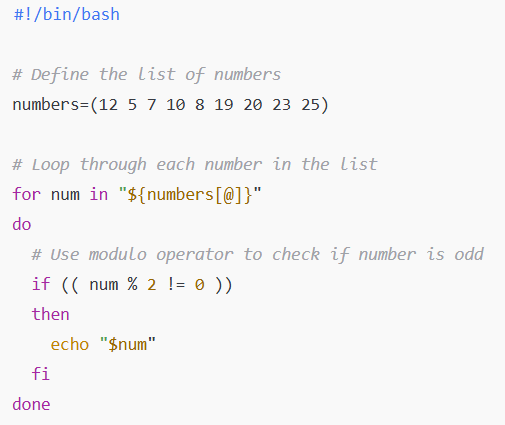
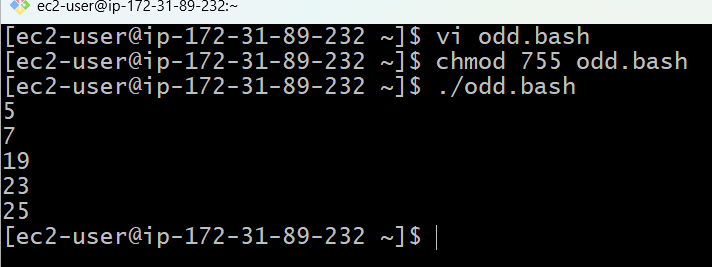
1) Bash script to print odd numbers from the list. (12,5,7,10,8,19,20,23,25)

Logic:



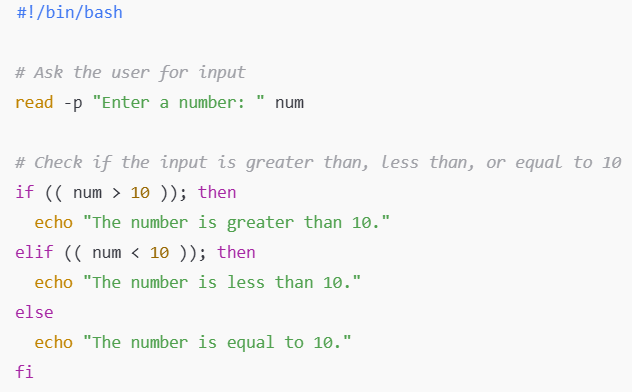
**🧠 Explanation**

1. #!/bin/bash
   * This is called a shebang. It tells the system to use the **bash shell** to interpret this script.
2. numbers=(12 5 7 10 8 19 20 23 25)
   * Declares an **array** called numbers with the given list of values.
3. for num in "${numbers[@]}"
   * A **loop** that iterates through each element of the numbers array.
   * ${numbers[@]} means all elements of the array.
4. if (( num % 2 != 0 ))
   * This uses **arithmetic evaluation** to check if the number is **not divisible by 2**.
   * % is the modulo operator.
   * num % 2 != 0 means the number is **odd**.
5. echo "$num"
   * Prints the **odd number** to the terminal.



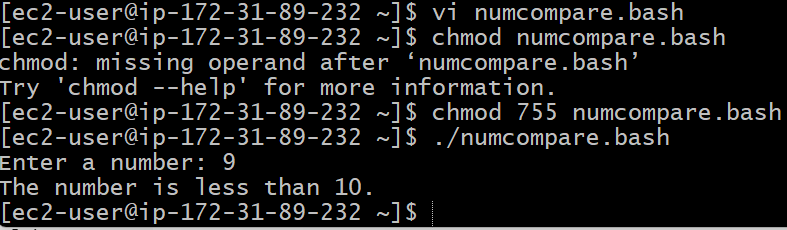
2) Bash script to take a input from user and check if it is greater than or less than 10

Logic:



**🧠 Explanation**

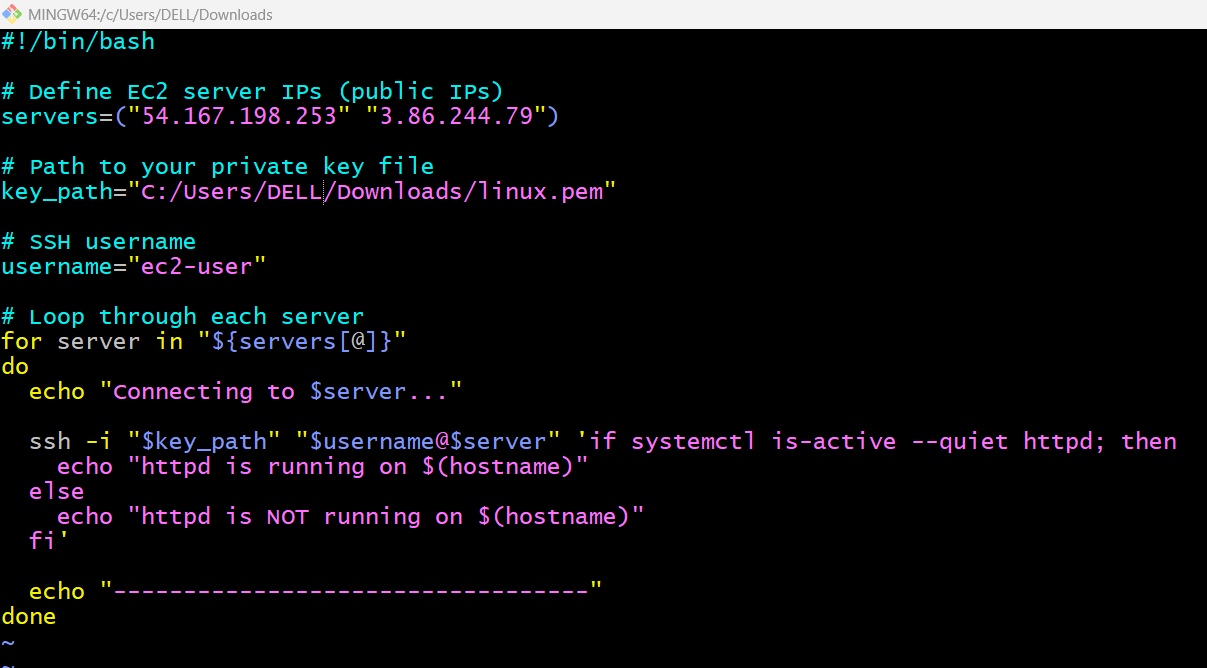
1. #!/bin/bash
   * This tells the system to use the **bash shell** to interpret the script.
2. read -p "Enter a number: " num
   * read takes **input** from the user and stores it in the variable num.
   * -p option shows a **prompt message** before taking input.
3. if (( num > 10 )); then
   * This checks if the number entered is **greater than 10**.
4. elif (( num < 10 )); then
   * This checks if the number is **less than 10**.
   * elif stands for **"else if"**.
5. else
   * This part executes if the number is **equal to 10**.
6. echo
   * Prints the appropriate message to the screen.



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

Note: First create a bash script file in gitbash (with out connecting to any ec2 machine)

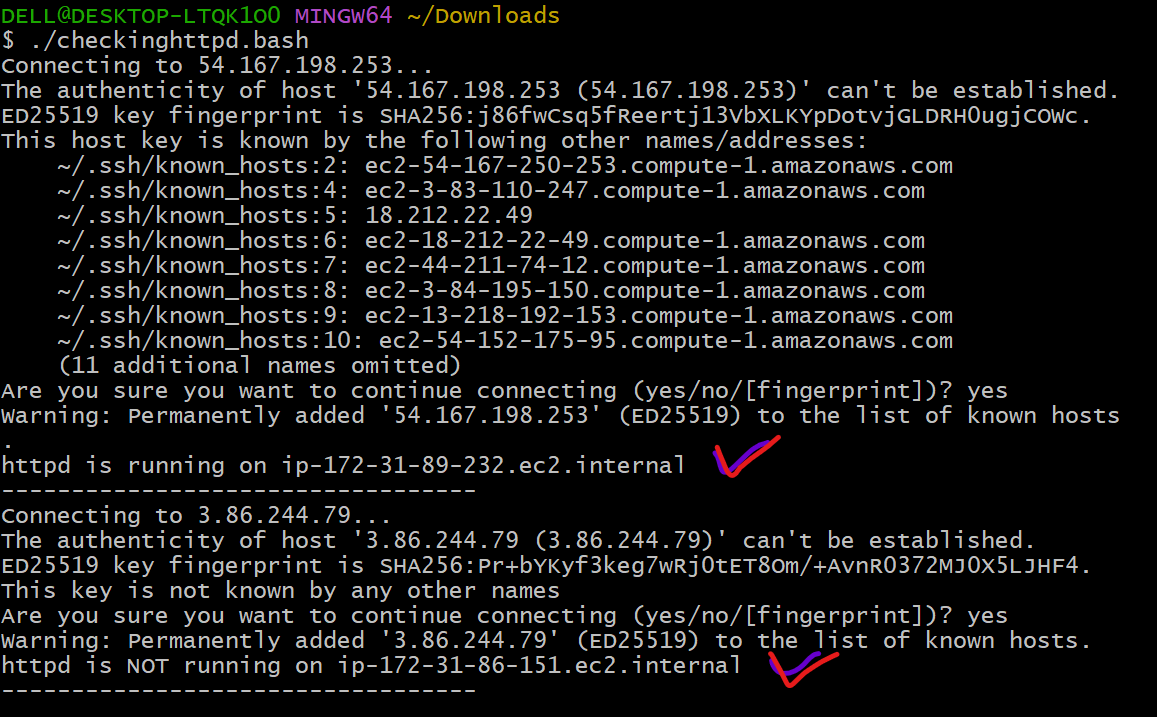
Filename : checkinghttpd.bash



Logic:

**🧠 Explanation Line-by-Line**

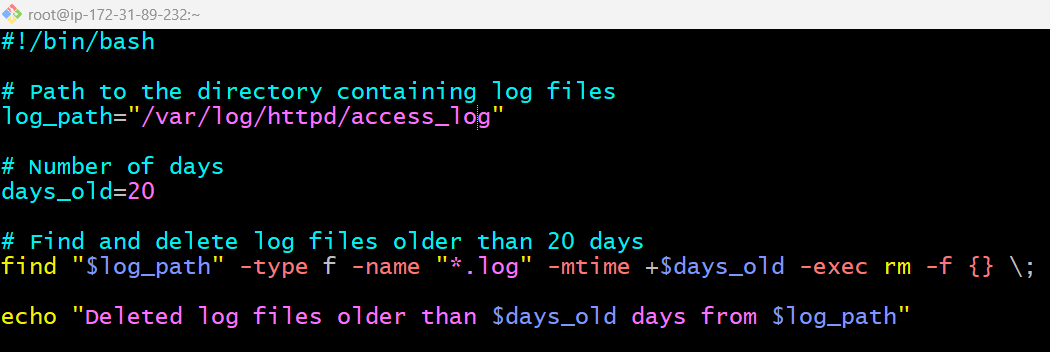
1. #!/bin/bash
   * Tells the system this script should be run with bash.
2. servers=("54.167.198.253" "3.88.50.10")
   * These are the **public IPs** of your EC2 instances that you'll connect to.
3. key\_path="path/to/my-key.pem"
   * The full path to your **EC2 SSH key** file.
   * Example: /c/Users/yourname/Downloads/my-key.pem (in Git Bash on Windows).
4. username="ec2-user"
   * The default user for Amazon Linux. If you're using Ubuntu, change it to ubuntu.
5. ssh -i "$key\_path" "$username@$server"
   * Connects to the server using your private key.
6. 'if systemctl is-active --quiet httpd; then ... fi'
   * Checks if the httpd service is active **on the remote server**.
7. $(hostname)
   * Gets the remote machine’s hostname for clarity in output.
8. echo "----------------------------------"
   * Separates the output for each server.



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

Logic: logfiles.bash





**🧠 Explanation Line-by-Line**

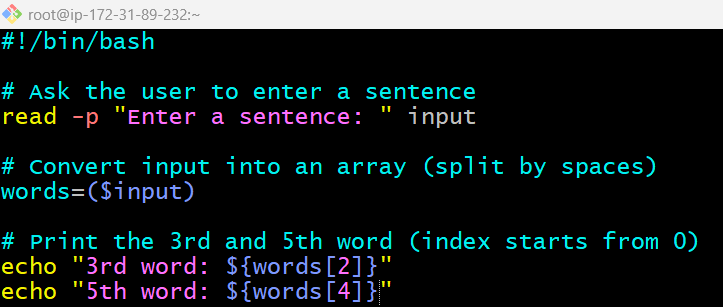
1. #!/bin/bash
   * Runs the script using the **bash shell**.
2. log\_path="/var/log/myapp"
   * Set this to the path where your log files are stored.
   * Example: /var/log/httpd, /home/ec2-user/logs, etc.
3. days\_old=20
   * The age threshold — files older than this number of days will be deleted.
4. find "$log\_path" -type f -name "\*.log" -mtime +$days\_old -exec rm -f {} \;
   * find: searches for files.
   * -type f: finds only regular files.
   * -name "\*.log": filters files ending in .log.
   * -mtime +20: finds files **modified more than 20 days ago**.
   * -exec rm -f {} \;: deletes each file found ({} is replaced with the filename).
5. echo "Deleted log files older than $days\_old days from $log\_path"
   * Displays a summary message after deletion.

Output:



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

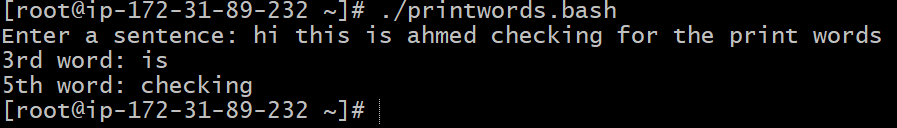
Logic: printwords.bash

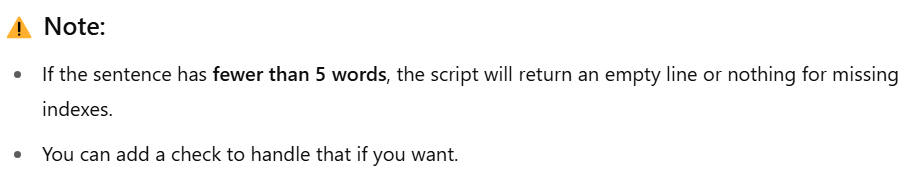


**🧠 Explanation Line-by-Line**

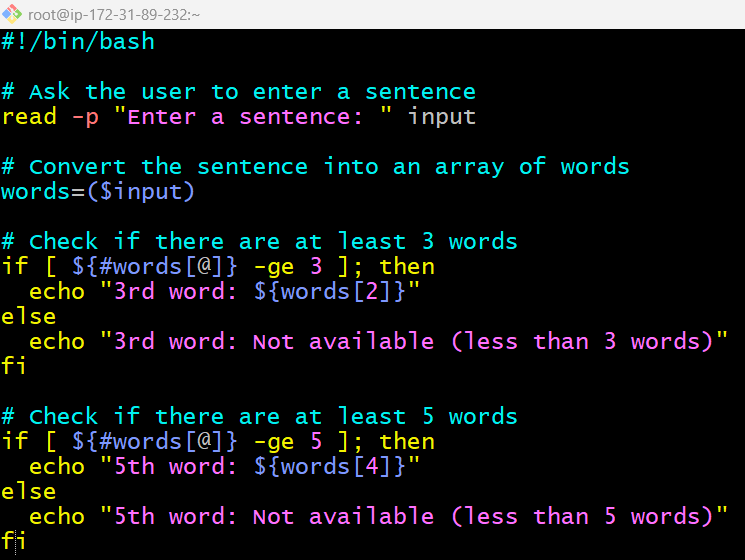
1. #!/bin/bash
   * Indicates the script is a **bash** script.
2. read -p "Enter a sentence: " input
   * Prompts the user to enter a sentence and stores it in the variable input.
3. words=($input)
   * This splits the input sentence into an array named words, using spaces as separators.
4. ${words[2]}
   * Refers to the **3rd word** (arrays in bash start from index 0).
5. ${words[4]}
   * Refers to the **5th word**.

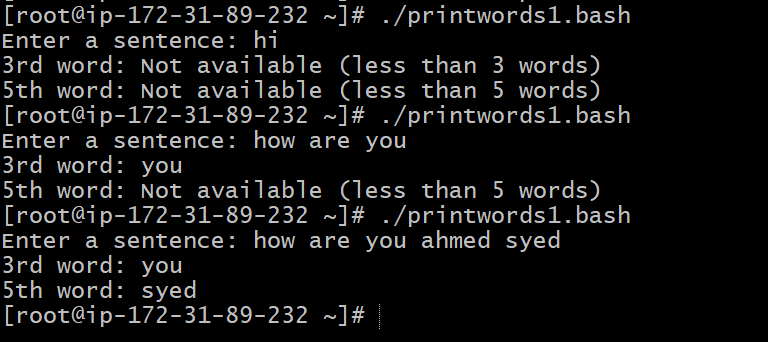
Output:





Logic: printwords1.bash

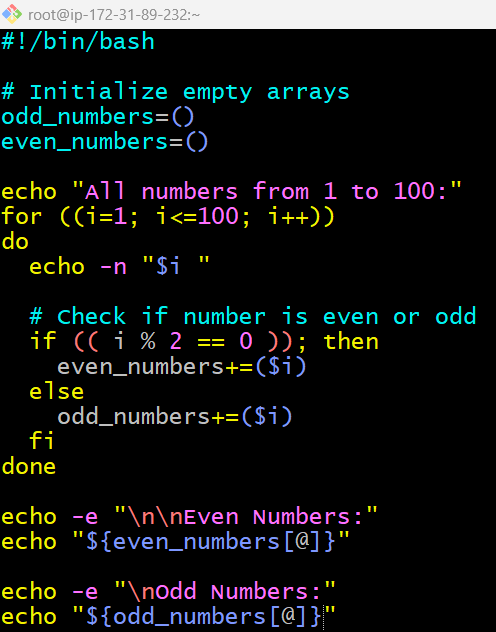




**🧠 What’s New?**

* ${#words[@]} gives the **number of words** entered.
* We check if there are **at least 3** and **at least 5** words before printing the 3rd and 5th words.
* Gives a **friendly message** if words are missing.

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

Logic: printnumberandcheckevenandodd.bash

**🧠 Explanation**

* odd\_numbers=() and even\_numbers=() → initialize empty arrays.
* for ((i=1; i<=100; i++)) → loops from 1 to 100.
* if (( i % 2 == 0 )) → checks if the number is divisible by 2 (even).
* +=($i) → appends the number to the appropriate array.
* echo "${array[@]}" → prints all elements of an array.

Output:

