 

**Placement Empowerment Program**

***Cloud Computing and DevOps Centre***

**Write a Shell Script to monitor Logs**

Create a script that monitors server logs for errors and alerts you.

Name: Mohammed Aneeq M Department:CSE



**Introduction**

Server logs are crucial for monitoring the health and performance of a server. These logs contain vital information about the server's activities, including errors, warnings, and other system messages. Monitoring these logs for errors in real-time is essential for ensuring that any issues are quickly identified and addressed. Writing a shell script to monitor server logs is an effective way to automate this task, saving time and reducing the chances of overlooking critical issues.

**Objective**

The objective of this task is to:

1. Create a shell script that monitors server logs for errors.
2. Parse the logs for keywords like "error" or "fail".
3. Trigger alerts whenever an error is detected in the logs.
4. Use basic shell commands like tail, grep, and echo to achieve this.

**Step-by-Step Procedure**

**1. Create a Shell Script**

Open a terminal on your Linux system and create a new shell script file:

touch monitor\_logs.sh

Make the script executable:

chmod +x monitor\_logs.sh

**2. Open the Script for Editing**

Use a text editor to open the script:

nano monitor\_logs.sh

**3. Define the Log File to Monitor**

Specify the log file you want to monitor. For example, if you are monitoring system logs, you can use /var/log/syslog or /var/log/messages for general system logs. Alternatively, you could monitor a custom application log. Add this line at the top of the script to define the log file:

LOG\_FILE="/var/log/syslog"

**4. Use the tail Command to Monitor Logs in Real Time**

The tail command allows you to view the last few lines of a file. With the -f flag, it will continuously display new log entries as they are added:

tail -f $LOG\_FILE

**5. Search for Specific Error Keywords Using grep**

Use the grep command to search for specific keywords such as "error", "fail", or other patterns that indicate problems. For example, to search for the word "error":

tail -f $LOG\_FILE | grep --line-buffered "error"

The --line-buffered option ensures that each line is processed as soon as it's added to the log.

**6. Add Conditional Logic to Send Alerts**

If the script detects an error in the log, it can trigger an alert. For example, you can use the echo command to print an alert message in the terminal. Alternatively, you can send an email or other notifications. Below is a simple example of an alert in the terminal:

LOG\_FILE="/var/log/syslog"

ERROR\_KEYWORD="error"

tail -f $LOG\_FILE | while read LINE

do

if echo "$LINE" | grep -q "$ERROR\_KEYWORD"; then

echo "Error detected: $LINE"

# Here you could send an email or trigger another notification

fi

done

This script continuously reads the log file. When it finds a line containing the word "error", it outputs an alert in the terminal.

**7. Sending Email Notifications (Optional)**

If you'd prefer to receive an email when an error is detected, you can use the mail command to send an email. Make sure you have mail configured on your system. Here's an example of how you could modify the script to send an email:

LOG\_FILE="/var/log/syslog"

ERROR\_KEYWORD="error"

EMAIL="your-email@example.com"

tail -f $LOG\_FILE | while read LINE

do

if echo "$LINE" | grep -q "$ERROR\_KEYWORD"; then

echo "Error detected: $LINE" | mail -s "Log Error Alert" $EMAIL

fi

done

Replace your-email@example.com with your actual email address.

**8. Test the Script**

Save your script and run it to test its functionality. Ensure the script is running and monitoring the log file as expected. You can simulate errors in the log to verify that alerts are triggered properly.

./monitor\_logs.sh

**9. Monitor the Output**

As the script runs, it will output an alert in the terminal each time an error is detected in the log file. If you've configured email notifications, you'll also receive an email each time an error is found.

**Conclusion**

By following the steps outlined above, you have created a shell script to monitor server logs for errors and alert you in case an issue arises. This script leverages powerful Linux commands such as tail and grep to parse log files in real-time, making it an essential tool for system administrators or developers who need to quickly detect and respond to errors. Additionally, the flexibility of shell scripting allows you to easily customize this script to suit your specific logging and notification needs.