Getting Started with PowerShell

Module Overview

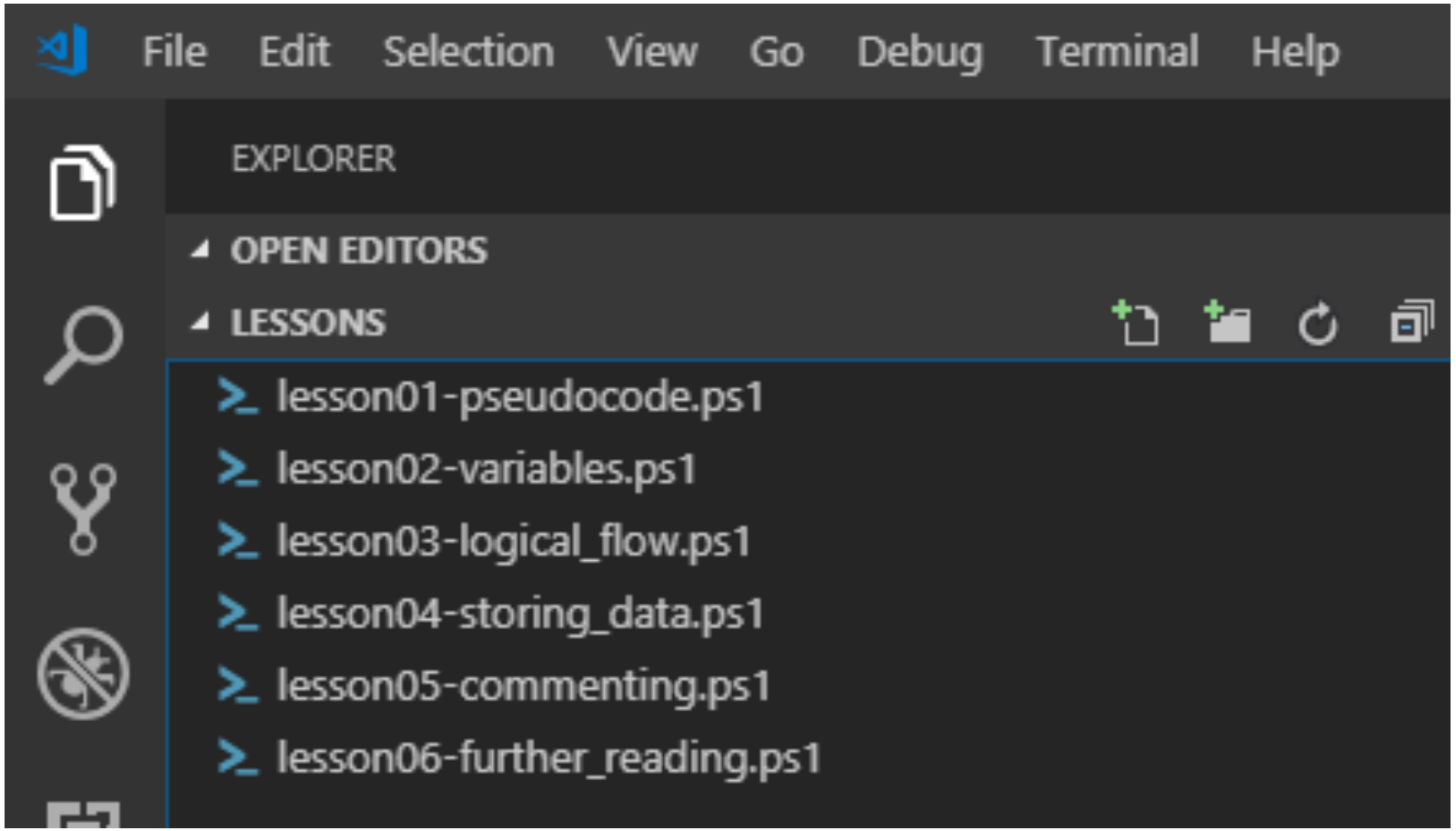
This module will cover the following:

- PowerShell Basics
  - Cmdlet, Parameters, Aliases, Getting Help
- Variables
- Pipeline
- Modules
- Rubrik SDK for PowerShell

Module Setup

- Establish a connection to the Rubrik Build environment
- Launch the Visual Studio Code (VSCode) application

1. Click on the **File** Menu
2. Select **Open Folder**
3. Navigate to “getting-started-with-powershell”
4. Open the “Lessons” folder



