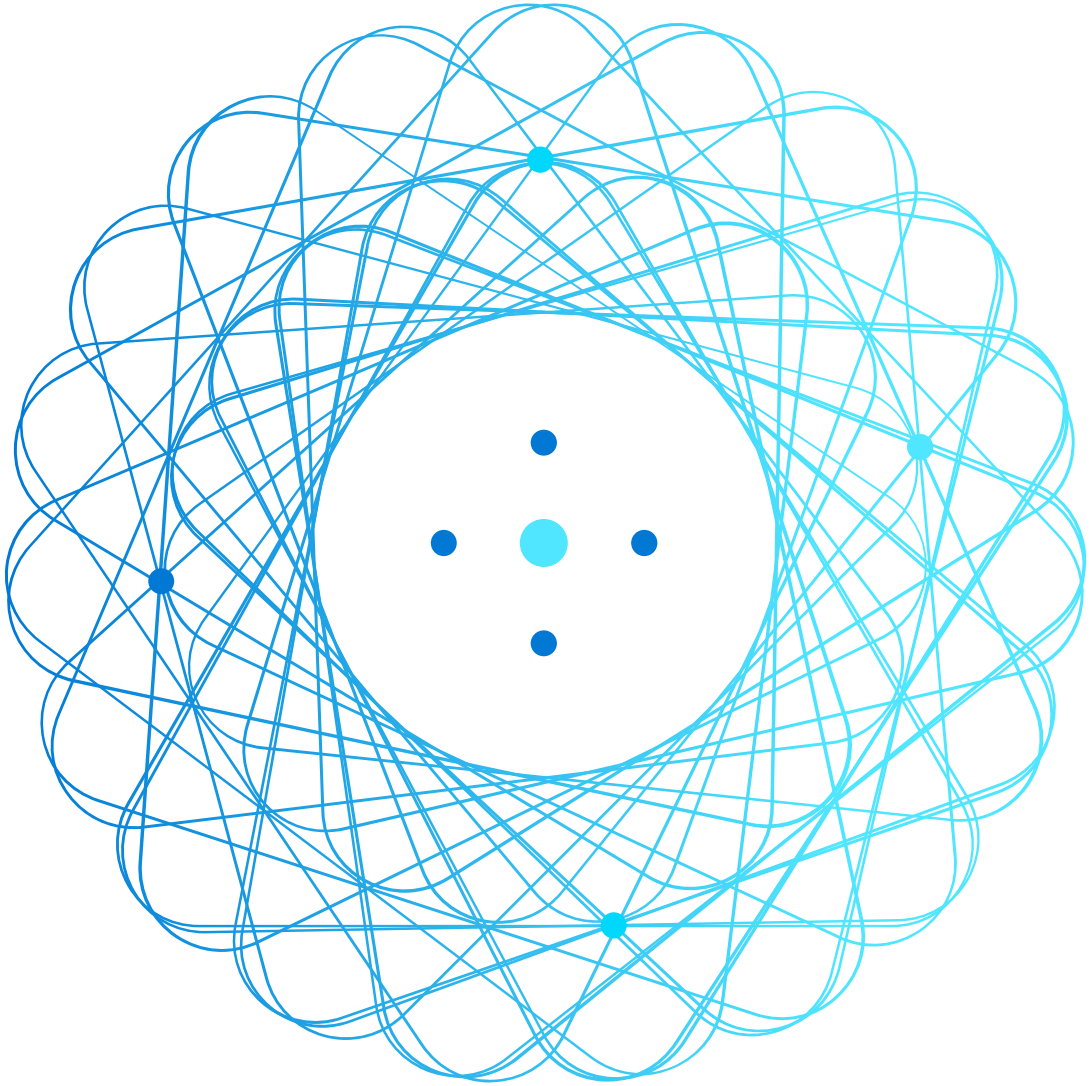
Microsoft Azure

AZ-040 Automating Administration with PowerShell

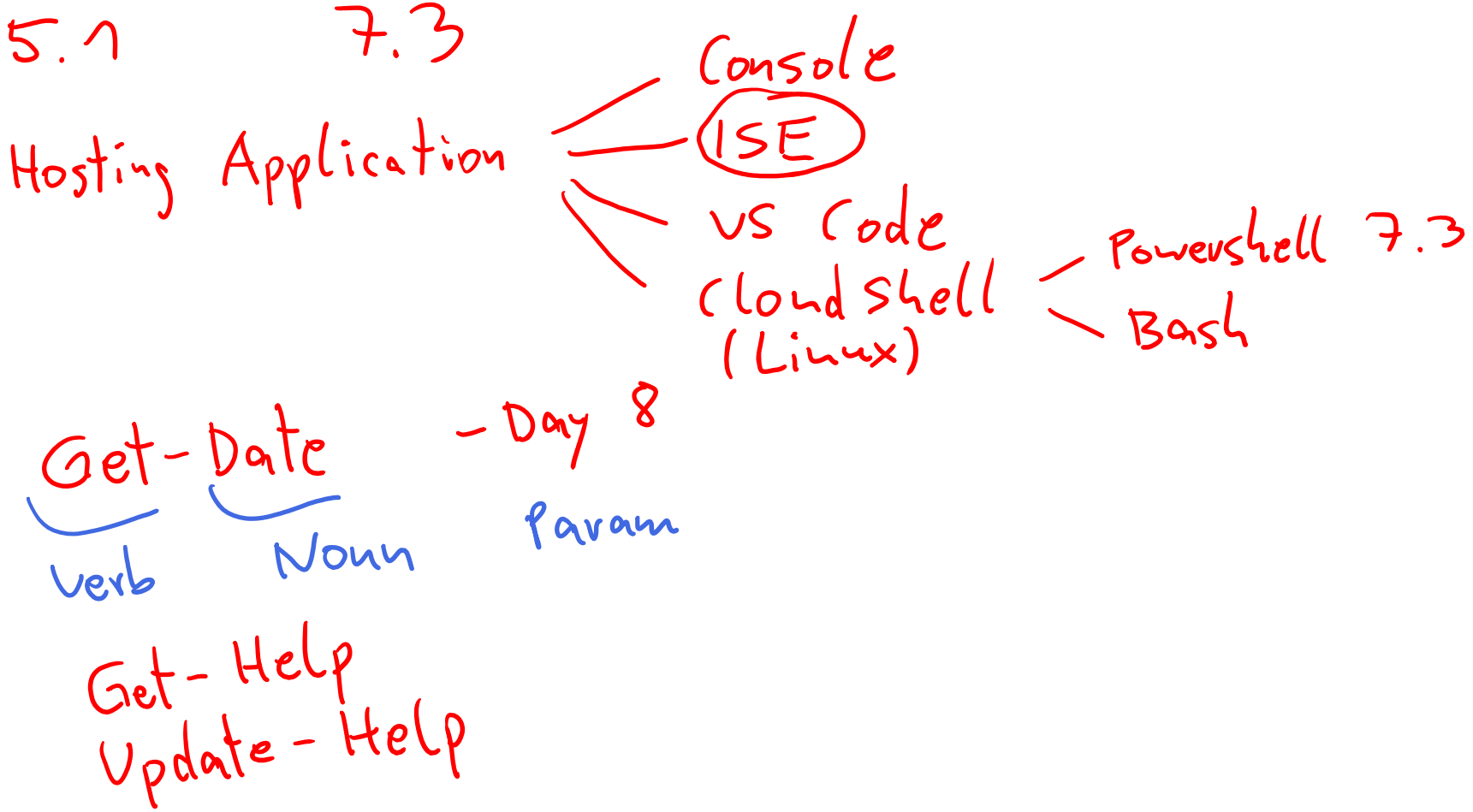


Course outline

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

Skillable

Learning Path 1: Get started with Windows PowerShell



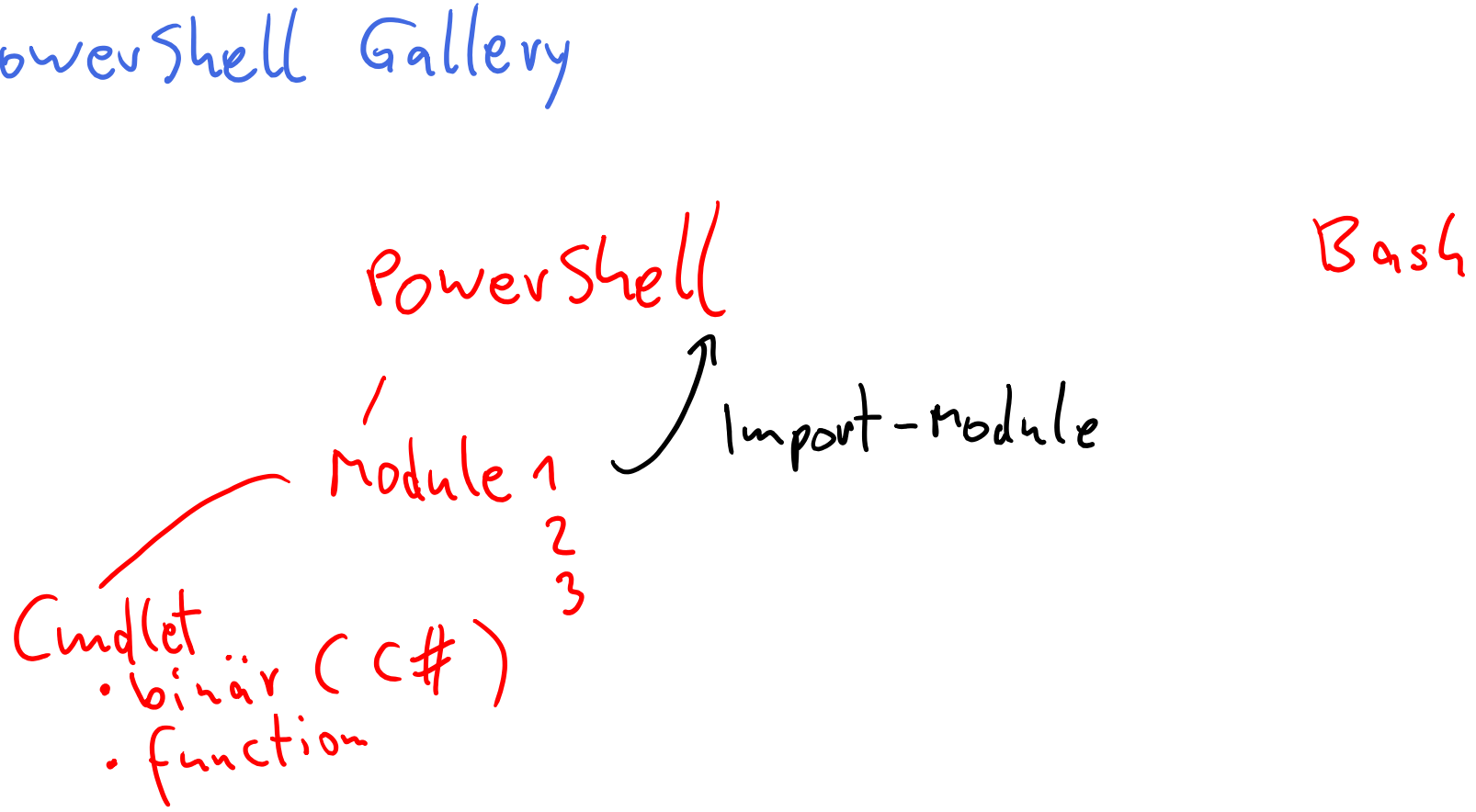
Overview

This Learning Path introduces you to PowerShell and provides an overview of its functionality. You'll learn to open, configure, and run commands by using PowerShell on Windows. You'll also learn about PowerShell's built-in help system, which assists you with using the various PowerShell components and commands.

Modules:

- Review Windows PowerShell
- Understand the command syntax in Windows PowerShell
- Find commands and get help in Windows PowerShell

Review Windows PowerShell



Module overview

In this Module, you'll learn about PowerShell's system requirements and how to open and configure commonly used host applications, including the Windows PowerShell console and Windows PowerShell Integrated Scripting Environment (ISE). Finally, you'll discover how you can use Microsoft Visual Studio Code (VS Code) to develop PowerShell scripts.

Units:

- Windows PowerShell introduction
- Windows PowerShell versions
- Windows PowerShell applications
- Considerations when using PowerShell
- Configuring the PowerShell console
- Demonstration: Configuring the console
- Configuring the ISE
- Demonstration: Configuring the ISE
- Using Visual Studio Code with PowerShell

Windows PowerShell Introduction

- PowerShell is an automation solution that consists of:
  - A command-line shell.
  - A scripting language.
  - A configuration management framework.
- Commands include:
  - Cmdlets
  - Functions
  - Filters
  - Workflows
- PowerShell was originally built on the .NET framework and only worked on Windows operating systems.
- It currently uses the .NET Core and can run on Windows, MacOS, and Linux platforms.

Windows PowerShell versions

Version	Release date	Notes
PowerShell 7.2	November 2021	Built on .NET 6.0.
PowerShell 7.1	November 2020	Built on .NET 5.0.
PowerShell 7.0	March 2020	Built on .NET Core 3.1.
PowerShell 6.0	September 2018	Built on .NET Core 2.0. First release that's installable on Windows, Linux, and macOS.
PowerShell 5.1	August 2016	Released in Windows 10 Anniversary Update and Windows Server 2016 and as part of Windows Management Framework (WMF) 5.1.
PowerShell 5.0	February 2016	Integrated in Windows 10 version 1511. Released in Windows Management Framework (WMF) 5.0. Can be installed on Windows Server 2008 R2, Windows Server 2012, Windows 10, Windows 8.1 Enterprise, Windows 8.1 Pro, and Windows 7 SP1.
PowerShell 4.0	October 2013	Integrated in Windows 8.1 and Windows Server 2012 R2. Can be installed on Windows 7 SP1, Windows Server 2008 SP1, and Windows Server 2012.

Windows PowerShell versions (Slide 2)

Version	Release date	Notes
PowerShell 3.0	October 2012	Integrated in Windows 8 and Windows Server 2012. Can be installed on Windows 7 SP1, Windows Server 2008 SP1, and Windows Server 2008 R2 SP1.
PowerShell 2.0	July 2009	Integrated in Windows 7 and Windows Server 2008 R2. Can be installed on Windows XP SP3, Windows Server 2003 SP2, and Windows Vista SP1.
PowerShell 1.0	November 2006	Installable on Windows XP SP2, Windows Server 2003 SP1, and Windows Vista. Optional component of Windows Server 2008.

Windows PowerShell applications

- Windows PowerShell console includes:
  - Basic command-line interface.
  - Maximum support for PowerShell features.
  - Minimal editing capabilities.
- Windows PowerShell ISE includes:
  - Script editor and console combination.
  - Rich editing capabilities.
- PowerShell Core doesn't support Windows PowerShell ISE. It uses VS Code with the PowerShell extension.

Lang Server PS
FG Debug
Extension

Considerations when using PowerShell

When using PowerShell, you should:

- Install and use PowerShell side-by-side with Windows PowerShell:
  - PowerShell uses a separate installation path and executable name (pwsh.exe).
  - PowerShell uses a separate PSModulePath, profile, and event logs.
  - You identify the PowerShell version by using **$PSVersionTable**.
- Run PowerShell using Administrative credentials:
  - 64-bit operating systems include both 64-bit and 32-bit versions of PowerShell.
  - The Windows title bar must display **Administrator** if you need administrative privileges in Windows PowerShell.
  - When UAC is enabled, you must right-click the application icon or activate its context menu to run as Administrator.
- Identify and modify the execution policy in PowerShell:
  - Use **Get-ExecutionPolicy** to identify the effective execution policy in PowerShell.
  - Be aware that **Restricted** is the default for Windows clients and **RemoteSigned** is the default for Windows servers.

