# PowerShell

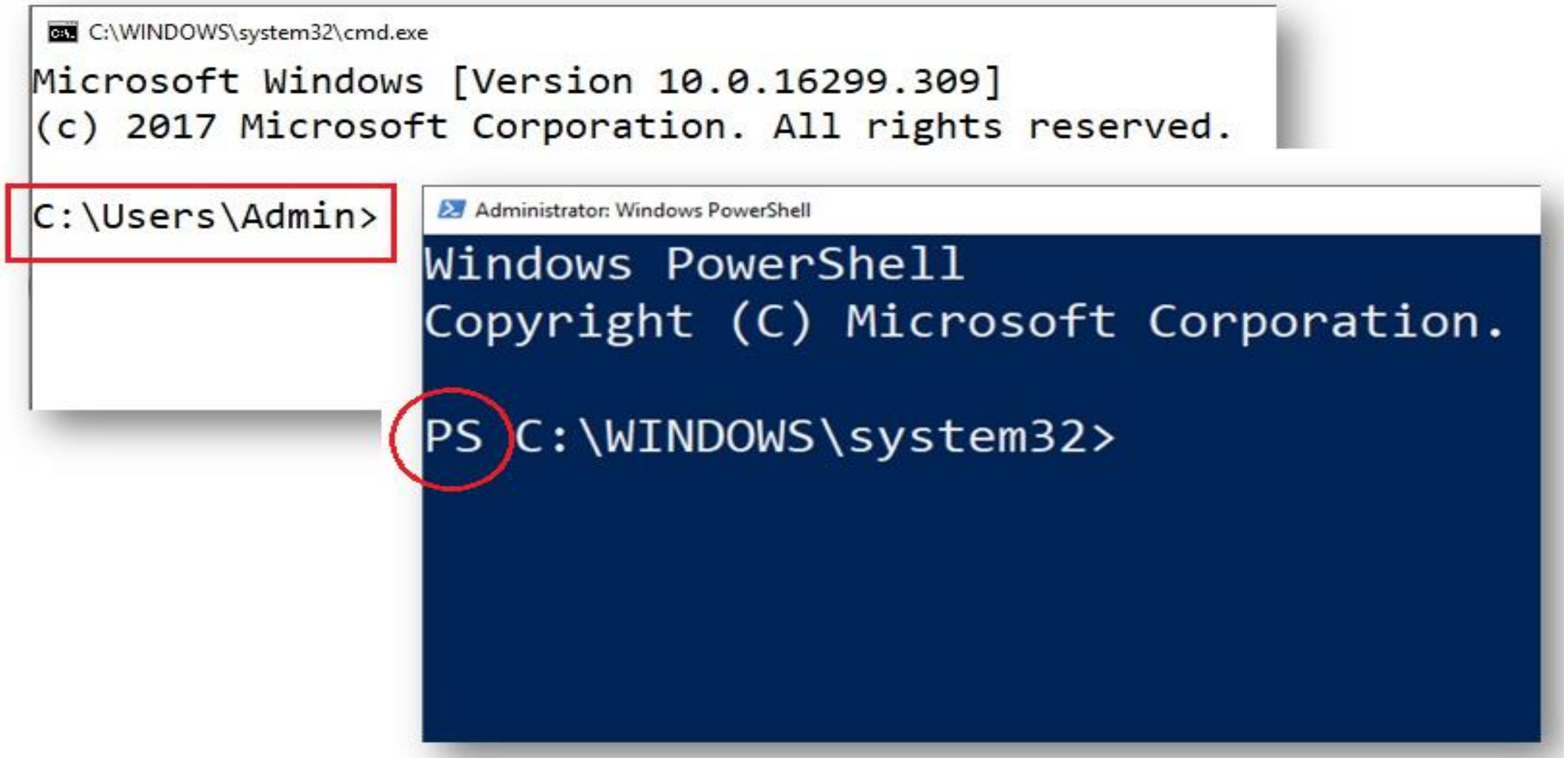
Trainer – Jitendra Singh Tomar

Microsoft

# What is PowerShell?

- PowerShell is a task automation and configuration management framework from Microsoft, consisting of a command-line shell and associated scripting language.
- Windows Command Shell
- Accepts Command from command line - **cmdlets**
- You can navigate the file system and the registry
- You can run scripts
- You can run script directly from the command line

# CMD vs PS


C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.16299.309]
(c) 2017 Microsoft Corporation. All rights reserved.
C:\Users\Admin>
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation.
PS C:\WINDOWS\system32>

# PowerShell Version

- **PowerShell v1** (PS 1.0)
  - In Nov, 2006 for XP, WinServer 2003 SP1, Windows Vista.
  - Optional in Win Server 2008.
- **PowerShell v2** (PS 2.0)
  - Integrated with Win7 & Win SVR 2008R2.
  - Included 240 cmdlets.
  - Includes below features:
    - PS remoting
    - Jobs, Advance functions
    - Debugging, ISE.
    - New APIs.