Example of the Lessons Folder

Learning PowerShell Basics

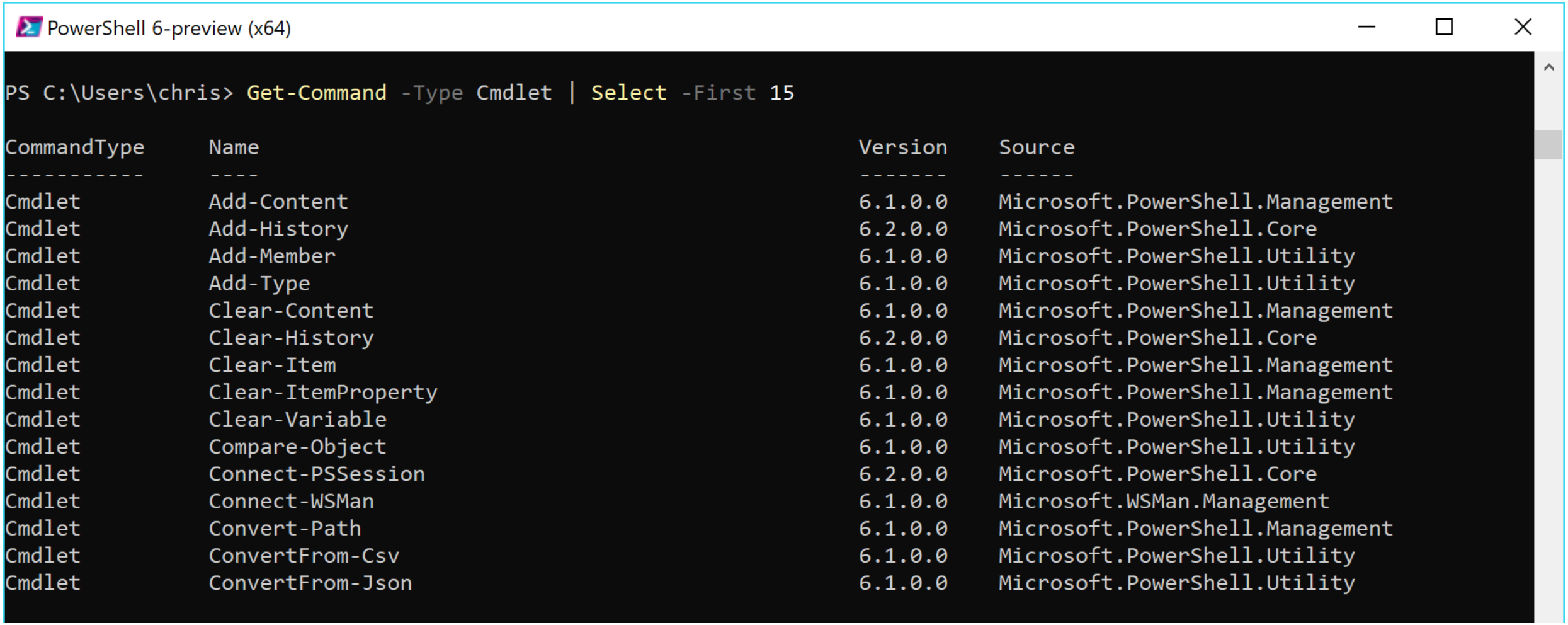
Learning Objectives

- Cmdlets
- Parameters
- Aliases
- Getting Help

What are Cmdlets?

A cmdlet (pronounced "command-let") is a lightweight PowerShell script that performs a single function.

- `Verb-Noun` format (*enforced*)
  - Singular verb and noun only. Don’t use plurals.
- Examples:
  - `Get-Computer`
  - `Get-Process`
  - `Get-ChildItem`
  - `Get-Module`


PowerShell 6-preview (x64)
PS C:\Users\chris> Get-Command -Type Cmdlet | Select -First 15
CommandType Name Version Source
Cmdlet Add-Content 6.1.0.0 Microsoft.PowerShell.Management
Cmdlet Add-History 6.2.0.0 Microsoft.PowerShell.Core
Cmdlet Add-Member 6.1.0.0 Microsoft.PowerShell.Utility
Cmdlet Add-Type 6.1.0.0 Microsoft.PowerShell.Utility
Cmdlet Clear-Content 6.1.0.0 Microsoft.PowerShell.Management
Cmdlet Clear-History 6.2.0.0 Microsoft.PowerShell.Core
Cmdlet Clear-Item 6.1.0.0 Microsoft.PowerShell.Management
Cmdlet Clear-ItemProperty 6.1.0.0 Microsoft.PowerShell.Management
Cmdlet Clear-Variable 6.1.0.0 Microsoft.PowerShell.Utility
Cmdlet Compare-Object 6.1.0.0 Microsoft.PowerShell.Utility
Cmdlet Connect-PSSession 6.2.0.0 Microsoft.PowerShell.Core
Cmdlet Connect-WSMan 6.1.0.0 Microsoft.WSMan.Management
Cmdlet Convert-Path 6.1.0.0 Microsoft.PowerShell.Management
Cmdlet ConvertFrom-Csv 6.1.0.0 Microsoft.PowerShell.Utility
Cmdlet ConvertFrom-Json 6.1.0.0 Microsoft.PowerShell.Utility

What are Parameters?

Allow users to select options or provide input in the form of “arguments”

- PowerShell processes parameters directly
- Encourages (but does not guarantee) that every cmdlet conforms to the standard.
- Parameter names always have a '-' prepended to them with a PowerShell command.
- Example:
  - `Get-Module -ListAvailable -Name "Rubrik"`

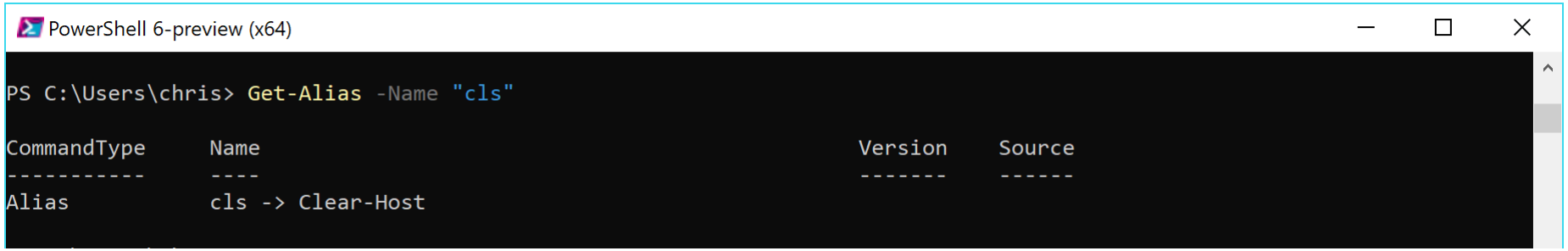
PowerShell 6-preview (x64)

```
PS C:\Users\chris> Get-Help Get-Module -Parameter *

-All

    Required?                    false
    Position?                    Named
    Accept pipeline input?       false
    Parameter set name           Available, Loaded
    Aliases                      None
    Dynamic?                     false


-CimNamespace <string>

    Required?                    false
    Position?                    Named
    Accept pipeline input?       false
    Parameter set name           CimSession
    Aliases                      None
    Dynamic?                     false


-CimResourceUri <uri>

    Required?                    false
    Position?                    Named
    Accept pipeline input?       false
    Parameter set name           CimSession
    Aliases                      None
    Dynamic?                     false
```

What are Aliases?

Aliasing associates a new name with another command.