Configuring the PowerShell console

- Select a font style, size, and color, and set the screen color so that text is easy to read and you can differentiate between often-confused characters, such as ` ' ({ [<.
- Modify screen size to maximize available space for output.
- Make sure that the screen buffer width is smaller than window width.
- Enable copy and paste.

" " double quotes
' ' single "
` back tic
\ back slash
/ slash

Demonstration: Configuring the console

In this demonstration, you'll learn how to:

1. Run the 64-bit console as Administrator.
2. Set a font family.
3. Set a console layout.
4. Start a transcript.

Configuring the ISE

- Two panes: script and console.
- One-pane and two-pane view options.
- **Command Add-on** displays available commands.
- Customization of font style, size, and color.
- Customization of screen color.
- Bundling of color selections into themes.
- Additional features include snippets, add-ins, and debugging.

Demonstration: Configuring the ISE

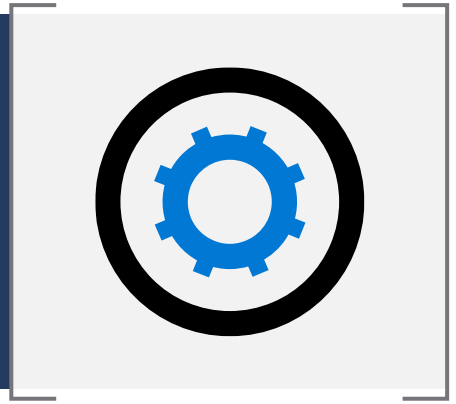
In this demonstration, you'll learn how to:

1. Run the ISE as Administrator.
2. Configure the pane layout.
3. Dock and undock the command pane.
4. Configure the font size.
5. Select a color theme.

Using Visual Studio Code with PowerShell

- Visual Studio Code is a Microsoft Script Editor that offers a similar experience to the PowerShell ISE when you're using the PowerShell extension add-on.
- Supports the following:
  - PowerShell Core 6, 7, and newer for Windows, macOS, and Linux.
  - PowerShell 5.1 for Windows.
- Supports ISE mode, which enables:
  - Mapping keyboard functions in VS Code so they match those used in the ISE.
  - Replicating the VS Code user interface to resemble the ISE.
  - Enabling ISE-like tab completion.
  - Providing several ISE themes to make the VS Code editor look like the ISE.

Section break 2



Understand Windows PowerShell command syntax

Jeff Snover

Module overview

In this Module, you'll learn about the cmdlet structure and parameters for using Windows PowerShell cmdlets. You'll also learn how to use tab completion and display About files content.

Units:

- Cmdlet structure
- Parameters
- Tab completion
- About files
- Demonstration: Using About Files

Cmdlet structure

- The cmdlet's verb is the action the cmdlet performs, such as:
  - Get
  - Set
  - New
  - Add
  - Remove
- The cmdlet's noun is the resource the cmdlet affects, such as:
  - Service
  - Process
  - Use prefixes to group related nouns, including **AD**, **SP**, and **Az**