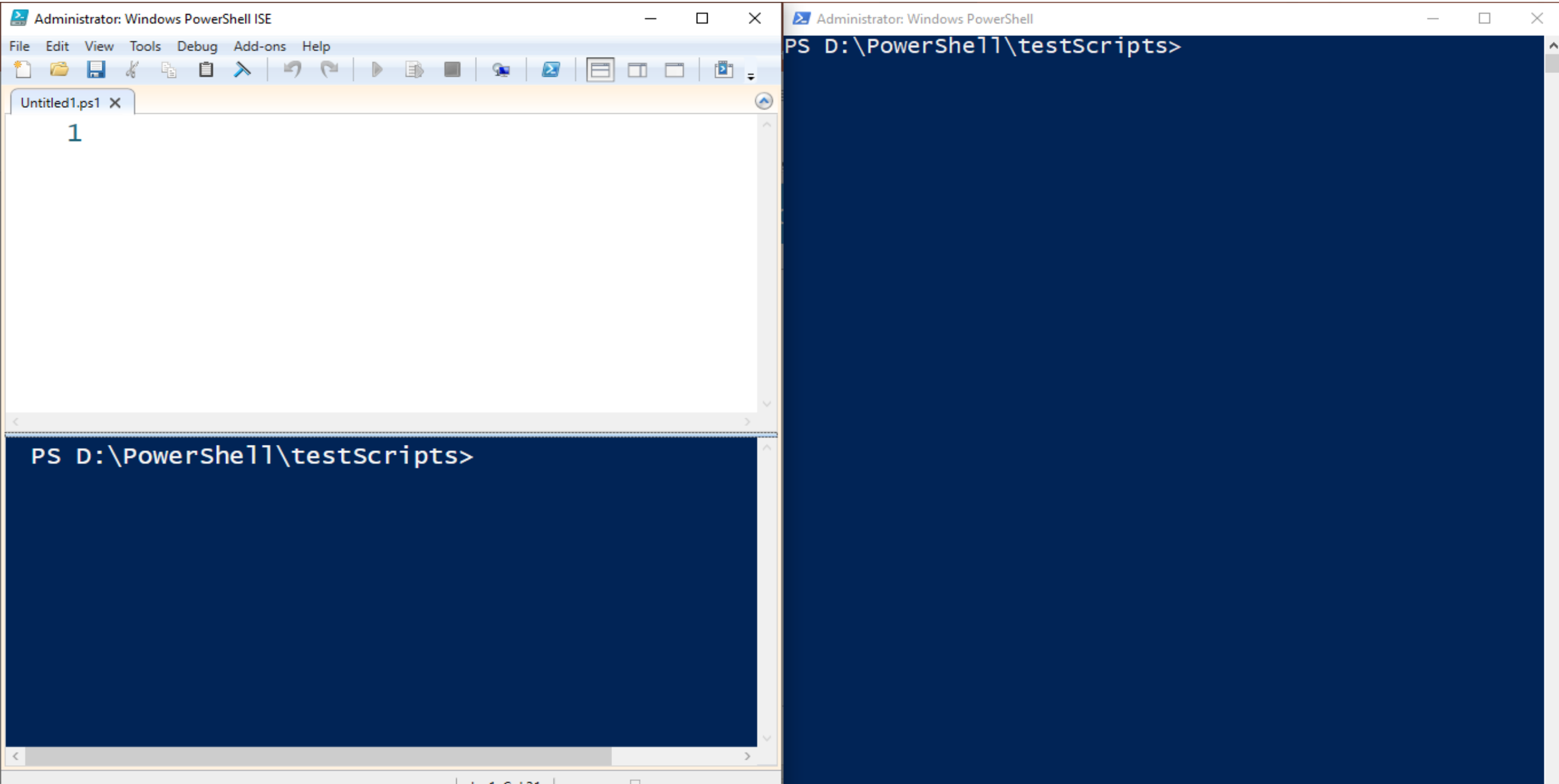
# PowerShell Version

- **PowerShell v3** (PS 3.0)
  - Integrated with Win 8 & Win SVR 2012.
  - Supports WinRM service. (5985/5986)
  - Job scheduling, session connectivity, help update, auto module detection.
- **PowerShell v4** (PS 4.0)
  - Integrated with Win 8.1 & Win SVR 2012R2.
  - Includes DSC, execution policy, save-help, ‘where’ & ‘foreach’.
- **PowerShell v5** (PS 5.0 or PS 5.1)
  - Integrated with Win10 & Win SVR 2016.
  - Includes Chocolatey's repository-based package management.

# PowerShell Version

- **PowerShell 5.1**
  - It was released along with the Windows 10 Anniversary Update.
  - PowerShell 5.1 is the first version to come in two editions of "Desktop" and "Core".
    - "*Desktop*" edition is the continuation of the traditional Windows PowerShell
    - "*Core*" edition runs on .NET Core and is bundled with Windows Server 2016 Nano Server
- **PowerShell Core 6.0**
  - Cross-platform, independent of Windows, free and open source.
  - It's a new edition of PS that is cross-platform (Windows, macOS, and Linux), open-source, and built for heterogeneous environments and the hybrid cloud.
  - Supports MacOS & Linux.
  - Setup renamed from "powershell.exe" to "pwsh.exe"

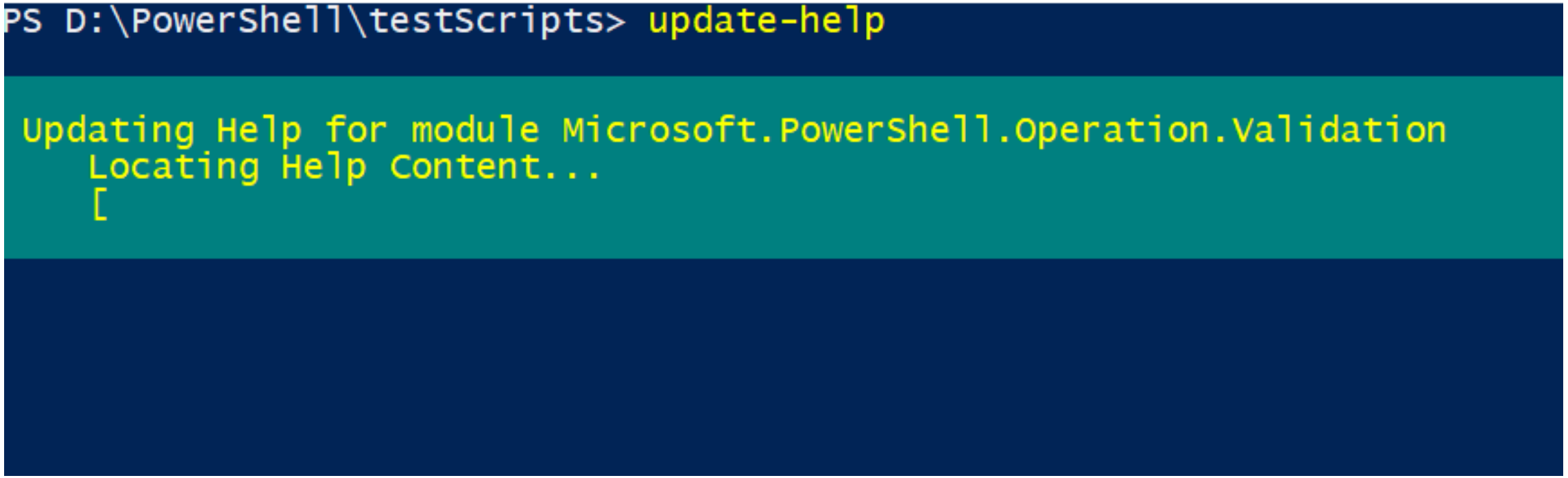
# PowerShell vs Integrated Scripting Environment (ISE)



# File extensions

1. **PS1** – Windows PowerShell script
2. **PSD1** – Windows PowerShell data file (for Version 2)
3. **PSM1** – Windows PowerShell module file (for Version 2)
4. **PS1XML** – Windows PowerShell format and type definitions
5. **CLIXML** – Windows PowerShell serialized data
6. **PSC1** – Windows PowerShell console file
7. **PSSC** – Windows PowerShell Session Configuration file

