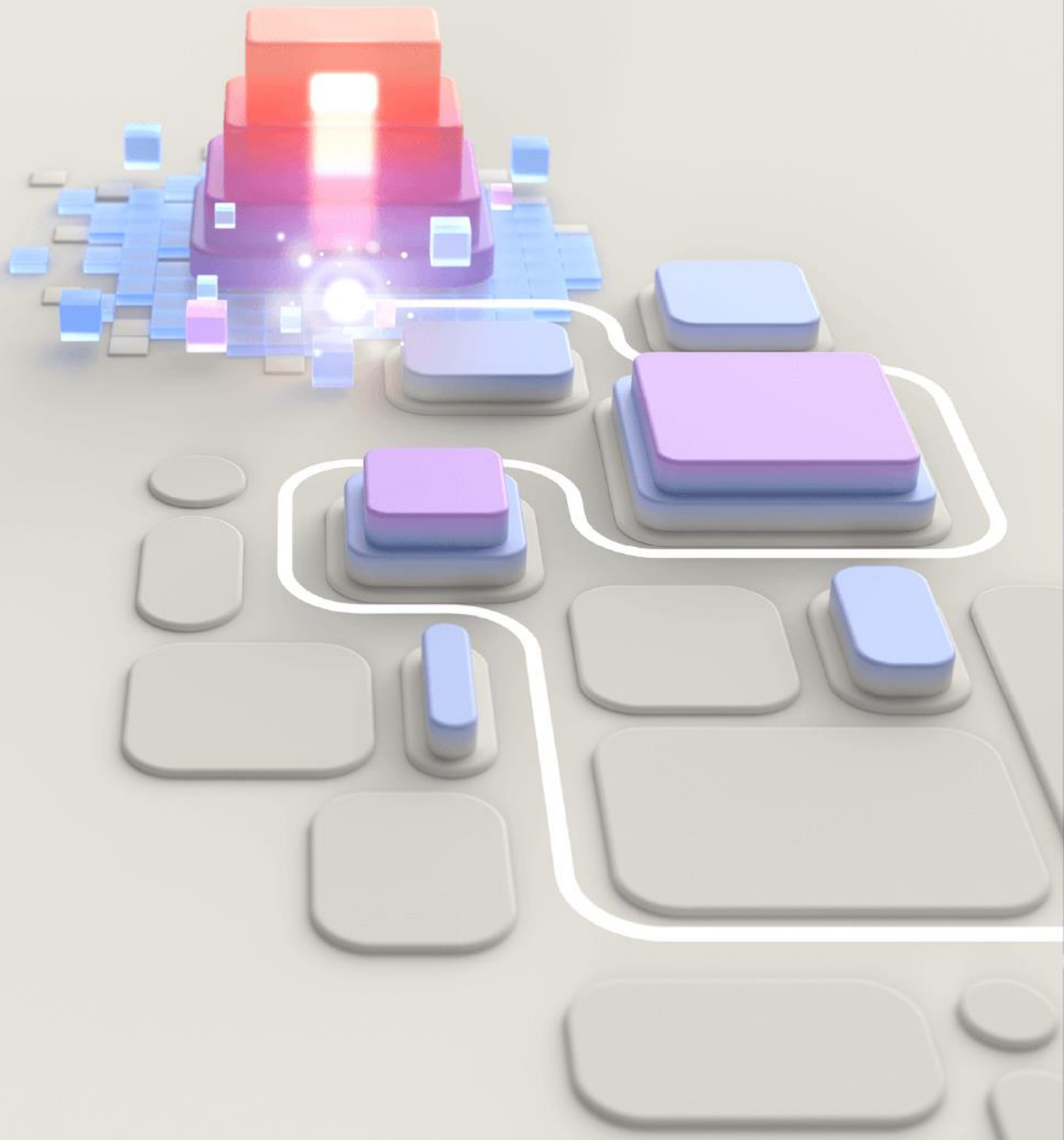
Microsoft

AZ-040 Automating Administration with PowerShell



Course outline

Learning Path 1: Getting started with Windows PowerShell
Learning Path 2: Windows PowerShell for local systems administration
Learning Path 3: Working with the Windows PowerShell pipeline
Learning Path 4: Using PSProviders and PSDrives
Learning Path 5: Querying management information by using CIM and WMI
Learning Path 6: Working with variables, arrays, and hash tables
Learning Path 7: Windows PowerShell scripting
Learning Path 8: Administering remote computers with Windows PowerShell
Learning Path 9: Managing Azure resources with PowerShell
Learning Path 10: Managing Microsoft 365 services with PowerShell
Learning Path 11: Using background jobs and scheduled jobs

