Lab 6 – Monitoring and Automation with PowerShell

Course: NET-2220 – Server Management
Student: Clayton Holden
Date Completed:

---

1. Purpose

The purpose of this lab was to create an automated monitoring solution for Windows Server 2022 using PowerShell and Task Scheduler. The goal was to collect service status, recent system events, and high-CPU processes, then generate a daily health report that can be used by IT operations or SOC teams for routine system checks.

---

2. Monitoring Folder Setup

A dedicated monitoring directory was created to store scripts and generated reports:

```
C:\Scripts\Monitoring
C:\Scripts\Monitoring\Logs
```

Permissions were applied to ensure that Administrators retained full control.
This provided a clean location for both automation scripts and exported health reports.

---

3. PowerShell Script Creation

A script named **ServerMonitor.ps1** was created in the Monitoring folder.
The script collects:

- Services that are *not* running

- The 20 most recent System event log entries
- The top 10 processes by CPU usage

It timestamps each report and writes results to:

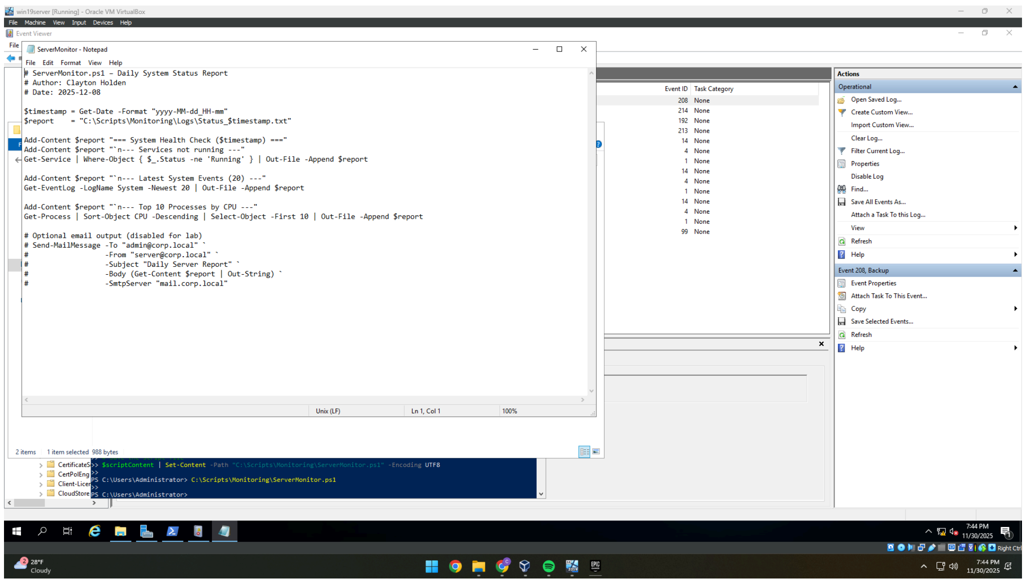
```
C:\Scripts\Monitoring\Logs\Status_<date>.txt
```

A portion of the script is shown below:

```
# ServerMonitor.ps1 — Daily System Status Report
$timestamp = Get-Date -Format "yyyy-MM-dd_HH-mm"
$report    = "C:\Scripts\Monitoring\Logs\Status_$timestamp.txt"

Add-Content $report "=== System Health Check ($timestamp) ==="
Add-Content $report "`n--- Services not running ---"
Get-Service | Where-Object { $_.Status -ne 'Running' } | Out-File
-Append $report

Add-Content $report "`n--- Latest System Events (20) ---"
Get-EventLog -LogName System -Newest 20 | Out-File -Append $report

Add-Content $report "`n--- Top 10 Processes by CPU ---"
Get-Process | Sort-Object CPU -Descending | Select-Object -First 10 |
Out-File -Append $report
```

The script was executed manually to verify that a report file was created successfully.

Evidence:



```
# ServerMonitor.ps1 – Daily System Status Report
# Author: Clayton Holden
# Date: 2025-12-08

$timestamp = Get-Date -Format "yyyy-MM-dd_HH-mm"
$report    = "C:\Scripts\Monitoring\Logs\Status_$timestamp.txt"

Add-Content $report "=== System Health Check ($timestamp) ==="
Add-Content $report "`n--- Services not running ---"
Get-Service | Where-Object { $_.Status -ne 'Running' } | Out-File -Append $report

Add-Content $report "`n--- Latest System Events (20) ---"
Get-EventLog -LogName System -Newest 20 | Out-File -Append $report

Add-Content $report "`n--- Top 10 Processes by CPU ---"
Get-Process | Sort-Object CPU -Descending | Select-Object -First 10 | Out-File -Append $report

# Optional email output (disabled for lab)
# Send-MailMessage -To "admin@corp.local" `
#                  -From "server@corp.local" `
#                  -Subject "Daily Server Report" `
#                  -Body (Get-Content $report | Out-String) `
#                  -SmtpServer "mail.corp.local"
```

```
>> $scriptContent | Set-Content -Path "C:\Scripts\Monitoring\ServerMonitor.ps1" -Encoding UTF8
>>
PS C:\Users\Administrator> C:\Scripts\Monitoring\ServerMonitor.ps1
>>
PS C:\Users\Administrator>
```

