Assignment :11-08-2025

Name : Lalit Joshi ,Email : [Joshilalit2275@gmail.com](mailto:Joshilalit2275@gmail.com)

Topic 1: PSADT Folder Structure and Its Importance

he **PowerShell App Deployment Toolkit (PSADT)** uses a well-organized folder hierarchy to manage deployment packages and scripts efficiently. This structured approach helps in easy management, customization, and reuse of deployment components, making the application deployment process more streamlined and less error-prone.

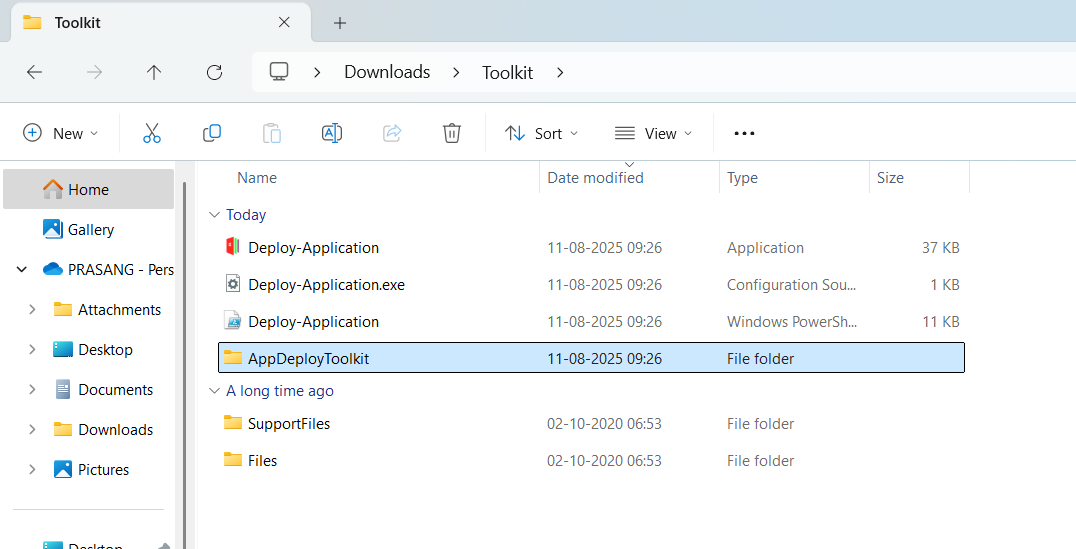
**Key Folders:**

* **Toolkit/**: Contains core PSADT module files including scripts, manifests, and language resources; essentially the engine of PSADT.
* **Examples/**: Holds example deployment scripts serving as templates for new deployments.
* **Files/**: Stores the application installation files such as .msi, .exe, or .appx.
* **SupportFiles/**: Contains supporting files like configuration files, certificates, or helper scripts required during deployment.
* **Deploy-Application.ps1**: The main PowerShell script orchestrating the deployment process, customized per application.
* **Deploy-Application.exe**: A wrapper to run the PowerShell script with the correct execution policy.

**Additional Highlights:**

* PSADT supports customization for branding (banners, icons), UI, and scripts.
* Version control (e.g., Git) is recommended to track changes and maintain deployment scripts.
* The modular and flexible structure introduced in PSADT v4 makes managing and customizing deployments easier.

Understanding and using this folder structure effectively allows IT professionals to create, deploy, and manage application deployments with greater ease and consistency.



Topic 2:psadt .exe application deployment process

When you download PSADT, you’ll get a folder with:

Deploy-Application.ps1 ← Main script you edit

Deploy-Application.exe ← Wrapper to run the .ps1

Toolkit\ ← PSADT core functions

Files\ ← Place for your application installers

AppDeployToolkitConfig.xml

AppDeployToolkitMain.ps1

For deploying an .exe app, you’ll mainly work with:

* \**Files\** → put your .exe installer here.
* **Deploy-Application.ps1** → edit this to install/uninstall.

**2. Place Your .exe Installer**

Example:

CopyEdit

.\Files\7z2501-x64.exe

This keeps things organized and lets PSADT find the installer easily.

**3. Edit Deploy-Application.ps1**

Inside the script, find the **##\*============================[ INSTALLATION ]============================\*##** section and add:

powershell

CopyEdit

##\*============================[ INSTALLATION ]============================\*##

# Install 7-Zip silently

Show-InstallationWelcome -CloseApps '7zFM' -AllowDefer

Show-InstallationProgress -StatusMessage 'Installing 7-Zip, please wait...'

Execute-Process -Path "$dirFiles\7z2501-x64.exe" -Parameters "/S"

Show-InstallationPrompt -Message 'Installation complete!' -ButtonRightText 'OK'

For uninstall, in the **##\*============================[ UNINSTALLATION ]============================\*##** section:

powershell

CopyEdit

##\*============================[ UNINSTALLATION ]============================\*##

# Example uninstall (if your exe supports /uninstall or /S)

Execute-Process -Path "$dirFiles\7z2501-x64.exe" -Parameters "/S" # or actual uninstall command

**4. Run the Deployment**

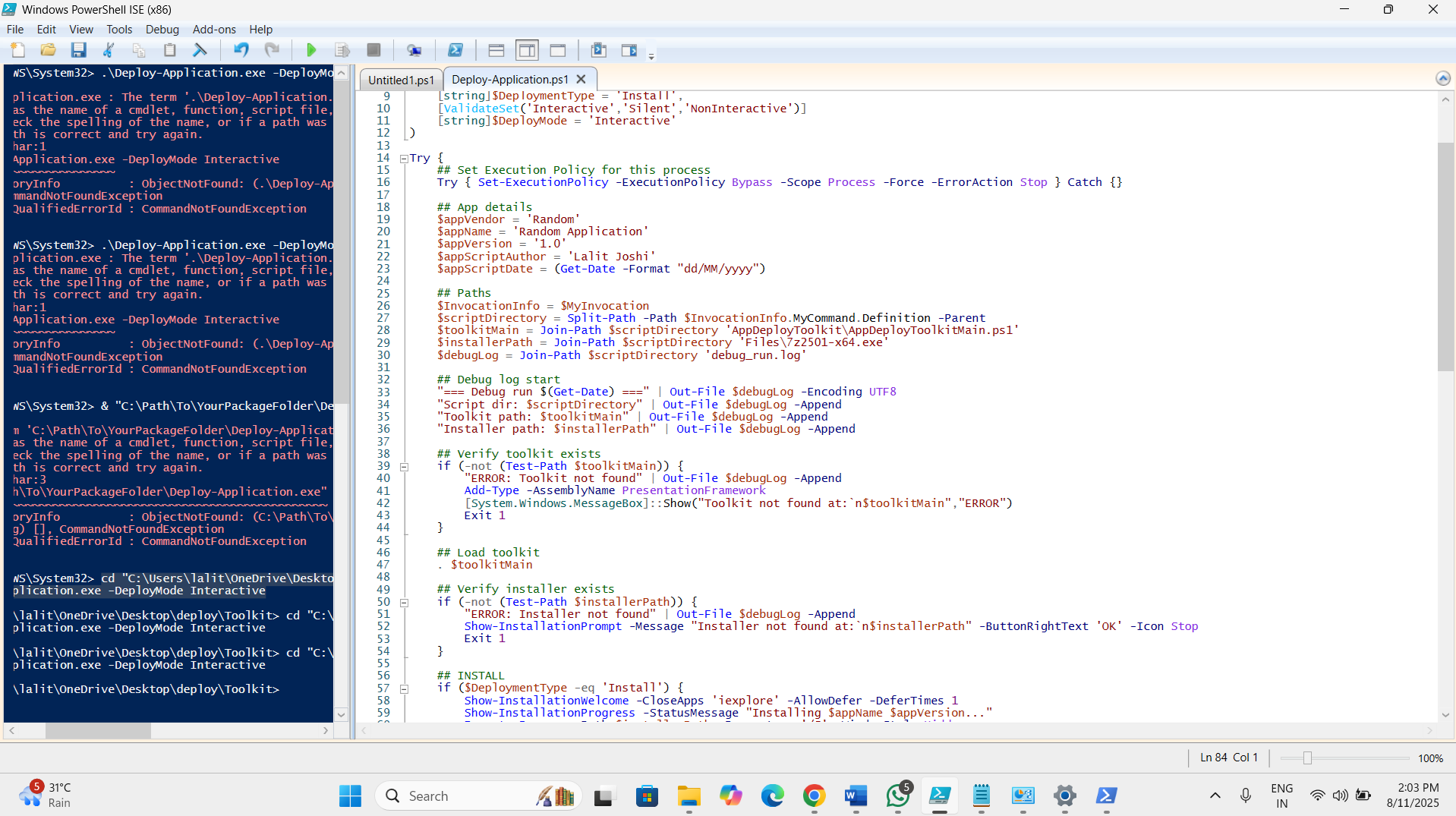
You have **two ways**:

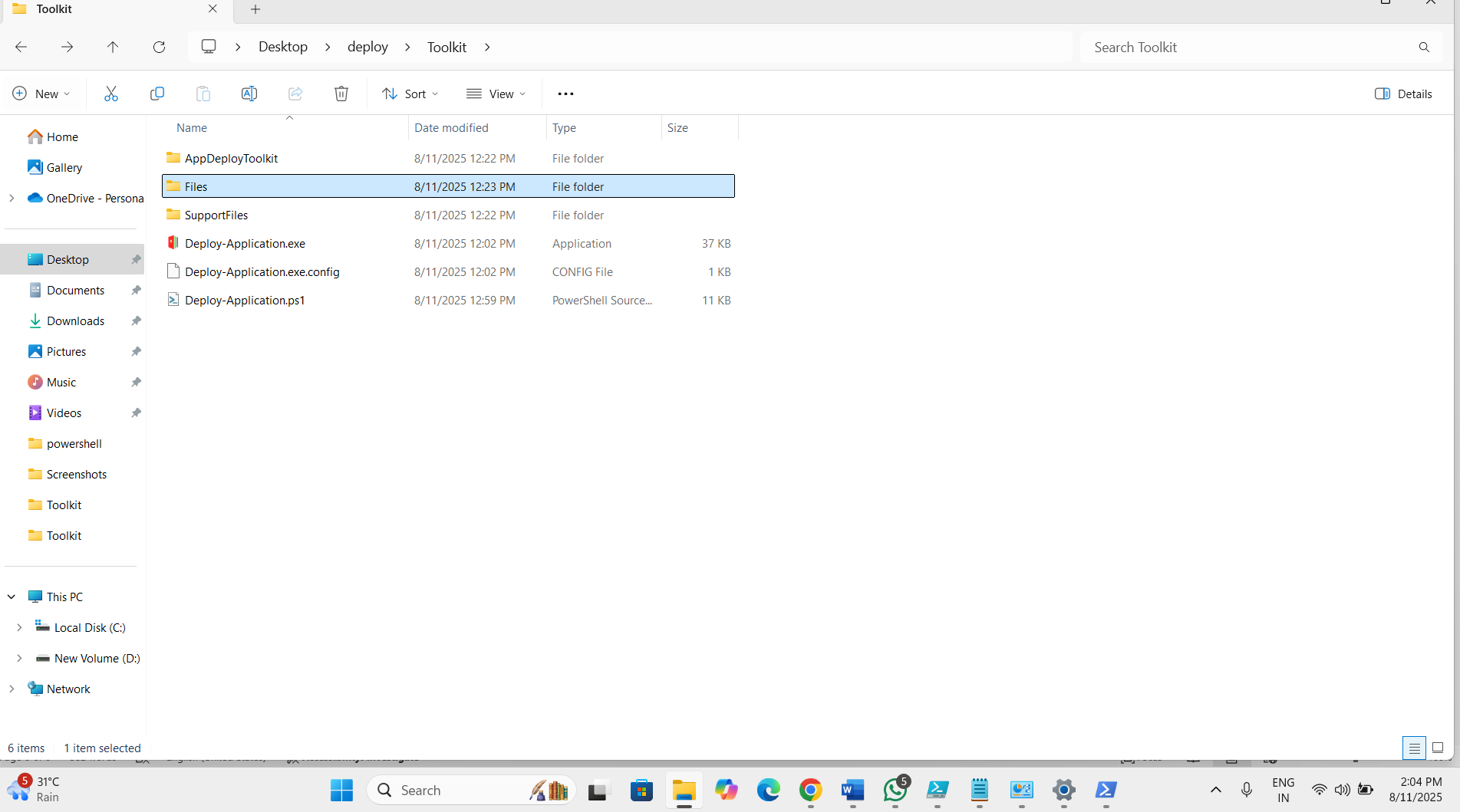
* **PowerShell directly:**

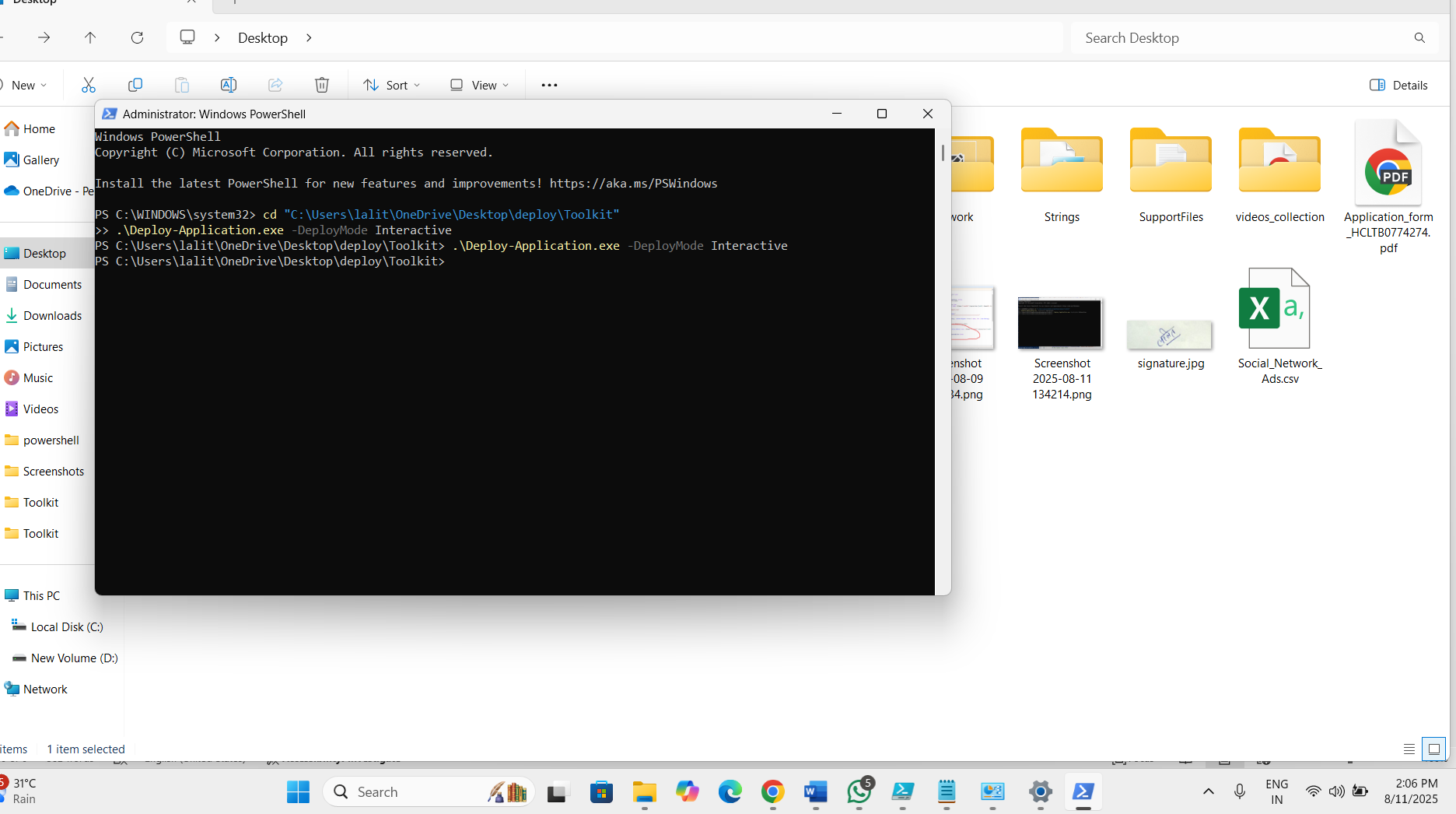
powershell

CopyEdit

.\Deploy-Application.ps1 -DeploymentType Install







Topic 3: PSAppDeployToolkit

 **What it is:**  
A framework that simplifies enterprise app deployments by providing ready-to-use functions, UI elements, and template scripts.

 **Key Template Script – Deploy-Application.ps1:**

* Used for **installing** or **uninstalling** apps.
* Located at C:\AdminStudio Shared\PowerShellTemplate.
* Customizable in PowerShell ISE.
* "Install" steps are split into **Pre-Install**, **Install**, and **Post-Install** phases.
* Relies on AppDeployToolkitMain.ps1 for core logic.

 **Creating Deployments:**

* **v3 template:**

powershell

CopyEdit

New-ADTTemplate -Destination C:\Temp\MyAppDeployment -Name "MyOldAppDeployment" -Version 3

* **v4 template:**

powershell

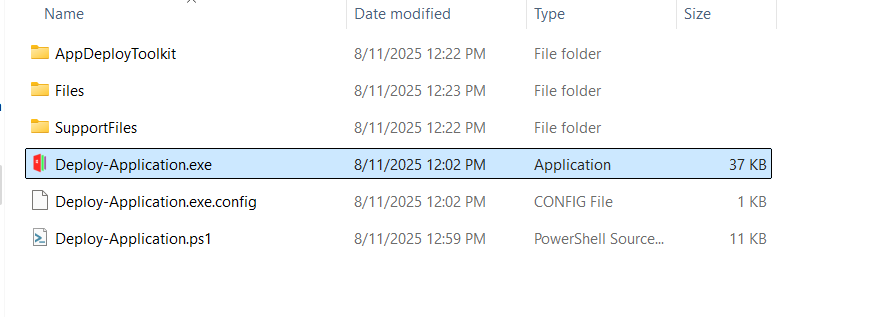
CopyEdit

New-ADTTemplate -Destination C:\Temp\MyAppDeployment -Name "MyAppDeployment"

 **Additional Resources:**  
Templates can also be downloaded from the **PSAppDeployToolkit GitHub releases** page (v3 and v4 zip files).

 **Customization:**

* You can modify parameters, add commands, and tweak the UI.
* Acts as a base for building more complex deployment scenarios.



Topic 4:PSAppDeployToolkit Configuration

 **What is it?**  
PSAppDeployToolkit is a scripting framework that makes application deployment easier by allowing customization of the deployment process, UI, logging, and more.

 **Configuration File:**  
The main configuration is done by editing the **AppDeployToolkitConfig.xml** file located in the Toolkit\AppDeployToolkit folder.

 **Configuration Steps:**

1. **Download and Extract** the PSAppDeployToolkit zip file.
2. **Locate** the AppDeployToolkitConfig.xml file.
3. **Edit** the XML file to customize:
   * Toolkit options (admin rights, temp paths, logging)
   * UI elements (banner, logos, icons)
   * MSI install options (log paths, parameters)
   * General UI settings (balloon notifications, timeouts, exit codes)
   * UI messages and languages
4. **Save and apply** changes, which then affect all deployment scripts using the toolkit.

 **Benefits:**

* **Centralized Configuration:** One place to configure for all scripts.
* **Consistency:** Uniform UI and deployment behavior.
* **Customization:** Tailored deployment experience.
* **Simplified Scripting:** Less repetitive coding needed per deployment.

Topic 5: MSI/MSP Logging in psadt

* **Purpose:**  
  PSADT provides built-in functions (Execute-MSI and Execute-MSP) to handle MSI (Windows Installer) and MSP (patch) deployments with logging, error handling, and user interaction.
* **Key Logging Parameters:**
  + **-LogName**: Sets the name of the log file (adds .log if missing).
  + **-LogPath**: Defines where the log file will be saved (defaults to PSADT’s standard log directory if not specified).
  + **-LogVerbosity**: Controls detail level of logging; options include Verbose (detailed), Informational, and Error (only errors).
* **Usage Examples:**
  + **MSI Install:**

powershell

CopyEdit

Execute-MSI -Action Install -Path "C:\MyApps\MyPackage.msi" -LogName "MyPackageInstallLog" -LogVerbosity Verbose

* + **MSP Patch:**

powershell

CopyEdit

Execute-MSI -Action Patch -Path "C:\MyApps\MyPatch.msp" -LogName "MyPatchLog"

* + **MSI Uninstall:**

powershell

CopyEdit

Execute-MSI -Action Uninstall -Path "[PRODUCTCODE]" -LogName "MyPackageUninstallLog"

* **Customizing Log Location:**
  + Set the variable $configToolkitLogDir in the AppDeployToolkitConfig.xml file to specify a custom folder for logs.
  + Alternatively, define $configToolkitLogDir early in Deploy-Application.ps1 or modify AppDeployToolkitMain.ps1.
* **Important Considerations:**
  + Default logs are saved under the PSADT working directory, but setting a dedicated or network folder is recommended for easier access.
  + Use -LogVerbosity to adjust log detail based on needs.
  + Error handling can be controlled with -ContinueOnError to either continue or stop deployment on errors.
  + Logs are essential for troubleshooting MSI/MSP deployment issues effectively.