

Placement Empowerment Program

Cloud Computing and DevOps Centre

Write the Shell Script to Monitor Logs : Create a script that monitors server logs for errors and alert you

Name: Samraj K

Department:CSE

Introduction

Log files play a critical role in IT systems, as they record activities and events generated by applications, servers, and network devices. Monitoring these logs helps identify issues such as errors, warnings, and suspicious activities that may require immediate attention. Automating the monitoring process ensures efficiency and reduces the risk of missing critical information.

This PoC demonstrates the creation of a **PowerShell script** to monitor logs in real-time. The script will detect specific keywords (like "error") in a log file and alert the user when such events occur.

Overview

This project involves writing and running a PowerShell script that continuously scans a log file for specific keywords. The script:

1. Reads the log file in real time.
2. Matches new entries in the log against predefined keywords (e.g., "error").
3. Triggers an alert when a match is found.

This solution is lightweight and practical for system administrators and IT professionals to monitor logs on Windows systems.

Objective

The objective of this project is to:

1. Automate the process of monitoring log files for critical events.
2. Learn how to create and execute PowerShell scripts on a Windows system.
3. Demonstrate real-time detection of keywords like "error" in log files.
4. Enhance troubleshooting efficiency by providing immediate alerts for critical events.

Importance

1. Proactive Issue Detection

By monitoring logs in real time, this project helps detect errors and issues as they occur, reducing downtime and improving system reliability.

2. Learning Automation Tools

This project introduces PowerShell scripting, a powerful automation tool, to beginners. It provides hands-on experience with practical applications.

3. Cost-Effective Solution

Using PowerShell eliminates the need for expensive third-party monitoring tools while still achieving effective log monitoring.

4. Time Efficiency

Automation saves significant manual effort in scanning logs, allowing IT professionals to focus on resolving issues.

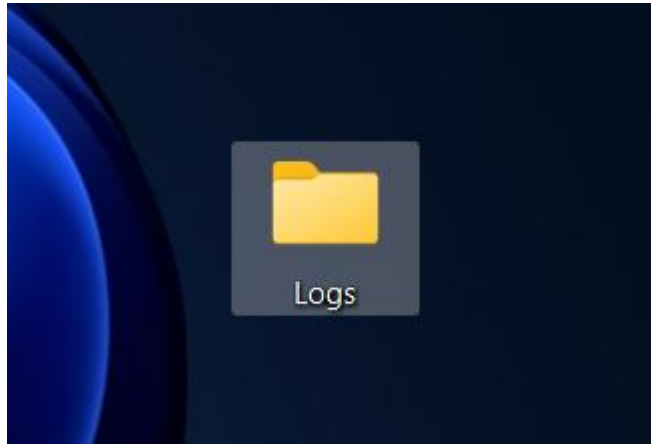
5. Scalability

The script can be adapted to monitor multiple log files or handle complex use cases, making it a foundational step toward advanced automation.