---

4. Task Scheduler Automation

A scheduled task named **"Server Monitoring Daily"** was created using Task Scheduler with the following configuration:

- Trigger: Daily at 08:00 AM
- Action:
  - Program: `powershell.exe`

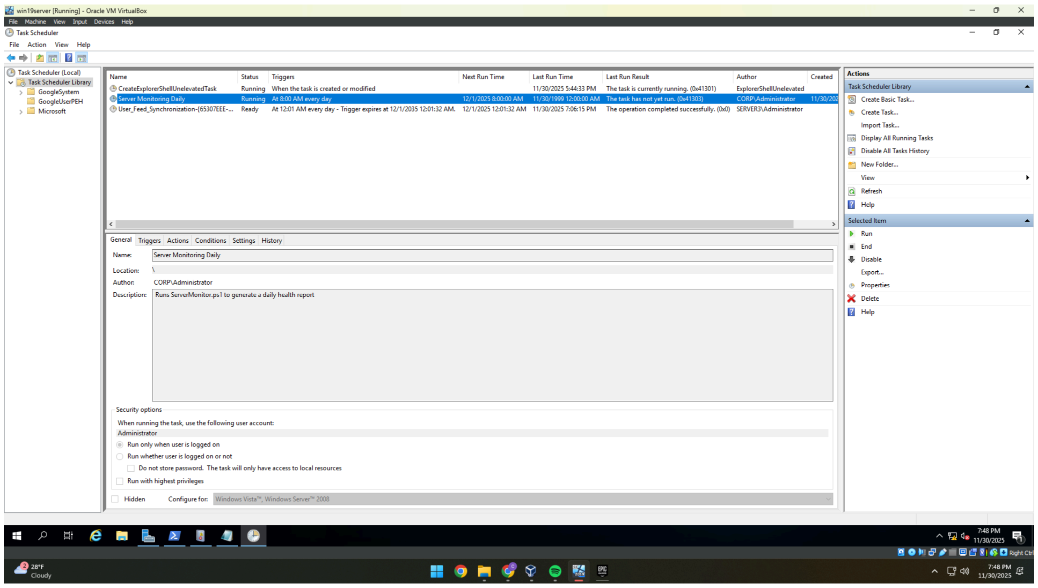
Arguments:

```
-ExecutionPolicy Bypass -File
"C:\Scripts\Monitoring\ServerMonitor.ps1"
```

  -

The task was run manually to confirm that a new output file appeared in the Logs directory and that the script executed as expected.

Evidence:



Name	Status	Triggers	Next Run Time	Last Run Time	Last Run Result	Author	Created
CreateExplorerShellUnelevatedTask	Running	When the task is created or modified		11/30/2025 5:44:33 PM	The task is currently running. (0x41301)	ExplorerShellUnelevated	
Server Monitoring Daily	Running	At 8:00 AM every day	12/1/2025 8:00:00 AM	11/30/1999 12:00:00 AM	The task has not yet run. (0x41303)	CORP\Administrator	11/30/202
User_Feed_Synchronization-{65307EEE-...	Ready	At 12:01 AM every day - Trigger expires at 12/1/2035 12:01:32 AM.	12/1/2025 12:01:32 AM	11/30/2025 7:06:15 PM	The operation completed successfully. (0x0)	SERVER3\Administrator	

General | Triggers | Actions | Conditions | Settings | History

Name: Server Monitoring Daily

Location: \

Author: CORP\Administrator

Description: Runs ServerMonitor.ps1 to generate a daily health report

Security options

When running the task, use the following user account:

Administrator

[✓ Run only when user is logged on]

