**1. Process Check and Delete**

**Definition:**  
In PowerShell and PSAppDeployToolkit, process checking is used to verify if a specific application is running, and process deletion (termination) is used to close it before performing an action like installation or uninstallation.

**Example Code:**

If (Get-Process -Name "notepad" -ErrorAction SilentlyContinue) {

Write-Host "Notepad is running. Closing..."

Stop-Process -Name "notepad" -Force

}

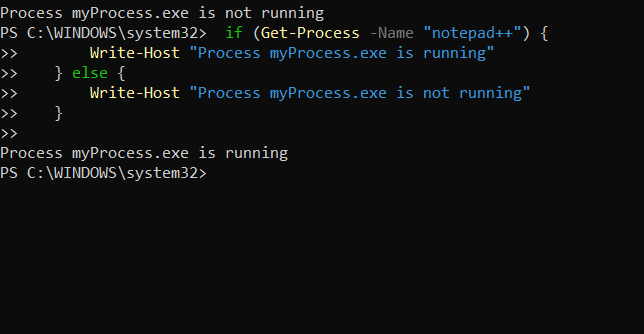
Else {

Write-Host "Notepad is not running."

}

**Use Cases:**

* Closing an app before updating it.
* Preventing file lock errors during installation.



**2. Loop-Based Scripting Operations**

**Definition:**  
Loops in PowerShell allow repetitive execution of code until a condition is met.

**Types:**

* for loop
* foreach loop
* while loop
* do...while loop

**Example Code:**

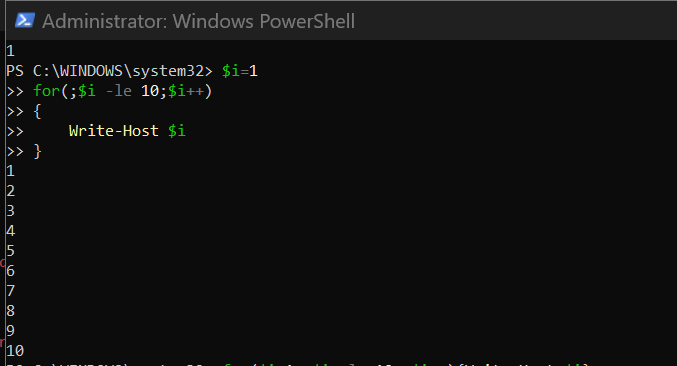
for ($i = 1; $i -le 5; $i++) {

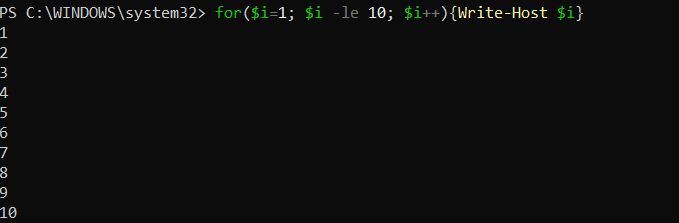
Write-Host "Count: $i"

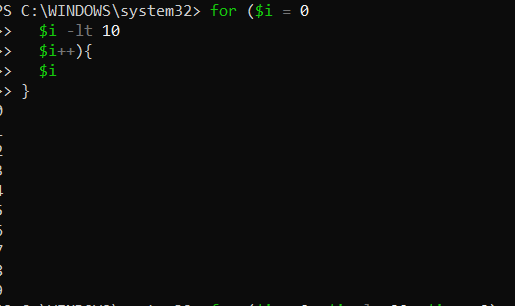
}

**Use Cases:**

* Running the same command for multiple files.
* Automating repetitive admin tasks.