Learning Path 4: Work with PowerShell providers and PowerShell drives in Windows PowerShell

Learning objectives

- Connect with data stores using PowerShell providers
- Use PowerShell drives in PowerShell

Overview

PSProvider and PSDrive are two technologies that let you work with many forms of storage by using the same commands and techniques that you use to manage the file system:

- *PSProvider*: A Windows PowerShell adapter that makes some form of storage resemble a hard drive
- *PSDrive*: An actual connection to a form storage

Modules:

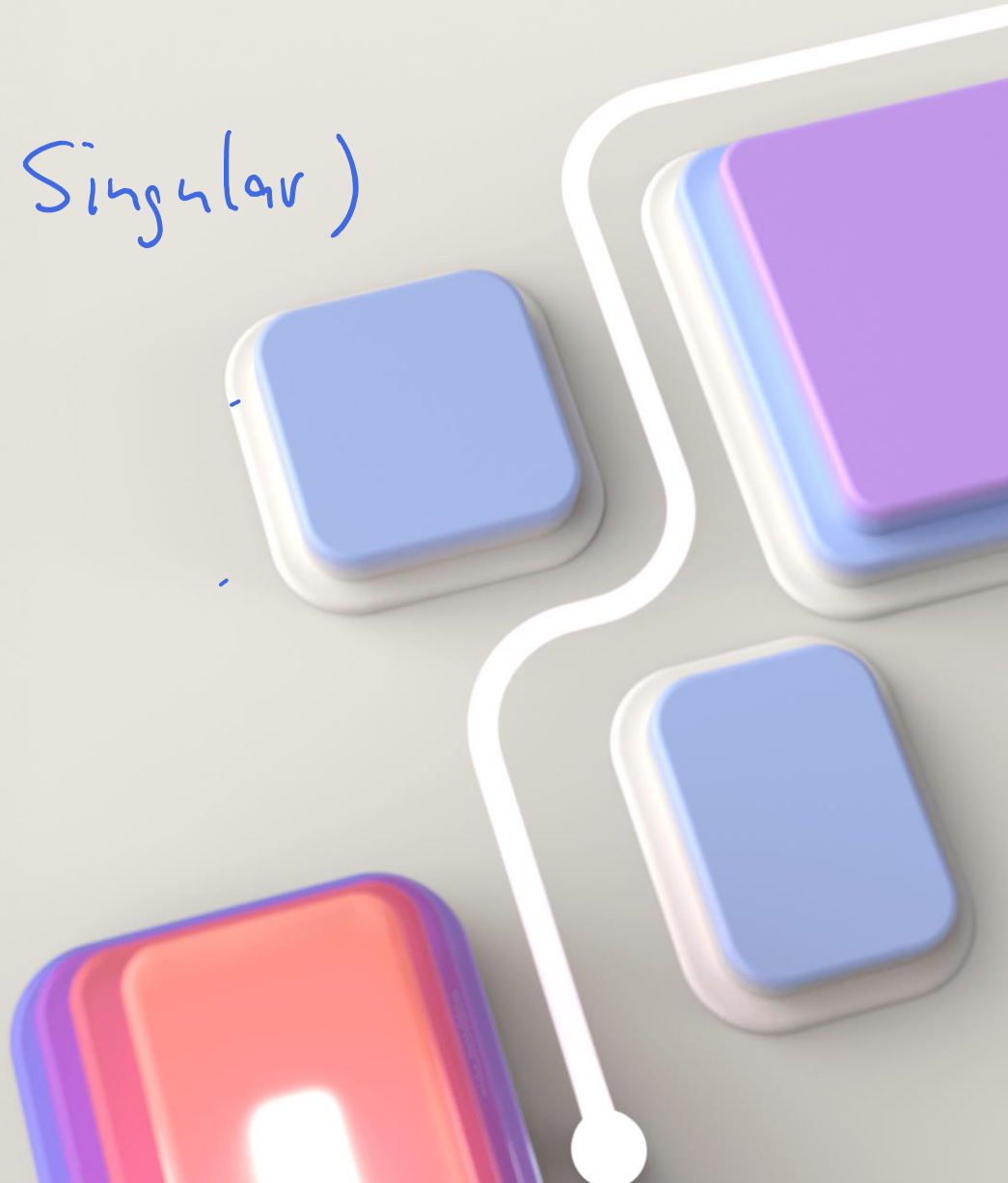
- Connect with data stores using PowerShell providers
- Use PowerShell drives in PowerShell

