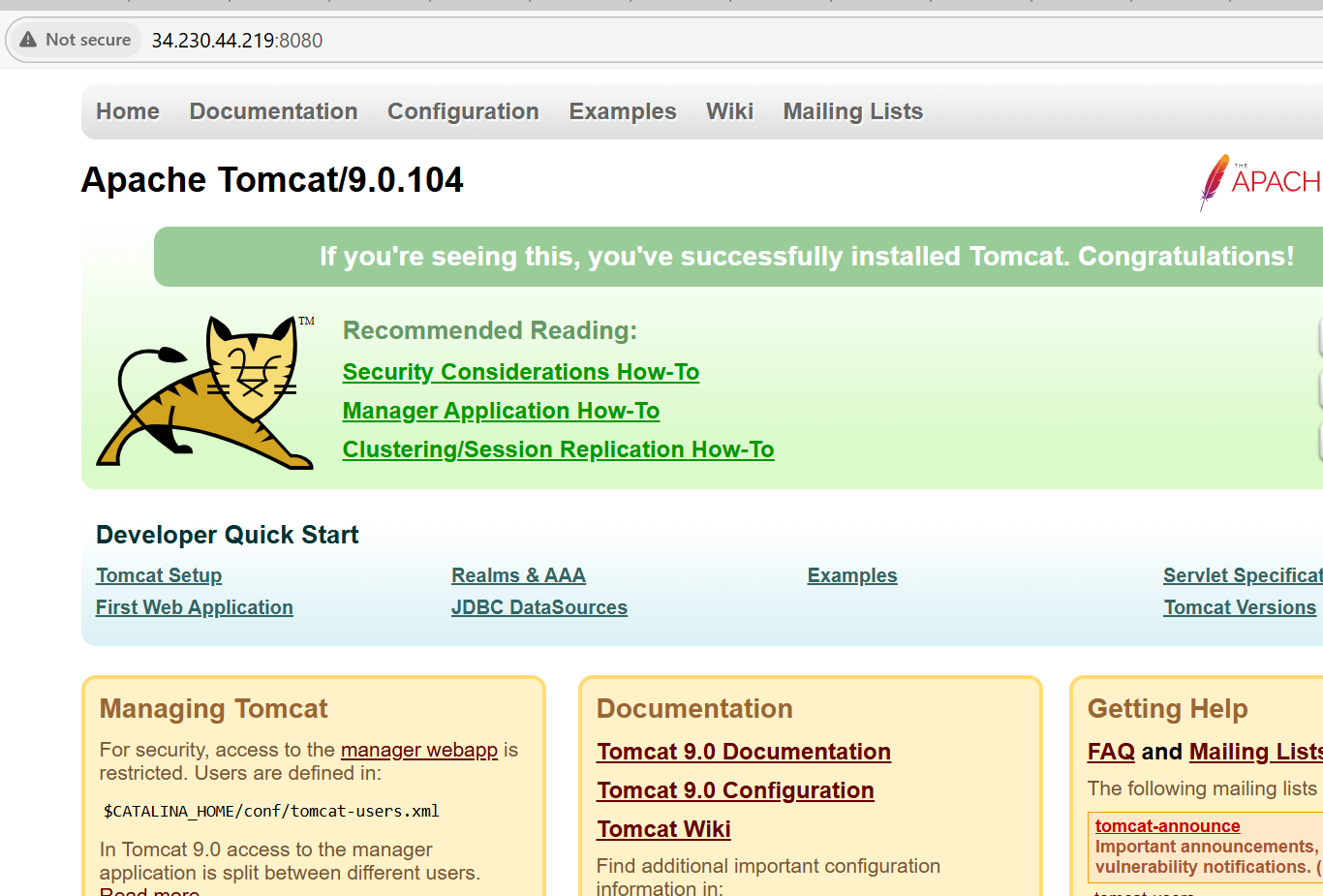
And also using port number and checked in browser with the help of public ip address:port number and it is running

So file execution is successful.

5) Create a bash script to install ApacheTomcat in ec2 server.







Bash script to install the Apache tomcat

Sudo yum update –y : To update the system packages

Sudo yum install java –y : To install the java in the system as it is pre-requisite to install the Apache tomcat

Wget https:// for to download the tar file of to install from the internet

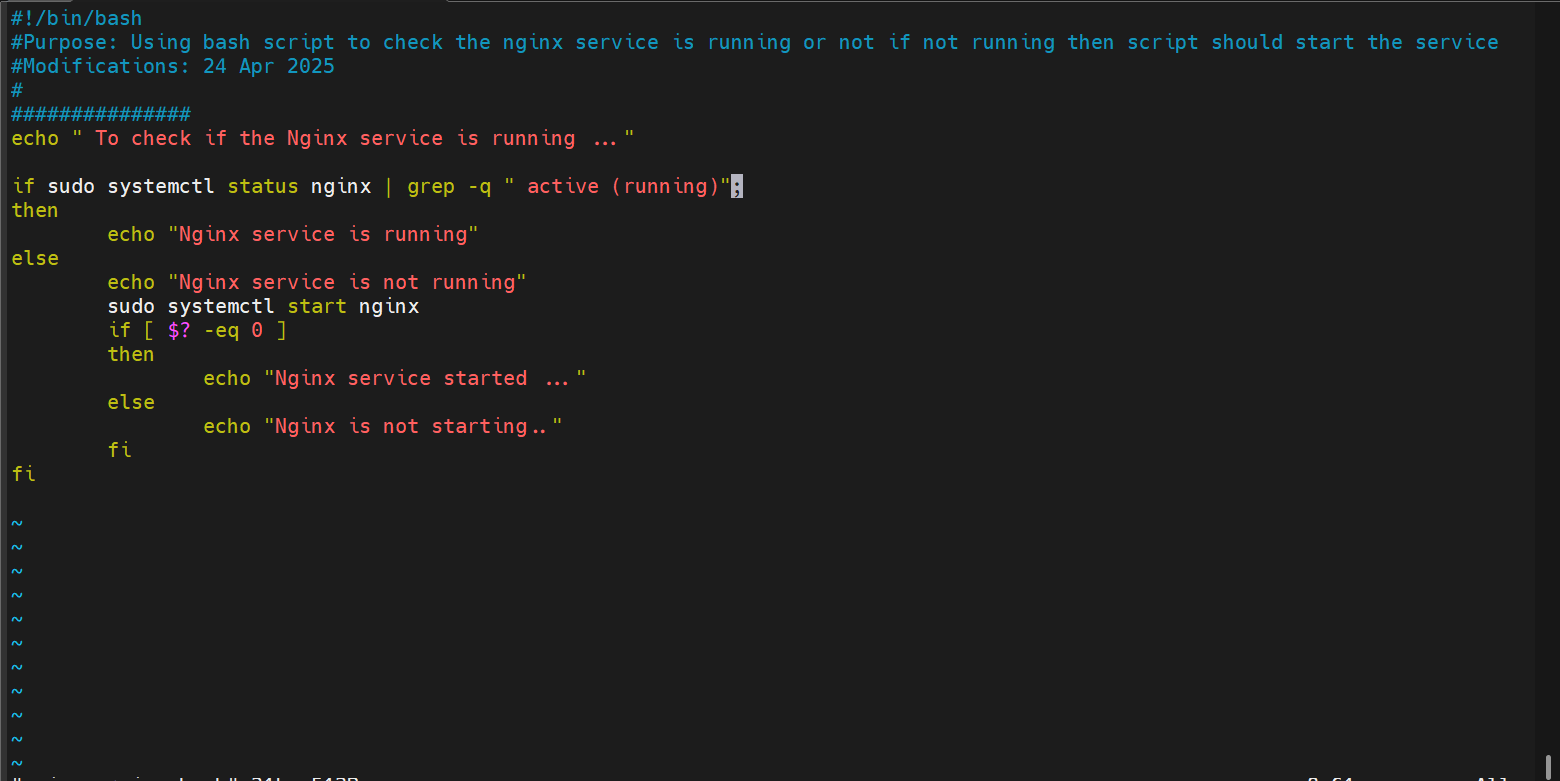
tar cvf apache-tomcat.tar : To extract the tar file of apache tomcat