**3. Set-IniValue**

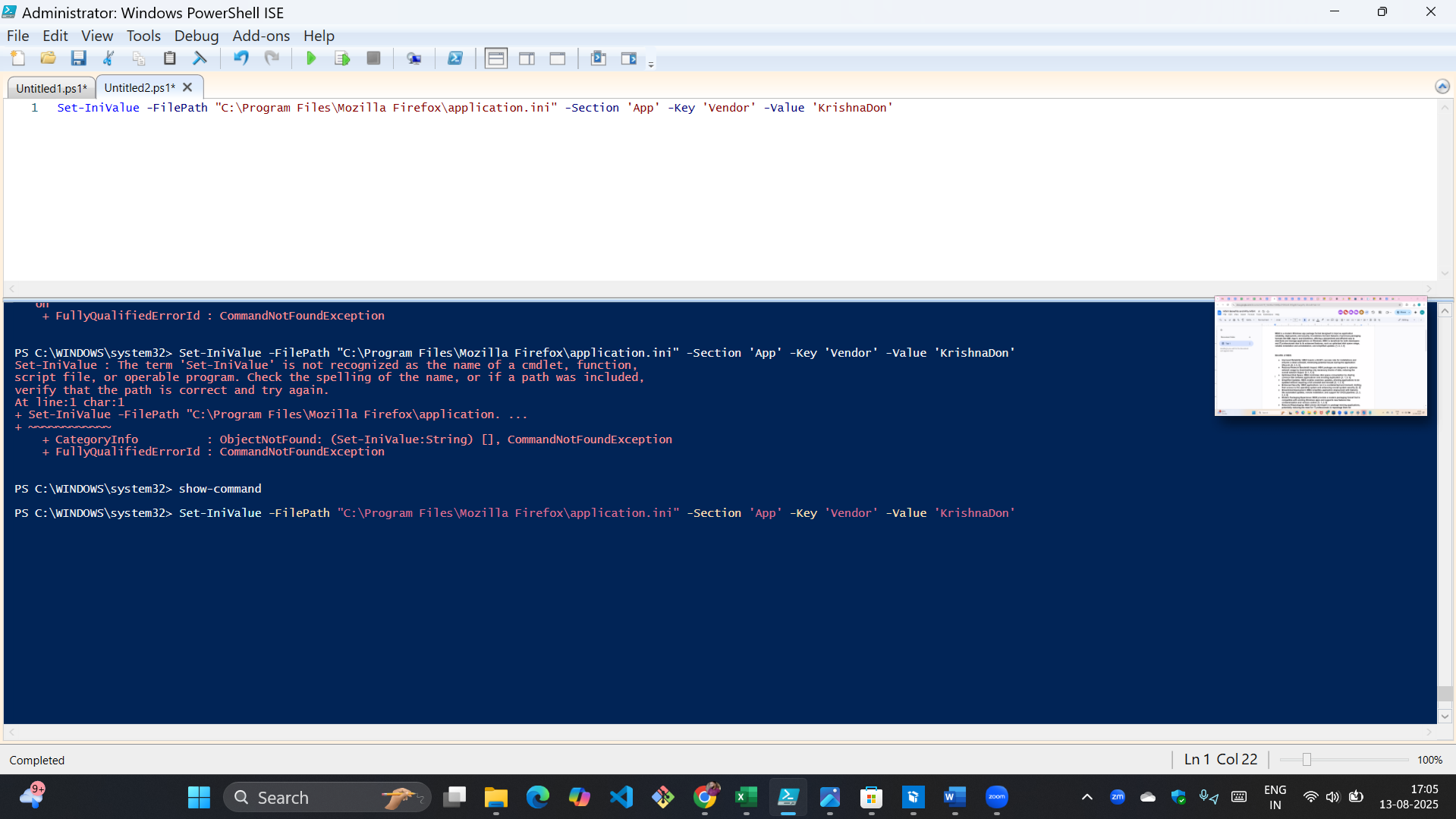
**Definition:**  
Set-IniValue is used to modify a key-value pair in an INI configuration file.  
PSAppDeployToolkit includes functions to read and write INI files for easy configuration management.

**Example Code:**

Set-IniValue -FilePath "$dirFiles\config.ini" -Section "General" -Key "Language" -Value "en"

**Use Cases:**

* Changing application settings before installation.
* Updating deployment parameters dynamically.