Run whether user is logged on or not

Do not store password. The task will only have access to local resources

Run with highest privileges

Hidden Configure for: Windows Vista™, Windows Server™ 2008

---

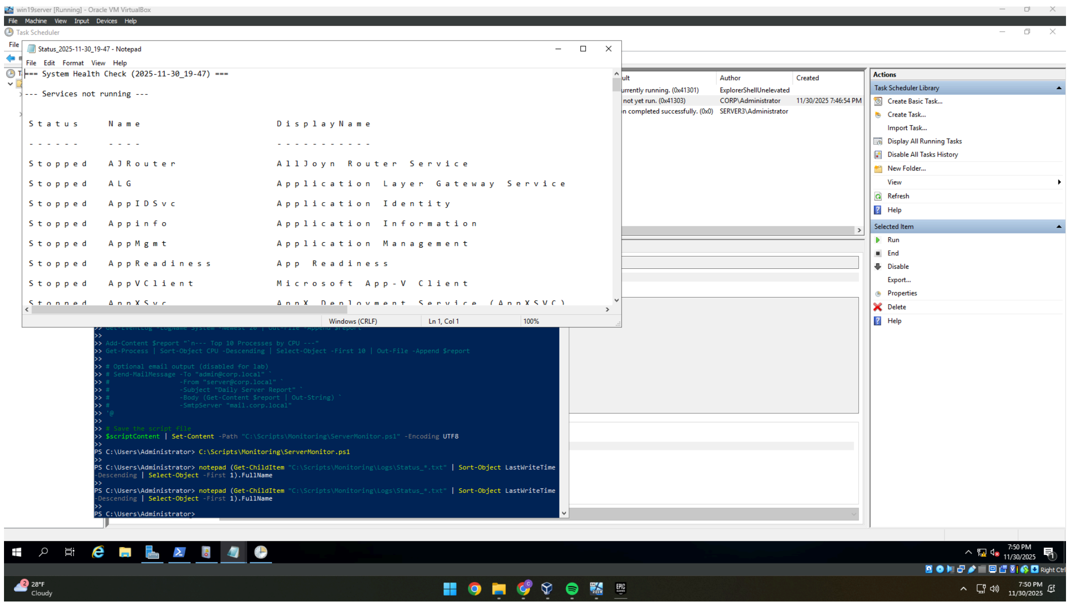
5. Report Validation

The generated report file contained:

- A timestamp
- A list of services not running
- The 20 most recent System log events
- The top 10 CPU-consuming processes

This confirmed that the script ran correctly and collected the required operational data.

Evidence:



```
=== System Health Check (2025-11-30_19-47) ===

--- Services not running ---

Status    Name              DisplayName
------    ----              -----------
Stopped   AJRouter          AllJoyn Router Service
Stopped   ALG               Application Layer Gateway Service
Stopped   AppIDSvc          Application Identity
Stopped   Appinfo           Application Information
Stopped   AppMgmt           Application Management
Stopped   AppReadiness      App Readiness
Stopped   AppVClient        Microsoft App-V Client
Stopped   AppXSvc           AppX Deployment Service (AppXSVC)
```

```
>>
>> Add-Content $report "`n--- Top 10 Processes by CPU ---"
>> Get-Process | Sort-Object CPU -Descending | Select-Object -First 10 | Out-File -Append $report
>>
>> # Optional email output (disabled for lab)
>> # Send-MailMessage -To "admin@corp.local" `
>> #                  -From "server@corp.local" `
>> #                  -Subject "Daily Server Report" `
>> #                  -Body (Get-Content $report | Out-String) `
>> #                  -SmtpServer "mail.corp.local"
>> '@
>>
>> # Save the script file
>> $scriptContent | Set-Content -Path "C:\Scripts\Monitoring\ServerMonitor.ps1" -Encoding UTF8
>>
PS C:\Users\Administrator> C:\Scripts\Monitoring\ServerMonitor.ps1
>>
PS C:\Users\Administrator> notepad (Get-ChildItem "C:\Scripts\Monitoring\Logs\Status_*.txt" | Sort-Object LastWriteTime
-Descending | Select-Object -First 1).FullName
>>
PS C:\Users\Administrator> notepad (Get-ChildItem "C:\Scripts\Monitoring\Logs\Status_*.txt" | Sort-Object LastWriteTime
-Descending | Select-Object -First 1).FullName
>>
PS C:\Users\Administrator>
```

win19server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Task Scheduler
Status_2025-11-30_19-47 - Notepad
File Edit Format View Help
Windows (CRLF) Ln 1, Col 1 100%
Actions
Task Scheduler Library
Create Basic Task...
Create Task...
Import Task...
Display All Running Tasks
Disable All Tasks History
New Folder...
View
Refresh
Help
Selected Item
Run
End
Disable
Export...
Properties
Delete
Help
7:50 PM 11/30/2025

---

6. Summary

Lab 6 successfully automated server health monitoring using PowerShell and Task Scheduler. The automation provides consistent, timestamped health reports without requiring manual checks. This type of monitoring is essential for ongoing security auditing, troubleshooting, and operational visibility. The script can also be extended with alerts or email notifications for use in production SOC or IT environments.

---

7. Reflection

This lab demonstrated how PowerShell automation can simplify routine administrative tasks by producing predictable, repeatable system-health reports. By scheduling the script to run daily, the server can generate consistent logs that support both operations and compliance requirements. This type of automation provides a foundation for proactive monitoring and can be expanded to include alerting or integration with SIEM platforms.