- Most of the old CMD.exe commands are available as aliases.
- Example:
  - `cls = Clear-Host`
  - `gmo = Get-Module`


PowerShell 6-preview (x64)
PS C:\Users\chris> Get-Alias -Name "cls"
CommandType Name Version Source
Alias cls -> Clear-Host

Getting Help

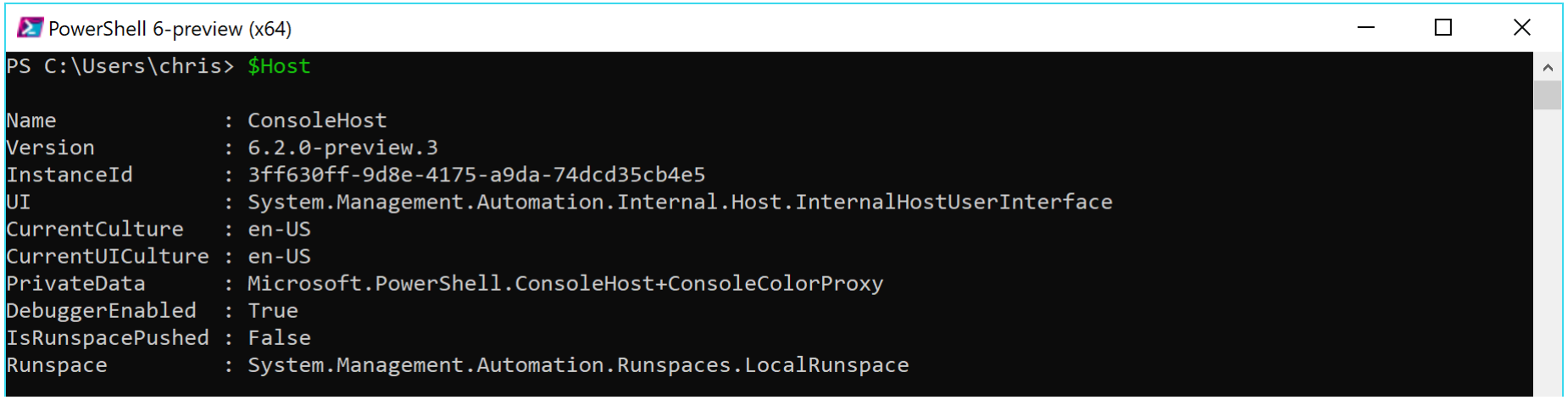
All cmdlets have help associated with them.
Similar to a “man” (manual) page in Linux.

- Use `Get-Help <cmdlet>` to view help information.
- Helpful parameters
  - *Examples*: Displays only the name, synopsis, and examples.
  - *Details*: Adds parameter descriptions and examples to the basic help display
  - *Full*: Displays the whole help topic for a cmdlet.
- Use `Update-Help` to refresh your local copy of help files.

Variables and Pipeline

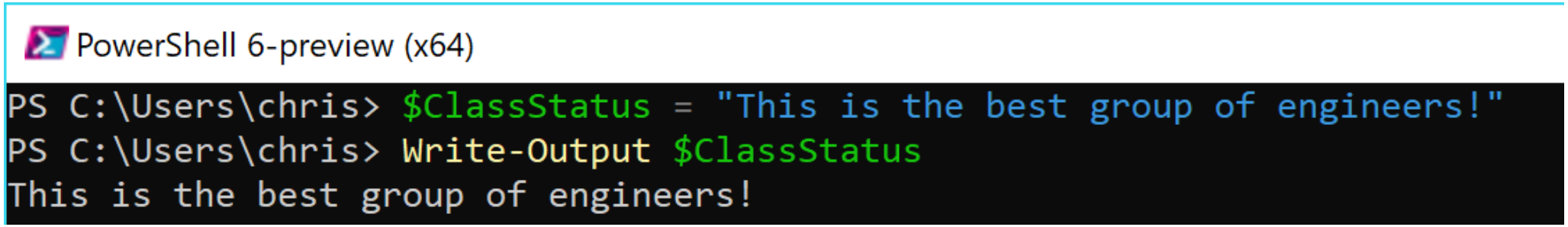
What are Variables?

A storage location paired with an associated symbolic name, which contains some known or unknown quantity of data referred to as a value.



How do I declare a variable?