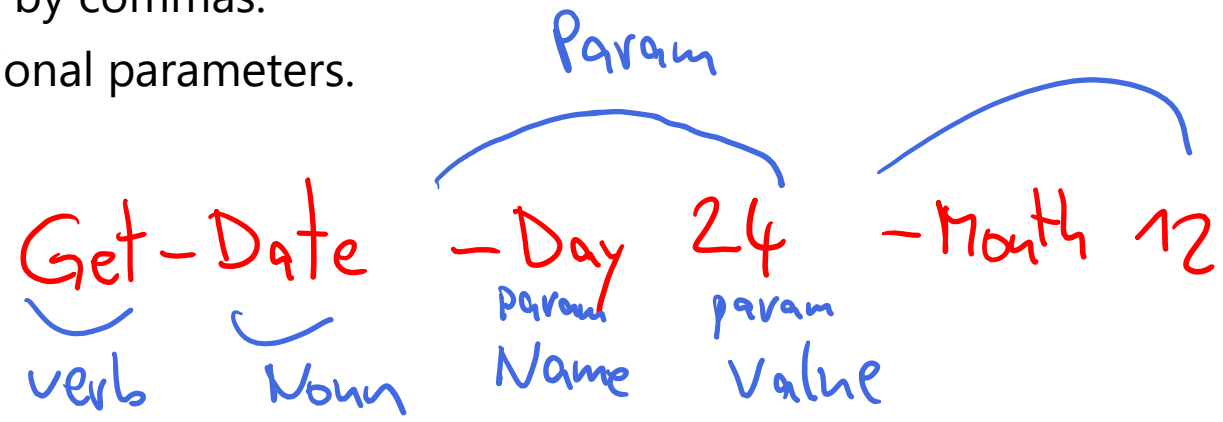
Get-Verb

Singular

Parameters

- Parameters modify the action of a cmdlet.
- Names are entered starting with a dash (-).
- Parameters can be optional or required:
  - You'll receive prompts for required parameters, if needed.
- Some accept multiple values, separated by commas.
- Parameter names are optional for positional parameters.

[]
Array



Help

[-Name]
↑
optional
= positional param

Get-Service -Name Bits
Get-Service "Bits" ✓

Tab completion

- Enables you to enter a few characters of a cmdlet or a parameter in the ISE or console and then press the Tab key on the keyboard.
- Allows you to enter cmdlet, parameter, variable, and path names more quickly and accurately.
- Helps you to discover cmdlets and parameters.
- Supports the use of wildcards.

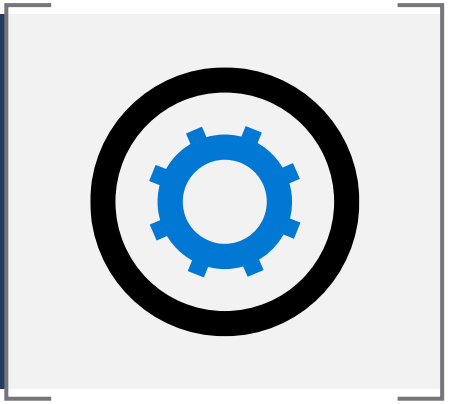
About files

- Provide documentation for global shell techniques, concepts, and features.
- Start with **about_**.
- Review list by running **Get-Help about***.
- You'll need to read many of these files to complete several upcoming lab exercises.

Demonstration: Using About files

In this demonstration, you'll learn how to review About help file topics.

Section break 3



Find commands and get help in Windows PowerShell

Module overview

In this Module, you'll learn how to find Windows PowerShell cmdlets for performing specific tasks. This Module provides strategies and tools for finding cmdlets based on the actions they perform and the features or technologies they manage.

You'll also learn how to use **Get-Help** to retrieve detailed information about a cmdlet and its parameters.

Units:

- What are modules?
- Finding cmdlets
- What are aliases?
- Demonstration: Using aliases
- Using **Show-Command**
- Using **Get-Help**
- Demonstration: Reviewing Help
- Interpreting the help syntax
- Updating help

What are modules?

- Modules:
  - Are containers for related cmdlets.
  - Are provided as part of management tools for various software packages.
  - Must be loaded into your current session.
  - May only support specific operating systems.
- Windows PowerShell version 3.0 and newer support autoloading.
- Windows PowerShell and PowerShell Core support different module paths as indicated by the *$Env:PSModulePath* environment variable.

Finding cmdlets

- Use **Get-Command** and **Get-Help**, both of which support wildcards.
- Use **–Noun**, **–Verb**, and **–Module** parameters with **Get-Command**.
- **Get-Help** can also search help files if no match is found when searching command names.
- Use the **Find-Command** and the **PowerShellGet** module to find modules and commands from the PowerShell Gallery.

What are aliases?

- Familiar batch commands include:
  - **Dir**
  - **Cd**
  - **Mkdir**
  - **Type**
- These are really aliases to Windows PowerShell commands.
- External commands such as **ping.exe** and **ipconfig.exe** all work as usual.
- Windows PowerShell commands often have a different syntax, even if accessed by an alias that matches an older command name.

