**Today Assignment – Date: August 11,2025**

**Name: 34733-Prasang Kumar Singh**

**Gmail:** [**thakurprasang23@gmail.com**](mailto:thakurprasang23@gmail.com)

**Topic :- The importance of PowerShell App**

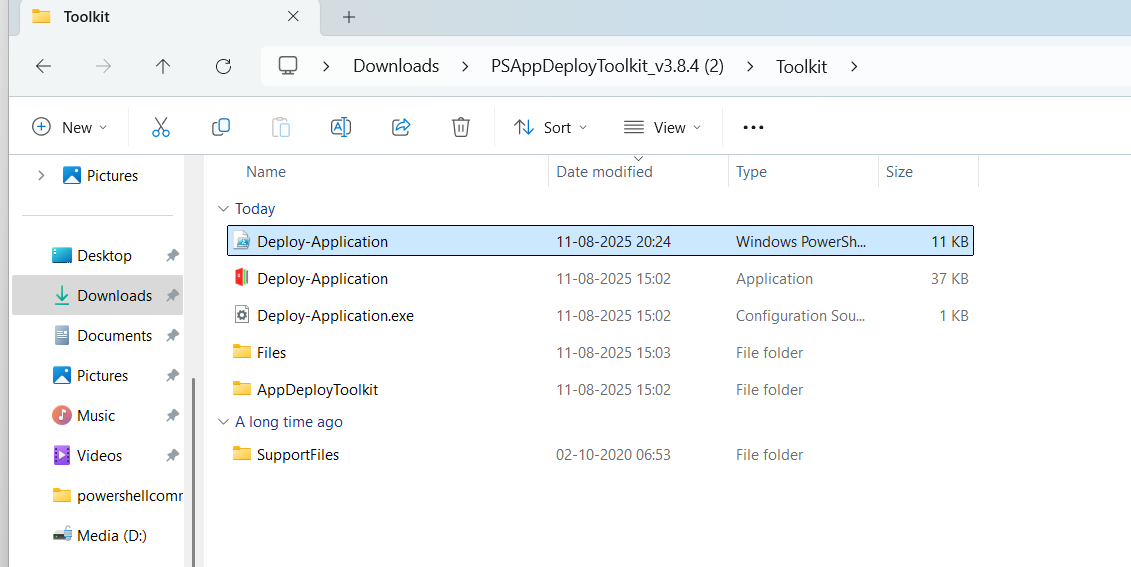
The PowerShell App Deployment Toolkit (PSADT) uses a clear and logical folder structure to organize deployment packages and scripts. This layout simplifies management, customization, and reusability of deployment components, making the application deployment process more efficient and reducing the chances of errors.

Main Folders and Files:

* Toolkit/ – Contains the core PSADT engine, including essential scripts, module files, manifests, and language resources that make the toolkit work.
* Examples/ – Provides sample deployment scripts that can be used as starting points or templates for new projects.
* Files/ – Stores the actual application installers, such as .msi, .exe, or .appx files.
* SupportFiles/ – Holds additional resources required during deployment, such as configuration files, security certificates, or helper scripts.
* Deploy-Application.ps1 – The primary PowerShell script that controls the deployment sequence and is tailored for each specific application.
* Deploy-Application.exe – An executable wrapper that launches the PowerShell script with the correct execution policy applied.

Additional Points:

* PSADT allows customization of branding elements, such as banners, logos, and user interface components.
* Using version control tools like Git is recommended for tracking updates and maintaining deployment scripts.
* Starting from PSADT v4, the modular and flexible framework makes it simpler to manage and adapt deployments.



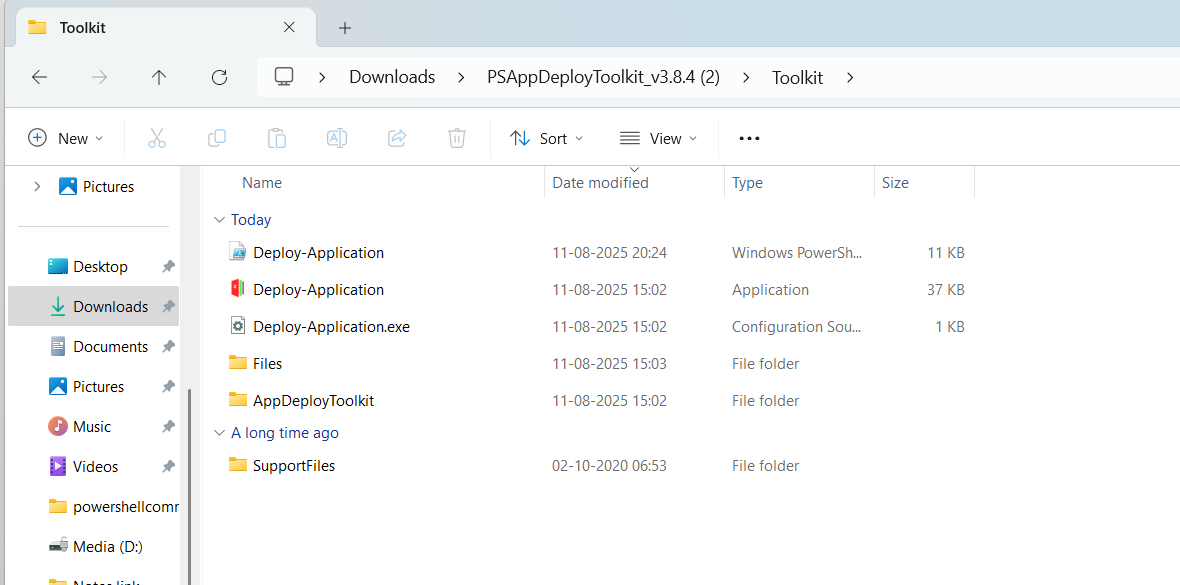
When you download the PSADT package, it comes with the following structure:

* Deploy-Application.ps1: This is the main script that you customize according to your deployment needs.
* Deploy-Application.exe: This executable acts as a launcher that runs the PowerShell script.
* Toolkit: Contains the core functions of the PSADT framework.
* Files: A folder designated for storing the application installers you want to deploy.
* AppDeployToolkitConfig.xml: The configuration file for the toolkit.
* AppDeployToolkitMain.ps1: The primary PowerShell script that supports deployment operations.

For deploying an application with an .exe installer, your primary focus will be on these two parts:

* The Files folder, where you place your .exe installation package.
* The Deploy-Application.ps1 script, where you specify the logic to install or uninstall the application.

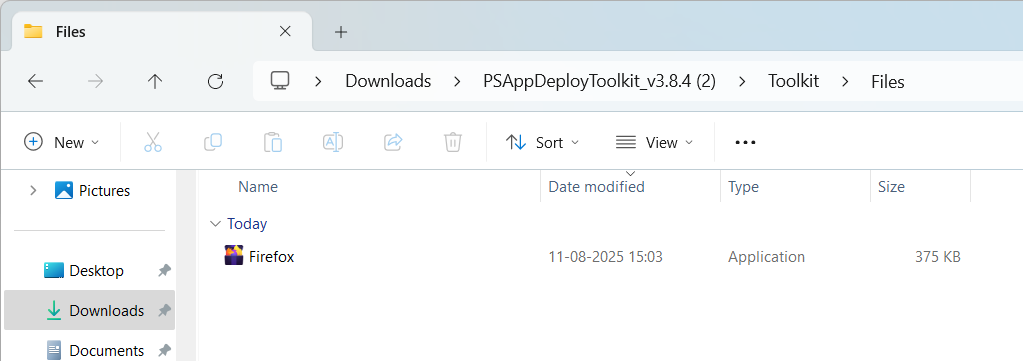
This setup allows you to manage your application deployment by placing the installer in the appropriate folder and customizing the main deployment script to control how the installation or removal process operates.



**2. Place Your .exe Installer**

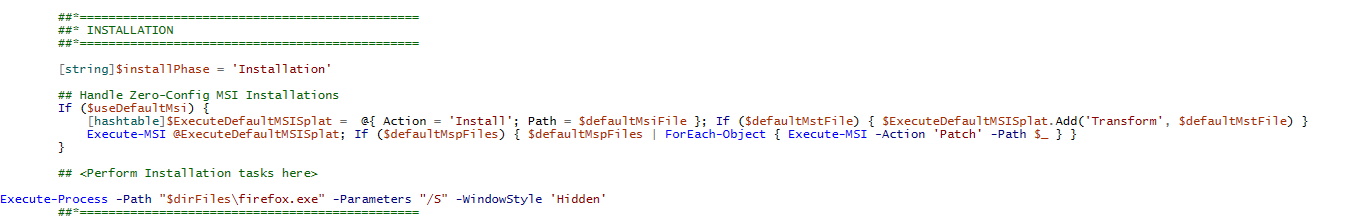
Example:

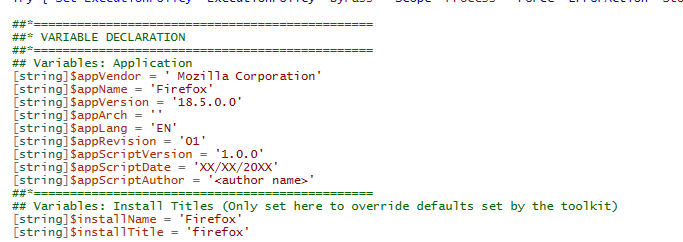
C:\Users\thaku\Downloads\PSAppDeployToolkit\_v3.8.4 (2)\Toolkit\Files



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

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

****



Inside the Deploy-Application.ps1 script, locate the section marked:

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

and add the following commands to silently install Firefox:

**# Install Firefox silently**

Show-InstallationWelcome -CloseApps '' -AllowDefer

Show-InstallationProgress -StatusMessage 'Installing Firefox, please wait...'

Execute-Process -Path "$dirFiles\firefox.exe" -Parameters "/S"

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

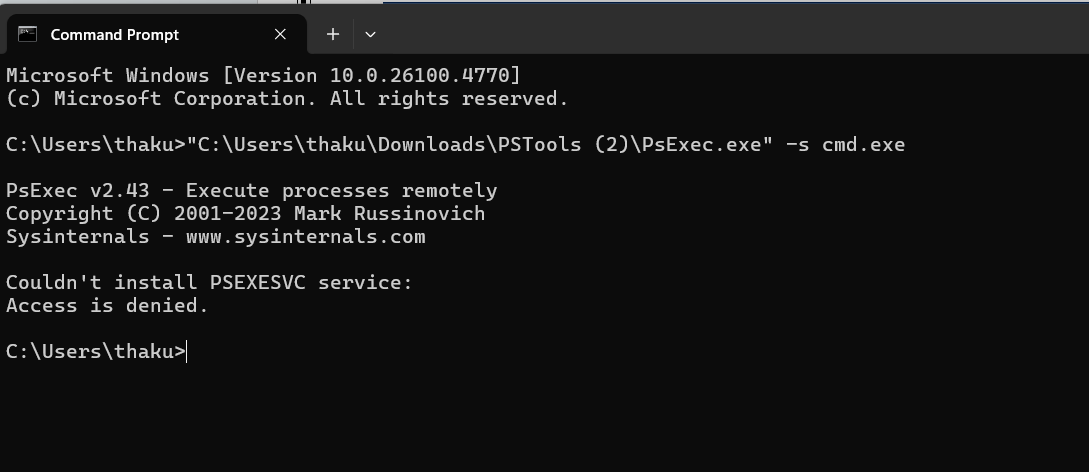
For the uninstall process, go to the section marked:

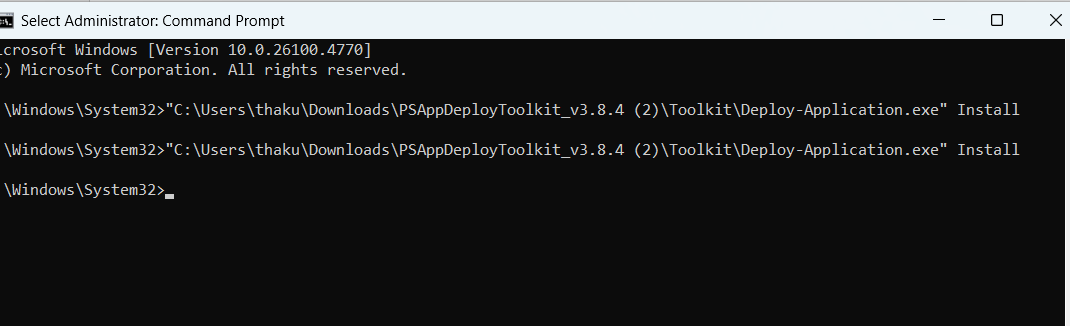
**##*============================[ UNINSTALLATION ]============================*##**

and add this command assuming your Firefox installer supports silent uninstall parameters like /S:

**6. Run the Deployment in System Context**

When deploying applications, they often need to run in **SYSTEM context** so they install for all users and have full permissions. Sometimes, installers behave differently in SYSTEM context compared to running as a normal user.  
Before deploying, I test the installation locally on a VM by running it in SYSTEM context using **PsExec**:





**Topic 3: PSAppDeployToolkit Overview**

PSAppDeployToolkit (PSADT) is a PowerShell framework that helps in deploying applications.  
It includes ready‑made functions, user prompts, and a standard script template.

Main Script – Deploy-Application.ps1

* Used for installing and uninstalling applications.
* The script is divided into Pre‑Install, Install, and Post‑Install phases.
* Works with AppDeployToolkitMain.ps1 for core operations.

Creating a Deployment Template

* **For version 3:**

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

* **For version 4:**

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

Templates can also be downloaded from the GitHub releases page and customised as needed.

**Topic 4: Toolkit Configuration**

The toolkit is configured using the AppDeployToolkitConfig.xml file located in the Toolkit\AppDeployToolkit folder.

Configuration Steps

1. Download and extract PSADT.
2. Open AppDeployToolkitConfig.xml.
3. Adjust settings for:
   * Admin rights, temp paths, log folder
   * UI elements like banners or icons
   * MSI defaults such as parameters and log file location
   * Notifications, timeouts, and exit codes
4. Save the file to apply changes across all deployments.

Benefits

* Central control of all deployments
* Consistent UI and behaviour
* Less repetitive scripting

**Topic 5: MSI/MSP Logging in PSADT**

PSADT provides Execute-MSI and Execute-MSP commands to install, uninstall, or patch MSI files with logging and error handling.

Logging Parameters

* **-LogName** → Name of the log file.
* **-LogPath** → Location to store the log.
* **-LogVerbosity**→ Level of detail (Verbose, Informational, Error).

Examples

* **Install MSI:**

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

* **Patch MSP:**

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

* Uninstall MSI:

Execute-MSI -Action Uninstall -Path "{PRODUCT-CODE}" -LogName "UninstallLog"

**Custom Log Location**

* Change $configToolkitLogDir in AppDeployToolkitConfig.xml
* Can also be set inside Deploy-Application.ps1 or AppDeployToolkitMain.ps1