# PowerShell App Deployment Toolkit (PSADT)

### 1. What is PSADT?

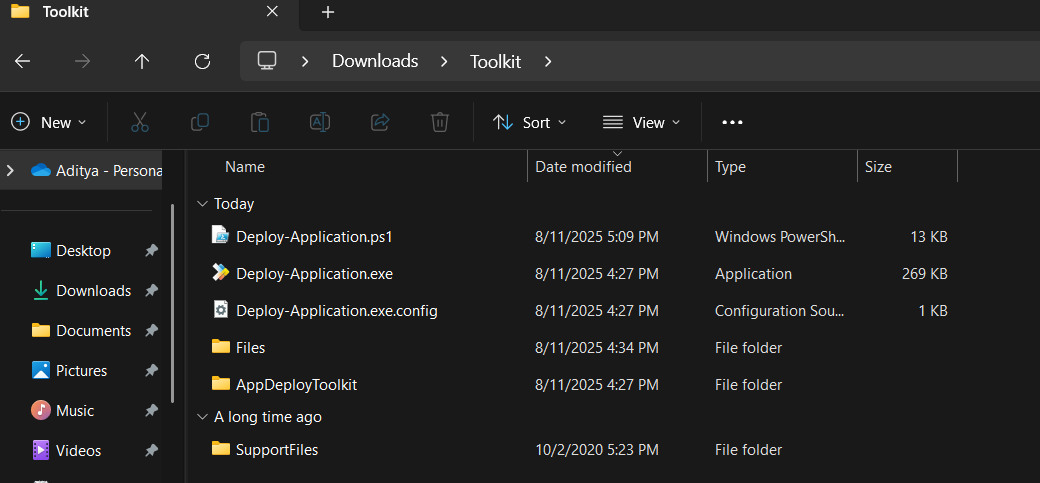
* PowerShell App Deployment Toolkit (PSADT) is a toolkit designed to simplify application deployment in corporate or large-scale environments.
* It uses PowerShell scripts to automate install, update, and uninstall processes consistently.
* Built-in functions handle common tasks like logging activity, prompting users, closing conflicting apps, and running installers silently without user interaction.
* This framework reduces errors and saves time compared to writing deployment scripts from scratch.

### 2. Tool Folder Structure

When you download PSADT, the folder structure includes:

* **AppDeployToolkit:** Core scripts and function definitions powering the toolkit.
* **Files:** Your actual application installers (MSI, EXE, MSP, etc.) and other files needed during deployment.
* **SupportFiles:** Additional helper files like icons, config templates, or custom resources.
* **Deploy-Application.ps1:** The key script you edit to control what happens during deployment (install/uninstall logic).
* **Deploy-Application.exe / .cmd:** Executables that launch the deployment script with specific parameters.

Understanding this layout helps you organize deployment assets cleanly.



### 3. Configuration Basics

* Most customization happens inside Deploy-Application.ps1. Here you define which actions occur during installation or removal, including any prep or cleanup steps.
* You can tweak AppDeployToolkitConfig.xml to adjust global settings like where logs save, how progress windows behave, or timeout durations.
* This separation makes it easy to maintain consistent deployments across multiple applications.

### 4. Template Script Overview

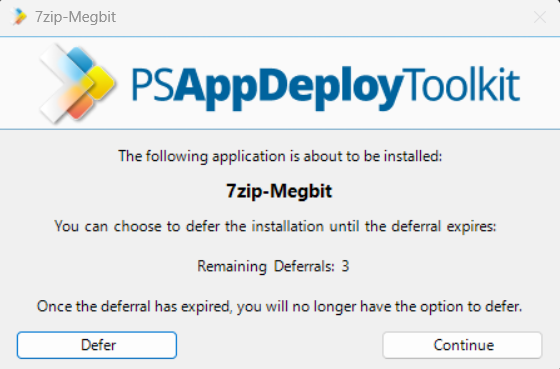
* The main deployment script runs in phases:
* **Pre-Installation:** Close apps, show messages, prep system.
* **Installation:** Run your MSI, EXE, or MSP installers silently.
* **Post-Installation:** Cleanup and notify users.
* **Uninstallation:** Steps for safely removing software.

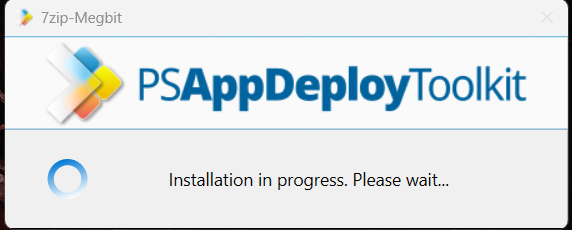
**Usage example:**

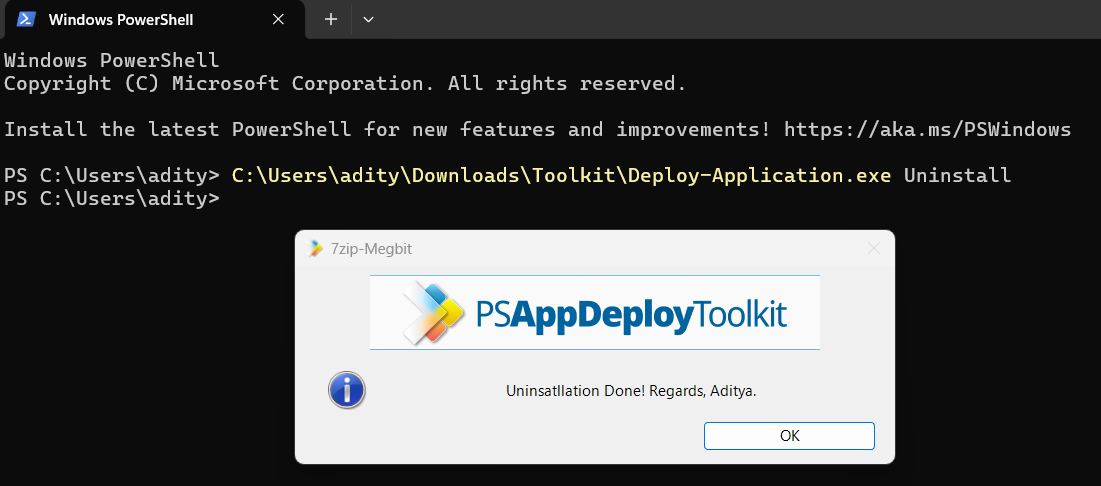
* To install: Deploy-Application.exe Install
* To uninstall: Deploy-Application.exe Uninstall

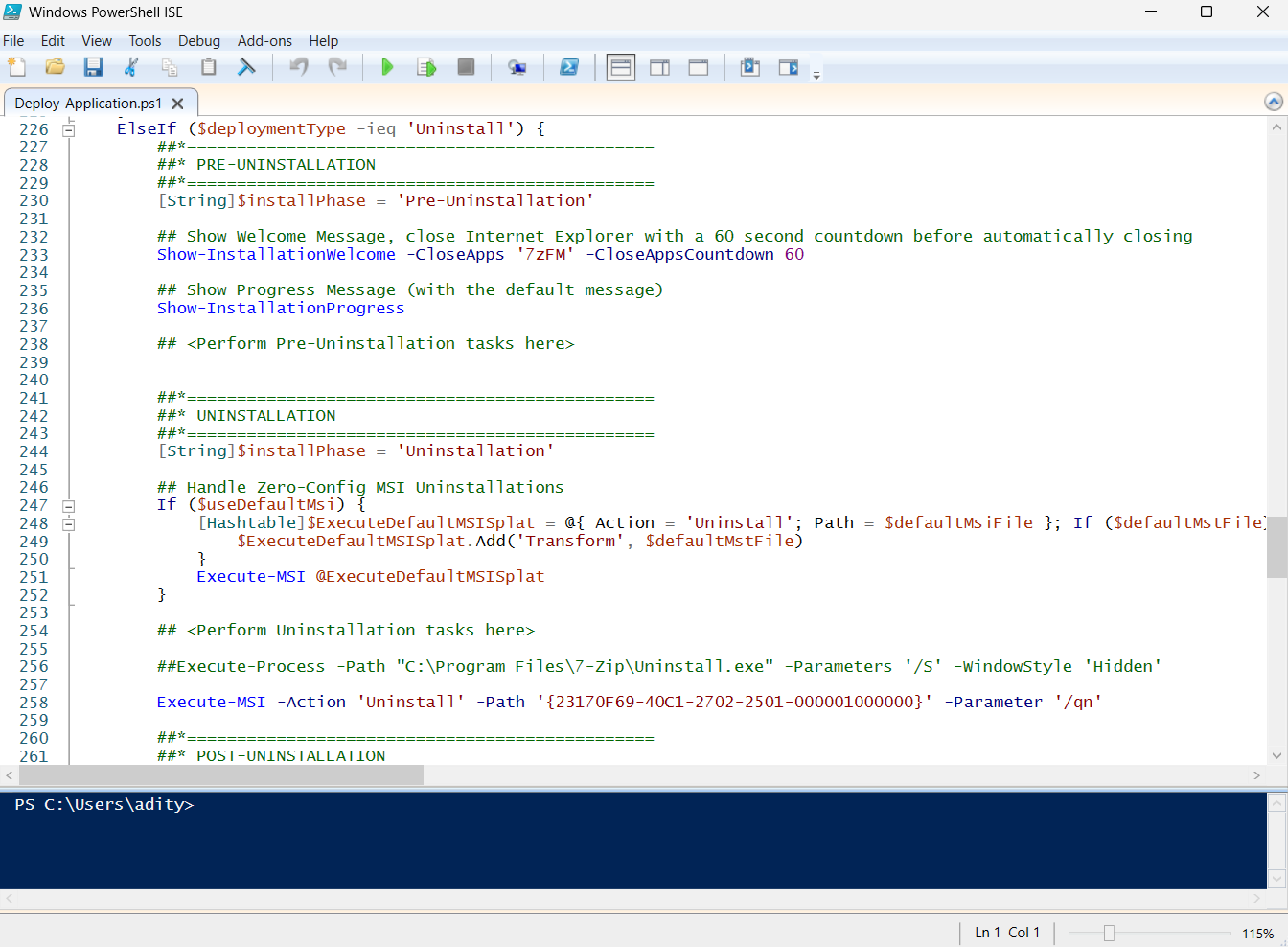
This helps admins deploy or remove apps with consistent commands.

Run installation from PowerShell or CMD: C:\Users\adity\Downloads\Toolkit\Deploy-Application.exe Install





Run uninstallation: C:\Users\adity\Downloads\Toolkit\Deploy-Application.exe Uninstall  




### 5. Predefined Environment Variables

To make scripting easier, PSADT provides variables you can use without hardcoding:

* $appName — The friendly name of the application being deployed.
* $installPhase — Identifies if the script is running pre-install, install, or post-install code.
* $dirFiles — Path to the folder containing your installation files.
* $dirSupportFiles — Path to the SupportFiles folder with helpers/resources.

These keep your script clean and adaptable if folder locations change.

### 6. Core Deployment Commands

PSADT wraps common deployment tasks in simple commands:

* **Execute-MSI:** Installs or uninstalls MSI packages with built-in logging and silent options.
* **Execute-Process:** Runs EXE files, batch scripts, or any executable installer with parameters you specify.
* **Execute-MSP:** Applies Microsoft patch files quietly and logs the outcome.
* **Install-MSUpdates:** Installs Windows updates (MSU or CAB files), useful during patch cycles.

Using these commands ensures uniform behavior and logging across deployments.

### 7. Execute-MSI: How It Works

* To install an MSI package, use:  
   Execute-MSI -Action Install -Path 'MyApp.msi'
* To uninstall by product code:  
   Execute-MSI -Action Uninstall -Path '{PRODUCT-CODE-GUID}'
* The toolkit automatically adds silent install parameters and logs the process, so you don’t need to handle that manually.

This cuts down errors and keeps deployment quiet and clean.

### 8. Execute-Process for EXE Installers

* When dealing with EXE installers or scripts, use:  
   Execute-Process -Path 'setup.exe' -Parameters '/silent /norestart'
* This command lets you pass any parameters your installer requires, enabling unattended installs or custom scripts.
* It’s versatile for batch files or other executable tools, not just EXEs.

### 9. Execute-MSP for Patching

* To deploy MSP patch files:  
   Execute-MSP -Path 'update.msp'
* This runs patches with the correct silent switches and logs results.
* Essential for applying software updates or fixes during deployment workflows.

### 10. Installing Windows Updates

* To install OS updates during deployment:  
   Install-MSUpdates -MSUFile 'Windows10-KB123456.msu'
* Supports MSU and CAB file formats.
* Great for integrating security or system patches into your software rollout process.