Placement Empowerment Program

Cloud Computing and DevOps Centre

Sheel Script To Monitor Logs

Name: Samuel James Billy Graham

Department: CSE

Introduction and Overview

Log monitoring is an essential process in **system administration, DevOps, and security** to track system activities, detect errors, and troubleshoot issues. Automating log monitoring using a **Shell Script** helps administrators efficiently **analyze logs, identify failures, and trigger alerts** when required.

This **Proof of Concept (PoC)** will guide you through:

- Writing a **Shell Script** to monitor logs in real-time.
- Filtering log messages based on keywords (e.g., "ERROR", "CRITICAL").
- Automating alerts when specific logs appear.

By following this process, you will gain hands-on experience with **Shell scripting, log analysis, and system monitoring**.

Objectives

The goal of this project is to:

✅ Understand the importance of log monitoring.
✅ Create a **Shell Script** to track logs in real-time.
✅ Filter logs based on error keywords.
✅ Trigger alerts when issues are detected.
✅ Automate log analysis for efficient troubleshooting.

Importance of Log Monitoring

📌 **Detects System Issues** – Identifies errors, warnings, and critical failures.

📌 **Enhances Security** – Monitors unauthorized access attempts.

📌 **Automates Alerts** – Sends notifications when errors occur.

📌 **Reduces Downtime** – Quick detection leads to faster issue resolution.

📌 **Improves Troubleshooting** – Provides insights into system behavior.

Step-by-Step Overview

Step 1: Identify the Log File to Monitor

- Common system log files:
  - **/var/log/syslog** → General system logs.
  - **/var/log/auth.log** → Authentication logs.
  - **/var/log/nginx/access.log** → Nginx access logs.
  - **/var/log/nginx/error.log** → Nginx error logs.
- Choose the log file based on your requirement.

Step 2: Create a Shell Script for Log Monitoring

1. Open the terminal and create a new script file:

CopyEdit

```
nano monitor_logs.sh
```

2. Add the following script:

CopyEdit

```
#!/bin/bash
```

```
# Define the log file to monitor
LOG_FILE="/var/log/syslog"

# Define the keyword to monitor (e.g., ERROR, CRITICAL, FAILED)
KEYWORD="ERROR"

# Define the alert message
ALERT_MESSAGE="ALERT: '$KEYWORD' found in logs!"

# Monitor the log file in real-time and trigger an alert when the keyword appears
tail -F "$LOG_FILE" | while read LINE
do
  if [[ "$LINE" == *"$KEYWORD"* ]]; then
    echo "$ALERT_MESSAGE"
    echo "$ALERT_MESSAGE" | mail -s "Log Alert" admin@example.com  # Send email alert
  fi
done
```

3. Save and exit (CTRL + X, then Y and Enter).

Step 3: Make the Script Executable

Run the following command to give execution permission:

CopyEdit

```
chmod +x monitor_logs.sh
```

Step 4: Run the Log Monitoring Script

Execute the script to start monitoring logs in real-time:

CopyEdit

```
./monitor_logs.sh
```

It will continuously check for logs containing the **"ERROR"** keyword and trigger an alert if detected.

Step 5: Automate the Script Using Cron Jobs (Optional)

To run the script automatically every **10 minutes**, add it to **Cron Jobs**:

1. Open the crontab file:

CopyEdit

```
crontab -e
```

2. Add the following line:

CopyEdit

```
*/10 * * * * /path/to/monitor_logs.sh
```

3. Save and exit.
   Now, the script will **run every 10 minutes** to check for errors.