Get-PSDrive

Noun

(immer im Singular)

Use PowerShell drives in PowerShell



Module overview

In this Module, you'll learn about PSProviders, which are adapters that connect Windows PowerShell to data stores. Providers give you an easier-to-understand and consistent interface for working with these data stores. This makes learning to work with the providers simpler and allows you to reuse scripts as you change the underlying technologies with which you are working.

Units:

- What are Windows PowerShell providers?
- Different provider capabilities
- Accessing provider help
- Demonstration: Reviewing provider help

What are Windows PowerShell providers?

Windows PowerShell providers:

- Adapt data stores to resemble hard drives inside Windows PowerShell.
- Allow management by using familiar file-system management commands.
- More complex than managing by using technology-specific commands.
- Good solution for dynamic or extensible technologies where all manageable components can't be known in advance.
- Examples of providers:
  - **Registry**
  - **Alias**
  - **Environment**
  - **FileSystem**
  - **Function**
  - **Variable**

dir HKLM:

dir C:

dir variable:

Different provider capabilities

- Providers have varying capabilities depending on the data that they manage.
- Use **Get-PSProvider** to review the capabilities of each provider.

Provider capability	Description
ShouldProcess	Supports the **-WhatIf** and **-Confirm** parameters
Filter	Supports queries with filtering
Include	Supports including data based on filtering
Exclude	Supports excluding data based on filtering
ExpandWildcards	Supports using wildcards for queries
Credentials	Supports using alternative credentials
Transactions	Supports combining multiple operating into a single transaction

Accessing provider help

1. Use the **Get-PSProvider** cmdlet to obtain a list of loaded providers:
   - To load a provider, you need to load the module that includes the provider

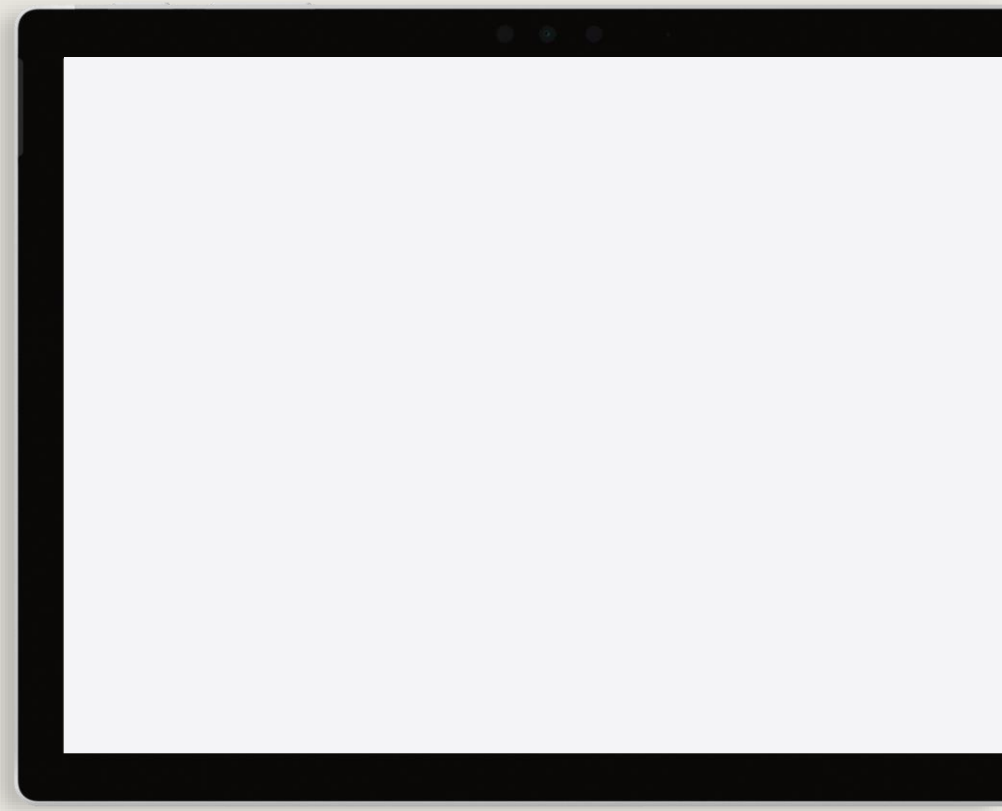
2. Provider help files use the naming format **about_*ProviderName*_Provider**:
   - To review the help for the FileSystem provider, run the following command:
     Get-Help about_FileSystem_Provider