# Updating PowerShell



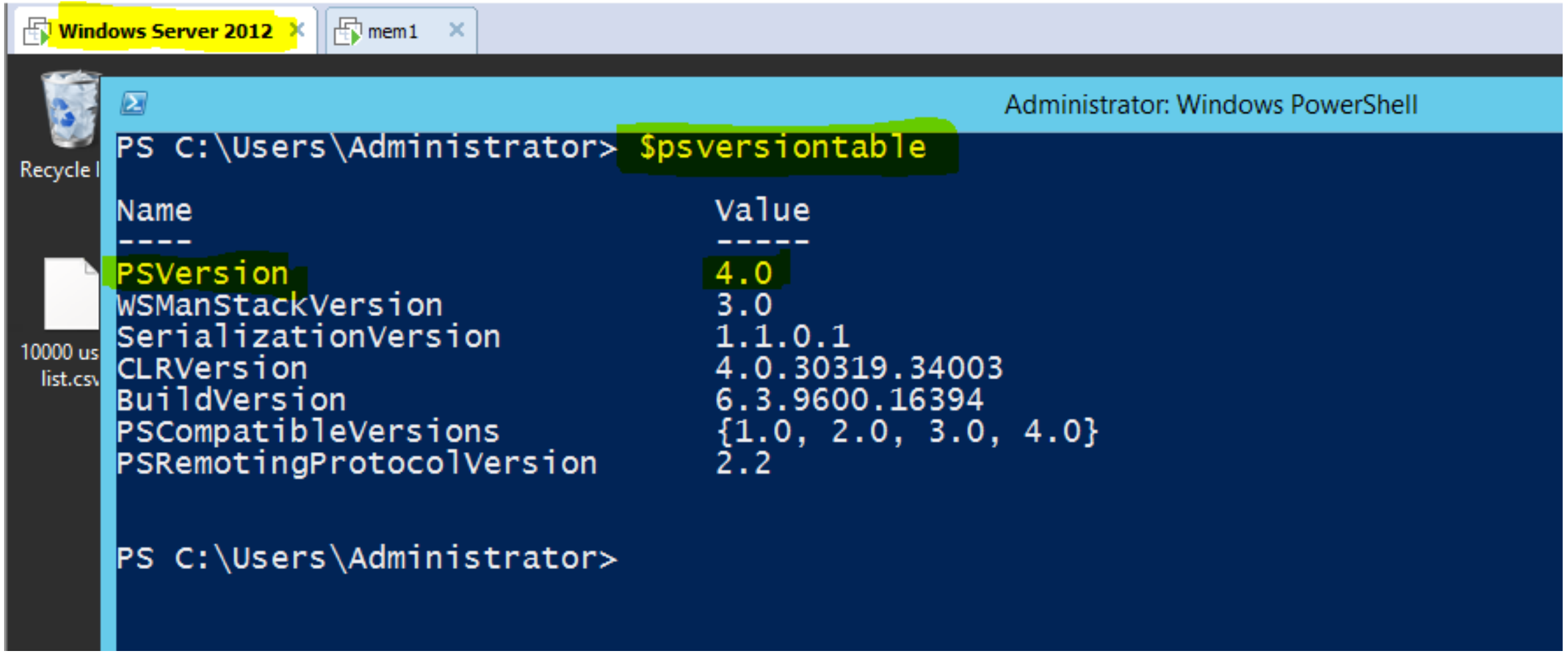
# Help command

```
PS D:\PowerShell\testScripts> help update-help

NAME
    Update-Help

SYNOPSIS
    Downloads and installs the newest help files on your computer.


SYNTAX
    Update-Help [[-Module] <String[]>] [[-UICulture] <CultureInfo[]>] [-Con
    [-Force] [-FullyQualifiedModule <ModuleSpecification[]>] [-LiteralPath
    [-UseDefaultCredentials] [-WhatIf] [<CommonParameters>]

    Update-Help [[-Module] <String[]>] [[-SourcePath] <String[]>] [[-UICult
    [-Credential <PSCredential>] [-Force] [-FullyQualifiedModule <ModuleSpe
    [-UseDefaultCredentials] [-WhatIf] [<CommonParameters>]


DESCRIPTION
    The Update-Help cmdlet downloads the newest help files for Windows Powe
    your computer. You can use the Get-Help cmdlet to view the new help fil
```

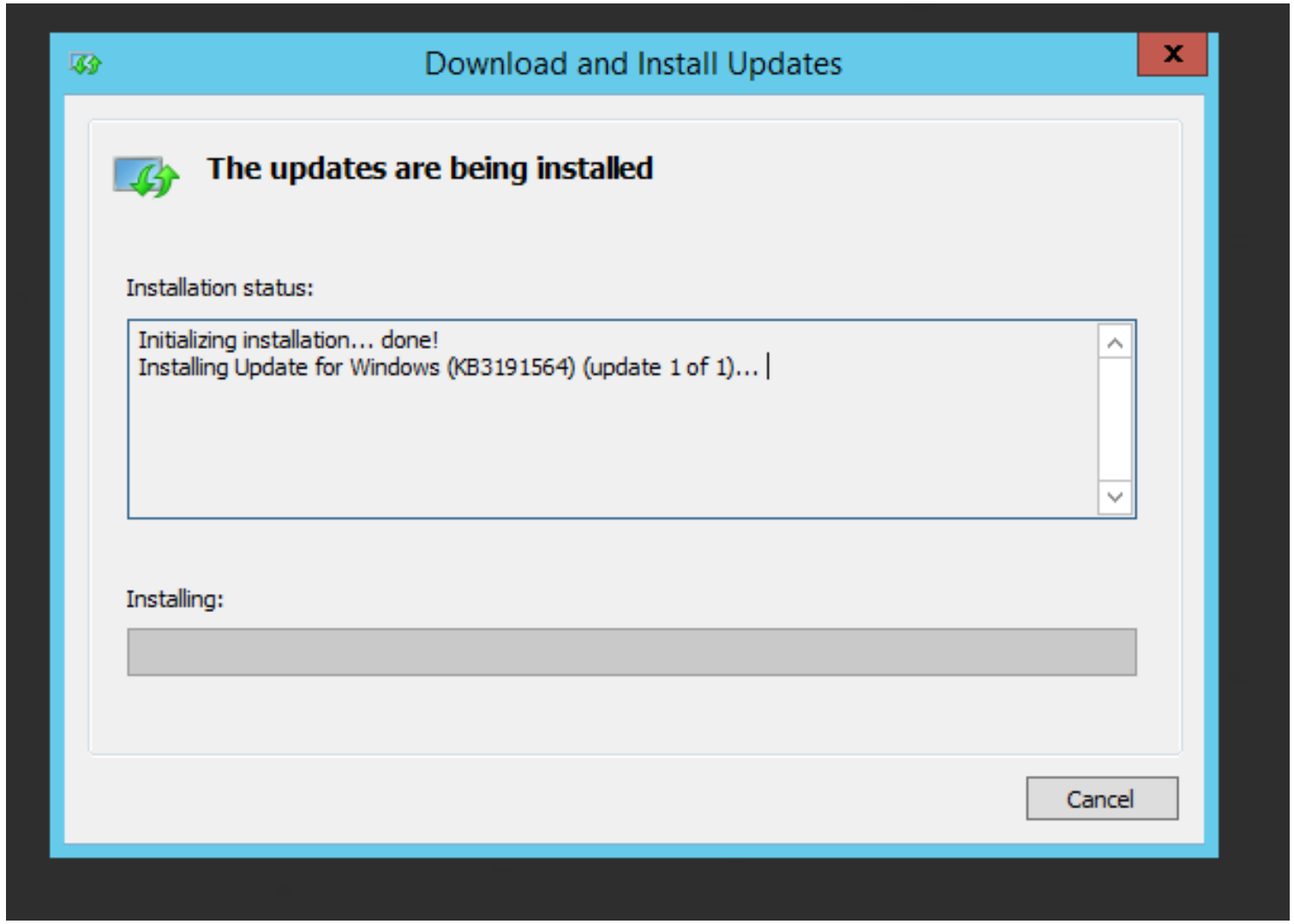
# Upgrade PowerShell

- V2 -> V4 (DotNet FW 4.0 required)
- V3 -> V4/V5 (DotNet FW 4.0 required, 4.5.0 required)
- V4 -> V5 (DotNet FW 4.5 required)

