Infrastructure as Code- Walkthrough

Installing PowerShell 7.x

Contents

Last Updated: 24 November 2022

A **walkthrough** is intended to bring you through a technical exercise. A walkthrough shows you how I completed a task in a particular context, on specific systems, at a point in time. The document is a good basic guide, but you should always confirm that the information is current and if a more recent best practice exists, you should make yourself aware of it.

Introduction

This walkthrough is provided to give you some more progress with learning PowerShell. Version 5.1 still comes preloaded in all MS operating systems, but dates to 2016. The latest version is 7.2 at time of writing and it must be manually installed. We will use 7.2 in this walkthrough.

Deliverables

You will be required to keep a list of commands with notes as to why you are using them as well as results of any testing you have done. You may be asked to present these notes as evidence of work completed or as part of a lab book. Check with your lecturer!

Prerequisites

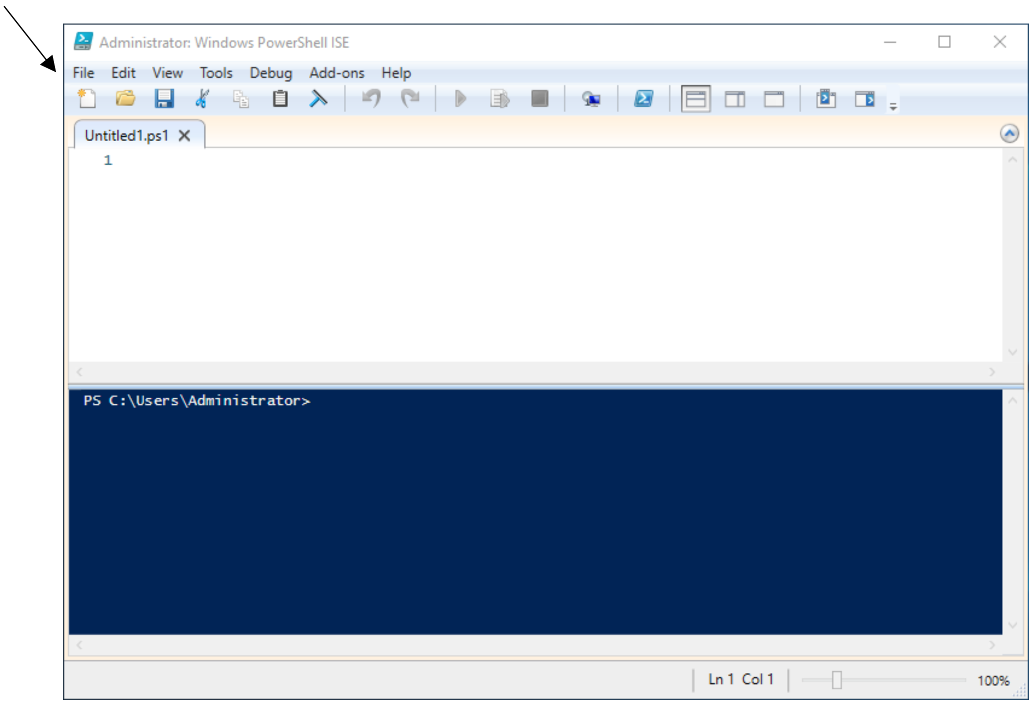
1. Ensure you have read the accompanying lecture notes before you begin.
2. I will carry out my exercises using a Windows 2019 VM Standard, Desktop Experience, running in Hyper-V.
3. You should have already installed Visual Studio Code as per my notes.
4. You have an existing GitHub account and a basic ability to use it.
5. You have created a directory to keep example files in and you are ready to code.
    a. On my VM, I am using **C:\Powershell**
    b. At the end of each session, I copy the files to **OneDrive\Powershell**

Running PowerShell 5.1

PowerShell 7.x does not come preinstalled; we need to download it. PowerShell 5.1 is built in living at **C:\Windows\System32\WindowsPowerShell**.

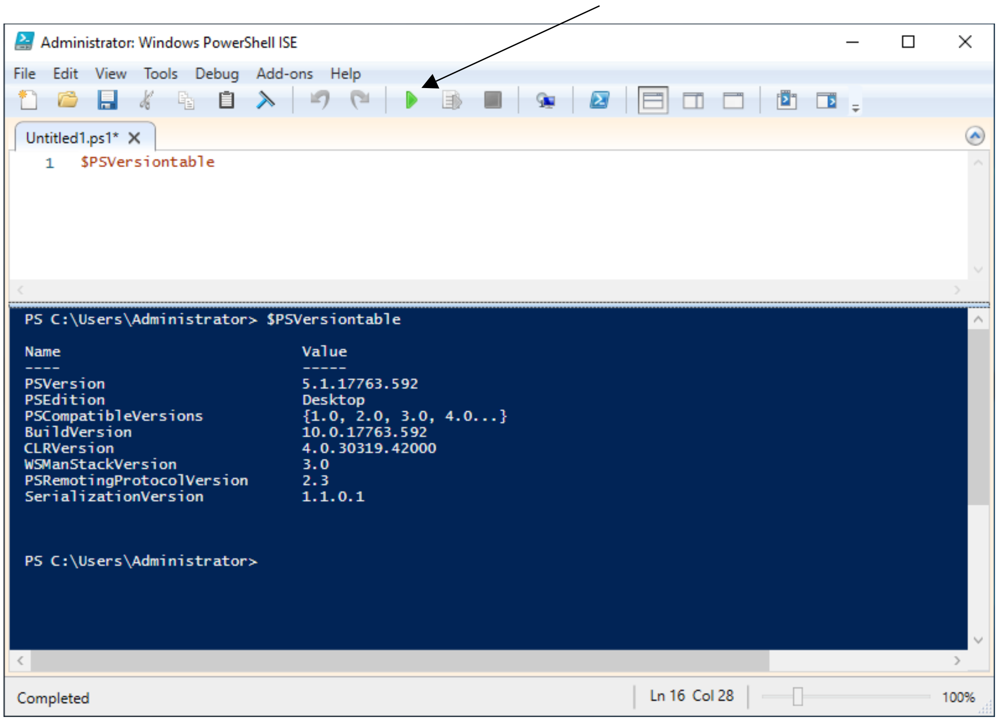
The original tool for working with PowerShell was the command prompt or an IDE called the ISE. In the search bar, type PowerShell and launch the Windows PowerShell ISE as administrator (right-click to do so).