**4. Toolkit Logging Location and How It’s Configured**

**Default Location:**

C:\Windows\Logs\Software

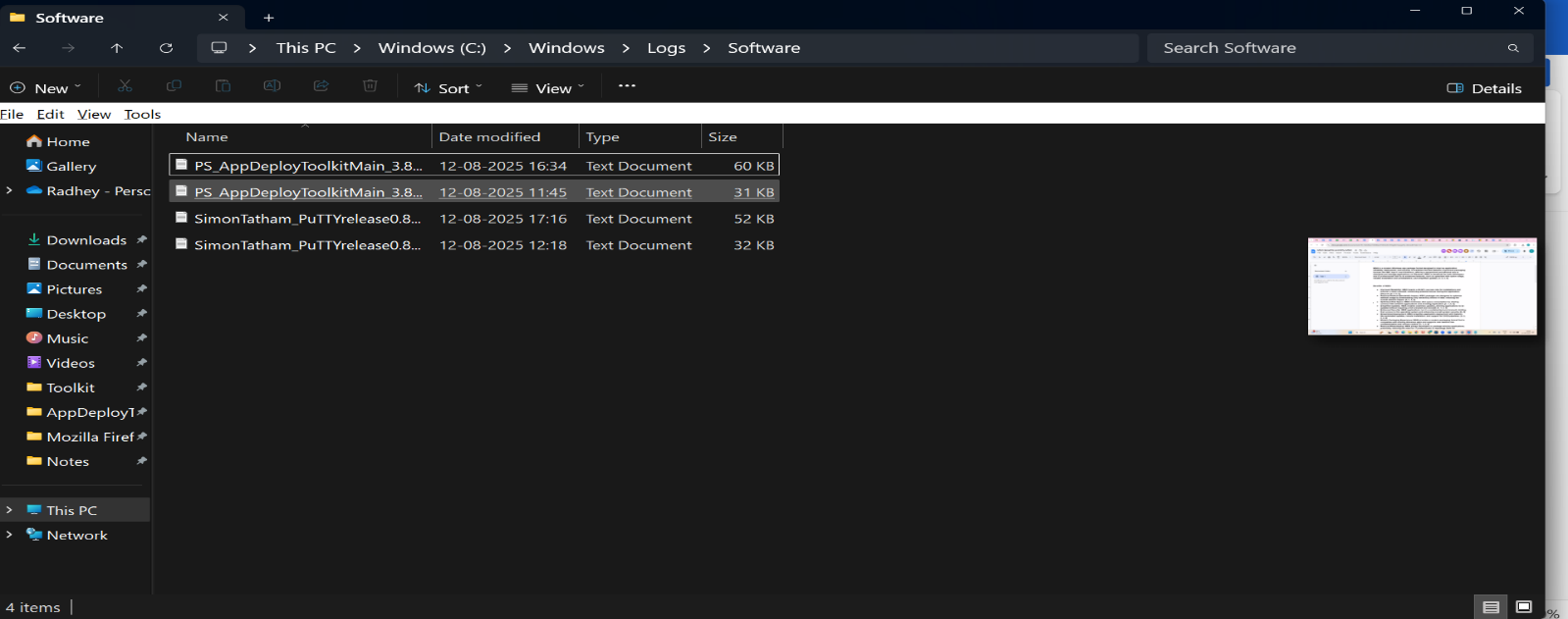
Each run of Deploy-Application.ps1 creates a detailed log file here.

