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: Harshana Perianayaki B   Department : IT

INTRODUCTION

In any server environment, logs play a crucial role in monitoring system health, identifying issues, and ensuring smooth operation. A log monitoring script automates the process of checking logs for critical errors and notifying administrators in real-time. This helps in quick troubleshooting and minimizing downtime.

OVERVIEW

This document outlines a shell script that monitors server log files for errors and alerts administrators whenever an issue is detected. The script reads the log file in real-time, searches for specific error keywords, and triggers alerts through console messages, sound notifications, or email alerts.

The script is particularly useful for monitoring logs generated by web servers, application servers, and database systems. It continuously checks the log file for new entries, reducing the need for manual log inspection.

OBJECTIVES

The primary goal of this shell script is to:

- Automate the process of scanning server logs.
- Detect critical errors such as "ERROR," "FATAL," "CRITICAL," and "EXCEPTION."
- Provide real-time alerts through different notification methods.
- Reduce manual intervention and enhance system reliability.
- Improve the response time for resolving system issues.

IMPORTANCE OF AUTOMATED DEPLOYMENT

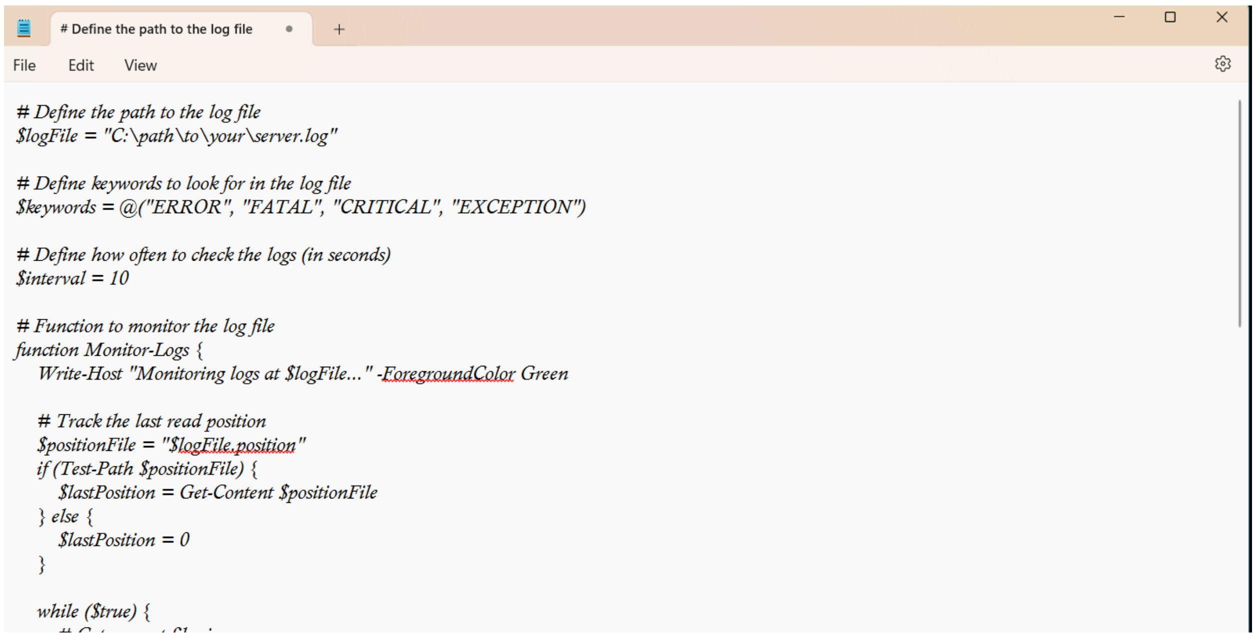
Monitoring logs manually is inefficient and prone to human error. A shell script automates this process, ensuring that no critical error goes unnoticed. Key benefits include:

- **Enhanced System Reliability**: Quick detection and resolution of errors prevent system failures.
- **Proactive Monitoring**: Helps administrators take preventive actions before a minor issue escalates.
- **Time Efficiency**: Reduces the effort spent manually reviewing logs.
- **Real-time Alerts**: Notifies administrators instantly, allowing for faster troubleshooting.

STEP-BY-STEP OVERVIEW

Since Windows does not support traditional shell scripting (`.sh`), the best approach is to use **PowerShell**. Below is a **PowerShell script** that monitors a log file for errors and alerts you if any issues occur.

PowerShell Script to Monitor Logs for Errors



```
# Define the path to the log file
$logFile = "C:\path\to\your\server.log"

# Define keywords to look for in the log file
$keywords = @("ERROR", "FATAL", "CRITICAL", "EXCEPTION")

# Define how often to check the logs (in seconds)
$interval = 10

# Function to monitor the log file
function Monitor-Logs {
    Write-Host "Monitoring logs at $logFile..." -ForegroundColor Green

    # Track the last read position
    $positionFile = "$logFile.position"
    if (Test-Path $positionFile) {
        $lastPosition = Get-Content $positionFile
    } else {
        $lastPosition = 0
    }

    while ($true) {
```