3. Cmdlets that work with providers use the **Item** and **ItemProperty** nouns:
   - To list cmdlets that work with providers, run the following command:
     Get-Command *-Item,*-ItemProperty

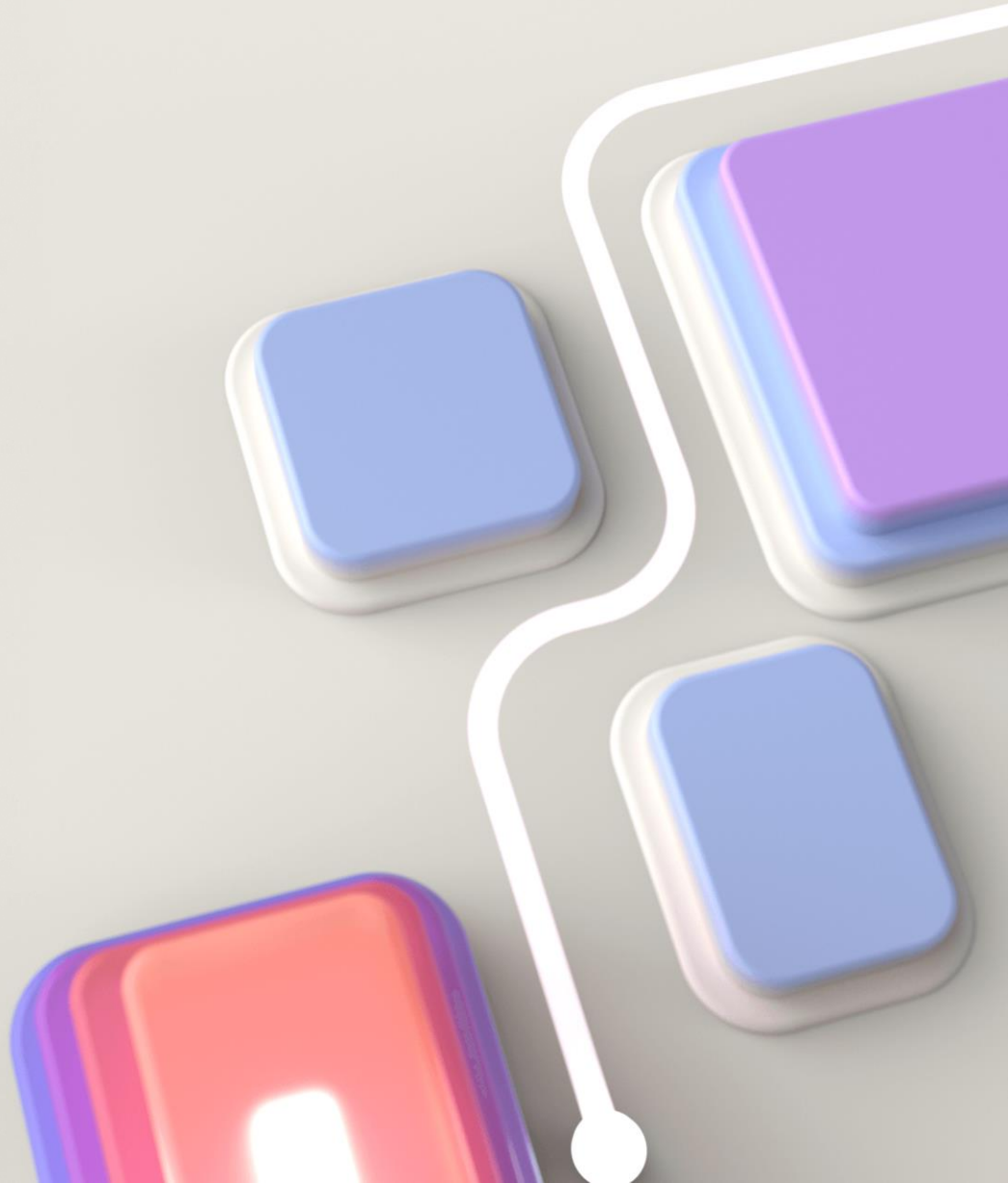
Demonstration: Reviewing PSProvider help