In View, I click to show the script pane.


Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1
Ln 1 Col 1
100%

```
PS C:\Users\Administrator>
```

If I type **$PSVersiontable** into the script window and click **Run**, I can check the version.


Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1*
Completed
Ln 16 Col 28
100%

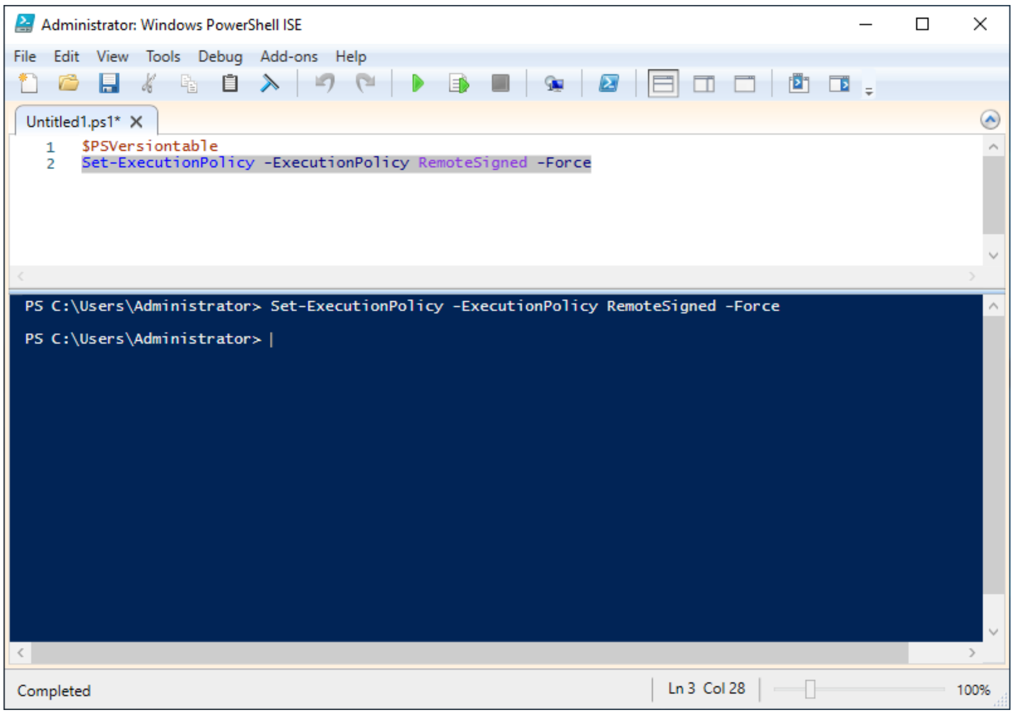
```
$PSVersiontable
```

```
PS C:\Users\Administrator> $PSVersiontable

Name                           Value
----                           -----
PSVersion                      5.1.17763.592
PSEdition                      Desktop
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
BuildVersion                   10.0.17763.592
CLRVersion                     4.0.30319.42000
WSManStackVersion              3.0
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1


PS C:\Users\Administrator>
```

Execution Policy

By default, PowerShell will not let me run scripts, even though I am administrator.

I must manually set the execution policy. If I highlight a command, I can click **Run Selection** to run just that command.


Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1*
Completed
Ln 3 Col 28
100%

