

# Placement Empowerment Program

## *Cloud Computing and DevOps Centre*

### Day 06-Log File Monitor&Alert Script

Create a script to monitor log files(e.g.,/var/log/syslog)

In real-time and alert when specific keyword like  
“error”or “failed”appear

Name: Rogini.P

Department: ECE

# Introduction

In system administration, log monitoring is essential to **detect issues in real-time**. This Proof of Concept (PoC) uses Linux commands to track the system log file (**/var/log/syslog**) and alert the user when specific keywords are detected.

## Overview

This PoC demonstrates how to monitor Linux log files in real-time and trigger alerts when specific critical keywords such as "**error**", "**failed**", or "**critical**" appear.

Using a simple shell script with tools like **tail**, **grep**, and **echo**, system administrators can quickly detect abnormal system behavior and take immediate action. This solution is lightweight, efficient, and ideal for early detection of issues in production or development environments.

### Key steps in this PoC:

#### ✔ Open Terminal

Launch a terminal on your Linux system to begin the process.

#### ✔ Create the Monitoring Script

Use a text editor like nano to write a shell script (log\_monitor.sh) that monitors the system log file.

#### ✔ Use tail -f to Follow Logs

The script uses **tail -f /var/log/syslog** to track new log entries in real-time.

#### ✔ Filter for Keywords

Pipe the log entries through grep to detect keywords like **error**, **failed**, or **critical** using case-insensitive matching.

#### ✔ Trigger Alerts

If a match is found, the script displays an alert message and the matched log line on the terminal.

### ✓ **Make the Script Executable**

Change script permissions using **chmod +x log\_monitor.sh**.

### ✓ **Run the Script**

Start the log monitoring by executing the script (**./log\_monitor.sh**).

## **Objectives :**

### ✓ **Monitor System Logs in Real-Time**

Continuously observe `/var/log/syslog` to detect system activity and issues as they happen.

### ✓ **Detect Critical Events Automatically**

Search for specific keywords such as **"error"**, **"failed"**, and **"critical"** that indicate potential problems.

### ✓ **Trigger Immediate Alerts**

Notify users instantly via console output whenever a critical log entry is found.

### ✓ **Implement Lightweight Automation**

Use a simple shell script that runs efficiently without needing external tools or heavy monitoring solutions.

### ✓ **Lay the Foundation for Advanced Monitoring**

Establish a base that can be extended to send email alerts, system notifications, or integrate with monitoring tools like Nagios or Prometheus.

# Importance :

## ✓ **System Security & Stability**

Real-time log monitoring helps detect unauthorized access, service failures, and system errors as soon as they occur.

## ✓ **Proactive Troubleshooting**

By identifying critical keywords instantly, administrators can address issues before they escalate into bigger problems.

## ✓ **Time-Saving Automation**

Manual log checking is time-consuming. This script automates the process, increasing efficiency for system monitoring tasks.

## ✓ **Lightweight & Customizable**

Requires no third-party tools—runs with basic Linux utilities. It can also be customized for different log files or keywords.

## ✓ **Scalable for Production Use**

This basic setup serves as a foundation for building more advanced alerting systems (**e.g., email, Slack, or cloud alerts**).

# Step-by-Step Overview

## Step 1:Open Terminal

Launch a terminal window on your Linux system.

## Step 2:Create a Shell Script File

Create a new shell script named log\_monitor.sh:

```
rogini26@LAPTOP-H69F05A7:~$ nano log_monitor.sh
```

## Step 3:Write the Monitoring Script

In the nano editor,Paste the following code:

```
#!/bin/bash

# Log file to monitor
LOG_FILE="/var/log/syslog"

# Keywords to look for
KEYWORDS=("error" "failed")

# Function to check each line for keywords
check_keywords() {
    local line="$1"
    for keyword in "${KEYWORDS[@]}; do
        if echo "$line" | grep -iq "$keyword"; then
            echo "⚠️ALERT: Found keyword '$keyword' in log:"
            echo "$line"
        fi
    done
}

# Monitor the log file in real-time
echo "🐧 Monitoring $LOG_FILE for keywords: ${KEYWORDS[*]}"
tail -F "$LOG_FILE" | while read -r line; do
    check_keywords "$line"
done
```

## Step 4: Save and Exit

Press Ctrl + O → Enter (to save)

Press Ctrl + X (to exit)

## Step 5: Make the Script Executable

Back in the terminal:

```
rogini26@LAPTOP-H69F05A7:~$ chmod +x log_monitor.sh
```

This gives the script permission to run as a program.

## Step 6: Run the Script with sudo

Since you're monitoring a protected system log file, run the script using:

```
rogini26@LAPTOP-H69F05A7:~$ sudo ./log_monitor.sh
```

You'll be promoted to enter your Linux password.

## Step 7: Script Starts Monitoring

Once running, you'll see:

```
rogini26@LAPTOP-H69F05A7:~$ sudo ./log_monitor.sh
Monitoring /var/log/syslog for keywords: error failed
```

This means it's **actively watching the log file** in real time.

## Step 8:Test the Script(Optional)

If nothing is appearing,you can simulate a log message by typing:

You should see output like:

```
rogini26@LAPTOP-H69F05A7:~$ echo "This is a fake error message for testing" | sudo tee -a /var/log/syslog
[sudo] password for rogini26:
This is a fake error message for testing
```

## Outcomes:

### ✓ **Real-Time Log Monitoring**

You gain the ability to watch log files live and instantly detect system issues or failures.

### ✓ **Automated Alerts**

Critical events containing keywords like error, failed, or critical are immediately flagged without manual log inspection.

### ✓ **Improved Troubleshooting Speed**

Instant feedback helps reduce downtime by allowing you to act on issues as soon as they occur.

### ✓ **Hands-on Shell Scripting Experience**

Reinforces core Linux skills—especially in using tools like tail, grep, conditionals, and script automation.

### ✓ **Scalable for Production Environments**

Forms the base for future enhancements like sending email/SMS alerts, integrating with monitoring tools, or monitoring custom log files.