Beschrijving en Resultaten van het Testen

Wat We Hebben Getest?

We hebben een PowerShell-script gemaakt dat het CPU-gebruik van de computer in de gaten houdt. Dit script moet:

- De hoeveelheid CPU-gebruik goed meten.
- Deze informatie opslaan in een CSV-bestand (een soort spreadsheet).
- De informatie duidelijk laten zien in de console met verschillende kleuren, afhankelijk van het CPU-gebruik.
- Proberen opnieuw te schrijven naar het bestand als het tijdelijk niet kan.

Hoe We Hebben Getest

- Functionele Tests: We hebben gekeken of het script goed werkt door het verschillende keren uit te voeren.
- Stresstests: We hebben het script getest tijdens drukke momenten om te zien hoe het omgaat met hoge belasting.
- Gebruiksvriendelijkheid: We hebben gecontroleerd of de kleuren in de console duidelijk zijn en of de informatie makkelijk te begrijpen is.

Wat We Hebben Gevonden

- CPU-meting: Het script registreerde het CPU-gebruik nauwkeurig, zelfs bij drukte.
- CSV-loggen: De gegevens werden goed opgeslagen in het CSV-bestand, ook als het bestand in gebruik was.
- Kleurcodering: De kleurcodering werkte goed: rood voor hoog gebruik , geel voor gemiddeld gebruik en groen is voor laag gebruik
- Retry-mechanisme: Het script probeerde opnieuw te schrijven naar het bestand zonder te crashen.

Beschrijving en Resultaten van het Testen

Get-WmiObject Win32_Processor

Wat het doet: Dit commando haalt informatie op over de processor (CPU) van de computer. Het vertelt ons bijvoorbeeld hoe druk de CPU is en hoeveel logische cores (processors) er zijn.

Measure-Object -Property LoadPercentage -Average

Wat het doet: Dit commando berekent de gemiddelde waarde van een eigenschap. Hier gebruiken we het om het gemiddelde CPU-gebruik te berekenen.

Select-Object -ExpandProperty Average

Wat het doet: Dit commando haalt specifieke informatie uit de gegevens die we hebben opgehaald. In ons geval pakken we de gemiddelde waarde van het CPU-gebruik.

Get-Date -Format "yyyy-MM-dd HH:mm:ss"

Wat het doet: Dit commando haalt de huidige datum en tijd op en laat het in een duidelijk formaat zien. Dit is belangrijk voor de logbestanden, zodat we weten wanneer de gegevens zijn verzameld.

$logEntry | Add-Content -Path $csvFilePath -Encoding UTF8

Wat het doet: Add-Content voegt nieuwe informatie toe aan een bestaand bestand. In dit geval schrijven we de loggegevens naar een CSV-bestand, dat lijkt op een spreadsheet. We zorgen ervoor dat de tekst goed wordt opgeslagen.

Write-Host "[$currentDateTime] Hoger CPU gebruik gedetecteerd: $cpuLoad% - Aantal gebruikte cores: $usedCPUs" -ForegroundColor Red

Wat het doet: Dit commando toont informatie in de console (de terminal). We gebruiken het om te laten zien wat het huidige CPU-gebruik is. Met -ForegroundColor kunnen we de tekst een kleur geven, zoals rood.

Start-Sleep -Seconds 1

Wat het doet: Dit commando pauzeert het script voor het aantal seconden dat je opgeeft (hier 1 seconde). Dit is handig om te voorkomen dat het script te vaak gegevens verzamelt.

while ($true) { ... }

Wat het doet: Dit is een oneindige loop die ervoor zorgt dat het script blijft draaien. Het blijft CPU-gebruik meten totdat je het script handmatig stopt.

Beschrijving en Resultaten van het Testen

POWERPOINT

Overdracht naar de Beheerafdeling

Wat We Hebben Gepresenteerd

Bij de overdracht naar de beheerafdeling hebben we een PowerPoint-presentatie gemaakt met de volgende punten:

Inleiding

Uitleg over waarom we dit project hebben gedaan.

Technische Uitleg

Hoe het script werkt en wat het doet.

Voorbeeld van de kleurcodering in de console.

Testresultaten

Wat we hebben getest en de resultaten hiervan.

Gebruik van het Script

Hoe de beheerafdeling het script kan gebruiken en instellen.

Ondersteuning

Waar ze meer informatie en hulp kunnen vinden.

Teamwerk

Alle teamleden hebben hun deel van de presentatie voorbereid, zodat alles duidelijk was. We zorgden ervoor dat de presentatie makkelijk te begrijpen was voor de beheerafdeling.

Beschrijving en Resultaten van het Testen

Conclusie

Het testen was geslaagd en we hebben het project goed overgedragen aan de beheerafdeling. We zijn beschikbaar voor vragen en hulp bij het gebruik van het script. De presentatie hielp om de werking en het gebruik van het systeem duidelijk te maken.