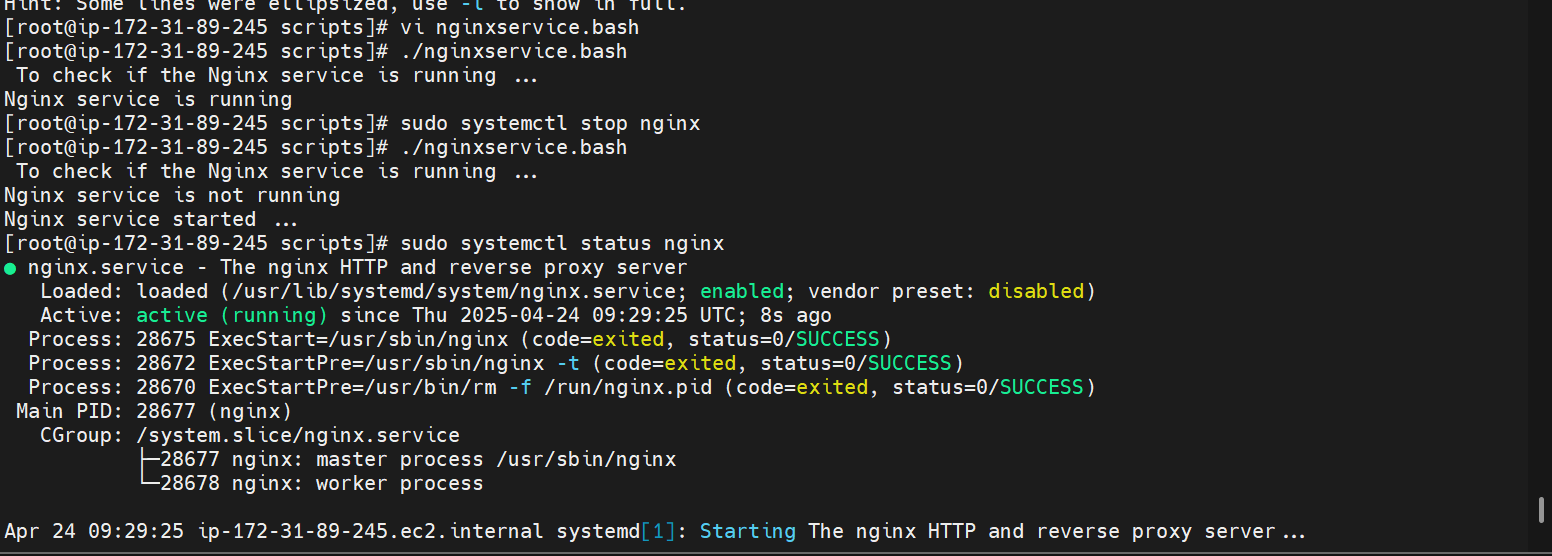
Mv : Is used to move the directotry in the tomcat directory.

Sudo /usr/local/tomcat/bin/startup.sh : To start the apache-tomcat service in the using startup.sh file

And the tomcat is running successfully and checked in the browser also.

6) Create a bash script to check list if nginx service is running or not,if not running then script should start the service.





To check the nginx is running or not

Sudo systemctl status nginx: It will give the status or nginx active or not

After this grep –q active (running): this will find the keyword in the running logs

Then if it matches the keyword

Then echo will print the output as “ nginx is running successfully “

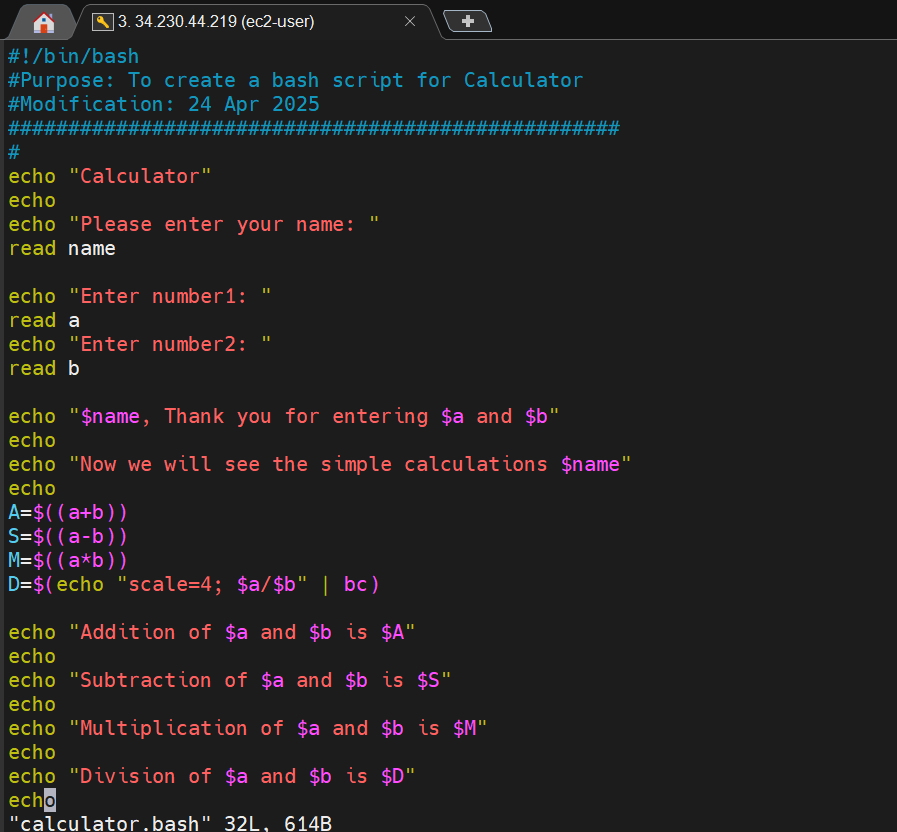
Else we it need to be started automatically

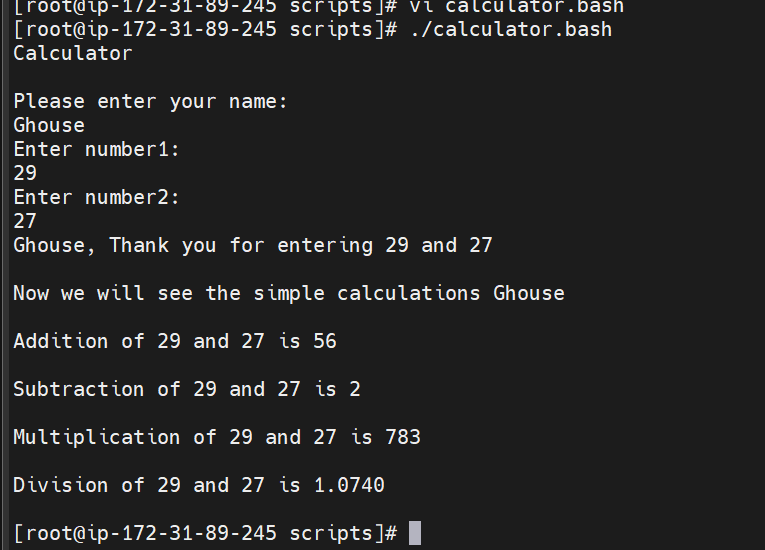
We used if condition using exit status $? -eq 0 that means previous command run successfully then it will execute the command to start the service

Sudo systemctl start nginx: To start the service

Then it will give the ouput as “Nginx started”

7) Create a bash script for calculator.