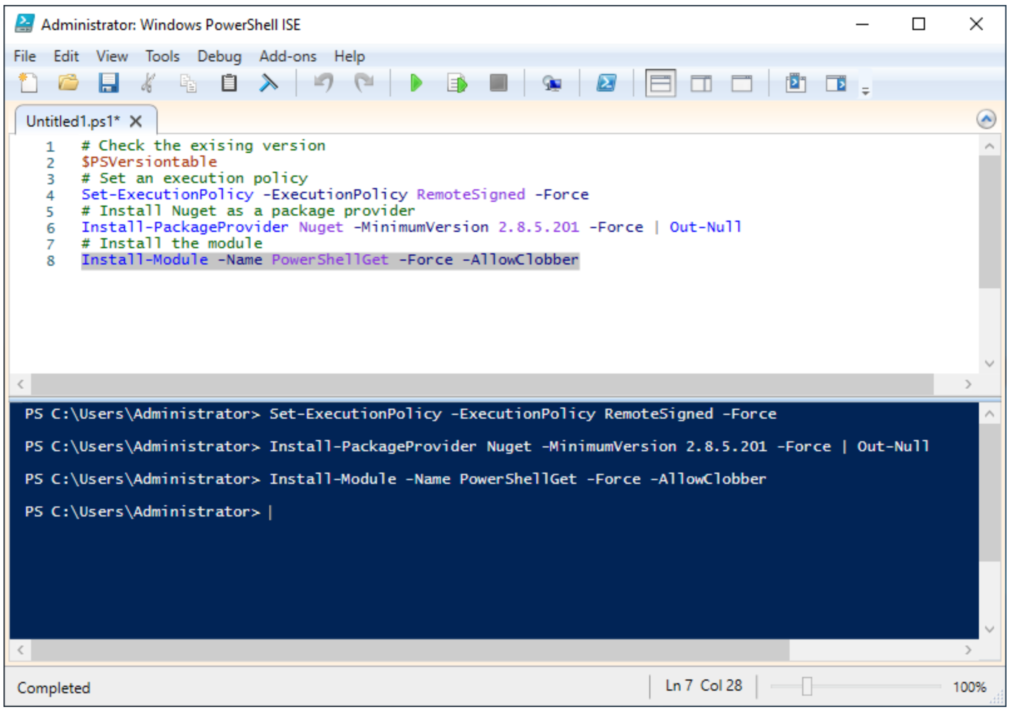
```
$PSVersiontable
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Force
```

```
PS C:\Users\Administrator> Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Force

PS C:\Users\Administrator> |
```

PowerShell Gallery

As with most code platforms these days, there is a repo!

The PowerShell Gallery [1] is this repo and you can download modules using PowerShellGet [2]. Read both references before you continue. To work with this gallery, you need some underlying tools and Nuget [3] is the package manager for .Net. I have started adding comments!


Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled1.ps1*

```
# Check the exising version
$PSVersiontable
# Set an execution policy
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Force
# Install Nuget as a package provider
Install-PackageProvider Nuget -MinimumVersion 2.8.5.201 -Force | Out-Null
# Install the module
Install-Module -Name PowerShellGet -Force -AllowClobber
```

```
PS C:\Users\Administrator> Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Force

PS C:\Users\Administrator> Install-PackageProvider Nuget -MinimumVersion 2.8.5.201 -Force | Out-Null

PS C:\Users\Administrator> Install-Module -Name PowerShellGet -Force -AllowClobber

PS C:\Users\Administrator> |
```

Script Folder

As we begin coding in PowerShell, keep a tab open to the documentation [4].

I am working on a VM, so I create a folder called C:\PowerShell. I will copy these scripts to my OneDrive whenever I finish an exercise. I add the line **mkdir c:\PowerShell** to my script and run it.

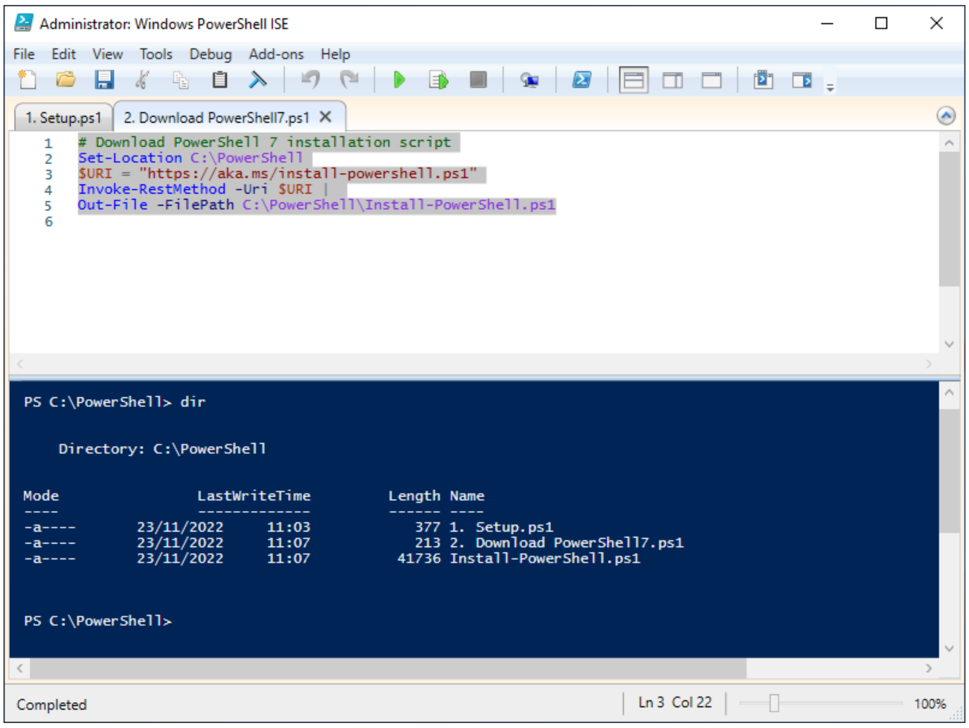
I save this script as **1. Setup.ps1**

Next, I open a new script window and save it as **2. Download PowerShell7.ps1**

Microsoft have a generic script they make available, I past this into the new script window and save. I run the script, no errors.

```
# Download PowerShell 7 installation script
Set-Location C:\PowerShell
$URI = "https://aka.ms/install-powershell.ps1"
Invoke-RestMethod -Uri $URI |
Out-File -FilePath C:\PowerShell\Install-PowerShell.ps1
```