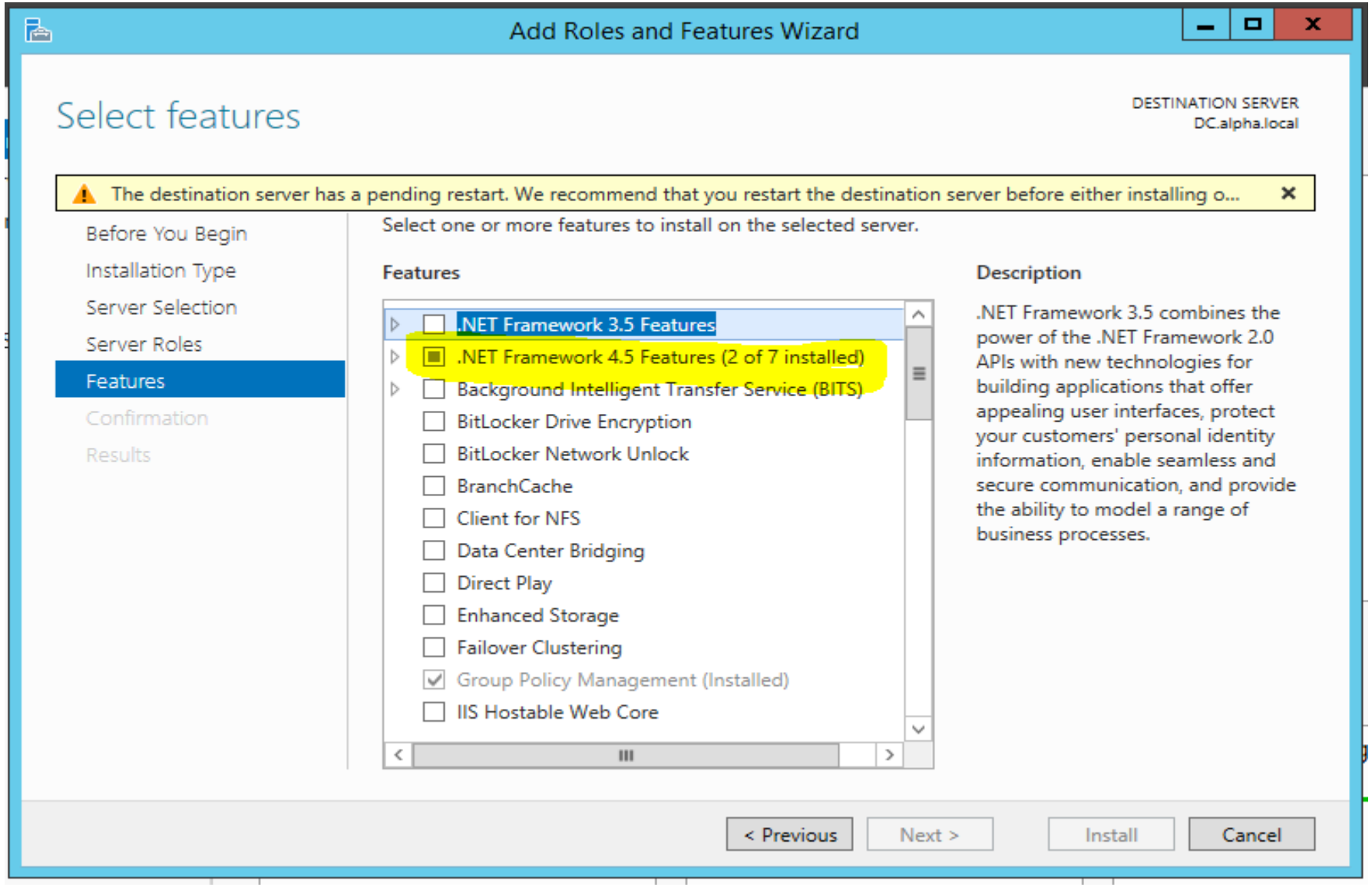
# Upgrade PowerShell (before upgrade)



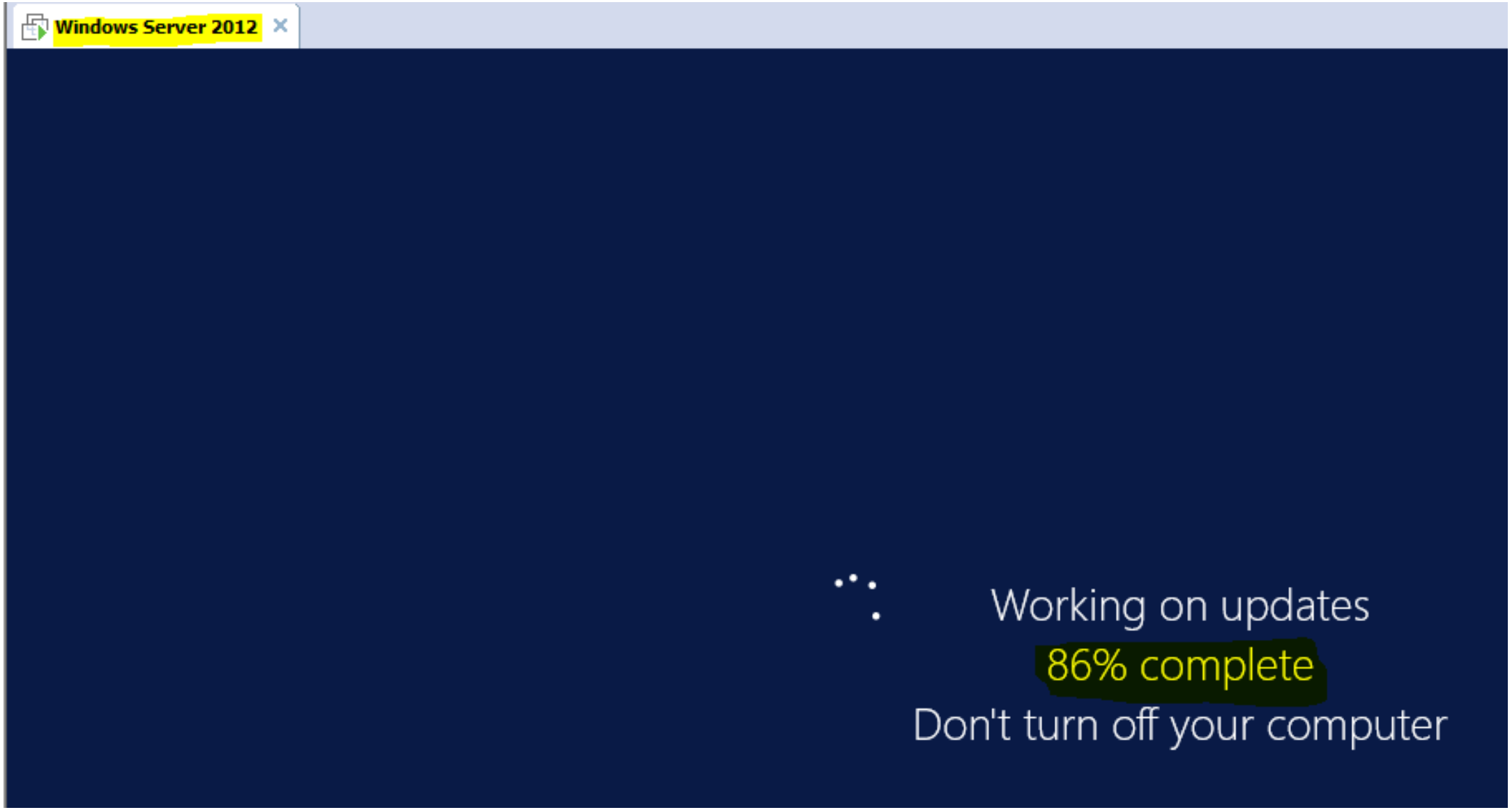
# Upgrade PowerShell (While upgrade)