Beschrijving en Resultaten van het Testen

```
$smtpServer = "smtp.gmail.com"
$smtpPort = 587
$fromEmail = "XXXX"
$password = "XXXX"  # Gebruik een app-specifiek wachtwoord
$toEmail = "XXXX"

# Zet het wachtwoord veilig
$securePassword = ConvertTo-SecureString $password -AsPlainText -Force
$credential = New-Object System.Management.Automation.PSCredential($fromEmail, $securePassword)

# CSV-bestand pad
$entDateTime = Get-Date -Format "yyyy-MM-dd_HH-mm-ss"
$csvFilePath = "cpu_usage_log_$entDateTime.csv"

# Controleer of het CSV-bestand al bestaat, anders maak het aan met headers
if (-not (Test-Path $csvFilePath)) {
    "Timestamp,CPU_Percentage,Used_CPUs" | Out-File -FilePath $csvFilePath -Encoding UTF8
}

# Infinite loop om het CPU-gebruik te monitoren
$monitorDuration = 60 # 2 minuten
$startTime = Get-Date
$endTime = $startTime.AddSeconds($monitorDuration)
$highCpuCount = 0

while ($true) {
    # Verkrijg het huidige CPU-gebruik via WMI
    $cpuLoad = Get-WmiObject Win32_Processor | Measure-Object -Property LoadPercentage -Average | Select-Object -ExpandProperty Average
    $usedCPUs = (Get-WmiObject Win32_Processor).NumberOfLogicalProcessors
    # Huidige datum en tijd
    $currentDateTime = Get-Date -Format "yyyy-MM-dd HH:mm:ss"

    # Log de gegevens live in CSV met retry-mechanisme
    $logEntry = "{0},{1},{2}" -f $currentDateTime, $cpuLoad, $usedCPUs
    $maxRetries = 5
    $retryCount = 0
    $success = $false

    while (-not $success -and $retryCount -lt $maxRetries) {
        try {
            $logEntry | Add-Content -Path $csvFilePath -Encoding UTF8 -ErrorAction Stop
            $success = $true
        } catch {
            $retryCount++
            Write-Host "Kan niet schrijven naar bestand, poging $retryCount van $maxRetries. Wacht 1 seconde..."
            Start-Sleep -Seconds 1
        }
    }

    # Print de huidige waarden met datum en tijd
    if ($cpuLoad -gt 95) {
        $highCpuCount++
        Write-Host "[$currentDateTime] Hoge CPU-belasting: $cpuLoad% - Aantal gebruikte cores: $usedCPUs" -ForegroundColor Red
    } elseif ($cpuLoad -gt 50) {
        Write-Host "[$currentDateTime] Gemiddelde CPU-belasting: $cpuLoad% - Aantal gebruikte cores: $usedCPUs" -ForegroundColor Yellow
        $highCpuCount = 0  # Reset de teller als de CPU onder de 95% gaat
    } else {
        Write-Host "[$currentDateTime] Laag CPU gebruik: $cpuLoad% - Aantal gebruikte cores: $usedCPUs" -ForegroundColor Green
        $highCpuCount = 0  # Reset de teller als de CPU onder de 95% gaat
    }

    # Controleer of de CPU langer dan 6 seconden boven de 95% is
    if ($highCpuCount -ge 6) {
        Send-MailMessage -From $fromEmail -To $toEmail -Subject "Hoge CPU-gebruik Waarschuwing" -Body "Hoge CPU-belasting gedetecteerd: $cpuLoad%." -SmtpServer $smtpServer -Port $smtpPort -Credential $credential -UseSsl
        Write-Host "E-mail succesvol verzonden!"
        $highCpuCount = 0

    }

    # Wacht 1 seconde
    Start-Sleep 1

    # Controleer of de monitoringstijd voorbij is
    if ((Get-Date) -ge $endTime) {
        Write-Host "Monitoringstijd voorbij. Creëer grafiek..."

        # Genereer de grafiek
        $data = Import-Csv -Path $csvFilePath
        $timestamp = Get-Date -Format "yyyyMMdd_HHmmss"
        $chartFilePath = "cpu_usage_chart_$timestamp.xlsx"

        # Creëer de grafiek
        try {
            $data | Export-Excel -Path $chartFilePath -WorksheetName 'CPU Usage' -AutoSize | Out-Null

            $excel = Open-ExcelPackage -Path $chartFilePath
            $worksheet = $excel.Workbook.Worksheets['CPU Usage']

            if ($worksheet) {

                $chartName = "CPUUsageChart_$timestamp"
                $chart = $worksheet.Drawings.AddChart($chartName, 'Line')
                $chart.SetPosition(5, 0, 1, 0)  # Set position of the chart
                $chart.SetSize(600, 400)  # Set size of the chart


                $chart.Series.Add("B2:B$($data.Count + 1)", "A2:A$($data.Count + 1)")  # CPU_Percentage
                $chart.XAxis.Title.Text = "Timestamp"
                $chart.YAxis.Title.Text = "CPU Percentage"
                $chart.Title.Text = "CPU Usage Over Time"
                $chart.Legend.Remove()
            } else {
                Write-Host "Worksheet is null. Cannot add chart."
            }


            Close-ExcelPackage $excel | Out-Null
            Write-Host "Grafiek succesvol toegevoegd aan Excel-bestand."
        } catch {
            Write-Host "Fout bij het openen of bewerken van het Excel-bestand: $_"
        }

        # Reset monitoring tijd
        $startTime = Get-Date
        $endTime = $startTime.AddSeconds($monitorDuration)
        Write-Host "Monitoring herstart na grafiek creatie."
    }
}
```

Beschrijving en Resultaten van het Testen