

**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: Gokul Bharathi P Department: ADS



# Introduction

Monitoring server logs is a critical task for system administrators and developers. Logs provide valuable insights into system operations, errors, and potential security issues. Automating log monitoring with a shell script helps detect errors or specific events in real time, enabling timely responses to critical issues. This document outlines the steps to create a shell script that monitors logs for errors and sends alerts.

# Objective

The objectives of this task are:

* Learn how to parse and analyze log files using shell commands.
* Understand the use of tools like grep and tail for monitoring.
* Write a shell script to monitor logs for specific keywords (e.g., "error") and trigger alerts.

**Step 1: Understanding the Requirements Key Concepts:**

* **Log Monitoring:** Regularly scanning log files for specific patterns or events.
* **Parsing Tools:** Using commands like grep to filter relevant information from logs.
* **Automation:** Scheduling scripts to run periodically or continuously.

**Tools/Commands:**

* tail - Continuously reads a log file for new entries.
* grep - Searches for specific patterns in text.
* echo or email utilities - Sends alerts when errors are detected.

# Step 2: Instructions 1. Create the Shell Script

Write a shell script that monitors a log file for specific keywords (e.g., "error"). Here's an example:

#!/bin/bash

# Specify the log file to monitor

LOG\_FILE="/var/log/system.log"

# Keyword to search for

KEYWORD="error"

# Monitor the log file in real-time tail -F "$LOG\_FILE" | while read LINE; do if echo "$LINE" | grep -i "$KEYWORD" > /dev/null; then # Send an alert (example: display a message or send an email) echo "Alert: Error detected in log file!"

# Uncomment the following line to send an email (configure mail settings first)

# echo "Error detected: $LINE" | mail -s "Log Alert" admin@example.com fi

done

# Test the Script

* Save the script to a file, e.g., log\_monitor.sh.
* Make the script executable:

chmod +x log\_monitor.sh

* Run the script:

./log\_monitor.sh

* Simulate an error in the log file to ensure the script works correctly.

# Automate the Script

• **Linux:** Use cron to schedule the script to run at specific intervals. crontab -e

Add the following line to run the script every minute:

\* \* \* \* \* /path/to/log\_monitor.sh

• **Windows:** Use Task Scheduler to run the script.

# Step 3: Best Practices

1. Tailor the script to monitor specific logs and keywords relevant to your system.
2. Test the script in a non-production environment before deploying.
3. Configure email alerts or integrate with monitoring tools like Nagios or Prometheus for advanced use cases.

# Conclusion

Automating log monitoring with a shell script ensures that critical events are detected in real time, minimizing downtime and improving system reliability. By following this guide, you can create a script to monitor logs effectively and adapt it to meet specific monitoring needs.