The command **dir** shows me a new script, **Install-PowerShell.ps1**


Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
1. Setup.ps1
2. Download PowerShell7.ps1
Completed
Ln 3 Col 22
100%

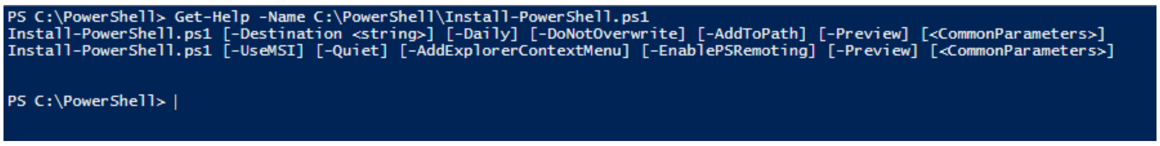
```
# Download PowerShell 7 installation script
Set-Location C:\PowerShell
$URI = "https://aka.ms/install-powershell.ps1"
Invoke-RestMethod -Uri $URI |
Out-File -FilePath C:\PowerShell\Install-PowerShell.ps1
```

```
PS C:\PowerShell> dir


    Directory: C:\PowerShell


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        23/11/2022     11:03            377 1. Setup.ps1
-a----        23/11/2022     11:07            213 2. Download PowerShell7.ps1
-a----        23/11/2022     11:07          41736 Install-PowerShell.ps1


PS C:\PowerShell>
```

I save the script and close it.

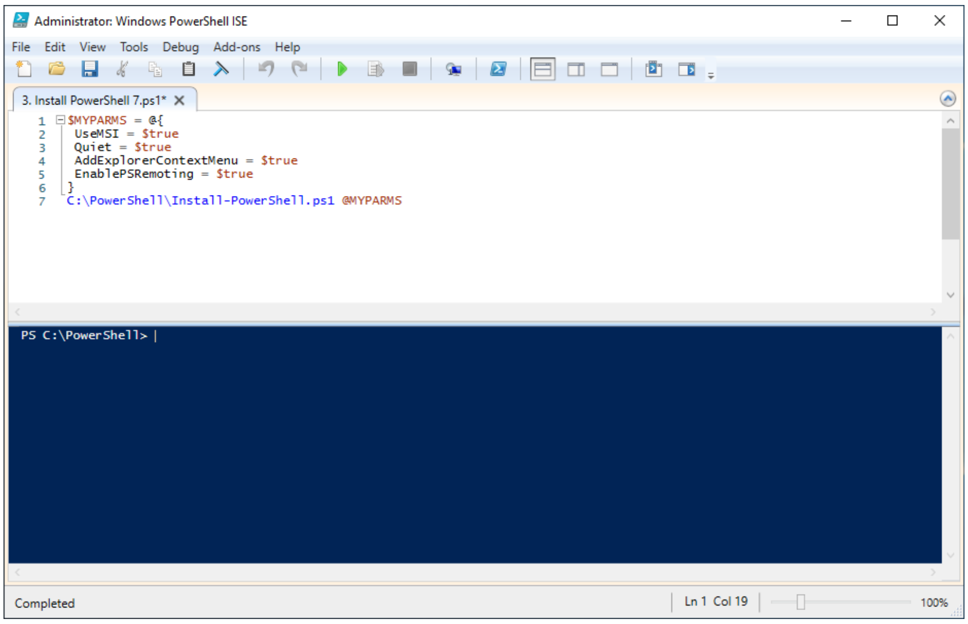
Install PowerShell 7

If I run the command **Get-Help -Name C:\PowerShell\Install-PowerShell.ps1** I can see how the script can be used. My computer added some modules automatically when I used **Get-Help**.



```
PS C:\PowerShell> Get-Help -Name C:\PowerShell\Install-PowerShell.ps1
Install-PowerShell.ps1 [-Destination <string>] [-Daily] [-DoNotOverwrite] [-AddToPath] [-Preview] [<CommonParameters>]
Install-PowerShell.ps1 [-UseMSI] [-Quiet] [-AddExplorerContextMenu] [-EnablePSRemoting] [-Preview] [<CommonParameters>]


PS C:\PowerShell> |
```

I could run a very long command line, but there is a more elegant way to do this in PowerShell. I create a new script as **3. Install PowerShell 7** and set up the parameters in a variable [5] and then pass then using ***@variable_name***.


Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
3. Install PowerShell 7.ps1*
Completed
Ln 1 Col 19
100%

```
$MYPARMS = @{
 UseMSI = $true
 Quiet = $true
 AddExplorerContextMenu = $true
 EnablePSRemoting = $true
}
C:\PowerShell\Install-PowerShell.ps1 @MYPARMS
```

```
PS C:\PowerShell> |
```

When I run the script, I note that everything downloads from GitHub!