Echo “Please enter your name:” :

Read name: It will ask the user to enter his name:

Echo “enter number 1 and 2: asking user to enter 2 numbers of his choice

After this

A=$((a+b))) ; will be for adding two numbers entered by user

S=$((a-b)); will be for subtracting the two numbers entered by user

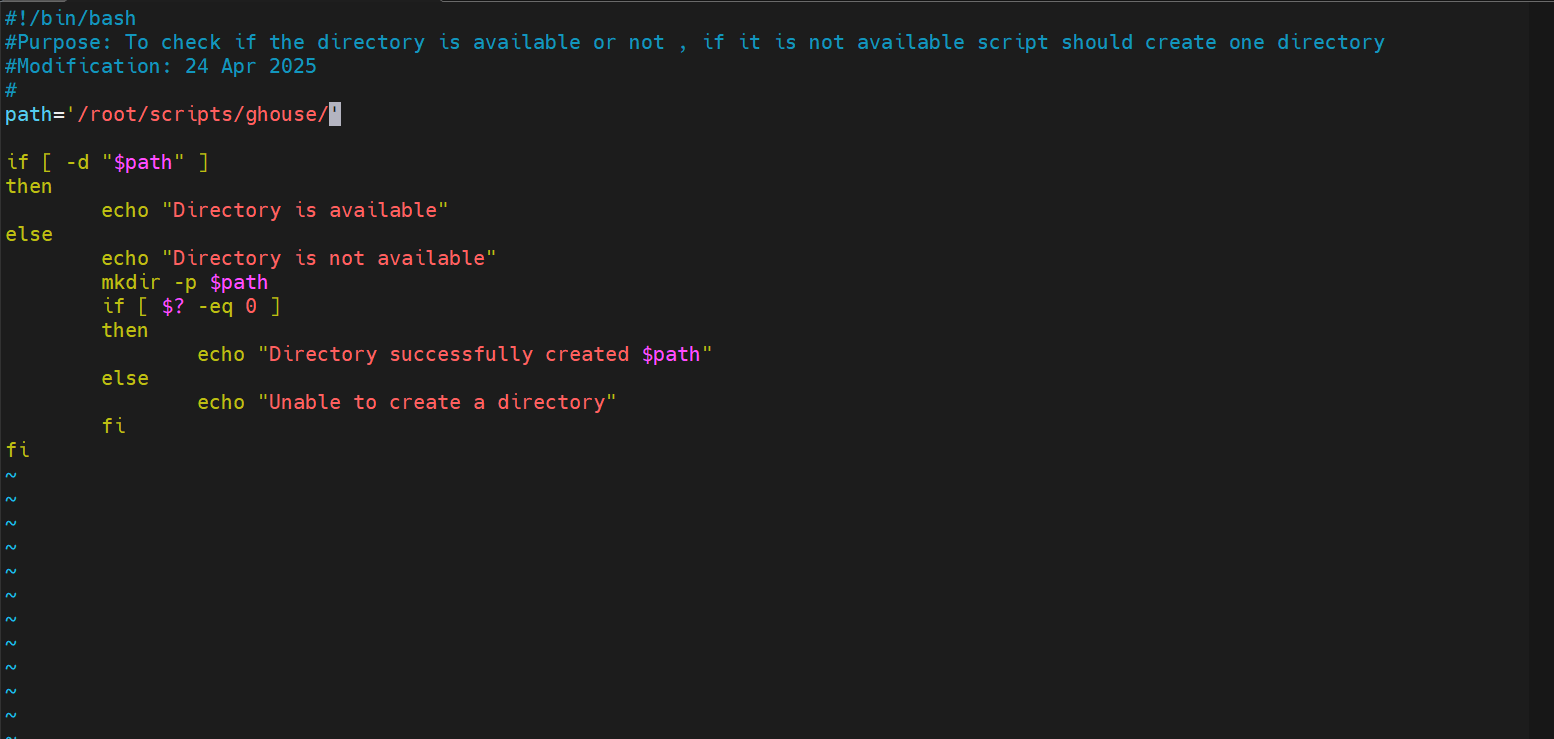
M=$((a\*\*b)) ; will be for multiplication

D=$(echo “scale=4; $a/$b | bc) ; will be for division of a and b and the to print the output upto 4 decimals

So finally echo will give the output of a and b’s addition, multiplication,subtraction and division using simple calculator

8) Create a bash script to check if directory is avaialble or not,if not then create a

directory.



Path is i have given certain directory in the variable

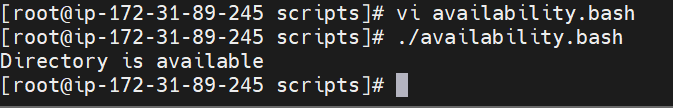
And if [-d “$path ]; this will check the if directory is exist or not

Then echo will print the output if it is exists ; the directory is available.

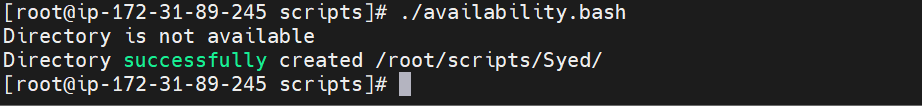
Else it should create the directory using

mkdir –p $path: to create new directory

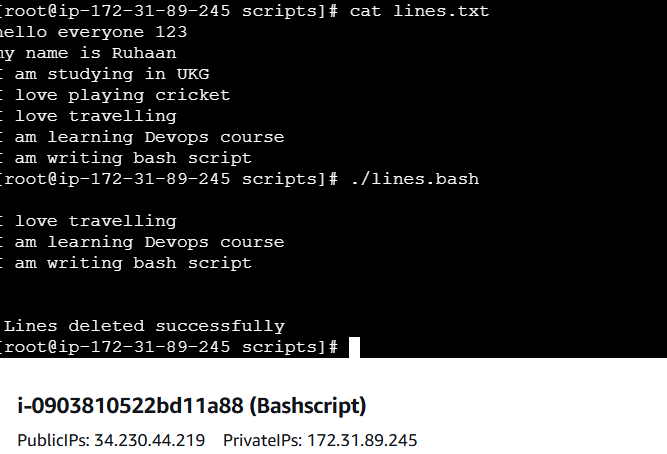
$? -eq 0 to satisfy the condition then only it will move to next command after creating the new directory







9) Create bash script to delete last 3 lines for a file.