Just start using it! ☺



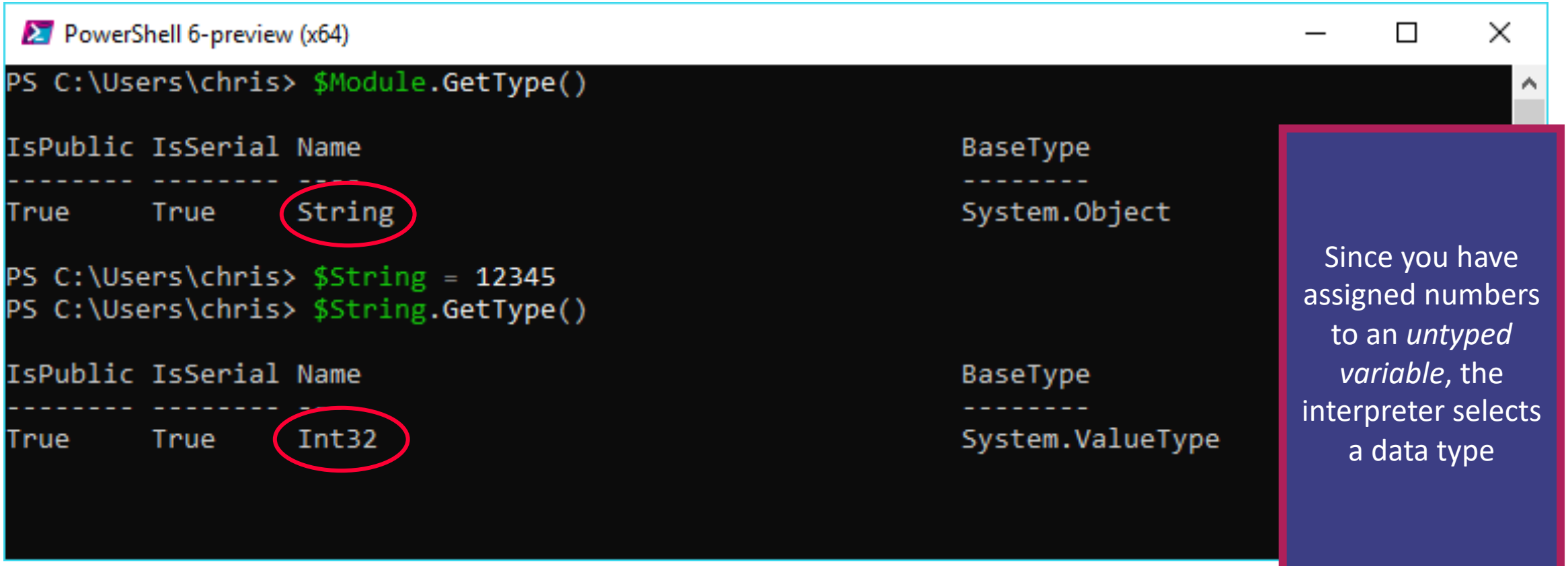
Variable Data Types

Always tell your code which data type to use for a variable.

Otherwise, it will make the variable's data type match the input that it sees.

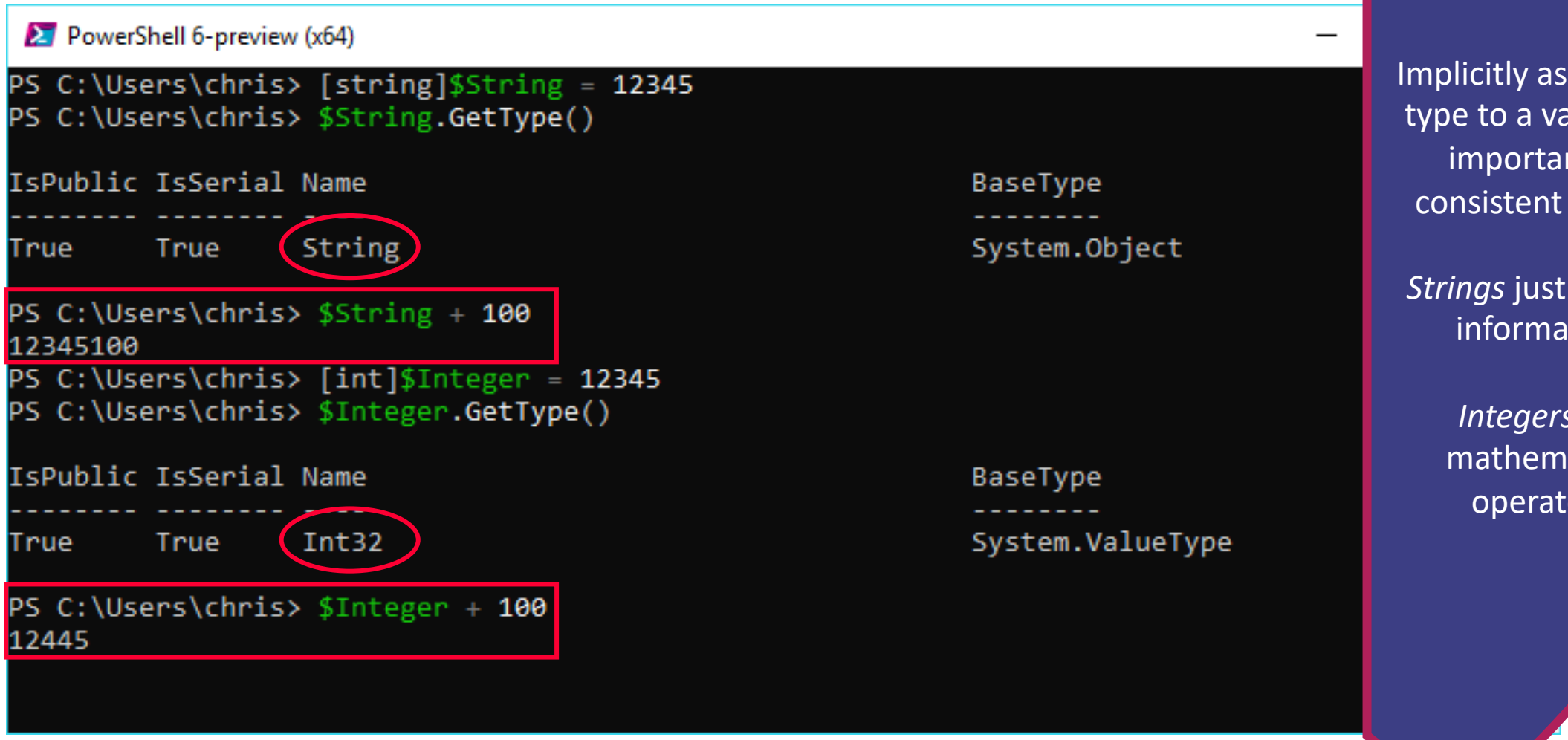
- String
- Int (Integer)
- Bool (Boolean)
- Float (Floating-point)
- Array
- Hashtable
- SecureString

Example – Untyped Variables



Since you have assigned numbers to an *untyped variable*, the interpreter selects a data type

Example – Typed Variables



Implicitly assigning a type to a variable is important for consistent results.

Strings just append information.

Integers use mathematical operators.

Variable Naming Standards

Different projects and languages use different capitalization for variables.

These will be caught when linting your code, or when working on a project that has defined a standard.