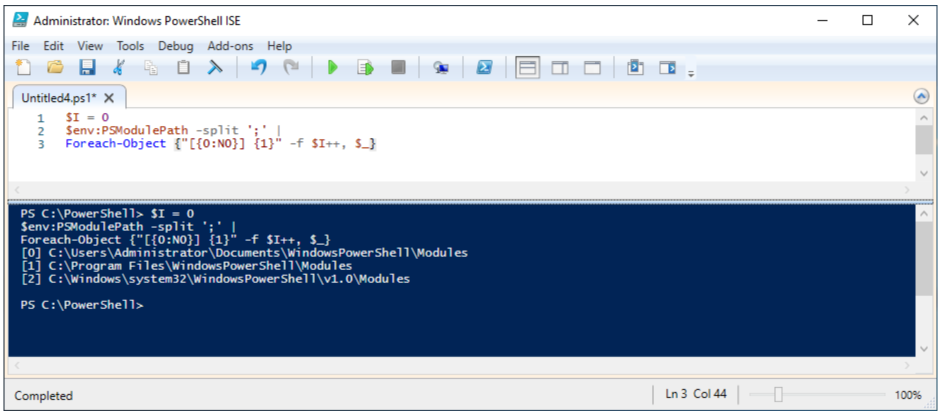
Verifying Installation

The commands in PowerShell are sorted into modules. You can manually import modules of use autoload. If a module is called which has not been installed, PowerShell will fetch it.

In PowerShell, we define variables using a $ symbol. Try the command **echo $env:PSModulePath**

The output is three separate paths, concatenated with a semicolon field separator.

I create a new script window 4**. Verify PowerShell 7**


Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Untitled4.ps1*

```
$I = 0
$env:PSModulePath -split ';' |
Foreach-Object {"[{0:N0}] {1}" -f $I++, $_}
```

```
PS C:\PowerShell> $I = 0
$env:PSModulePath -split ';' |
Foreach-Object {"[{0:N0}] {1}" -f $I++, $_}
[0] C:\Users\Administrator\Documents\WindowsPowerShell\Modules
[1] C:\Program Files\WindowsPowerShell\Modules
[2] C:\Windows\system32\WindowsPowerShell\v1.0\Modules

PS C:\PowerShell>
```

Completed
Ln 3 Col 44
100%

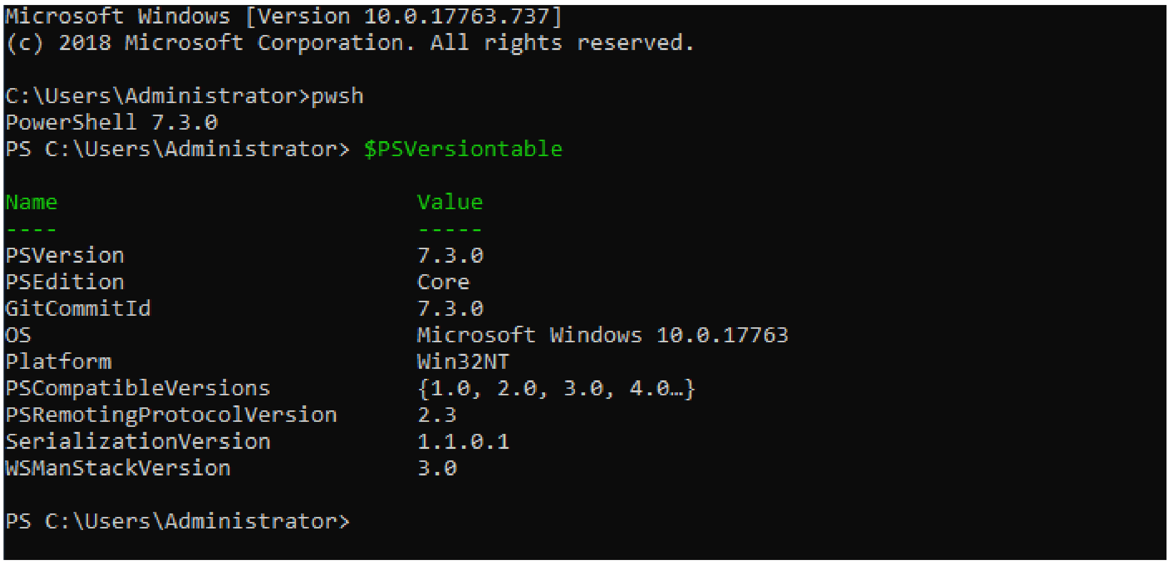
What is this script doing?

First it sets the variable $I = 0,

Then it uses a split function to break up and pipes the output using the **|** symbol to a for-each loop. The loop syntax will make more sense when we cover loops. But now we know the locations of the modules PowerShell uses.

Close ISE, we will not be using it again for now.