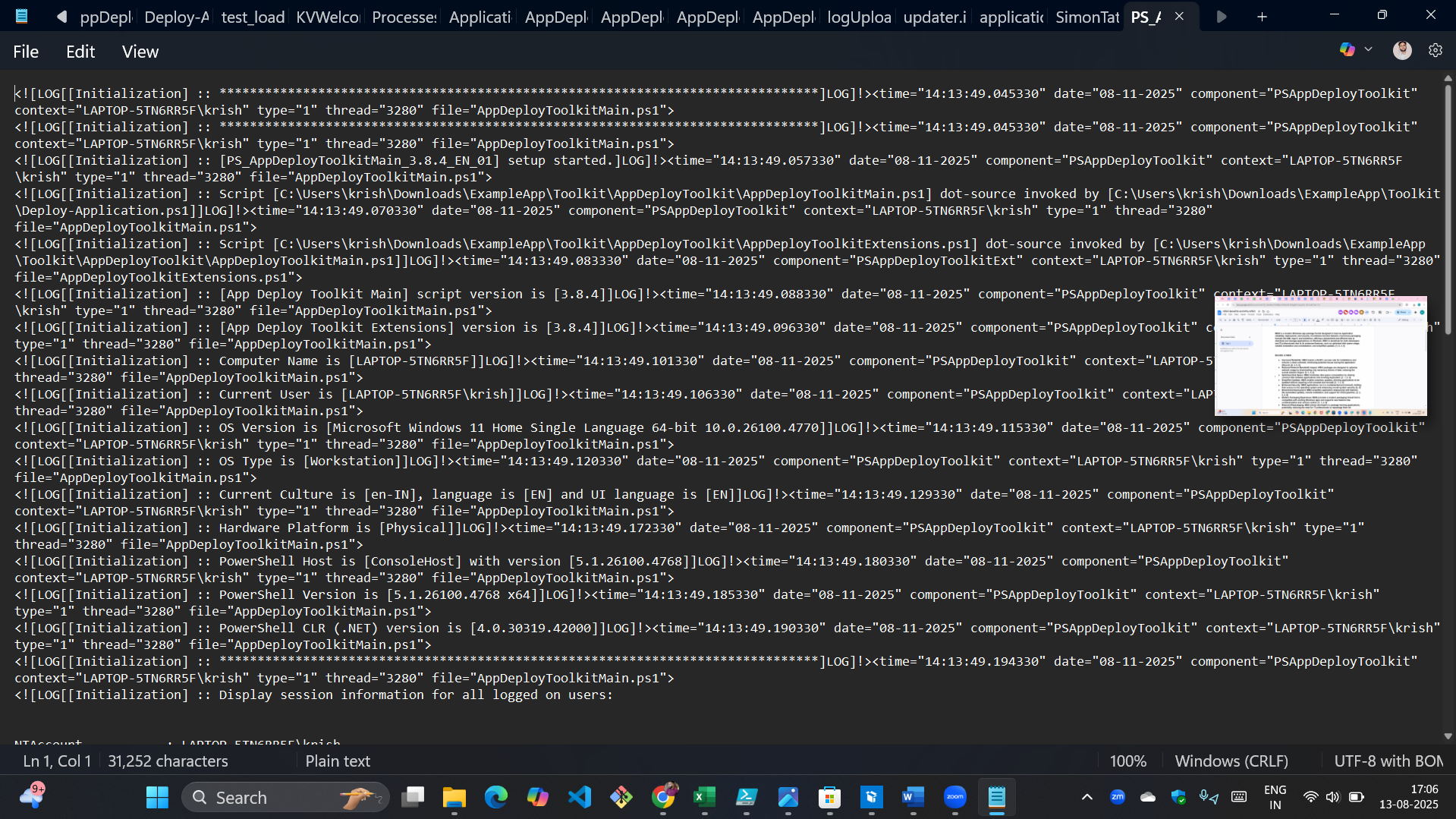
**Configured in:**

* Deploy-Application.ps1 (via $LogName variable)
* AppDeployToolkitMain.ps1 (default settings)

**Example:**

Write-Log "Installation started."