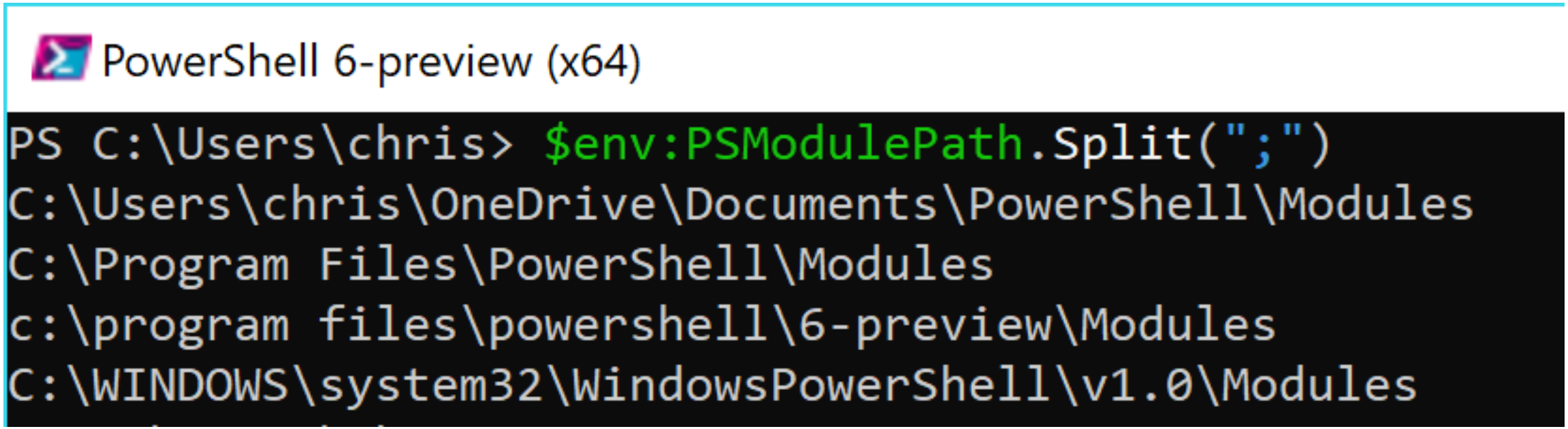
- Snake Case
  - my_variable_name
- Camel Case (Pascal Case)
  - MyVariableName
- Lower Camel Case
  - myVariableName

Environmental Variables

Variables that contain system-specific information.

Local to the system you are working on.

- Start with the `$env:` prefix
- Examples:
  - Environmental Path - `$env:PSModulePath`
  - All Users Profile - `$env:ALLUSERSPROFILE`
  - Computer Name - `$env:COMPUTERNAME`


PowerShell 6-preview (x64)
PS C:\Users\chris> $env:PSModulePath.Split(";")
C:\Users\chris\OneDrive\Documents\PowerShell\Modules
C:\Program Files\PowerShell\Modules
c:\program files\powershell\6-preview\Modules
C:\WINDOWS\system32\WindowsPowerShell\v1.0\Modules

Pipeline

The most powerful thing about PowerShell.

Long live object oriented shell languages!

Pipeline

Pipelines act like a series of connected segments of pipe.

Items moving along the pipeline pass through each segment.

- To create a pipeline in PowerShell, you connect commands together with the pipe operator "|".
  - The output of each command is used as input to the next command.
  - If any object from the input matches a parameter, it is automatically fed into the parameter.

PowerShell 6-preview (x64)

```
PS C:\Users\chris> $PROFILE.Split('\') | Format-Table -AutoSize

C:
Users
chris
OneDrive
Documents
PowerShell
Microsoft.PowerShell_profile.ps1
PS C:\Users\chris> $PROFILE.Split('\') | Select -Last 1
Microsoft.PowerShell_profile.ps1
PS C:\Users\chris>
```

1. Get the Profile data
2. Split it into objects when a backslash is found
3. Format those objects into a table that fits on the screen

1. Get the Profile data
2. Split it into objects when a backslash is found
3. Select the last “1” objects to return

Will these results be identical or different?

Using Parameters:

```
Get-Command -Module Rubrik -Verb 'Set’
```

Using the Pipeline:

```
Get-Command -Module Rubrik | Where-Object
{$_.Name -like "Set-*"}
```

Will these results be identical or different?

```
PS C:\> Get-Command -Module Rubrik -Verb 'Set'

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Function        Set-RubrikAvailabilityGroup                        4.0.0.220  Rubrik
Function        Set-RubrikBlackout                                 4.0.0.220  Rubrik
Function        Set-RubrikDatabase                                 4.0.0.220  Rubrik
Function        Set-RubrikHyperVVM                                 4.0.0.220  Rubrik
Function        Set-RubrikLogShipping                              4.0.0.220  Rubrik
Function        Set-RubrikManagedVolume                            4.0.0.220  Rubrik
Function        Set-RubrikMount                                    4.0.0.220  Rubrik
Function        Set-RubrikNASShare                                 4.0.0.220  Rubrik
Function        Set-RubrikNutanixVM                                4.0.0.220  Rubrik
Function        Set-RubrikSQLInstance                              4.0.0.220  Rubrik
Function        Set-RubrikSupportTunnel                            4.0.0.220  Rubrik
Function        Set-RubrikVM                                       4.0.0.220  Rubrik
```

Will these results be identical or different?

```
PS C:\> Get-Command -Module Rubrik | Where-Object {$_.Name -like "Set-*"}

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Function        Set-RubrikAvailabilityGroup                        4.0.0.220  Rubrik
Function        Set-RubrikBlackout                                 4.0.0.220  Rubrik
Function        Set-RubrikDatabase                                 4.0.0.220  Rubrik
Function        Set-RubrikHyperVVM                                 4.0.0.220  Rubrik
Function        Set-RubrikLogShipping                              4.0.0.220  Rubrik
Function        Set-RubrikManagedVolume                            4.0.0.220  Rubrik
Function        Set-RubrikMount                                    4.0.0.220  Rubrik
Function        Set-RubrikNASShare                                 4.0.0.220  Rubrik
Function        Set-RubrikNutanixVM                                4.0.0.220  Rubrik
Function        Set-RubrikSQLInstance                              4.0.0.220  Rubrik
Function        Set-RubrikSupportTunnel                            4.0.0.220  Rubrik
Function        Set-RubrikVM                                       4.0.0.220  Rubrik
```

Modules

What are Modules?