Step-by-Step Overview

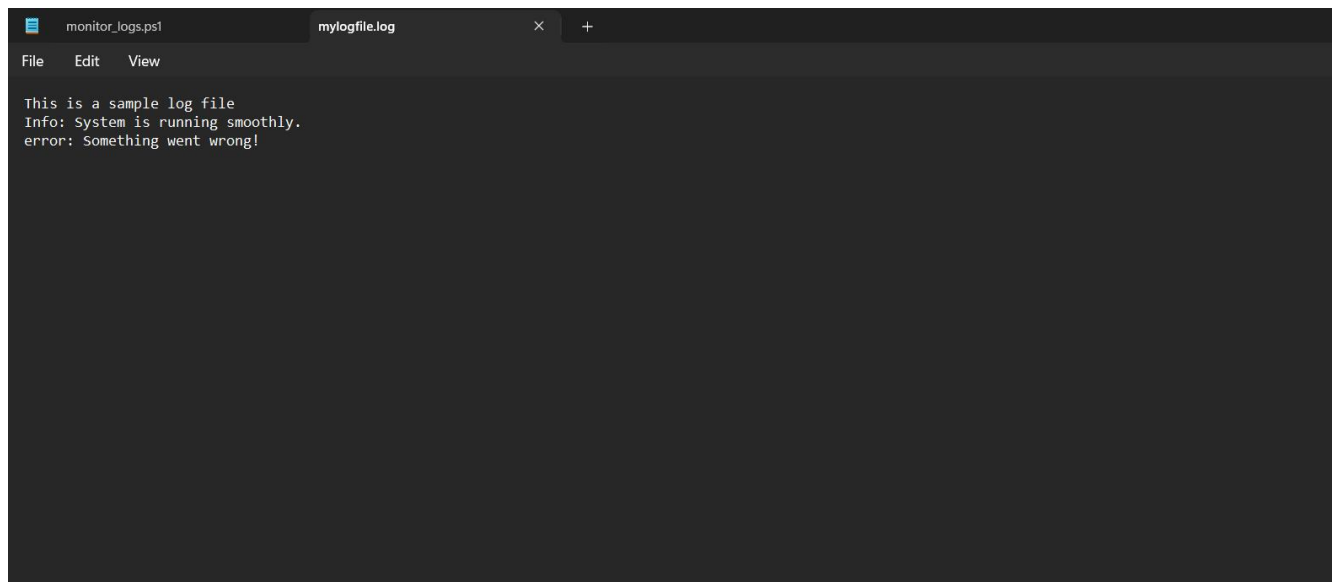
Step 1:

Create a Folder called logs for Your Logs and Script



Step 2:

Open Notepad and Add the following sample text to it and Save the file as **mylogfile.log** inside the logs folder

A screenshot of a Notepad window with a dark theme. The title bar shows two tabs: 'monitor_logs.ps1' and 'mylogfile.log'. The 'mylogfile.log' tab is active. The menu bar includes 'File', 'Edit', and 'View'. The text content of the file is as follows:

```
This is a sample log file
Info: System is running smoothly.
error: Something went wrong!
```

Step 3:

Open Notepad and Type the following PowerShell script into it and Set the \$LogFilePath address to the mylogfile.log which you saved in logs folder. Save the file as monitor_logs.ps1 inside the same logs folder

```
monitor_logs.ps1
File Edit View

# Define the path to the log file
$LogFilePath = "C:\Users\samni\Desktop\mylogfile.log"

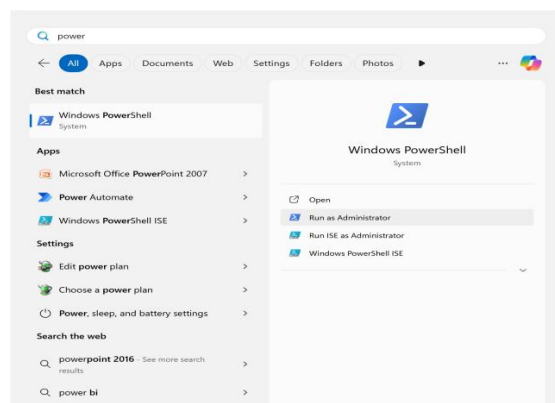
# Define the keyword to monitor
$Keyword = "error"

# Function to send an alert
Function Send-Alert {
    param([string]$Message)
    Write-Host "ALERT: $Message" -ForegroundColor Red
}

# Monitor the log file for new entries
Get-Content -Path $LogFilePath -Wait -Tail 0 | ForEach-Object {
    if ($_ -match $Keyword) {
        Send-Alert "Keyword '$Keyword' found in log: $_"
    }
}
```

Step 4:

Click the Windows Key and Search for Windows PowerShell and click Run as Administrator.