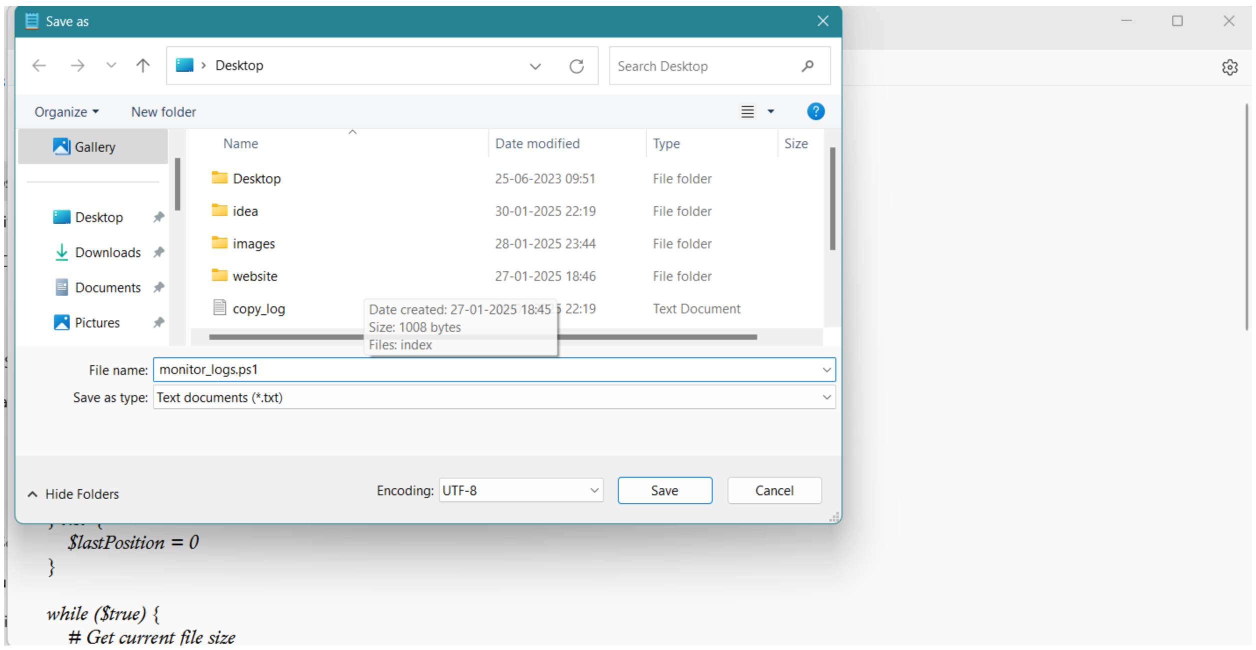
How It Works

1. **Monitors the specified log file** (`server.log`) for error keywords.

2. **Detects newly added log lines** since the last check.
3. **Alerts you** (console warning + sound beep) when an error is found.
4. **Runs continuously**, checking every **10 seconds**.

How to Use

1. **Save the script** as `monitor_logs.ps1`.



Name	Date modified	Type	Size
Desktop	25-06-2023 09:51	File folder	
idea	30-01-2025 22:19	File folder	
images	28-01-2025 23:44	File folder	
website	27-01-2025 18:46	File folder	
copy_log	22:19	Text Document	

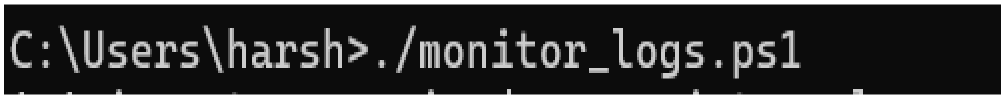
Date created: 27-01-2025 18:45
Size: 1008 bytes
Files: index

File name: monitor_logs.ps1
Save as type: Text documents (*.txt)
Encoding: UTF-8

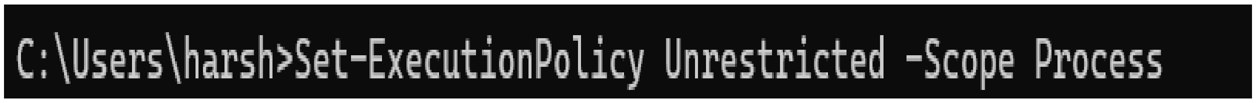
```
    $lastPosition = 0
}

while ($true) {
    # Get current file size
```

2. **Modify** `$logFile` with the actual path of your server log file.

3. **Run PowerShell as Administrator** and execute:



```
C:\Users\harsh>./monitor_logs.ps1
```



```
C:\Users\harsh>Set-ExecutionPolicy Unrestricted -Scope Process
```

Enhancements (Optional)

- **Send email alerts** when an error is found.
- **Write detected errors** into a separate log file.
- **Trigger a popup notification** in Windows.
- **Use Task Scheduler** to run automatically.

OUTCOME

By implementing this shell script, organizations can expect:

- **Improved System Stability**: Early detection and response to errors.
- **Faster Issue Resolution**: Alerts allow for immediate action.
- **Reduced Downtime**: Ensures continuous system operation with minimal disruptions.
- **Operational Efficiency**: Saves time by automating the log monitoring process.

This script serves as a valuable tool for IT administrators, DevOps engineers, and system administrators who need an efficient way to track server health and prevent unexpected failures.