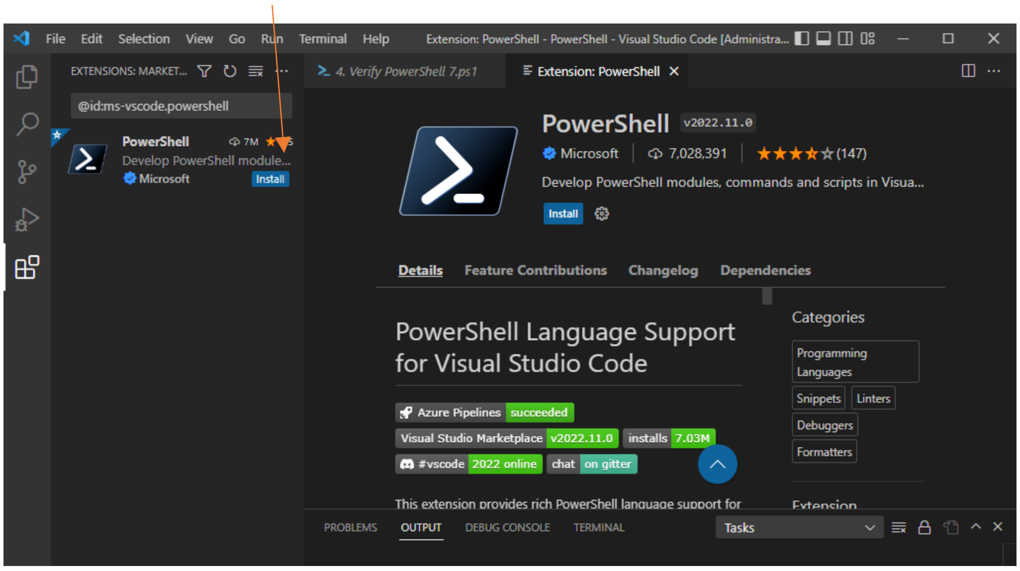
Open a command window and run PowerShell 7 by typing **pwsh**, then verify the version as we did before.



```
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>pwsh
PowerShell 7.3.0
PS C:\Users\Administrator> $PSVersiontable

Name                           Value
----                           -----
PSVersion                      7.3.0
PSEdition                      Core
GitCommitId                    7.3.0
OS                             Microsoft Windows 10.0.17763
Platform                       Win32NT
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0…}
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1
WSManStackVersion              3.0

PS C:\Users\Administrator>
```

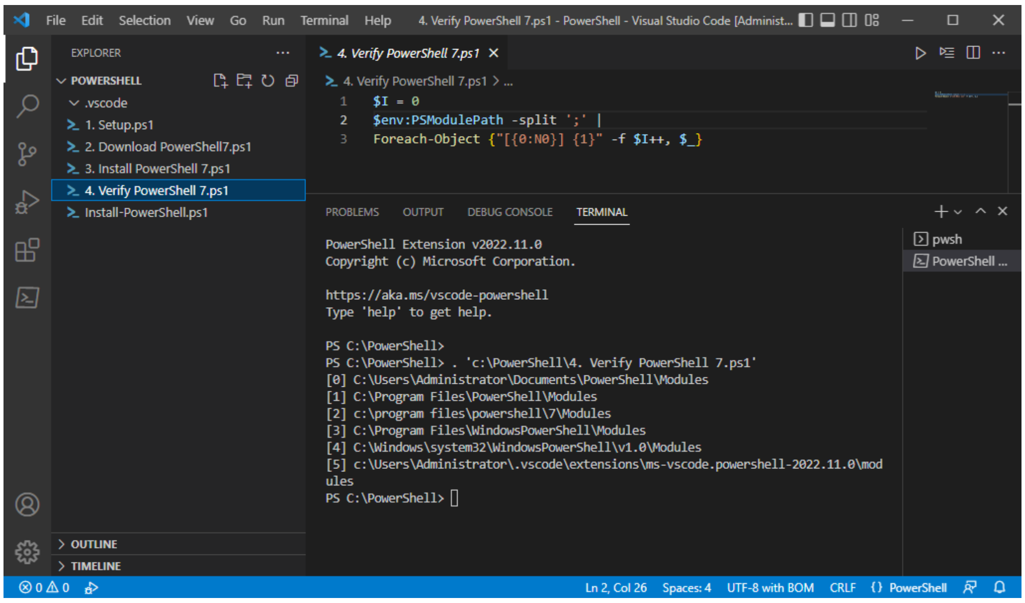
Visual Studio Code (VSC)

I download and install VSCode and when I open the c:\PowerShell folder, VSC recommends the PowerShell 7 extension. I install this.


Extension: PowerShell - PowerShell - Visual Studio Code [Administra...
EXTENSIONS: MARKET...
@id:ms-vscode.powershell
PowerShell
Develop PowerShell module...
Microsoft
Install
4. Verify PowerShell 7.ps1
Extension: PowerShell
PowerShell v2022.11.0
Microsoft | 7,028,391 | (147)
Develop PowerShell modules, commands and scripts in Visua...
Install
Details Feature Contributions Changelog Dependencies
PowerShell Language Support for Visual Studio Code
Azure Pipelines succeeded
Visual Studio Marketplace v2022.11.0 installs 7.03M
#vscode 2022 online chat on gitter
This extension provides rich PowerShell language support for
Categories
Programming Languages
Snippets Linters
Debuggers
Formatters
Extension
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Tasks

VSC offers to install PowerShell 7, we have already done this so I decline!

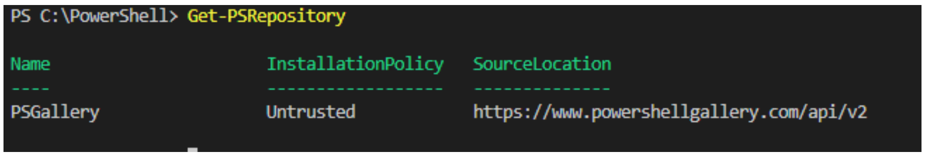
I load up script 4 from earlier and run it to test. VSC has made some changes, but everything works.