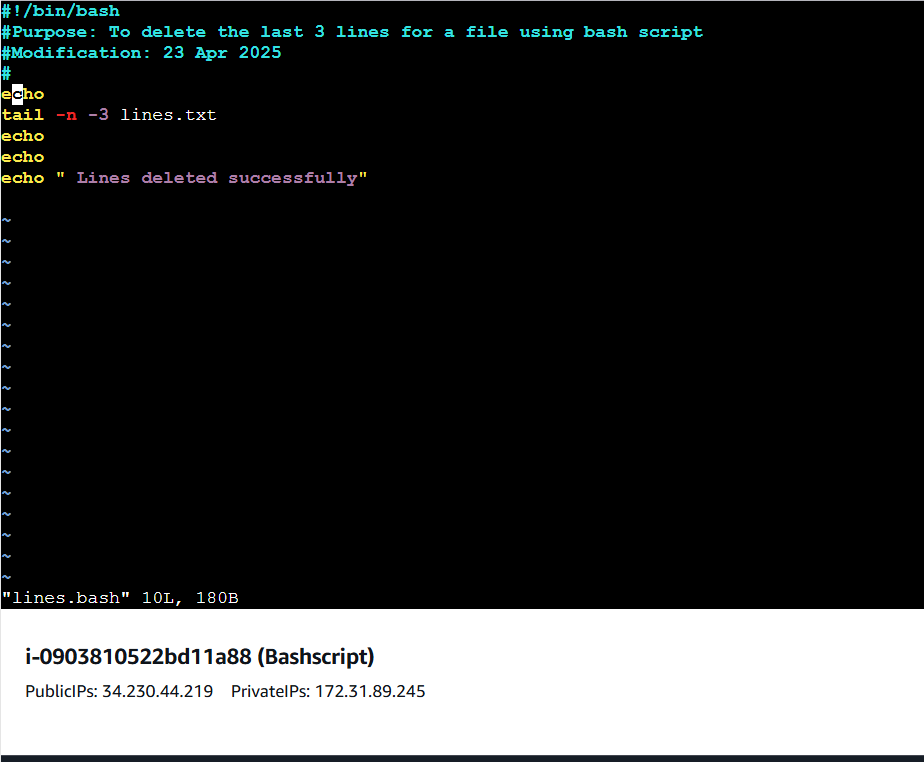
Created one file using vi lines.txt and entered some data in that file

Cat lines.txt ; to check the data what we have stored in the text file

To delete the last 3 lines i have used tail command

Tail –n –3 and the text file name

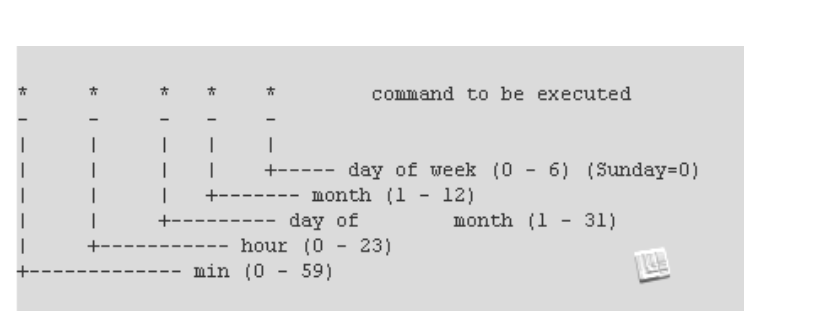
And it will give the output lines deleted successfully and we can check using cat command after successful execution of bash script

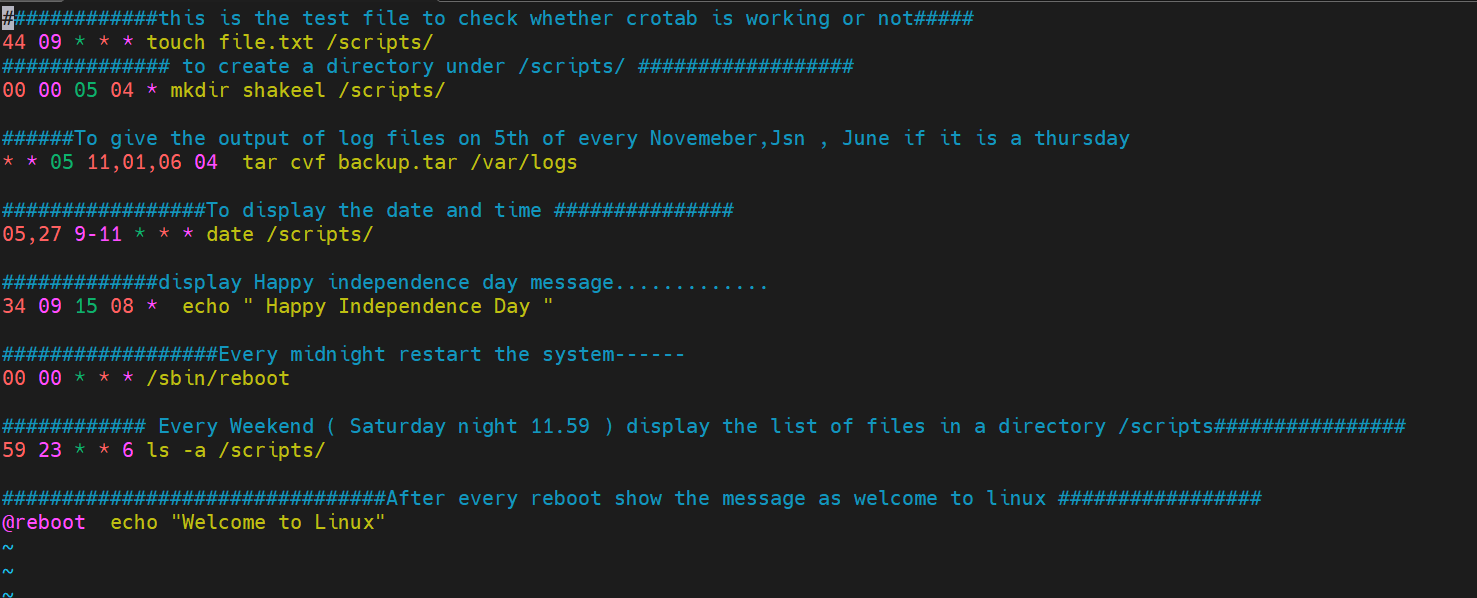


**Crontab Exercises**

**To edit the crontab : crontab –e**

**To check the crontab list: crontab -l**





1) April 5th Midnight

Midnight means : Minutes and hours will be as 00 00

Date mentioned : 5th

Month : April (04)

Day didn’t mention so \*

So the crontab syntax will be “00 00 05 o4 \*”

2) 5 th of Every November,Jan,June if it is a Thursday.

Minutes and hours: \* \*

Date : 5th

Month : November, Jan,June : 11,01,06

Day : thursday : 04

So the syntax will be : \* \* 05 11,01,06 04

3) At 05 and 27th minutes of 9,10,11 hours everyday.

Minutes : 27

Hours : 9,10,11

Date : 05

Month : \*

Everyday: \*

Syntax will be 27 9,10,11 05 \* \*

4) 34 min. of 9th hour on 15th Aug.

