PowerShell Everywhere: Unleashing the Power of Cross-Platform Scripting

Frank Lesniak

X: @franklesniak

Danny Stutz

X: @danny_stutz

Command Prompt

```
C:\Users\flesniak>whoami
Frank Lesniak (@franklesniak)
Tech enthusiast; Microsoft 365 consulting team lead at West Monroe

Experience:
Almost 20 years in consulting. PowerShellin' since 2006. Focused on corporate M&A.

Credentials:
MCSEs. Co-leads Chicago PowerShell Users Group. PowerShell + DevOps Global Summit speaker alumnus.
```

```
**********************************************************************************************
Ask Me About:

- PowerShell automation (duh!)
- Executing corporate divestitures and integrations - where I spent most of my consulting client time
- File server modernization - moving to Azure Files, Teams/SharePoint, etc.
- Retro video game software and FPGA emulation
- My upcoming home construction project

**********************************************************************************************
Contact:

- x.com/franklesniak
- bsky.app/profile/franklesniak.com
- linkedin.com/in/flesniak
- github.com/franklesniak
- infosec.exchange/@franklesniak
- flesniak ATSIGN westmonroe.com

C:\Users\flesniak>
```

Command Prompt

```
C:\Users\dstutz>whoami
Danny Stutz (@danny_stutz)
Computer nerd, M365 consulting and automation leader at West Monroe

Experience:
6 years working as a M365/Entra/Azure/AWS technical lead

Credentials:
West Monroe PowerShell Interest Group lead.
PowerShell + DevOps Global Summit alum

**********************************************************************************************************
Ask Me About:

- My mechanical keyboard! (Keychron K2, Boba U4's)
- Migration tool automation with PowerShell (ShareGate, MigrationWiz, azcopy, etc...)
- Threat hunting, M365/Entra/Azure/AWS security
- Executing corporate divestiture & integration migration work (where I spend the majority of my consulting time)
- My music taste (send me your favorite tunes on Discord or Slack!)

**********************************************************************************************************
Contact:

- x.com/danny_stutz
- linkedin.com/in/daniel-stutz
- github.com/danstutz
- dstutz ATSIGN westmonroe.com


C:\Users\dstutz>
```

Thank you Sponsors

Learning Objectives

1. Understand PowerShell's Cross-Platform Capabilities:

- Learn how PowerShell operates across various operating systems and hardware platforms, including Windows, macOS, Linux distributions, Azure Cloud Shell, Raspberry Pi, ARM devices, and historical platforms like Itanium.

Learning Objectives
(continued)

2. Gain Practical Skills for Cross-Platform Scripting:

- Identify key differences in PowerShell's behavior on different platforms.
- Acquire practical tips, tricks, and best practices for writing effective cross-platform scripts.

Learning Objectives
(continued)

3. Apply PowerShell in Diverse Environments:

- Be inspired to utilize PowerShell's full potential in various settings, from managing cloud infrastructure and automating IoT projects to experimenting with different hardware.
- Enhance both professional and personal projects by harnessing PowerShell's versatility.

Agenda

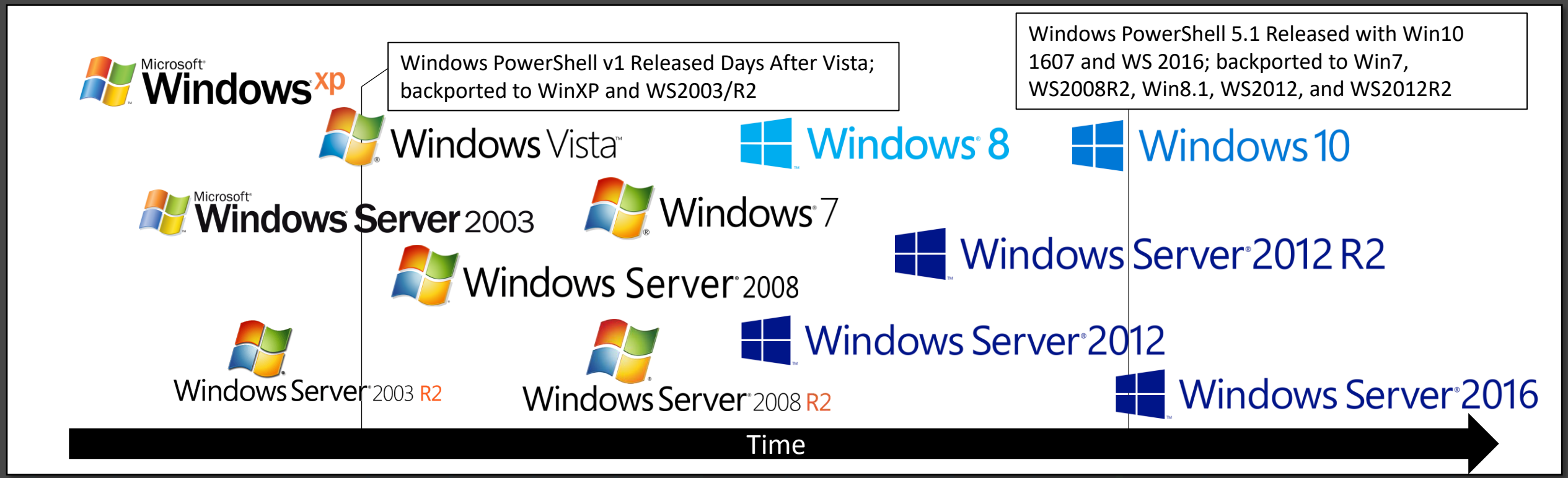
- Introduction
- Cloud Shell
- PowerShell in Linux Containers
- PowerShell on macOS
- PowerShell on Raspberry Pi and IoT Devices
- Alternative Processor Architectures
- Coding for Compatibility