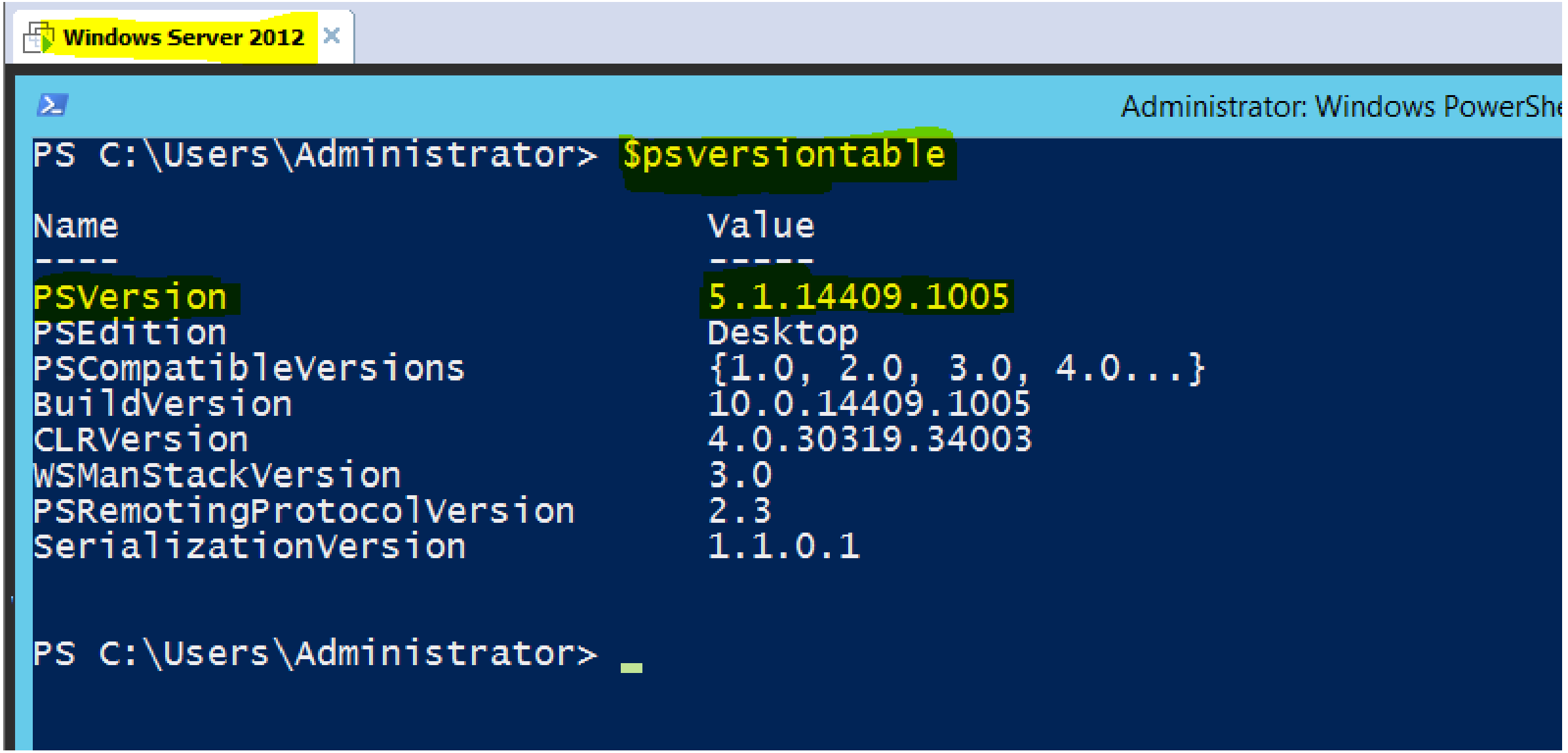
# Upgrade PowerShell (While upgrade)



# Upgrade PowerShell (While upgrade)



# Upgrade PowerShell (after upgrade)



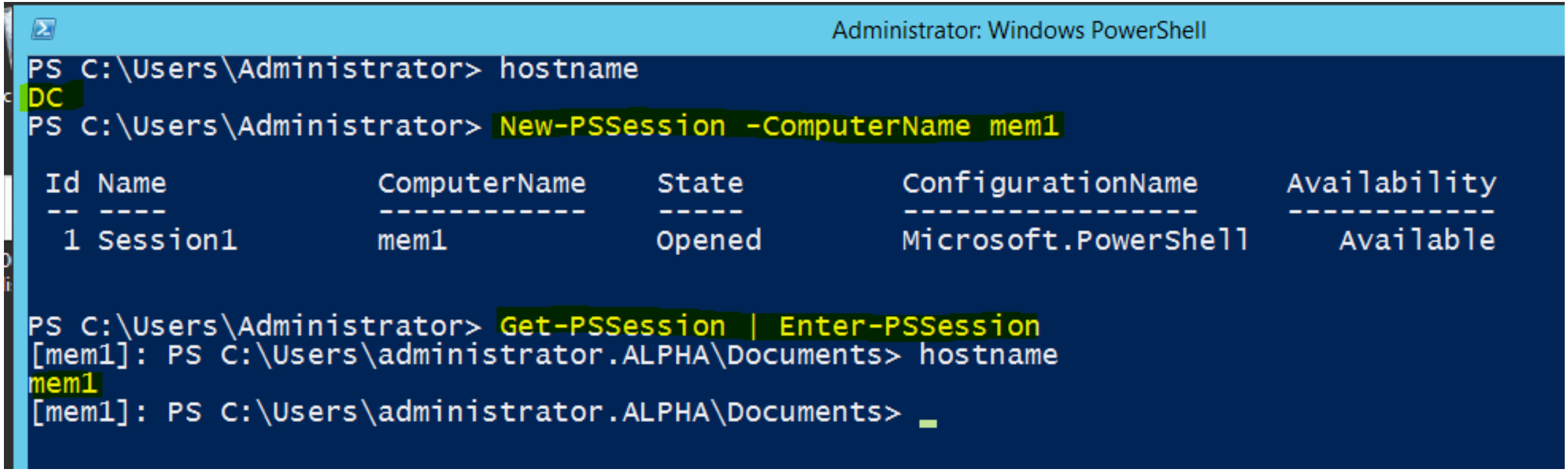
# Working locally with PS

```
PS D:\PowerShell\testScripts> Get-ChildItem


    Directory: D:\PowerShell\testScripts


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2/21/2019   7:33 AM                d4
d-----         1/21/2019   1:50 PM                mod-1
d-----         2/20/2019   9:17 AM                testPack
-a----  PS D:\PowerShell\testScripts> New-Item -Name testFile.txt -ItemType File
-a----


    Directory: D:\PowerShell\testScripts


Mode                 LastWriteTime         Length Name
----
-a----
```