**5. MSI/MSP Logging**

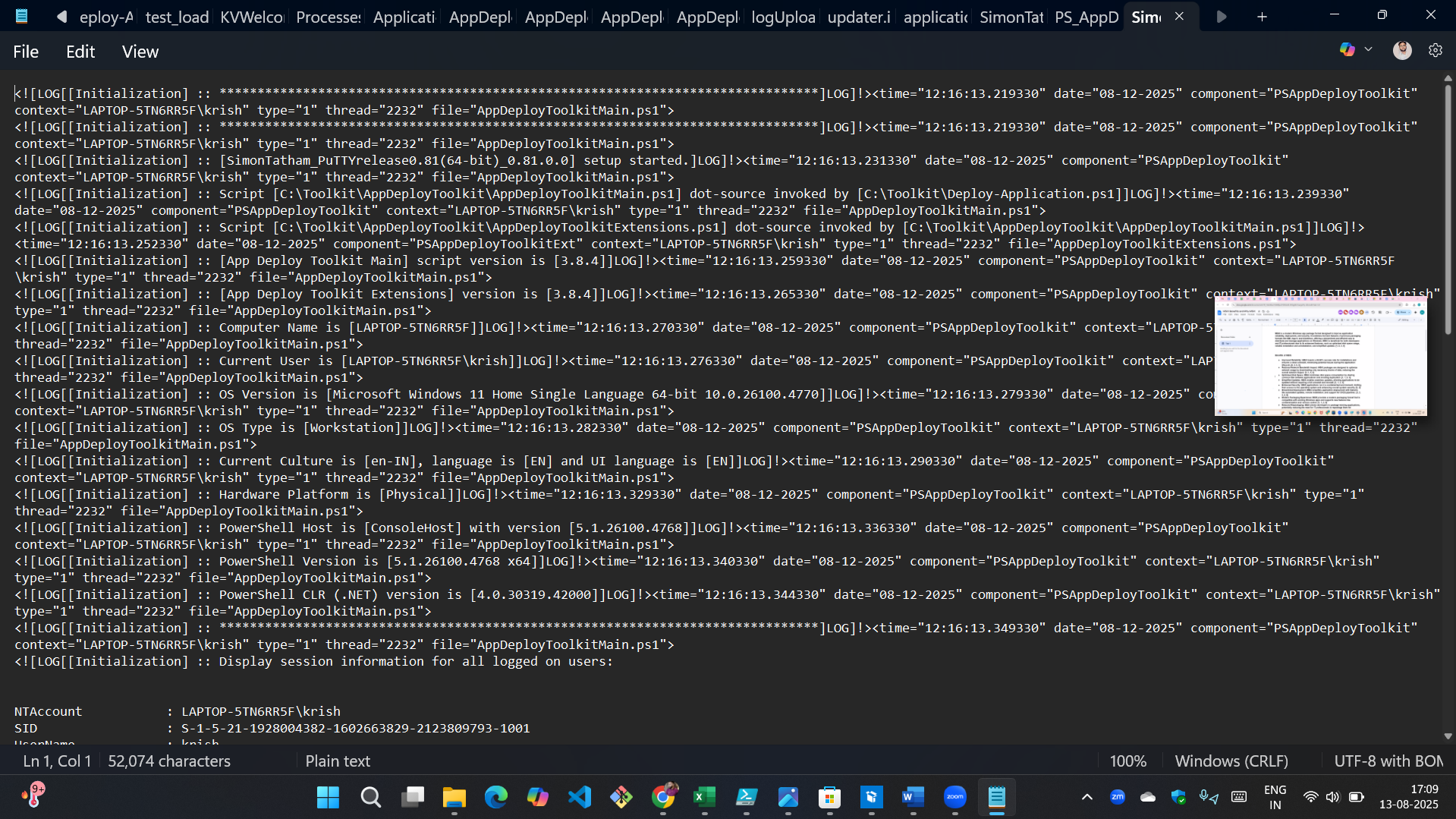
**Definition:**  
MSI and MSP installers support detailed logging to capture installation process output.

**PSADT Example:**

Execute-MSI -Action Install -Path "$dirFiles\MyApp.msi" -LogName "MyAppInstall.log"

**Standard PowerShell Example:**

Start-Process msiexec.exe -ArgumentList '/i MyApp.msi /qn /L\*v C:\Temp\msilog.txt' -Wait



**6. Individual Command Logging Within Script**

**Definition:**  
Using Write-Log to capture specific actions in the deployment process for better troubleshooting.

**Example:**

Execute-Process -Path "7zip.exe" -Parameters "/S"

Write-Log "7-Zip installation command executed."

**7. How to Catch Return Codes Within PowerShell**

**Definition:**  
Return codes indicate the success or failure of a command.  
In PowerShell, you can capture them with $LASTEXITCODE.

**Example:**

Execute-Process -Path "7zip.exe" -Parameters "/S"

Write-Host "Exit Code: $LASTEXITCODE"

If ($LASTEXITCODE -eq 0) {

Write-Host "Success"

}

Else {

Write-Host "Failed with code $LASTEXITCODE"

}

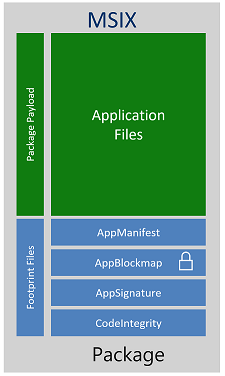
**8. MSIX Technology Fundamentals with Architecture**

**Definition:**  
MSIX is Microsoft’s modern packaging format that combines the best of MSI, AppX, and App-V.

**Key Points:**

* Single package format for all Windows 10/11 apps.
* Supports containerized deployment.
* Auto-updates built in.
* More secure (digital signing required).