In this demonstration, you'll learn how to:

1. Display the list of loaded providers
2. Load the **ActiveDirectory** module
3. Display the list of loaded providers again
4. Review help topics that include the term **Registry**
5. Display help for the **about_Registry_Provider** topic



Using PSDrives



Module overview

In this Module, you'll learn how to work with PSDrives—a specific form of storage that connects to Windows PowerShell by using a PSProvider. Understanding how to work with PSDrives enables you to use PSProviders successfully. In some cases (such as with the file system), PSDrives are the primary interface for working with the underlying data.

Units:

- What are PSDrives?
- Cmdlets for using PSDrives
- Working with the file system
- Demonstration: Managing the file system
- Working with the registry
- Demonstration: Managing the registry
- Working with certificates
- Working with other PSDrives

What are PSDrives?

1. A drive represents a connected data store.
2. Drives use a PSProvider to connect to the data store.
3. Drives have a name such as **C** or **Alias.**
4. Drive names don't include a colon (**:**).
5. Drive *paths* do include a colon, for example **C:**
6. You can:
   - Run the **Get-PSDrive** cmdlet for a list of drives.
   - Run the **New-PSDrive** cmdlet to map a new drive.
7. Windows PowerShell always starts with the same drives mapped.

Cmdlets for using PSDrives

- PSDrives contain items, child items, and item properties.
- To list cmdlets for managing PSDrive content, run the following command:
  Get-Command -Noun Item,ChildItem,ItemProperty
- Verbs commonly used for managing PSDrive contents include:
  - **New**
  - **Set**
  - **Get**
  - **Clear**
  - **Copy**
  - **Move**
  - **Remove**
  - **Rename**
  - **Invoke**
- To list cmdlets for managing PSDrive locations, run the following command:
  Get-Command -Noun Location

Working with the file system

1. **New-Item** creates files and folders:
   - Alternate commands (aliases): **mkdir, md, ni**
   - Requires the **-Path** and **-ItemType** parameters

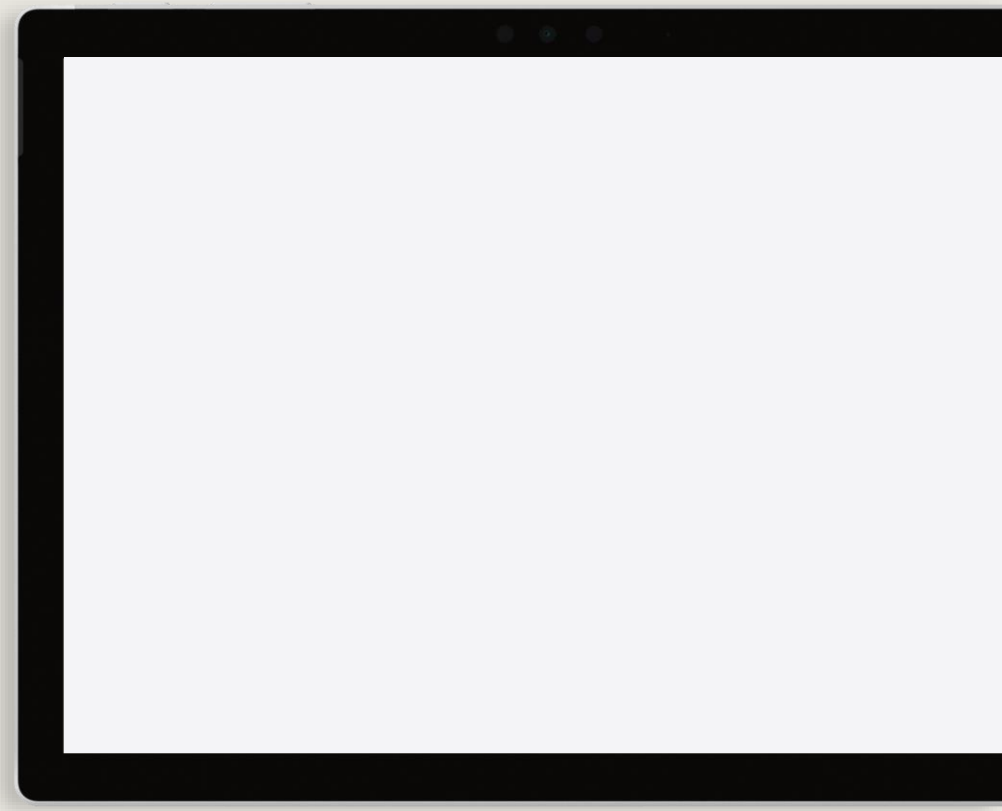
2. **Remove-Item** deletes files and folders:
   - Alternate commands (aliases): **del, erase, rd, ri, rm, rmdir**
   - Use -**Recurse** to delete files when deleting folders