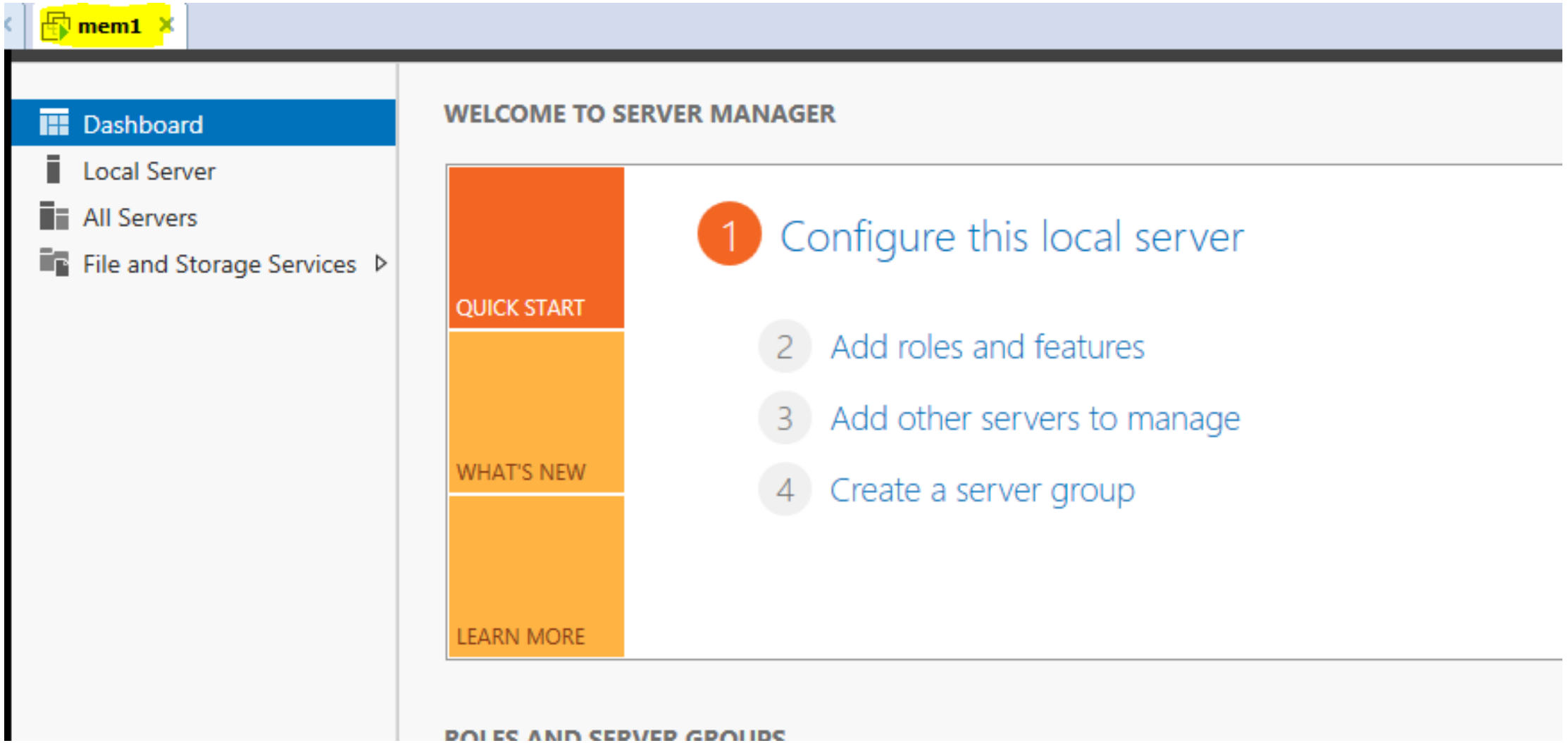
Administrator: Windows PowerShell

```
PS D:\PowerShell\testScripts> New-Item -Name testDir -ItemType Directory


    Directory: D:\PowerShell\testScripts


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         2/25/2019   9:46 PM                testDir
```