**Architecture:**

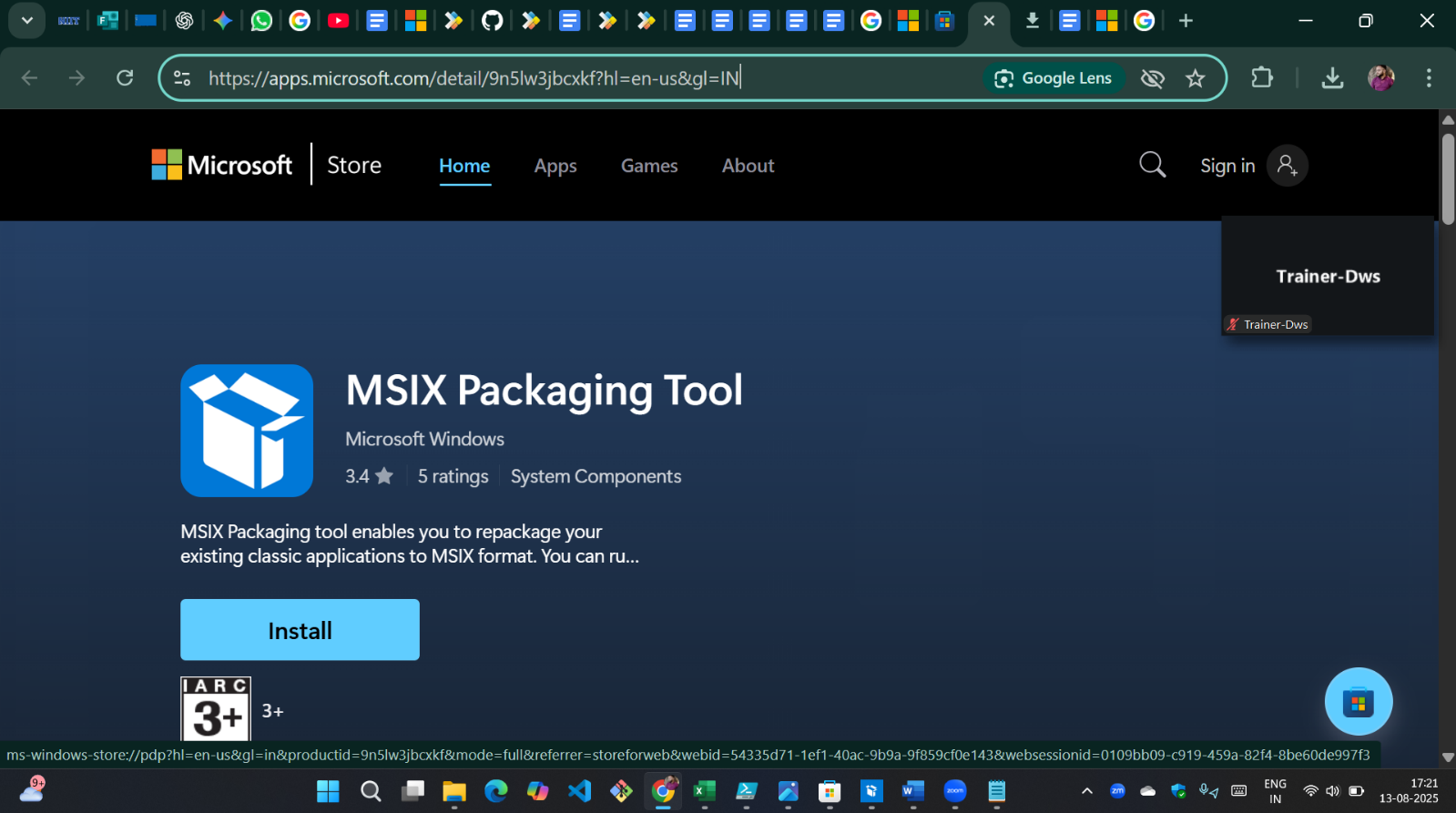
* **Package (.msix/.appx)** → Contains app binaries, manifest, and assets.
* **App Installer Service** → Handles deployment, updates, and removal.
* **Container Layer** → Isolates app from system for better security.

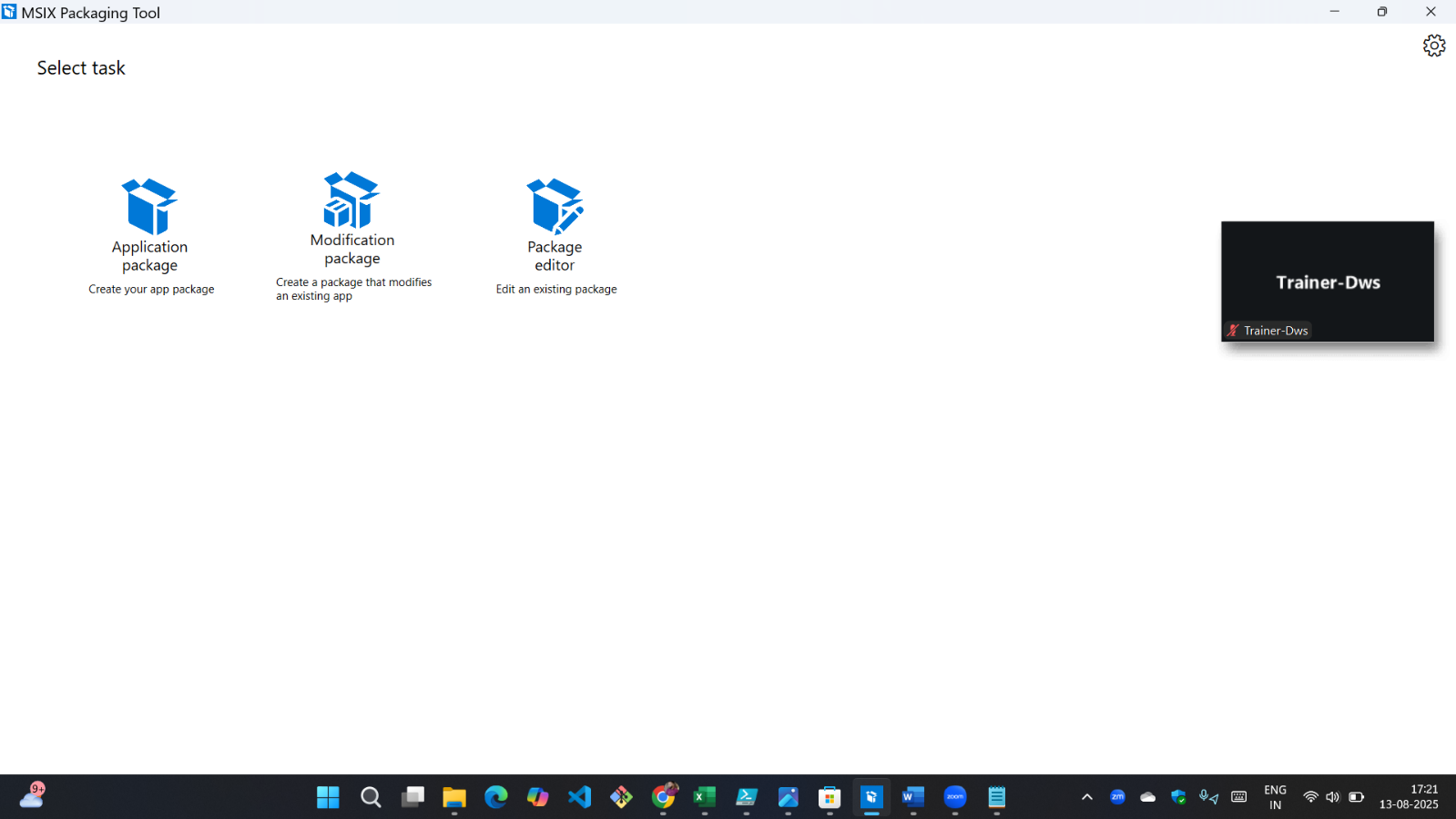
**Installation :-**

Go to this Link and install Msix Tool :-

<https://apps.microsoft.com/detail/9n5lw3jbcxkf?hl=en-us&gl=IN>

**Overview :-**

****

****