Demonstration: Using aliases

In this demonstration, you'll learn how to:

1. Find an alias for a cmdlet.
2. Find a cmdlet based on an alias you already know.
3. Create an alias.

Using Show-Command


```
Get-ADUser -Identity Ana
```

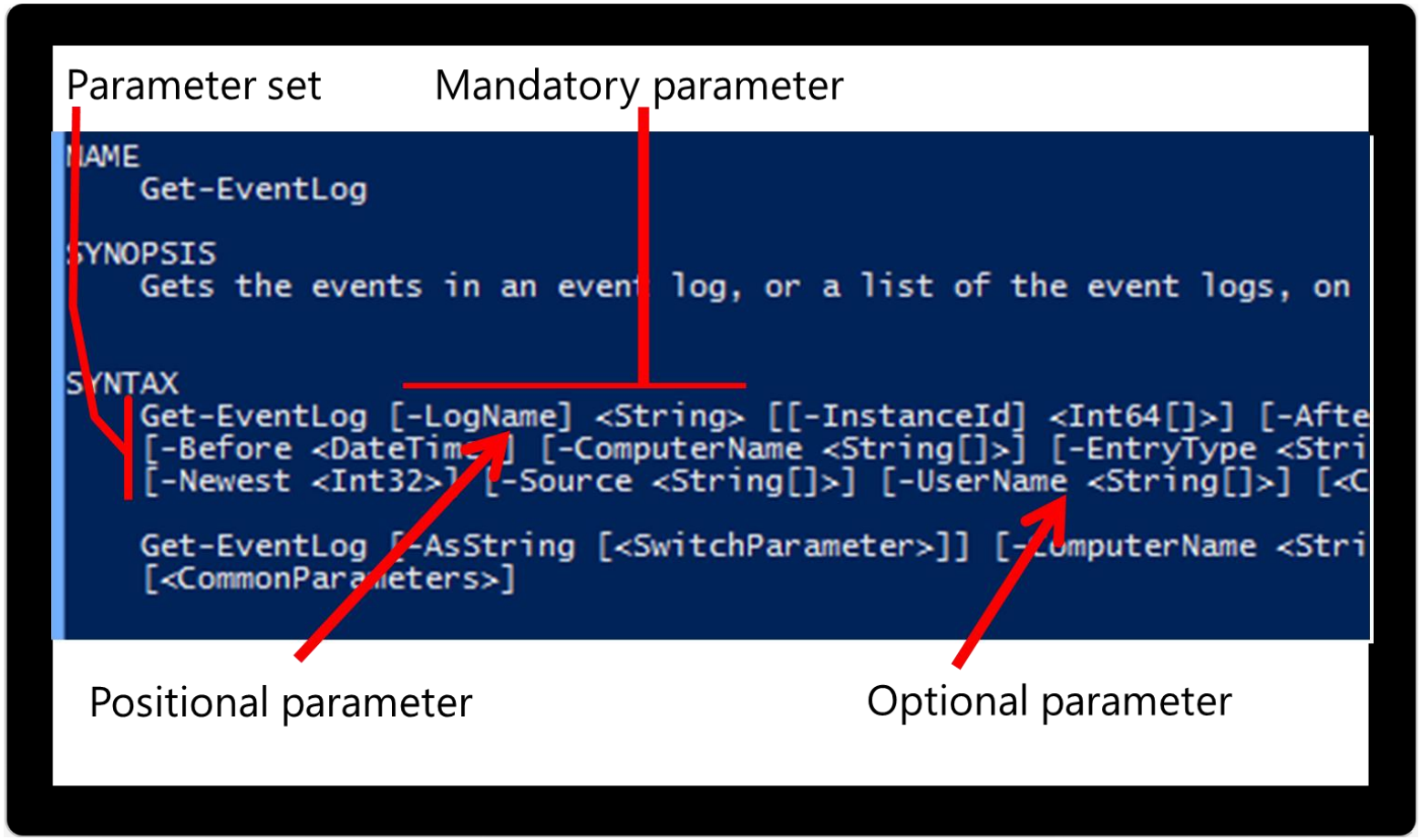
Using Get-Help

- Displays Windows PowerShell help content.
- You provide a cmdlet name to display help for a cmdlet.
- Supports wildcards.
- Parameters include:
  - *-Examples*
  - *-Full*
  - *-Online*
  - *-ShowWindow*
  - *-Parameter ParameterName*