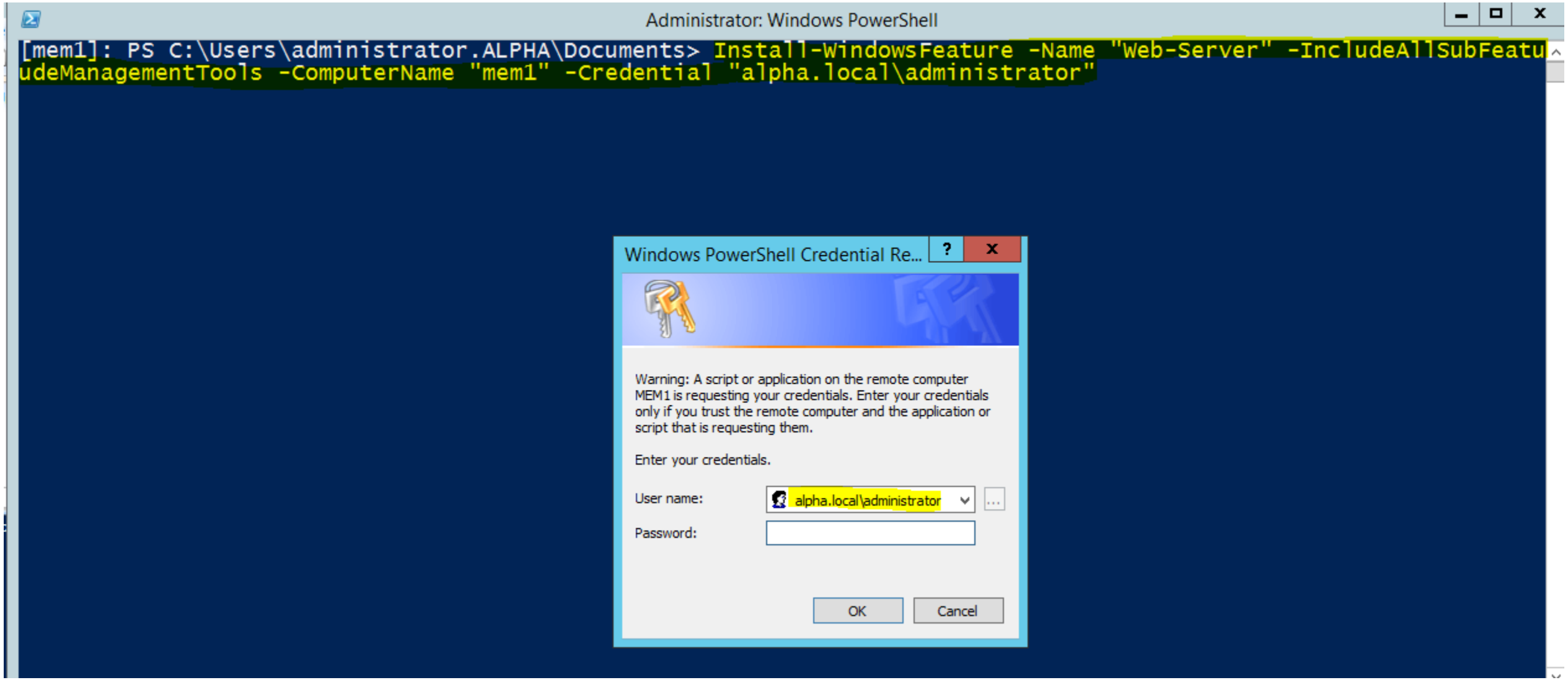
# Working remotely with PS



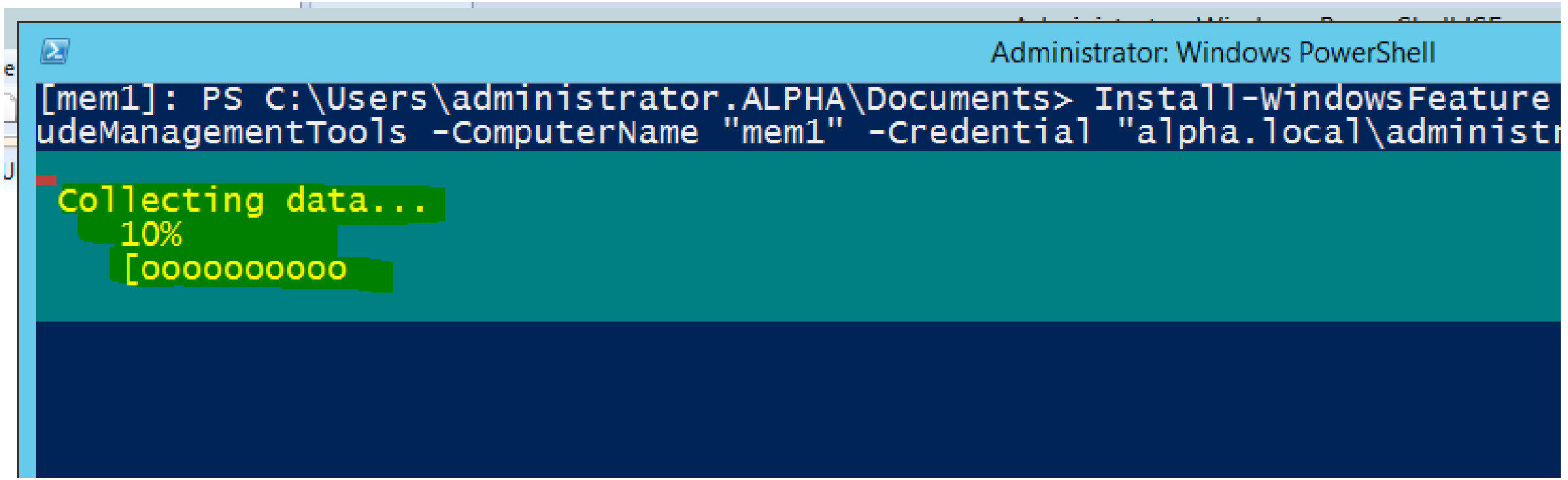
# Working remotely with PS – Before installing IIS webserver



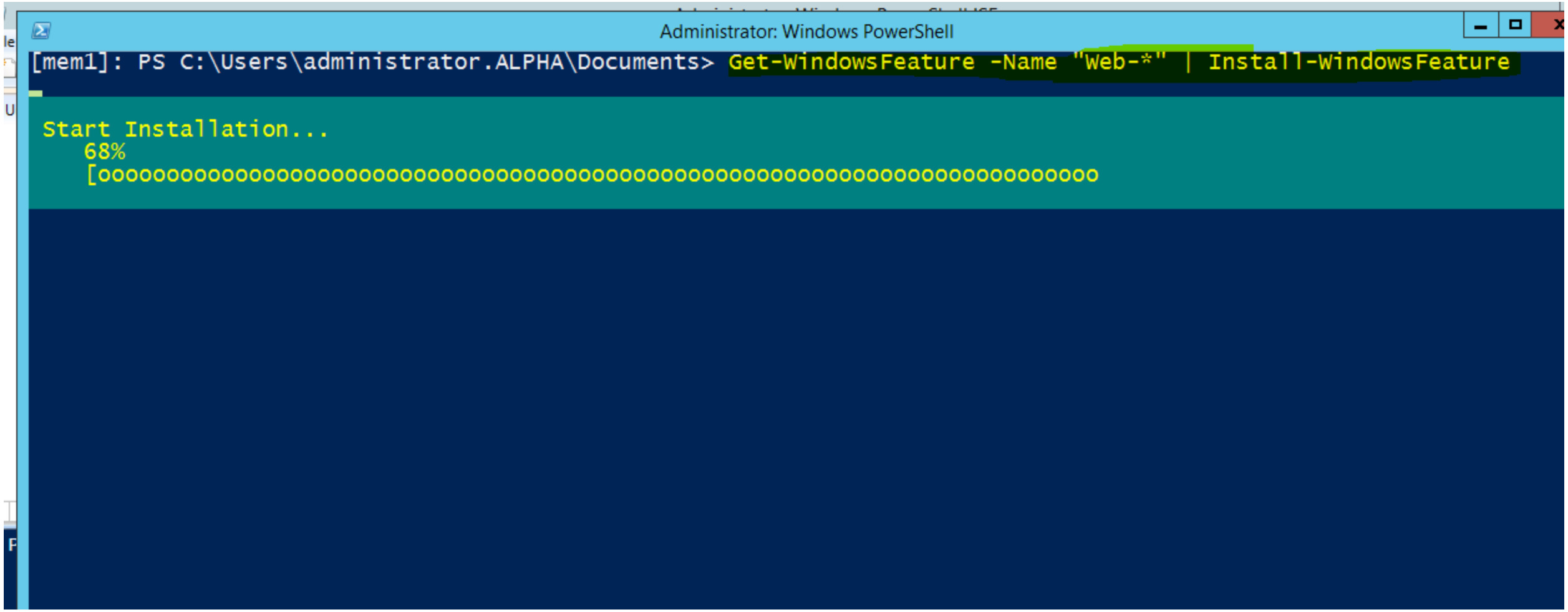
# Working remotely with PS – After installing IIS webserver