The year was 2018…

- Intel had just confirmed the Spectre/Meltdown vulnerabilities, the first in a series that slowed down our computers by 15-25%!
- Bitcoin had reached a record high

The year was 2018…

- And PowerShell – well, PowerShell only existed on Windows:



The year was 2018...

d Pow PowerShell only existed on Windows:

ed Days After Vista;
003/R2

Windows PowerShell 5.1 Released with Win10 1607 and WS 2016; backported to Win7, WS2008R2, Win8.1, WS2012, and WS2012R2

Windows 8
Windows 10
Windows 7
Windows Server 2012 R2
Windows Server 2008
Windows Server 2012
Windows Server 2008 R2
Windows Server 2016
Time

PowerShell Core 6

Then, on Jan 10, 2018, Microsoft released PowerShell Core 6.0, an open-source version of PowerShell that supported platforms other than Windows!

- Windows 7, Windows Server 2008 R2, and newer (side by side w/ PowerShell 5.1)
- macOS 10.12+
- Ubuntu 14.04, 16.04, and 17.04
- Debian 8.7+, and 9
- CentOS 7
- Red Hat Enterprise Linux 7
- OpenSUSE 42.2
- Fedora 25 and 26
- Arch Linux
- Kali Linux*
- Raspbian Stretch*
- Other Linux distributions?

* = Only Community-Supported (or “Experimental” Support)

Fast-Forward to Today

In general, PowerShell is supported until the OS reaches end of life/support or the version of PowerShell reaches its end of support

- Windows 10 22H2 + supported LTSB/LTSC releases (side by side w/ PowerShell 5.1)
- Windows 11 23H2 and newer + LTSC releases (side by side w/ PowerShell 5.1)
- Windows Server 2016 and newer (side by side w/ PowerShell 5.1)
- macOS 13 and newer
- Ubuntu 22.04 and newer long-term support releases; interim releases (24.10) may work but are limited to community support
- Debian 12
- Red Hat Enterprise Linux (RHEL) 8 and newer
- Alpine 3.20 and newer
- SLES and OpenSUSE*
- Arch Linux*
- Kali Linux*
- Raspberry Pi OS*
- Other Linux distributions?

Note that CentOS and Fedora were dropped; * = Only Community-Supported (or “Experimental” Support)

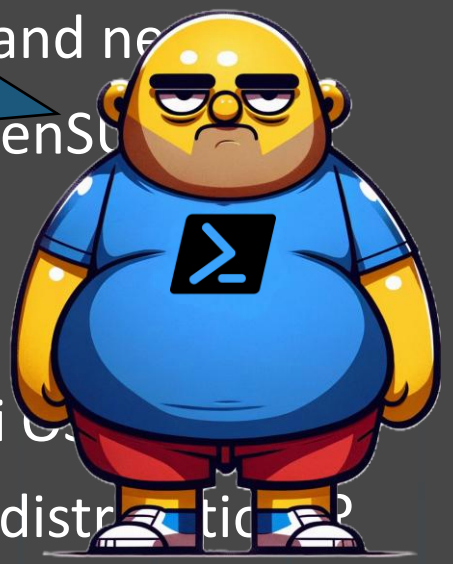
Fast-Forward to Today

In general, PowerShell is supported until the OS reaches end of life/support or the version of PowerShell reaches its end of support