A module is a package that contains PowerShell commands, such as cmdlets, providers, functions, workflows, variables, and aliases.

Why are Modules important?

- People who write commands can use modules to organize their commands and share them with others.
- People who receive modules can add the commands in the modules to their PowerShell sessions and use them just like the built-in commands.

What is in a Module?

- A Module is just a folder with some special files in it.
- The `New-ModuleManifest` cmdlet creates a new module manifest (.psd1) file, populates its values, and saves the manifest file in the specified path.

Module Manifest

Describes the module to the user and PowerShell Gallery.

<Module Name>.psd1

- Version
- Author
- Details on the script module (used to bootstrap the module)
- Requirements and dependencies
- Cmdlets contained within the module
- PowerShell Gallery data for listing

```
@{

# Script module or binary module file associated with this manifest.
RootModule = 'Rubrik.psm1'

# Version number of this module.
ModuleVersion = '4.0.0.237'

# Supported PSEditions
# CompatiblePSEditions = @()

# ID used to uniquely identify this module
GUID = 'a4cb0e3e-b1fe-4da8-9c75-d445e5f96cfb'

# Author of this module
Author = 'Chris Wahl'

# Company or vendor of this module
CompanyName = 'Rubrik'

# Copyright statement for this module
Copyright = '(c) 2015-2019 Rubrik, Inc. All rights reserved.'

# Description of the functionality provided by this module
Description = 'This is a community project that provides a Windows PowerShell module for managing and monitoring Rubrik''s Converged Data Management platform.'

# Minimum version of the Windows PowerShell engine required by this module
PowerShellVersion = '4.0'

# Name of the Windows PowerShell host required by this module
```

Script Manifest

Called by the module manifest to bootstrap the module.

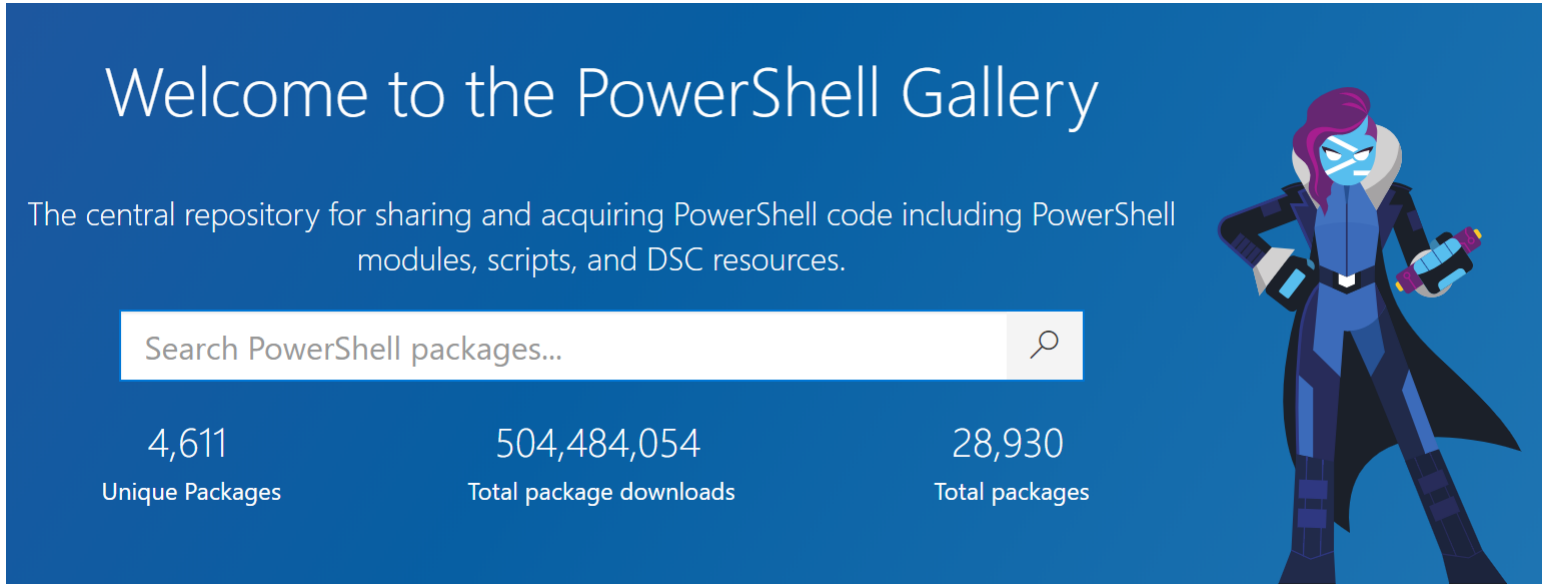
<Module Name>.psm1

- Parses through two sets of functions to load into the targeted session.
- **Private**
  - Cmdlets that are only usable by the module's functions and not the user.
  - Example: Helper functions.
- **Public**
  - Cmdlets that are published and visible to the user.
  - Example: `Connect-Rubrik, Get-RubrikSLA, New-RubrikMount`

```powershell
#Get public and private function definition files.
    $Public  = @( Get-ChildItem -Path $PSScriptRoot\Public\*.ps1 -ErrorAction SilentlyContinue )
    $Private = @( Get-ChildItem -Path $PSScriptRoot\Private\*.ps1 -ErrorAction SilentlyContinue )

#Dot source the files
    Foreach($import in @($Public + $Private))
    {
        Try
        {
            . $import.fullname
        }
        Catch
        {
            Write-Error -Message "Failed to import function $($import.fullname): $_"
        }
    }

# Export the Public modules
Export-ModuleMember -Function $Public.Basename
```

The PowerShell Gallery

- This is a “marketplace” for tons of Modules
  - https://www.powershellgallery.com/



4,970
Downloads

3
Downloads of 4.0.0.241
View full stats

4/1/2019
Last Published

Info

Project Site
License Info
Contact Owners
Report

Rubrik 4.0.0.241

This is a community project that provides a Windows PowerShell module for managing and monitoring Rubrik's Converged Data Management platform.