4. Verify PowerShell 7.ps1 - PowerShell - Visual Studio Code [Administ...
EXPLORER
POWERSHELL
.vscode
1. Setup.ps1
2. Download PowerShell7.ps1
3. Install PowerShell 7.ps1
4. Verify PowerShell 7.ps1
Install-PowerShell.ps1
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
pwsh
PowerShell ...
Ln 2, Col 26 Spaces: 4 UTF-8 with BOM CRLF PowerShell

```
$I = 0
$env:PSModulePath -split ';' |
Foreach-Object {"[{0:N0}] {1}" -f $I++, $_}
```

```
PowerShell Extension v2022.11.0
Copyright (c) Microsoft Corporation.

https://aka.ms/vscode-powershell
Type 'help' to get help.

PS C:\PowerShell>
PS C:\PowerShell> . 'c:\PowerShell\4. Verify PowerShell 7.ps1'
[0] C:\Users\Administrator\Documents\PowerShell\Modules
[1] C:\Program Files\PowerShell\Modules
[2] c:\program files\powershell\7\Modules
[3] C:\Program Files\WindowsPowerShell\Modules
[4] C:\Windows\system32\WindowsPowerShell\v1.0\Modules
[5] c:\Users\Administrator\.vscode\extensions\ms-vscode.powershell-2022.11.0\mod
ules
PS C:\PowerShell>
```

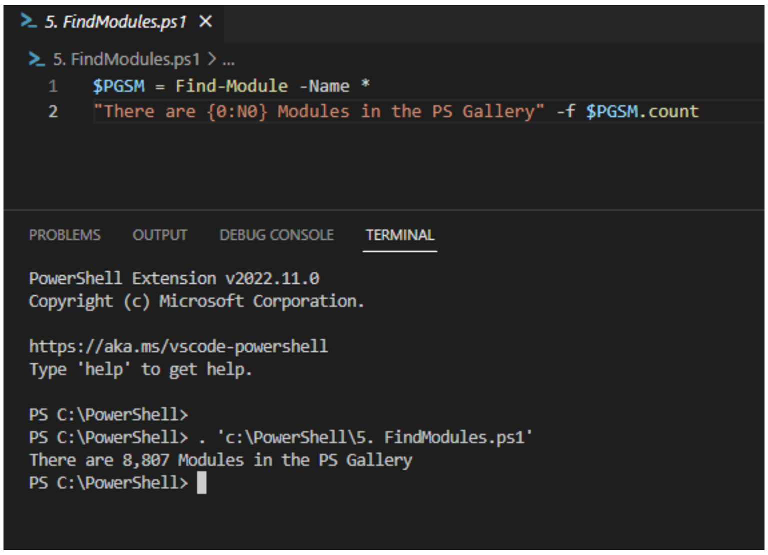
In the immediate window, I can check which repository PowerShell and VSC will use.



```
PS C:\PowerShell> Get-PSRepository

Name                      InstallationPolicy   SourceLocation
----                      ------------------   --------------
PSGallery                 Untrusted            https://www.powershellgallery.com/api/v2
```

A few more checks

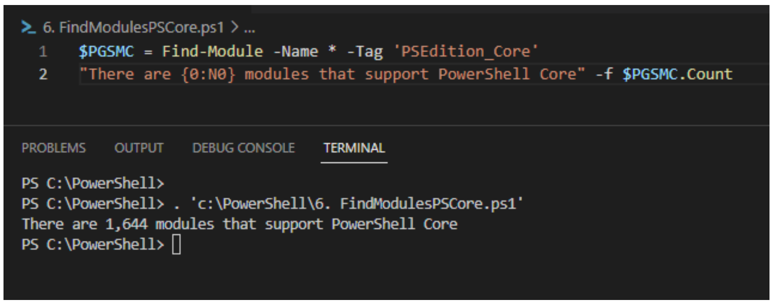
I wonder how many modules are available from the PS Gallery? I created **5. FindModules.ps1** and it took a while to run!



```
$PGSM = Find-Module -Name *
"There are {0:N0} Modules in the PS Gallery" -f $PGSM.count
```

```
PowerShell Extension v2022.11.0
Copyright (c) Microsoft Corporation.

https://aka.ms/vscode-powershell
Type 'help' to get help.

PS C:\PowerShell>
PS C:\PowerShell> . 'c:\PowerShell\5. FindModules.ps1'
There are 8,807 Modules in the PS Gallery
PS C:\PowerShell>
```