Minutes: 34

Hours: 09

Date 15

Month : Aug : 08

Day : \*

The syntax will be : 34 09 15 08 \*

5) Every midnight

Midnight time : 00 00

So every : \*

Syntax: 00 00 \* \* \*

6) Every Weekend ( Saturday night 11.59 )

Time 11:59

Minutes : 59

Hours 11 ( night ) : 23

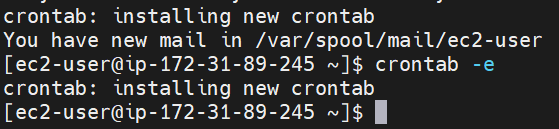
Date : \*

Month : \*

Day : Saturday

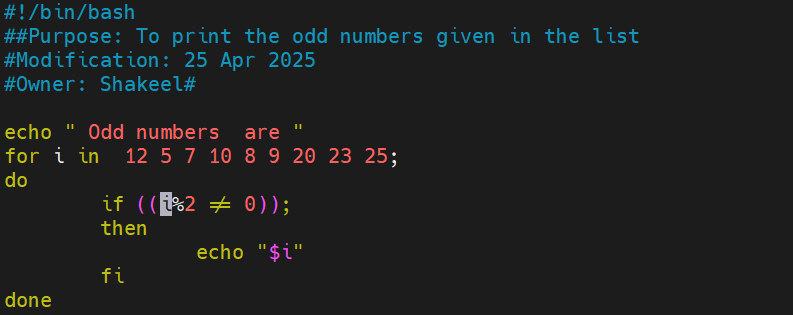
Syntaxt : 59 23 \* \* 06

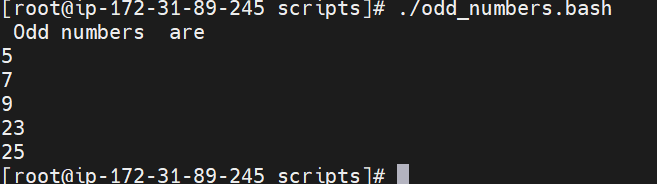
7) After every reboot



Syntax @reboot this will be direct command in crontab entries for every root

1) Bash script to print odd numbers from the list. (12,5,7,10,8,19,20,23,25)





To print the odd numbers from the given list we are using for loop here

for i in 12 5 7 10 8 9 20 23 25

It will take the input of every number in the list and enter the loop

So to check number is odd or not ; that means when we are dividing any number by 2 and the if the remainder is 0 then its not the odd number if we are not getting remainder is 0 it is the odd number

So to check this we are using if command with the condition as

If [ ((i %2 )) != 0]

Means i will be the input from the given list

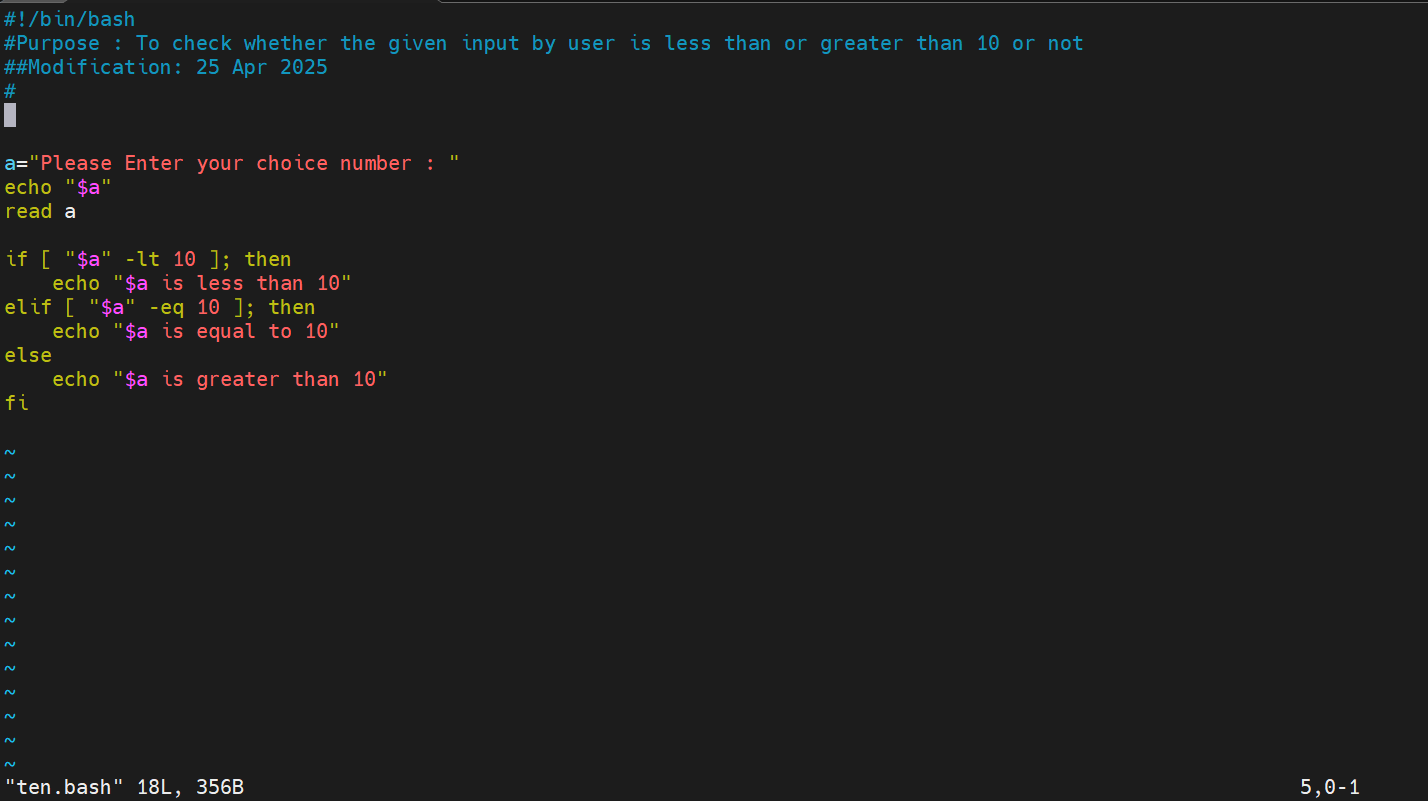
% will check for the remainder

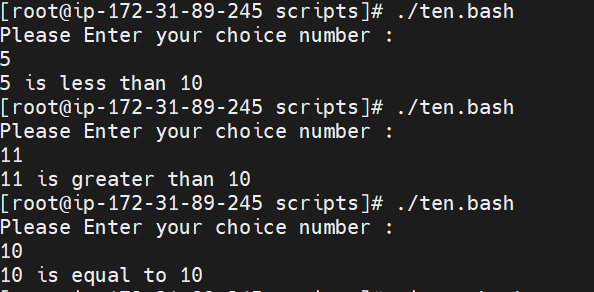
2 = to divide the numbers in the given list

!= arithmetic operator to check the not equal to 0

Once the above condition is true then it will give the output of odd numbers from the list

2. Bash script to take a input from user and check if it is greater than or less than 10





Echo $a ; it will as the user to enter any number of user’s choice

So to check the given number is less than or greater than 10 i have used if condition

If [ $a –lt 10 ] it will check the number is less than 10 or not

-lt is a comparison operator (less than)

Then if its true then it will give the output of given number is less than 10

If not then it will go for elif condition to check entered number is equal to 10 or not

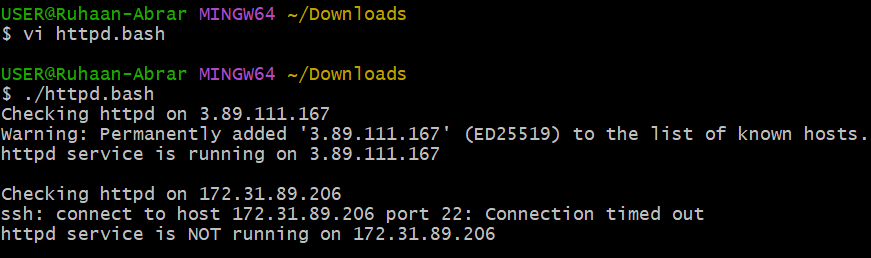
[ $a –eq 10 ]; checking for equal to or not