Minimum PowerShell version
4.0

Installation Options

Install Module | Azure Automation | Manual Download

Copy and Paste the following command to install this package using PowerShellGet More Info

```
PS> Install-Module -Name Rubrik
```

Author(s)
Chris Wahl

Copyright

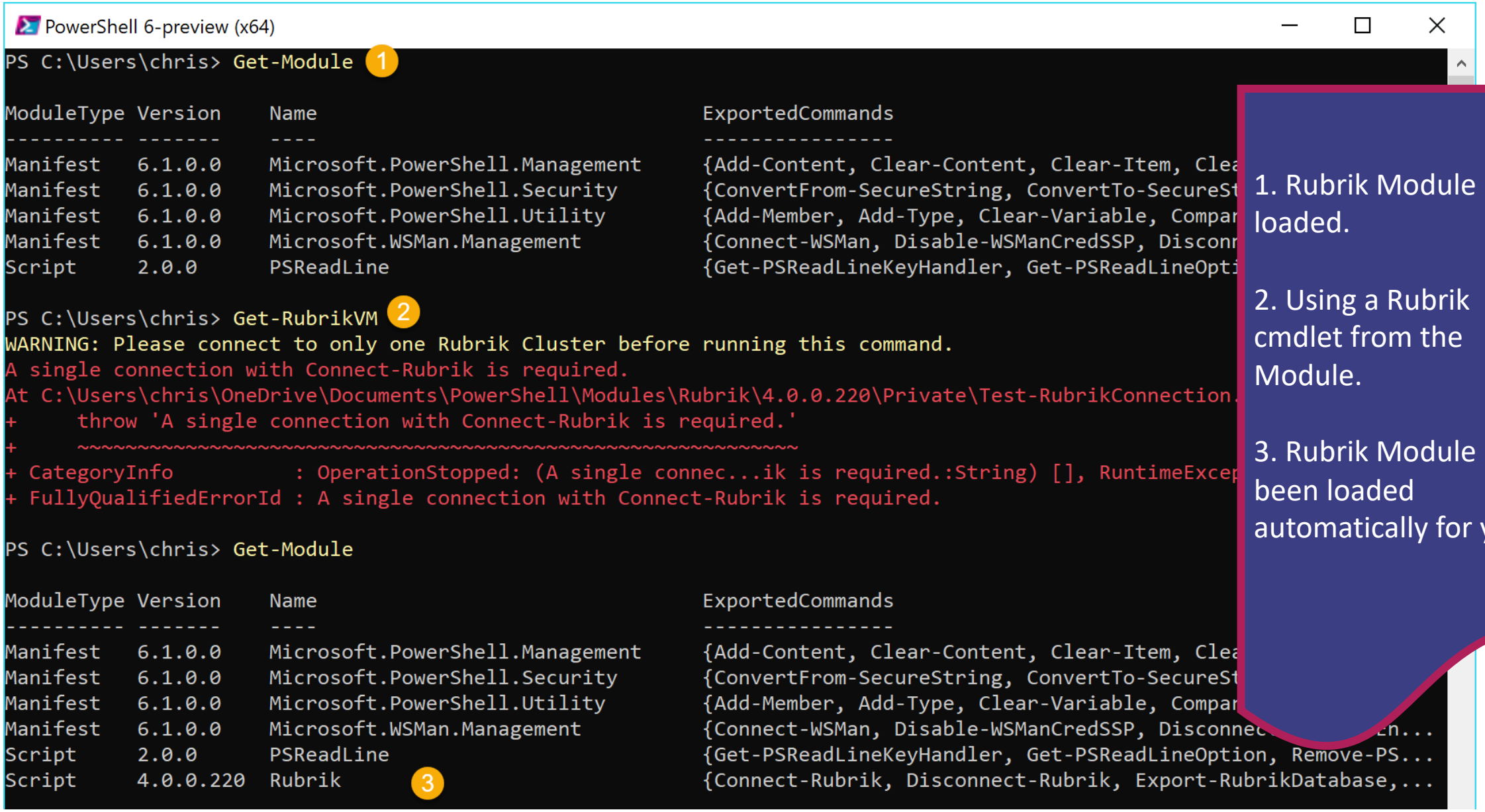
The Rubrik Module on the PowerShell Gallery

Cmdlets in a Module

A Module contains any number of custom created cmdlets that you can use in your environment.

Here are some of the Rubrik cmdlets that are available according to the PowerShell Gallery.

Connect-Rubrik Disconnect-Rubrik Export-RubrikDatabase Export-RubrikReport Get-RubrikAPIVersion Get-RubrikAvailabilityGroup Get-RubrikDatabase Get-RubrikDatabaseFiles Get-RubrikDatabaseMount Get-RubrikDatabaseRecoverableRange Get-RubrikFileset Get-RubrikFilesetTemplate Get-RubrikHost Get-RubrikHyperVVM Get-RubrikLDAP Get-RubrikLogShipping Get-RubrikManagedVolume Get-RubrikManagedVolumeExport Get-RubrikMount Get-RubrikNASShare Get-RubrikNutanixVM Get-RubrikOrganization Get-RubrikReport Get-RubrikReportData Get-RubrikRequest Get-RubrikSetting Get-RubrikSLA Get-RubrikSnapshot Get-RubrikSoftwareVersion Get-RubrikSQLInstance Get-RubrikSupportTunnel Get-RubrikUnmanagedObject Get-RubrikVCenter Get-RubrikVersion Get-RubrikVM Get-RubrikVMSnapshot Get-RubrikVolumeGroup Get-RubrikVolumeGroupMount Invoke-RubrikRESTCall Move-RubrikMountVMDK New-RubrikDatabaseMount New-RubrikFileset New-RubrikFilesetTemplate New-RubrikHost New-RubrikLDAP New-RubrikLogBackup New-RubrikLogShipping New-RubrikManagedVolume New-RubrikManagedVolumeExport New-RubrikMount New-RubrikNASShare New-RubrikReport New-RubrikSLA New-RubrikSnapshot New-RubrikVCenter New-RubrikVMDKMount New-RubrikVolumeGroupMount Protect-RubrikDatabase Protect-RubrikFileset Protect-RubrikHyperVVM Protect-RubrikNutanixVM Protect-RubrikTag Protect-RubrikVM Remove-RubrikDatabaseMount Remove-RubrikFileset Remove-RubrikHost Remove-RubrikLogShipping Remove-RubrikManagedVolume Remove-RubrikManagedVolumeExport Remove-RubrikMount Remove-RubrikNASShare Remove-RubrikReport Remove-RubrikSLA Remove-RubrikUnmanagedObject Remove-RubrikVCenter Remove-RubrikVolumeGroupMount Reset-RubrikLogShipping Restore-RubrikDatabase Set-RubrikAvailabilityGroup Set-RubrikBlackout Set-RubrikDatabase Set-RubrikHyperVVM Set-RubrikLogShipping Set-RubrikManagedVolume Set-RubrikMount Set-RubrikNASShare Set-