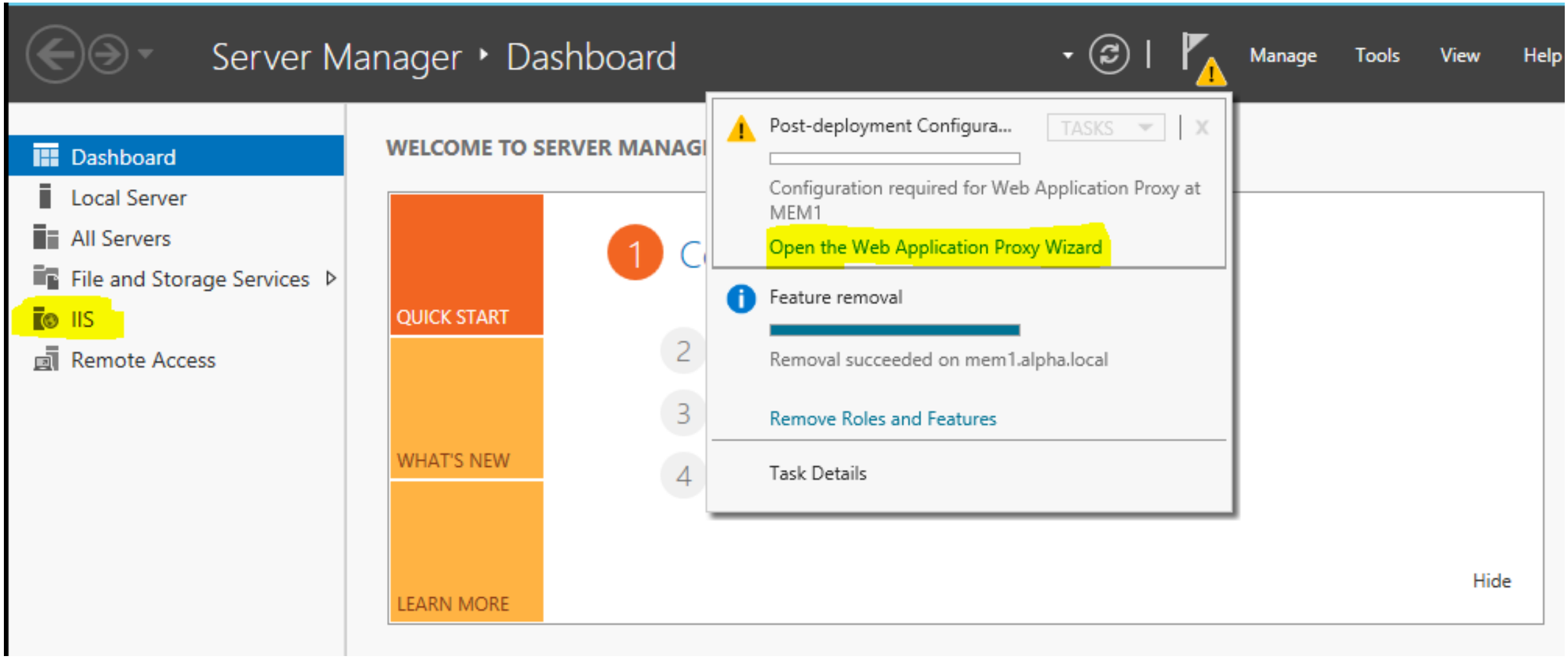
# Working remotely with PS – After installing IIS webserver



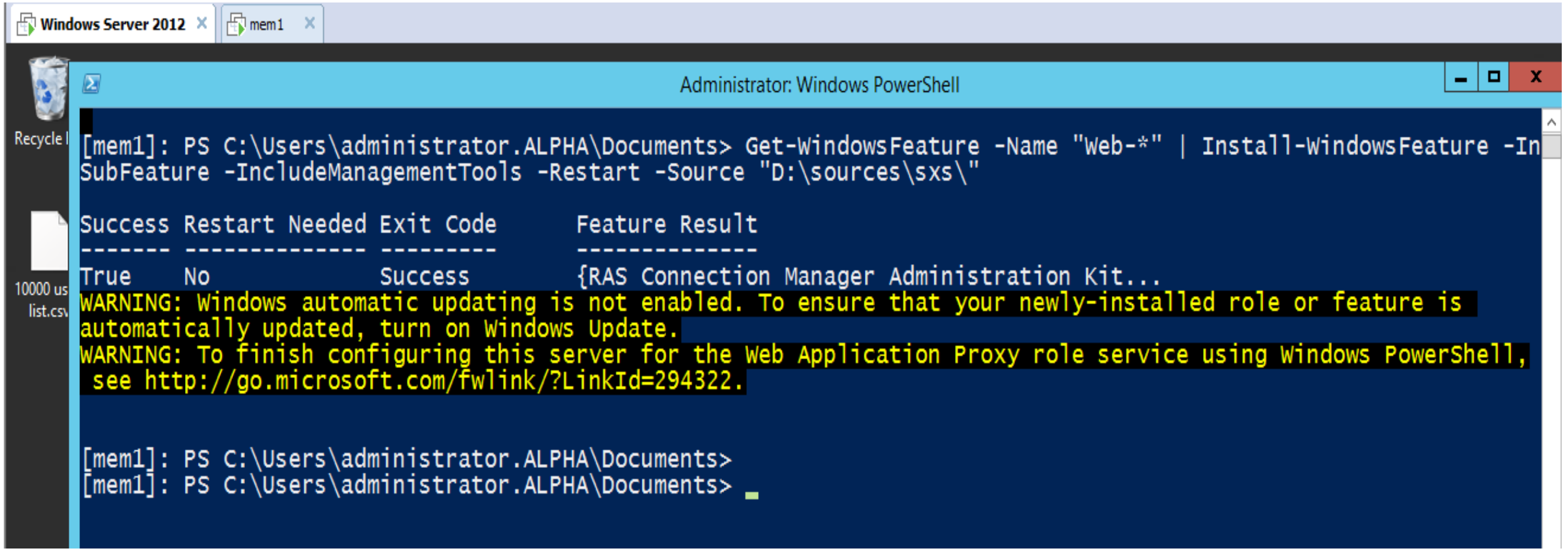
# Working remotely with PS – After installing IIS webserver



# Working remotely with PS – After installing IIS webserver



# Working remotely with PS – After installing IIS webserver



# Creating scripts with ISE.

```powershell
$os = Get-CimInstance -ClassName Win32_OperatingSystem -ComputerName localhost
$cs = Get-CimInstance -ClassName Win32_ComputerSystem -ComputerName localhost
$bios = Get-CimInstance -ClassName Win32_BIOS -ComputerName localhost

$properties = @{
'OSVersion' =$os.version;
'OSBuild' =$os.buildnumber;
'Mgfr' =$cs.manufacturer;
'Model' =$cs.model;
'BIOSSerial' =$bios.serialnumber
}

$obj = New-Object -TypeName PSObject -Property $properties
Write-Output $obj
```

```
PS D:\PowerShell\testScripts> D:\PowerShell\testScripts\converto-html.ps1


BIOSSerial : HRBLYX1
OSVersion  : 10.0.17763
Model      : Latitude E6430
OSBuild    : 17763
Mgfr       : Dell Inc.




PS D:\PowerShell\testScripts>
```

# Error handling with PS

```
 7   Get-WmiObject -ComputerName localhost -Class win32 _bios -ErrorAction SilentlyContinue
 8
 9
10
11
12
13
14
15
16   |
```

```
PS D:\PowerShell\testScripts> D:\PowerShell\testScripts\error.ps1

PS D:\PowerShell\testScripts>
```

# PS integration with other products

- Integration with several Microsoft products:
  - Microsoft Azure
  - Microsoft SQL server
  - Visual Studio
  - Office 365
  - Power BI
  - Windows Servers
  - AWS
  - Exchange server
  - Skype
  - SharePoint
  - DSC - Linux


That's all Folks!


Any
Questions?