3. **Get-Item** and **Get-ChildItem** retrieve files and folders:
   - Alternate commands (aliases): **gi, gci, dir, ls**
   - Supports -**Exclude, -Include**, and **-Filter**

Demonstration: Managing the file system

In this demonstration, you'll learn how to:

1. Use **Cd** to change location to **C:**
2. Use **Set-Location** to change location to **C:\Windows**
3. Create a new PSDrive that maps to **C:\Windows**
4. List files in the new WINDIR PSDrive
5. Create a new file in the WIDIR PSDrive



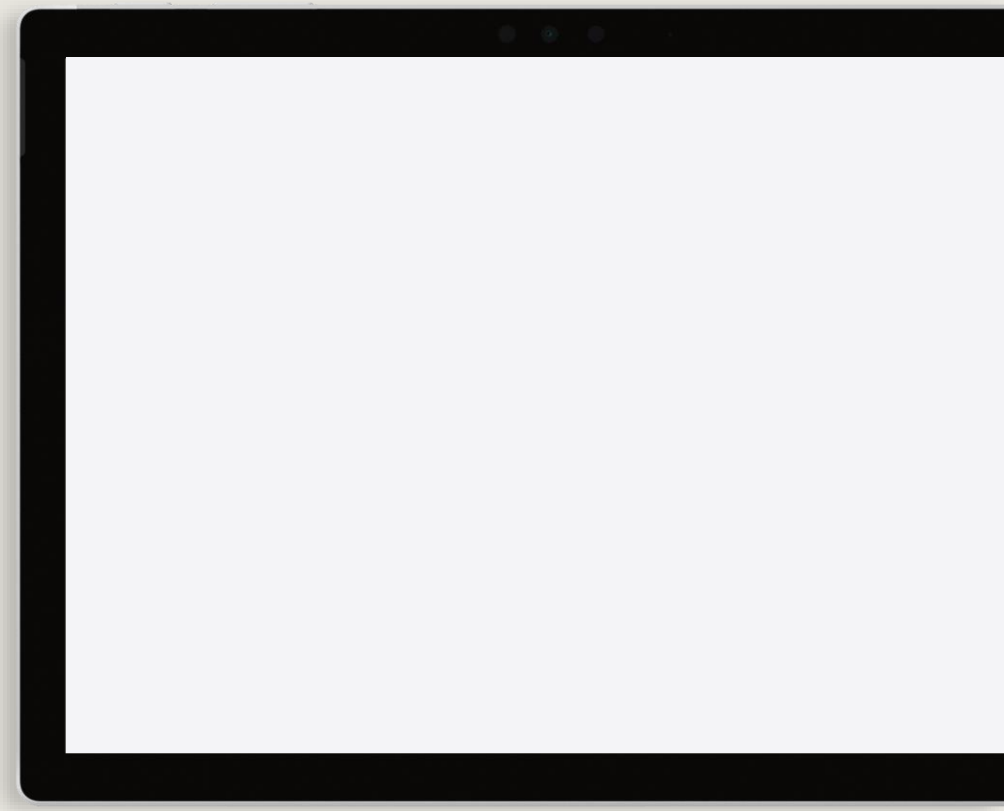
Working with the registry

1. There are two default PSDrives for registry locations:
   - **HKLM**
   - **HKCU**
2. Use the **Item** and **ChildItem** nouns to manage registry keys.
3. Use the **ItemProperty** noun to manage key values.
4. The **Invoke-Item** cmdlet is not supported.
5. The Registry provider supports transactions.