If yes it will print the output as entered number is eqaul to 10

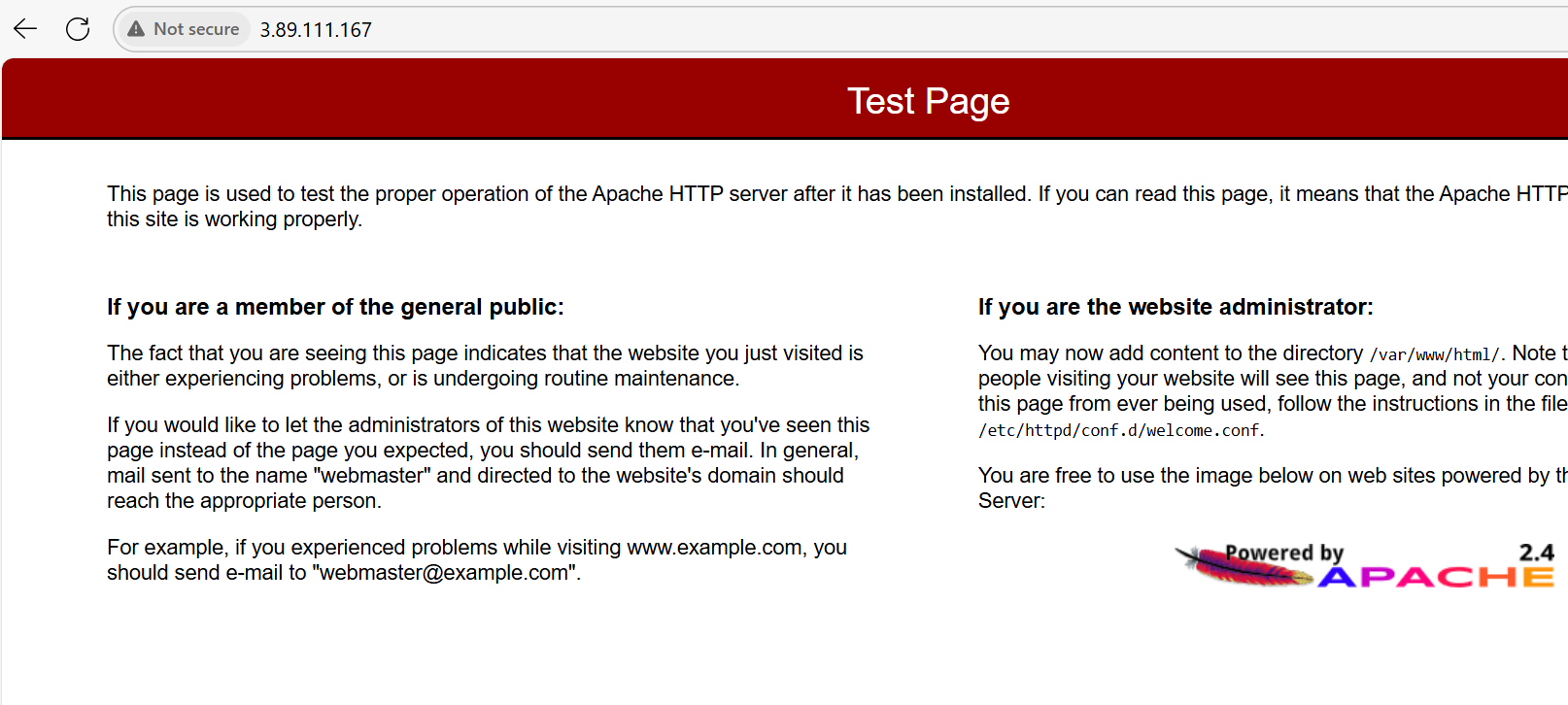
If not then it will go to next condition else to check greater than or not

[$a –gt 10 ]; -gt is for (greater than) if it is true then it will give the output as the entered number is greater than 10

3) Bash script to login to multiple servers and check if httpd service is running or not







**#!/bin/bash**

**This tells** the system to use bash to run the script.  
 It’s called a "shebang" (#!), and points to the Bash interpreter.

servers=("3.89.111.167" "172.31.89.206")

**We define a list (array) of IP addresses** here.  
 Each IP represents a server you want to connect to

key="/c/Users/user/Downloads/sabair\_test.pem"

**You define the path** to your .pem private key file,  
 which is used for SSH authentication.  
 This file is necessary to securely connect to your EC2 instances.

if [ ! -f "$key" ]; then

echo "ERROR: Private key not found at $key"

exit 1

fi

**This checks** if your .pem key file actually exists.

* If it **does not exist**, it prints an error message and **stops** (exit 1).
* If it **exists**, the script continues.

for server in "${servers[@]}"; do

**You start a loop**: for each server IP in your list, do the next set of actions.

${servers[@]} means "expand all elements of the array."

echo "Checking httpd on $server"

**Prints a message**: saying which server it’s checking now.

STATUS=$(ssh -i "$key" \

-o StrictHostKeyChecking=no \

-o ConnectTimeout=5 \

ec2-user@"$server" \

"systemctl is-active httpd" 2>/dev/null)

* ssh: Connect to the server using SSH
* -i "$key": Use the .pem private key for login
* -o StrictHostKeyChecking=no: Don’t ask about unknown hosts (skip the yes/no prompt)
* -o ConnectTimeout=5: Give up if the server doesn't respond in 5 seconds
* ec2-user@$server: Connect as the ec2-user to that server's IP
* "systemctl is-active httpd": Ask the server if httpd (Apache) is running
* 2>/dev/null: Hide error messages (makes output clean)
* STATUS=$(...): Save the result (active, inactive, or empty) into a variable STATUS.

if [[ "$STATUS" == "active" ]]; then

echo -e "\e[32mhttpd service is RUNNING on $server\e[0m"

else

echo -e "\e[31mhttpd service is NOT running on $server\e[0m"

fi

**This checks** if the service is active:

* If yes: print in **green** (\e[32m)
* If no: print in **red** (\e[31m)
* \e[0m resets color back to normal.

The -e in echo -e tells Bash to interpret escape sequences (\e).

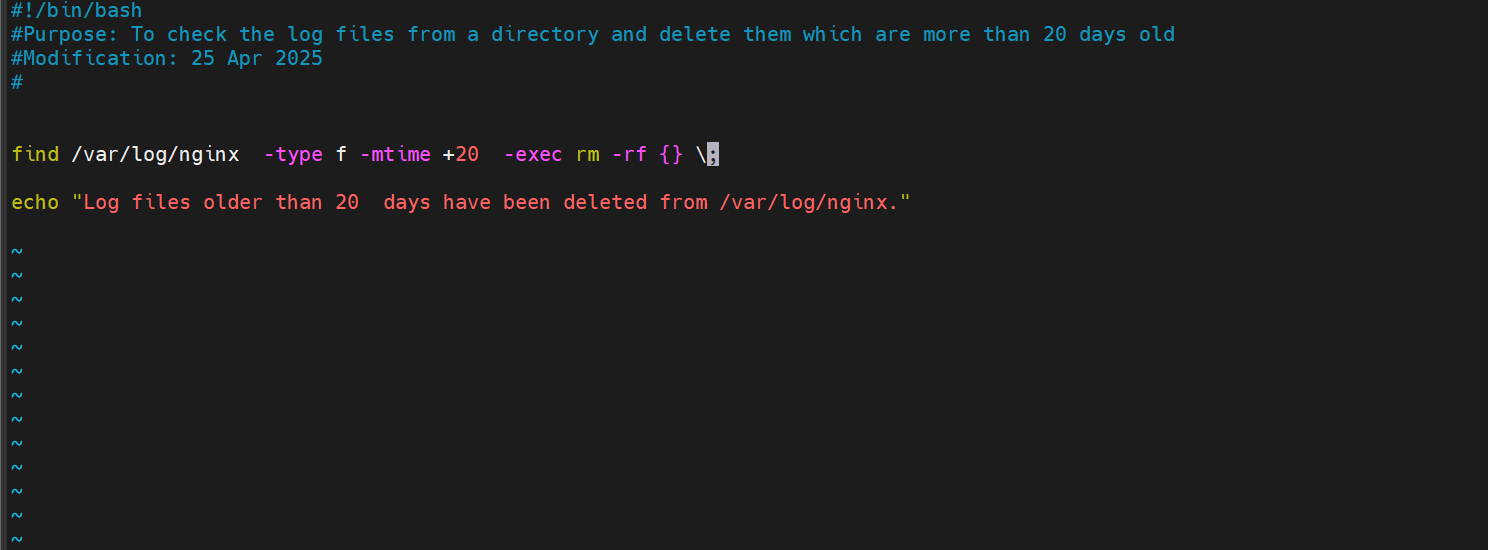
echo ""

Prints a blank line for clean spacing between server checks.

done

Once all servers have been checked, the script finishes.

