# 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: Aksha S Department: CSE

# Introduction

Log files are essential components of IT systems, capturing activities and events generated by applications, servers, and network devices. Regular monitoring of these logs helps detect errors, warnings, and potential security threats that require immediate action. Automating this process enhances efficiency and minimizes the chances of overlooking critical information.

This Proof of Concept (PoC) showcases a PowerShell script designed to monitor log files in real time. The script scans logs for specific keywords (such as "error") and promptly notifies the user when such events are detected.

# Overview

This project focuses on developing and executing a PowerShell script that actively monitors a log file for specific keywords. The script operates as follows:

1. Continuously reads the log file in real time.
2. Checks new log entries against predefined keywords (e.g., "error").
3. Sends an alert whenever a matching entry is detected.

This approach provides a lightweight and effective solution for system administrators and IT professionals to efficiently monitor logs on Windows system

# Objective

The goal of this project is to:

1. Automate log file monitoring to detect critical events efficiently.
2. Gain hands-on experience in creating and executing PowerShell scripts on Windows.
3. Implement real-time detection of keywords such as "error" in log files.
4. Improve troubleshooting by delivering instant alerts for critical issues.

# Importance

### 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.

### Learning Automation Tools

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

### Cost-Effective Solution

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

### Time Efficiency

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

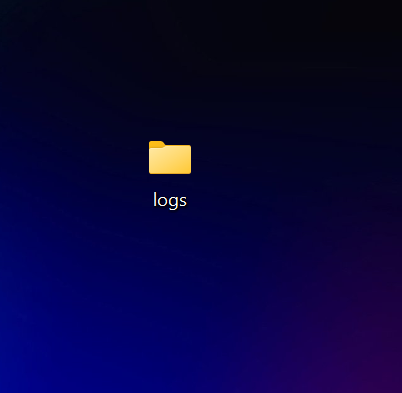
### Scalability

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

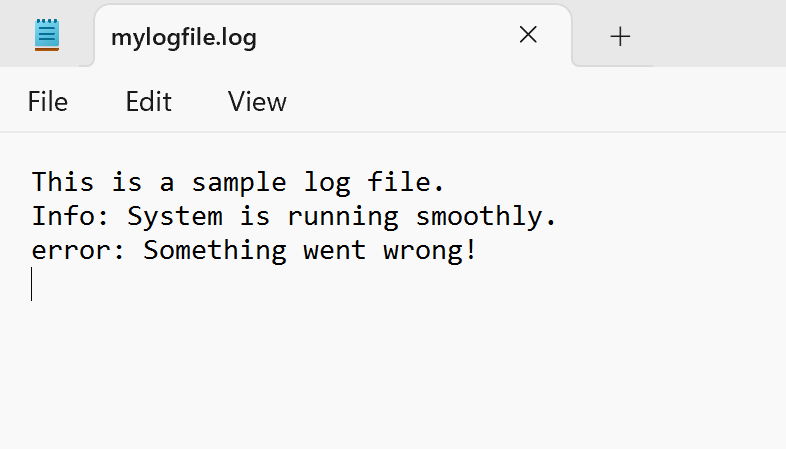
## Step 1:

Create a Folder called logs for Your Logs and Script



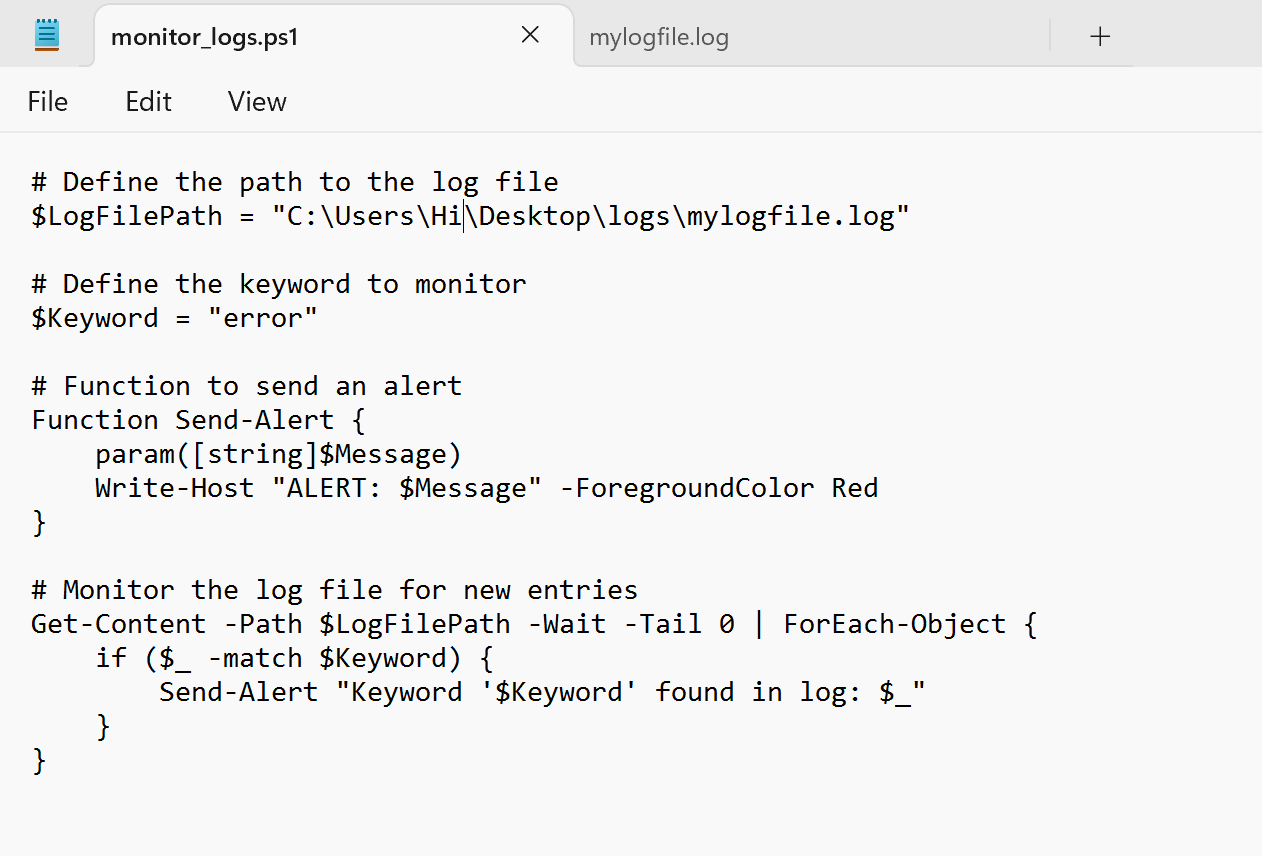
## Step 2:

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



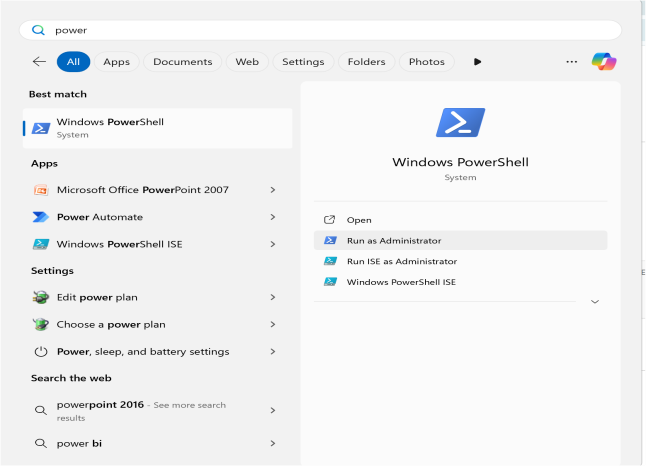
## Step 3:

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



## Step 4:

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

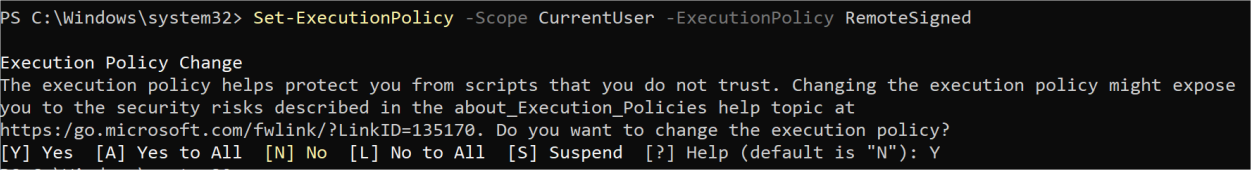


## Step 5:

Run the following command to allow script execution:

### Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned

When prompted, type Y and press Enter.



## Step 6:

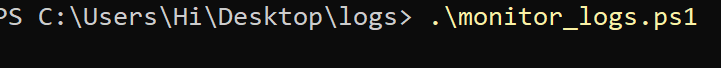
Navigate to the logs folder

Screenshot 2025-01-26 135858.png

## Step 7:

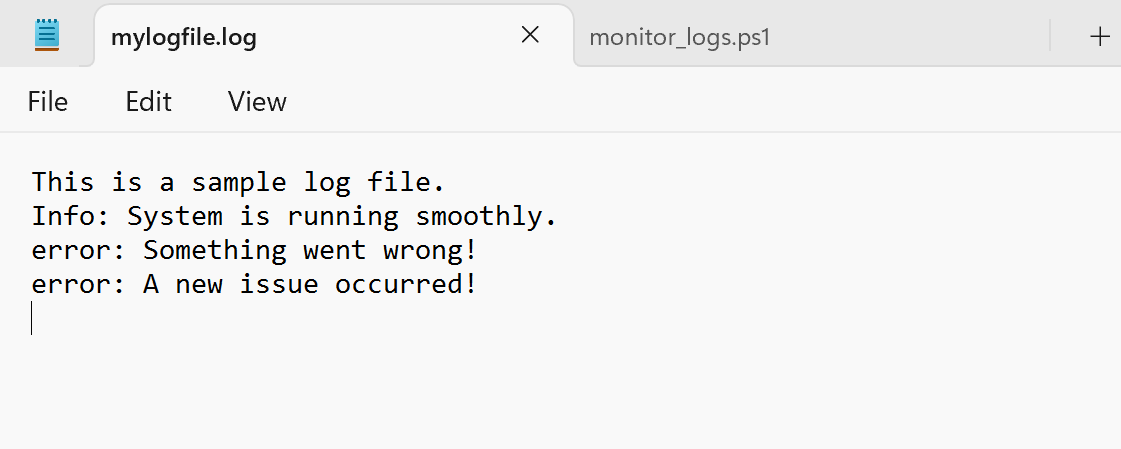
Run the script:

**.\monitor\_logs.ps1**

****

## Step 8:

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



## Step 9:

Check PowerShell — you should see an alert like:

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

**Screenshot 2025-01-26 140047.png**

## 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. Develop and run a PowerShell script that continuously monitors log files in real time.
2. Identify and alert on specific keywords (e.g., "error") to highlight critical system events.
3. Gain practical experience with PowerShell scripting and automation on Windows.
4. Recognize the significance of log monitoring for proactive system maintenance and troubleshooting.
5. Learn how to customize and expand the script for advanced monitoring in future projects.