Demonstration: Managing the registry

In this demonstration, you'll learn how to:

1. Set the working location to **HKLM:\Software**.
2. List the registry keys in **HKLM:\Software**.
3. Create a registry key.
4. Create a registry key value.
5. List registry key values.



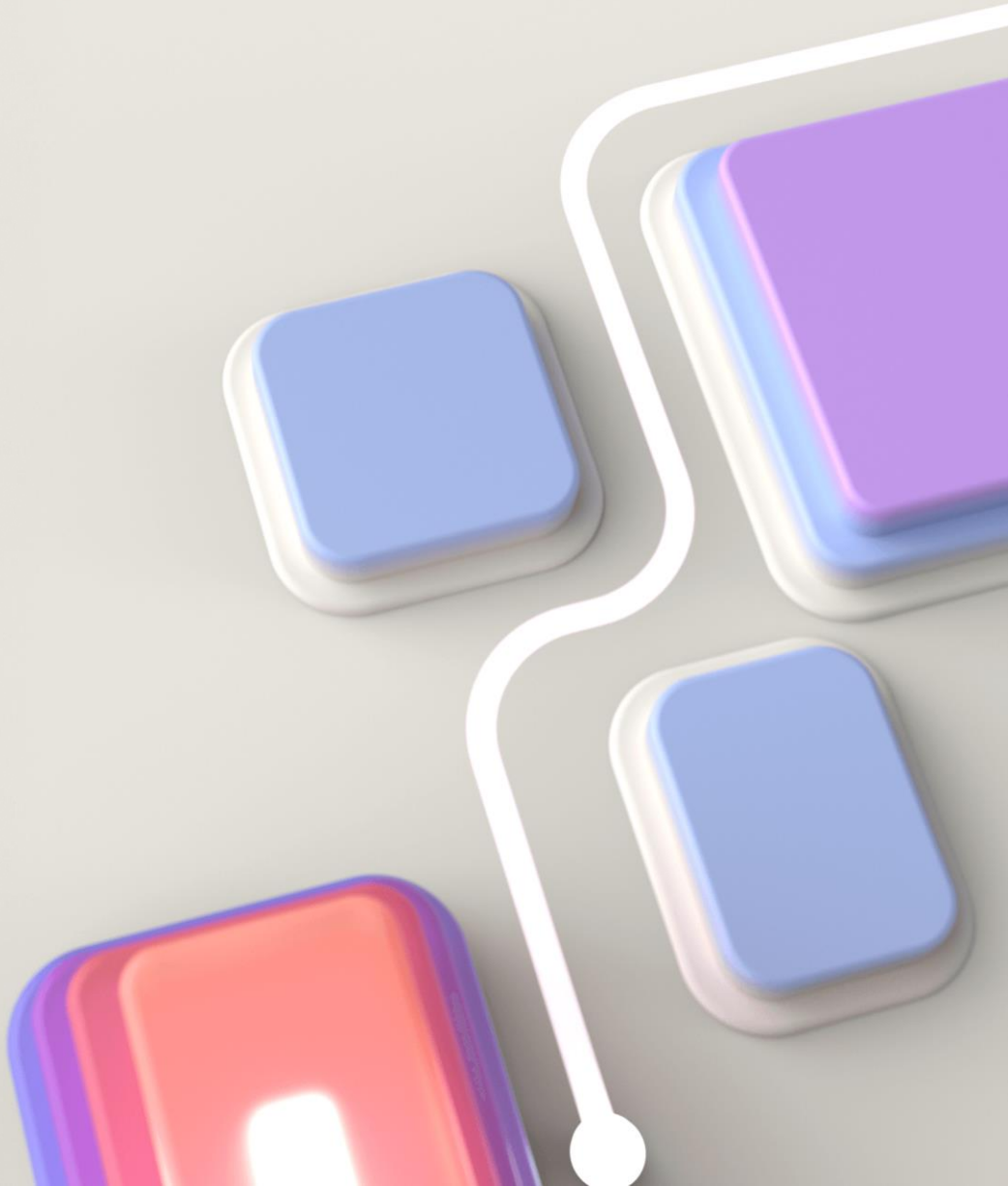
Working with certificates

- The Cert PSDrive provides access to local certificate stores:
  - **CurrentUser**
  - **LocalMachine**
- Manage certificates with the **Item** and **ChildItem** nouns.
- **Get-ChildItem** has unique parameters for Certificates:
  - **CodeSigningCert**
  - **DocumentEncryptionCert**
  - **DnsName**
  - **EKU**
  - **ExpiringInDays**
  - **SSLServerAuthentication**
- There are also cmdlets specifically for managing certificates.