Demonstration: Reviewing Help

In this demonstration, you'll learn how to use various options of the help system.

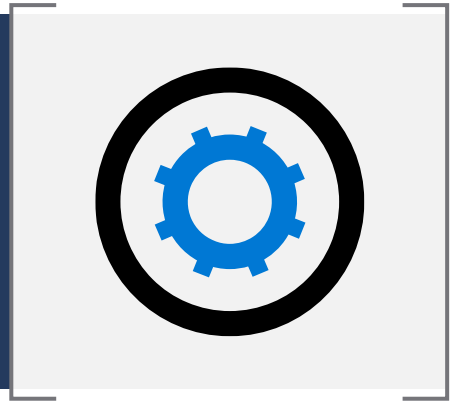
Interpreting the help syntax



Updating help

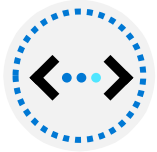
- Windows PowerShell 3.0 and newer versions don’t ship with help files.
- **Update-Help**:
  - Uses downloadable help content to update your local help.
  - Checks no more than once every 24 hours by default.
- **Save-Help** enables you to download help and save it to an alternate location accessible to computers that aren’t connected to the internet.

Section break 4



Lab: Configuring Windows PowerShell, and finding and running commands

Exercise 1: Configuring the Windows PowerShell console application



Exercise 2: Configuring the Windows PowerShell ISE application

Exercise 3: Finding and running Windows PowerShell commands

Exercise 4: Using About files

Sign-in information for the exercise(s):

Virtual machines:

- **AZ-040T00A-LON-DC1**
- **AZ-040T00A-LON-CL1**

Username: **Adatum\Administrator**

Password: **Pa55w.rd**

Lab scenario

You're an administrator and want to automate several tasks by using Windows PowerShell. You must ensure that you can successfully start the correct Windows PowerShell host applications and configure them for future use by customizing their appearance.

Additionally, you're preparing to complete several administrative tasks by using Windows PowerShell. You need to discover commands that you can use to perform those tasks, run several commands to begin performing those tasks, and learn about new Windows PowerShell features that'll enable you to complete those tasks.

Lab-review questions

Why might you configure alternative text colors in the ISE?

What causes a horizontal scroll bar in the Windows PowerShell console window?

What are some methods for finding commands, other than using **Get-Help** and **Get-Command**?

Lab-review answers

Why might you configure alternative text colors in the ISE?

Text and screen color are a matter of personal preference. However, some default ISE colors can be difficult to review, such as the default light gray used for curly brackets and other punctuation. Changing the colors can make these elements easier to review, helping you avoid errors.

What causes a horizontal scroll bar in the Windows PowerShell console window?

You'll observe a horizontal scroll bar when the screen buffer size is set to a value that is greater than the window size.

What are some methods for finding commands, other than using **Get-Help** and **Get-Command**?

- You can use the **Get-Module** command with the *-ListAvailable* parameter to search for available modules. The results of the **Get-Module** command include a partial list of commands. This can help you identify possible nouns, or you can use the module name as a parameter in **Get-Command**.
- You can use the **Get-Alias** command to identify the Windows PowerShell command that runs behind the scenes when you run a command that you used in **cmd.exe** or Linux environments and the command also works in Windows PowerShell.

References

[PowerShell Documentation]

[Install and Configure WMF 5.1]

[Visual Studio Code]

[Visual Studio Marketplace]

[Using Visual Studio Code for PowerShell Development]

[PowerShell Gallery]

[about_Profiles]