Step 5:

Run the following command to allow script execution:

When prompted, type Y and press Enter.

```
PS C:\Windows\system32> Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

Step 6:

Navigate to the logs folder

```
PS C:\Windows\system32> cd "C:\Users\samni\Desktop\Logs"
```

Step 7:

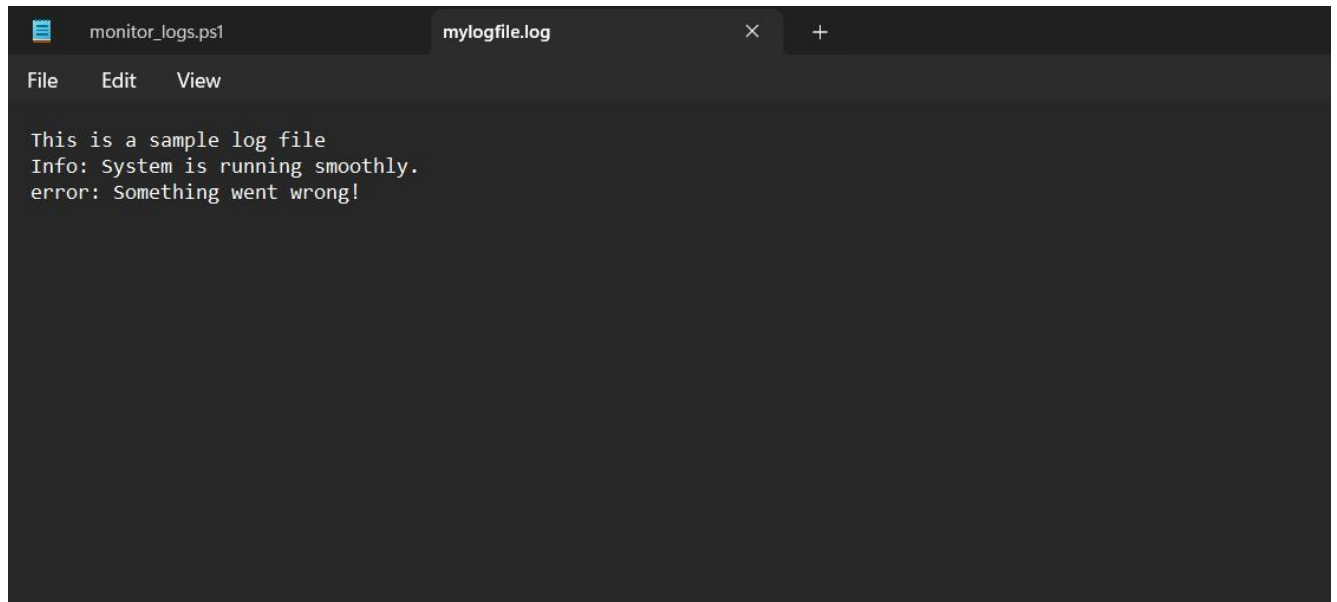
Run the script:

.\monitor_logs.ps1

```
PS C:\Users\samni\Desktop\Logs> .\monitor_logs.ps1|
```

Step 8:

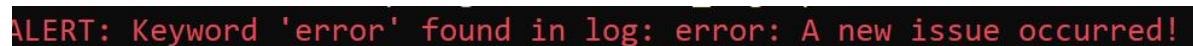
Open mylogfile.log in Notepad and Add a new line with the word "error" and Save the file.

A screenshot of a Notepad window with two tabs: 'monitor_logs.ps1' and 'mylogfile.log'. The 'mylogfile.log' tab is active. The text inside the Notepad is: 'This is a sample log file', 'Info: System is running smoothly.', and 'error: Something went wrong!'. The menu bar shows 'File', 'Edit', and 'View'.

Step 9:

Check PowerShell — you should see an alert like:

ALERT: Keyword 'error' found in log: error: A new issue occurred!

A screenshot of a PowerShell terminal window showing the alert message: 'ALERT: Keyword 'error' found in log: error: A new issue occurred!'. The text is red on a black background.

Explanation:

1. When the script is running, it continuously monitors the log file.
2. If any line containing the keyword "error" is added to the file, it immediately triggers an alert in PowerShell.

Outcome:

By completing this Proof of Concept (PoC), we will:

1. Successfully create and execute a PowerShell script to monitor log files in real time.
2. Detect and alert on predefined keywords (e.g., "error") to highlight critical events.
3. Gain hands-on experience with PowerShell scripting and automation on a Windows system.
4. Understand the importance of log monitoring in proactive system maintenance and troubleshooting.
5. Learn to customize and scale the script for more advanced monitoring scenarios in future projects.