Working with other PSDrives

1. Other PSDrives include:
   - **Alias**
   - **Env**
   - **Function**
   - **Variable**
   - **WSMan**
2. PSDrives created automatically when optional providers are loaded include:
   - **AD**
   - **IIS**

Lab – Using PSProviders and PSDrives with PowerShell



Lab: Using PSProviders and PSDrives with PowerShell

- **Exercise 1**: Creating files and folders on a remote computer
- **Exercise 2**: Creating a registry key for your future scripts
- **Exercise 3**: Creating a new Active Directory group

Sign-in information for the exercises:

- Virtual machines:
  - **AZ-040T00A-LON-DC1**
  - **AZ-040T00A-LON-SVR1**
  - **AZ-040T00A-LON-CL1**
- Username: **Adatum\Administrator**
- Password: **Pa55w.rd**

Lab scenario

You're a system administrator for the London branch office of Adatum Corporation. You must reconfigure several settings in your environment. You recently learned about PSProviders and PSDrives and how you can use them to access data stores. You've decided to use PSDrives to reconfigure these settings.

Lab-review questions

- How is a drive that you create using **New-PSDrive** different from a mapped drive letter to file share?
- When would you use PSDrives to access registry data instead of using Regedit?
- When creating common AD DS objects such as users and groups, is there another method that's easier than using the **AD** drive?

Lab-review answers

How is a drive that you create using **New-PSDrive** different from a mapped drive letter to file share?

- A drive mapping to a file share is used to make that file share available within all areas of Windows such as File Explorer. A PSDrive is accessible only within a Windows PowerShell prompt.

When would you use PSDrives to access registry data instead of using Regedit?

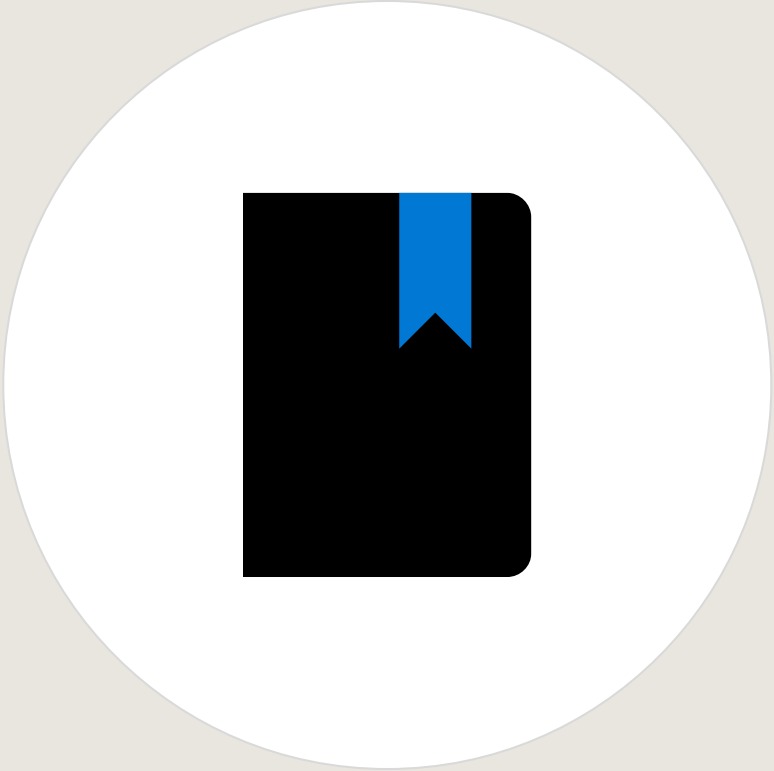
- You would use PSDrives when registry modification needs to be scripted for many computers to ensure reliability.

When creating common AD DS objects such as users and groups, is there another method that's easier than using the **AD** drive?

- The ActiveDirectory module includes cmdlets such as **Get-ADUser** and **New-ADGroup** that make it easier to manage common AD DS objects rather than using the **AD** drive.

References

Push-Location



Learning Path Recap

In this learning path, we:

- Learned about different Windows PowerShell providers and different provider capabilities.
- Learned how to work with PSDrives and PSProviders.

End of presentation