4) Bash script to check the log files from a path and delete files older than 20 days.



To check the log files from a directory and delete then which are more than 20 days old

In this I have used find command and one directory

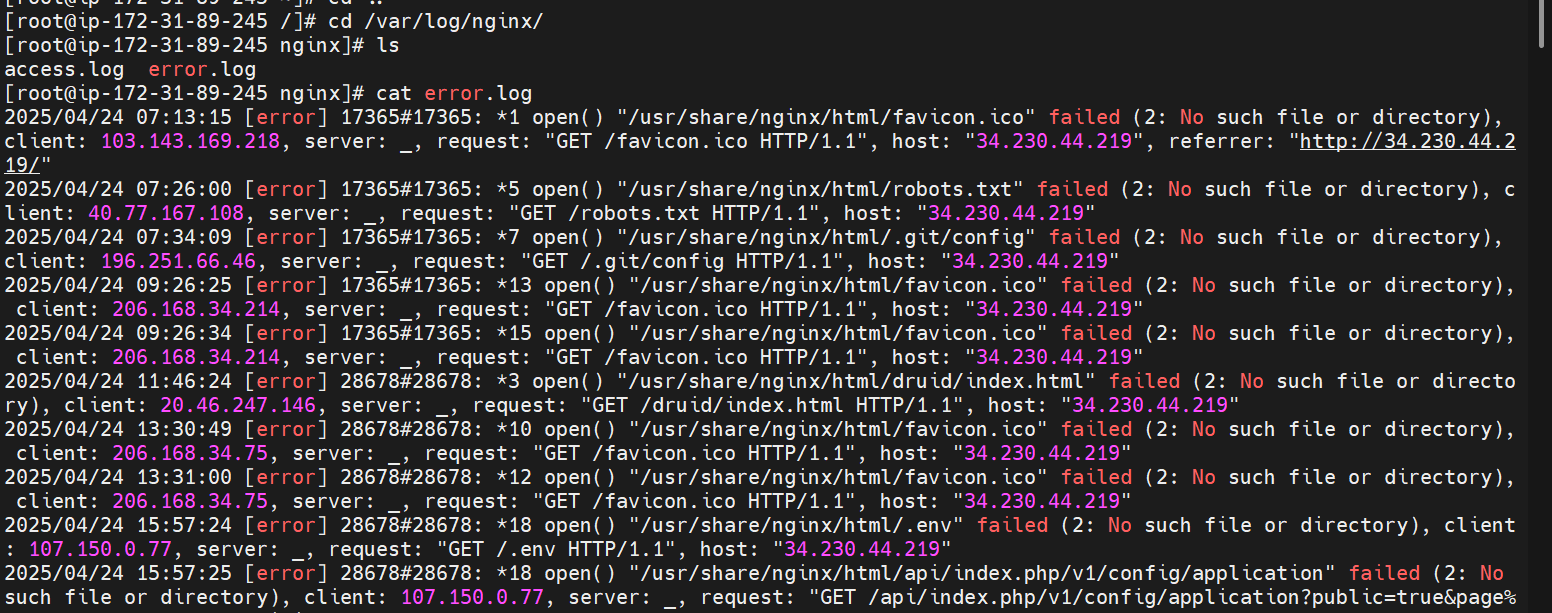
Find /var/log/nginx; it will find the log files in this particular directory

-type f ; it will check for all the log files

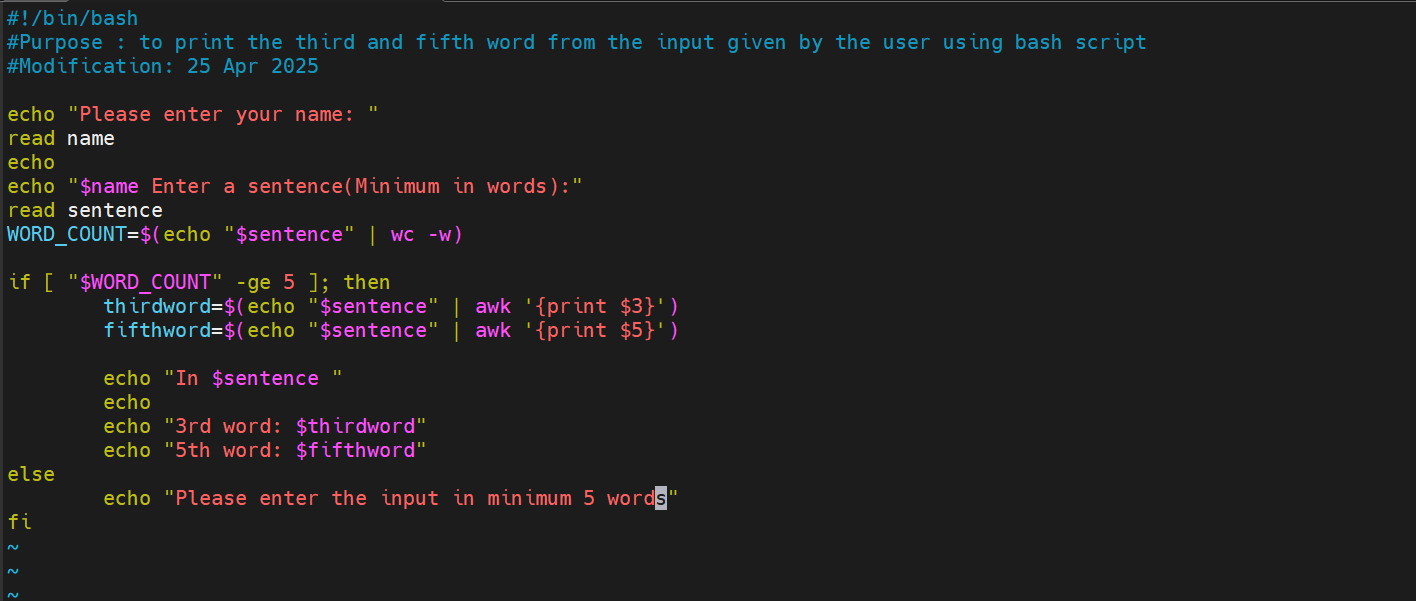
-mtime +20 ; the logs which are more than 20 days old

-exec rm –rf {}- it will delete the all log files with single command

\; : once the command is finished it will stop the command execution



5) Bash script to print 3rd word and 5th word from the given input of user



Read sentence ; it will ask the user to enter a sentence in minimum 5 words

$(echo “$sentence” ) it prints the input entered by user

Wc –w ; it will count the number of words in the sentence

To print the third and fifth word in the input , i have used if command with comparison operator

If ( wordcount –ge 5); it means if the number of words in the sentence greater than 5 , then only we have asked to print the 3rd and 5th number using awk command