How many of these modules support PSCore? I created **6. FindModulesPSCore.ps1**



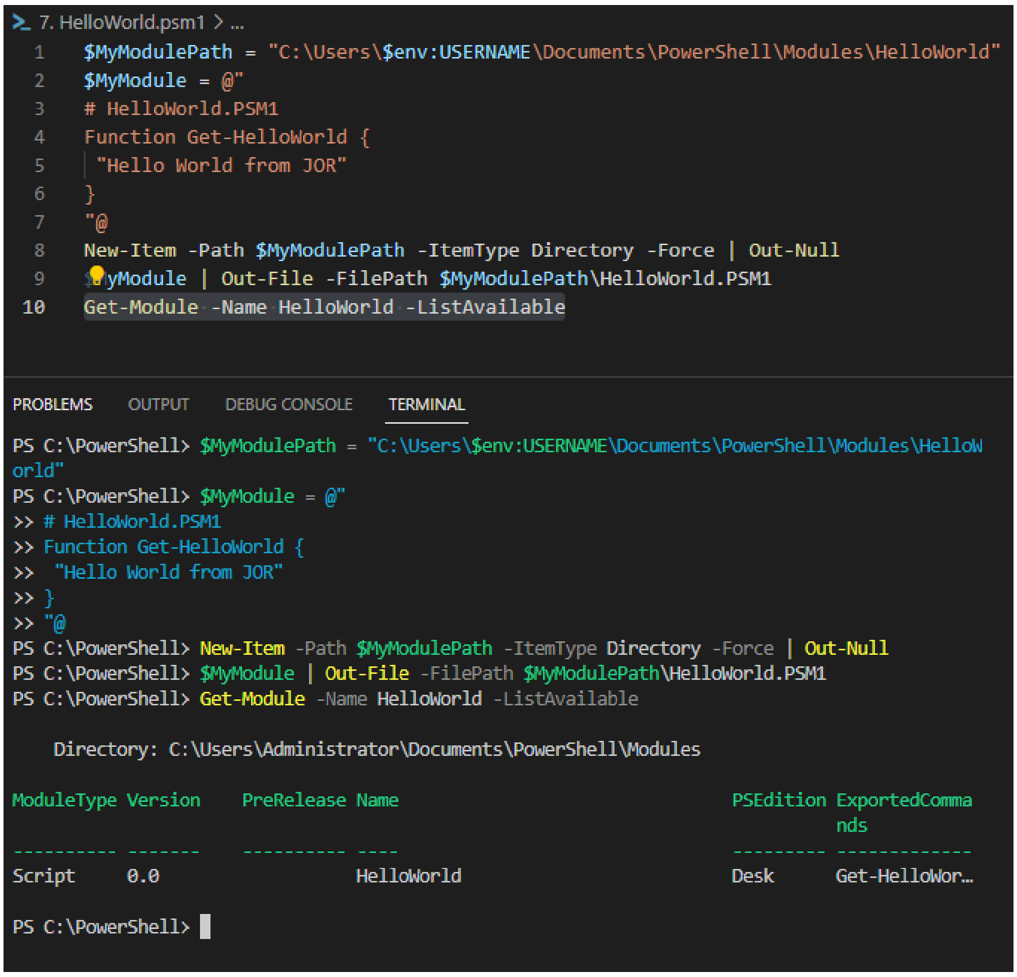
```
$PGSMC = Find-Module -Name * -Tag 'PSEdition_Core'
"There are {0:N0} modules that support PowerShell Core" -f $PGSMC.Count
```

```
PS C:\PowerShell>
PS C:\PowerShell> . 'c:\PowerShell\6. FindModulesPSCore.ps1'
There are 1,644 modules that support PowerShell Core
PS C:\PowerShell>
```

The command **get-module -ListAvailable** will give you some idea of the total modules available to you.

Creating a Module

I can create a simple module.

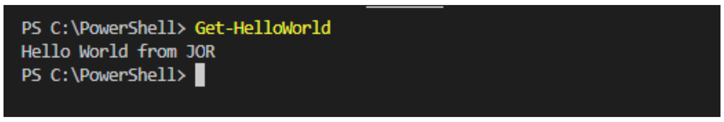


```
7. HelloWorld.psm1 > ...
$MyModulePath = "C:\Users\$env:USERNAME\Documents\PowerShell\Modules\HelloWorld"
$MyModule = @"
# HelloWorld.PSM1
Function Get-HelloWorld {
  "Hello World from JOR"
}
"@
New-Item -Path $MyModulePath -ItemType Directory -Force | Out-Null
$MyModule | Out-File -FilePath $MyModulePath\HelloWorld.PSM1
Get-Module -Name HelloWorld -ListAvailable
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\PowerShell> $MyModulePath = "C:\Users\$env:USERNAME\Documents\PowerShell\Modules\HelloW
orld"
PS C:\PowerShell> $MyModule = @"
>> # HelloWorld.PSM1
>> Function Get-HelloWorld {
>>  "Hello World from JOR"
>> }
>> "@
PS C:\PowerShell> New-Item -Path $MyModulePath -ItemType Directory -Force | Out-Null
PS C:\PowerShell> $MyModule | Out-File -FilePath $MyModulePath\HelloWorld.PSM1
PS C:\PowerShell> Get-Module -Name HelloWorld -ListAvailable

    Directory: C:\Users\Administrator\Documents\PowerShell\Modules

ModuleType Version    PreRelease Name                                PSEdition ExportedComma
                                                                               nds
---------- -------    ---------- ----                                --------- -------------
Script     0.0                   HelloWorld                          Desk      Get-HelloWor…

PS C:\PowerShell>
```

And now I can run it!



```
PS C:\PowerShell> Get-HelloWorld
Hello World from JOR
PS C:\PowerShell>
```

If I check the path C:\Users\Administrator\Documents\PowerShell\Modules\HelloWorld I see a new directory and file.

Finally

Now you have run

- PowerShell 5.1 using the terminal and ISE
- PowerShell 7.2 using the terminal and VSCode

This is a starting point!

Save your scripts to your OneDrive.

Set up a GitHub repo called PowerShell and make your PowerShell folder a Git repo. Sync your scripts up to GitHub and verify they are there.