1. Rubrik Module not loaded.

2. Using a Rubrik cmdlet from the Module.

3. Rubrik Module has been loaded automatically for you.

Modules support Auto Loading!

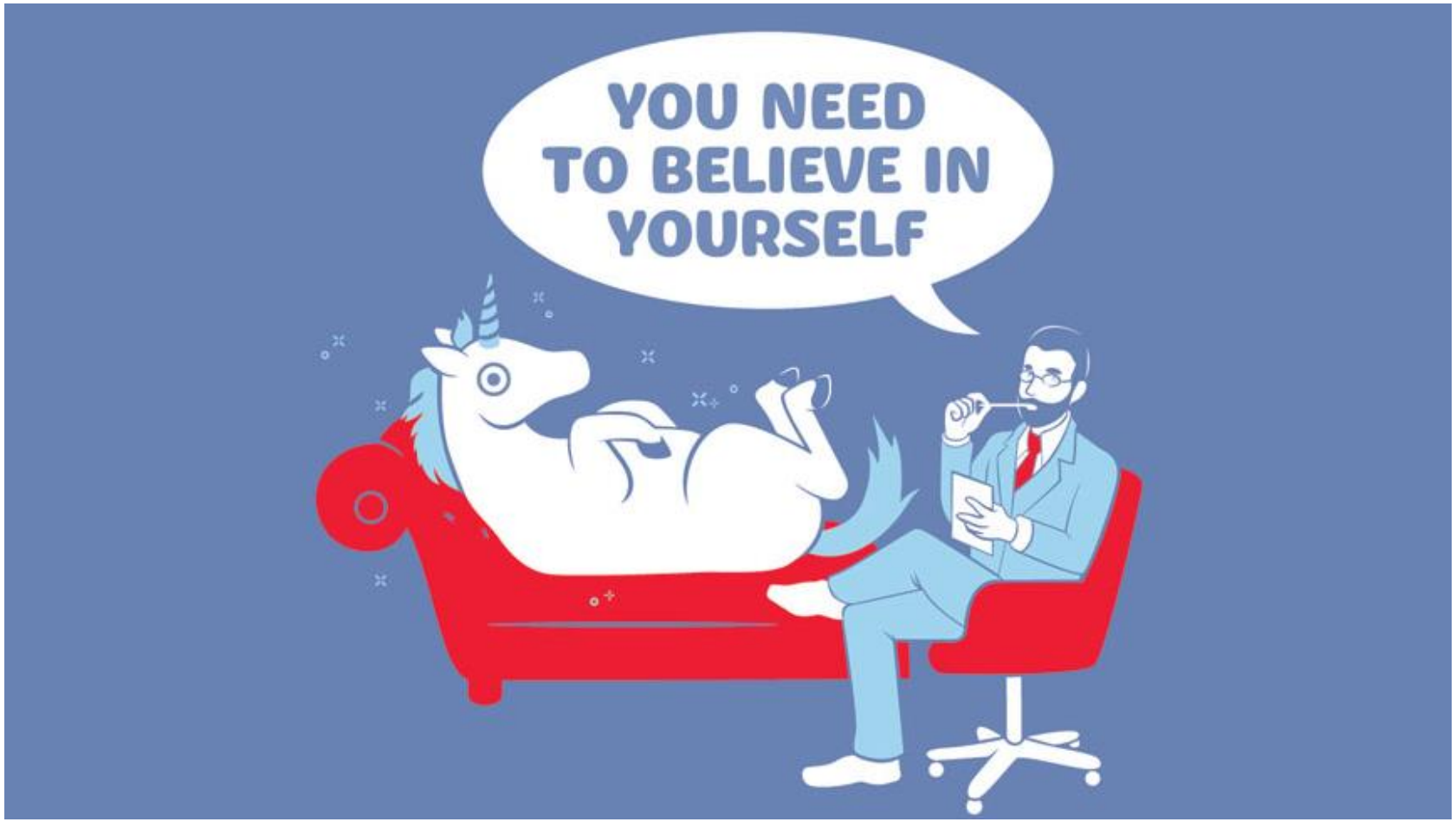
Rubrik SDK for PowerShell

What is the Rubrik SDK for PowerShell?

It’s a Module that is available as an open source project to anyone who wishes to interact with Rubrik.

Why did we create this SDK?

- Init in July 2015
  - Wanted to give sysadmins an easy way to control Rubrik.
  - Demonstrations for the lab for our Sales Engineering teams.
  - Cleanup scripts in the lab (remove erroneous SLA Domains, clean up live mounts).
  - Give customers solid examples on how to use PowerShell with an API.
  - VMware User Group meetups and events to teach PowerShell.
- Personal Challenge!
  - Figure out how to write a module for a vendor product.



Let’s get hands on with this Module!

Summary

Module Overview

- PowerShell Basics
  - Cmdlet, Parameters, Aliases, Getting Help
- Variables
- Pipeline
- Modules
- Rubrik SDK for PowerShell

build

Building the Future of Data Management