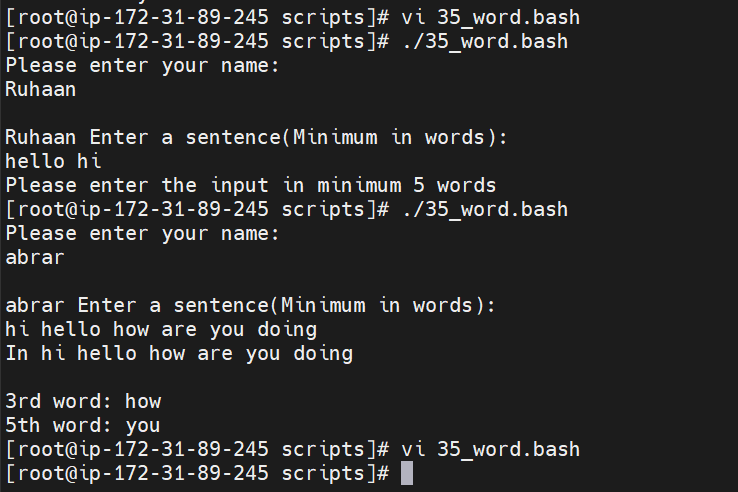
Awk {print $3 } - third word in the sentence

Awk {print $5}- fifth word in the sentence

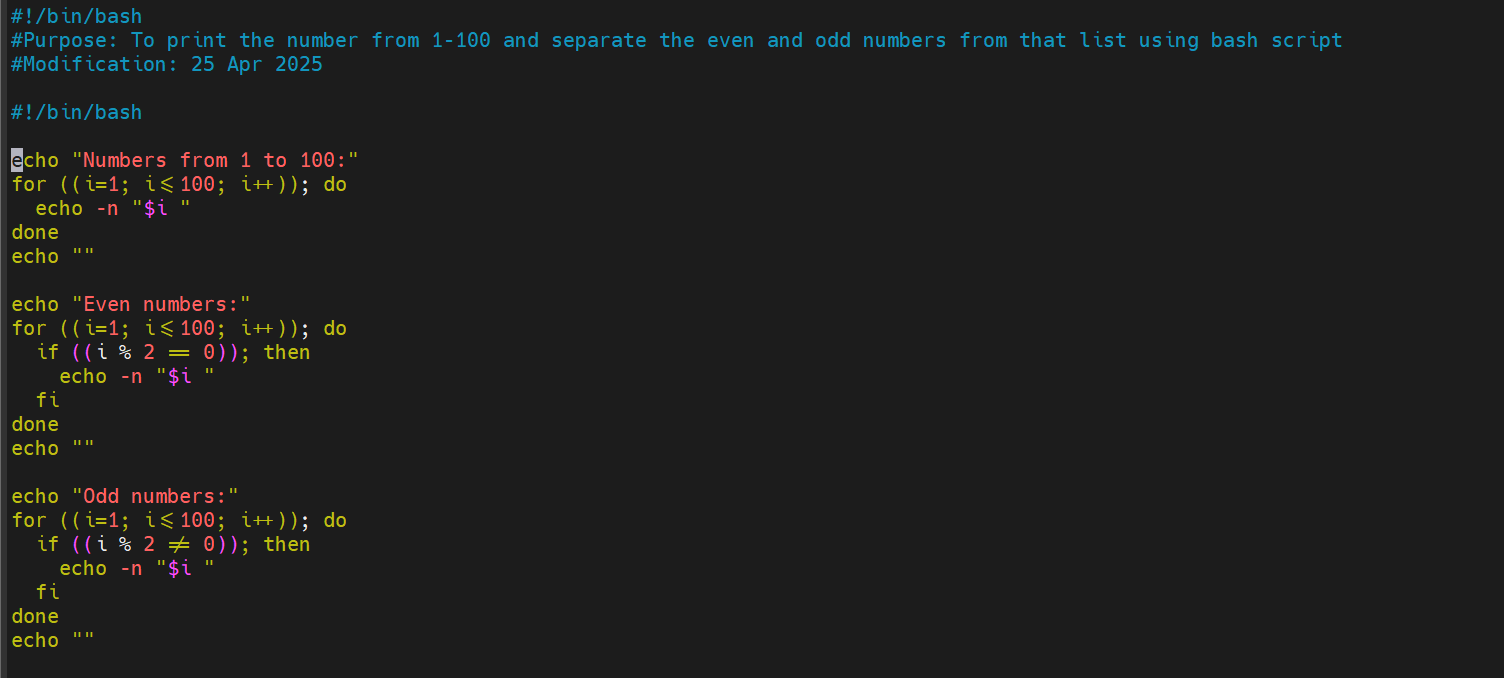
Once it is successful, then echo will print the 3rd and fifth word in the sentence

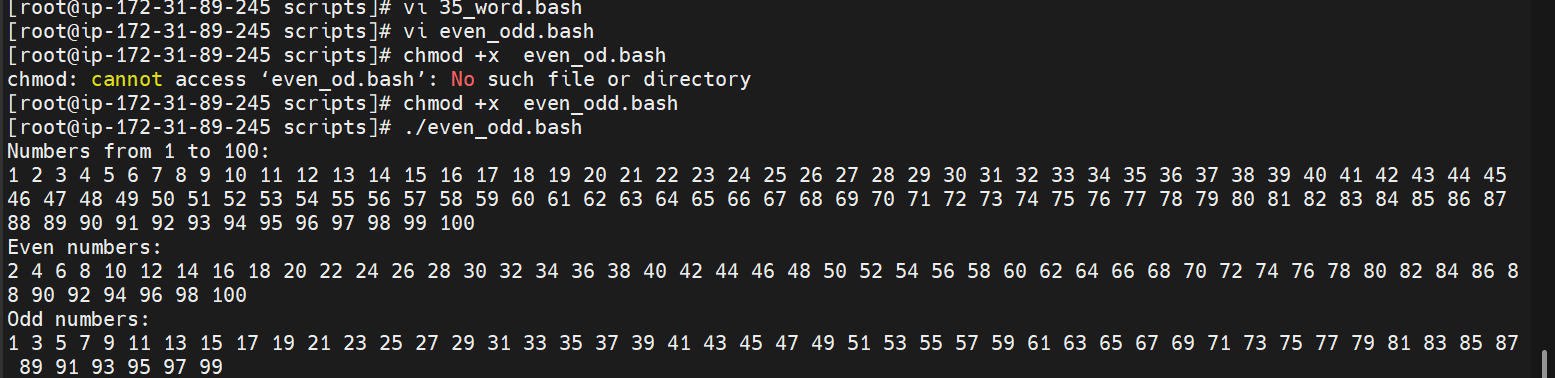
If not else will work then it will ask the user to enter the sentence with minimum 5 words

Fi ; to close the if condition.



6) Bash script to print numbers between 1 to 100 and then separate the odd numbers and even numbers





To print the numbers from 1-100 i have used for loop

For (( i=1; i –le 100; i++)) ; means the loop will starts from i=1 for the first time and it will satifsy the condition 1 less than or equal to 100 ; then it will print by echo command –n $i, then the loop will be continued for 2 , 3, 4 5, ....100.

So the complete numbers from 1-100 will be printed

To seprate the odd numbers from the output,

I have used if condition to check the odd or even

If [ $(i %2 ) == 0] ; if this satisfed that means number is divisible by 2 and gives the remainder zero then it print the even number starting from 2...100

Same will continue for odd numbers if the above condition is not satisfed it will give the output as odd numbers from the list printed already.