- Windows 10 22H2 + supported LTSB/LTSC releases (side by side w/ PowerShell 5.1)
- Windows 11 23H2 and newer + LTSC releases (side by side w/ PowerShell 5.1)
- Windows Server 2016 and newer (side by side w/ Po
- r
- U
  I
  r
  (
  I
  supp
- Debian 12
- Red Hat Enterprise Linux
  8 and newer
  19 and ne
  d OpenSU
  nux*
  ux*
  erry Pi O
- Other Linux distr tic P

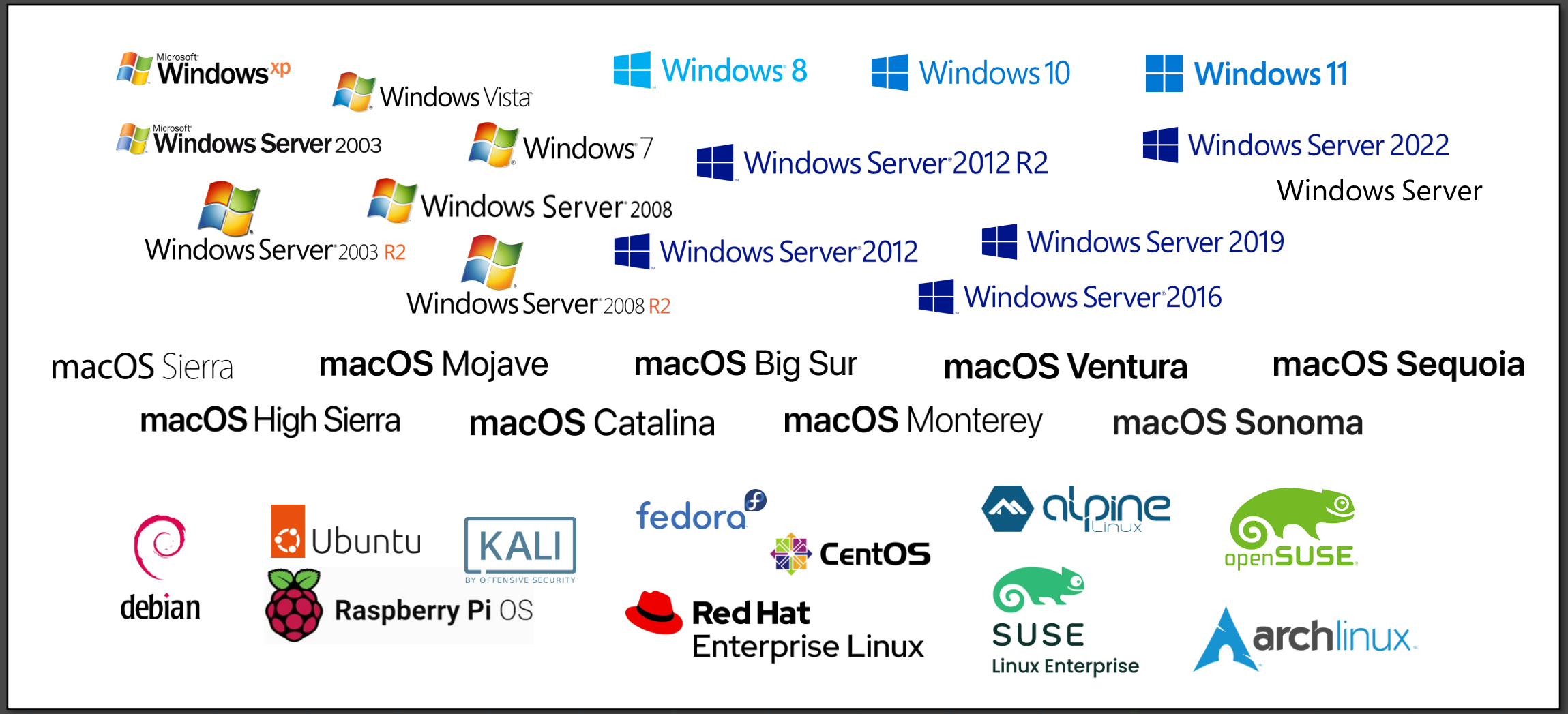
CentOS was dropped because Red Hat discontinued it in 2021.

If you were a *real* Linux sysadmin, you'd know that.



Note that CentOS and Fedora were dropped; * = Only Community-Supported (or "Experimental" Support)

PWSH Everywhere!



Benefits

- Brings a powerful object-oriented scripting language to macOS and Linux
- Unified scripting language – write once, use everywhere
- Increases productivity through consistent tooling and modules – at least in theory
- Improved collaboration/tool-sharing across teams; cost efficiency
- Reduced learning curve for people getting into scripting/automation
- Good backward compatibility (usually)

Use Cases

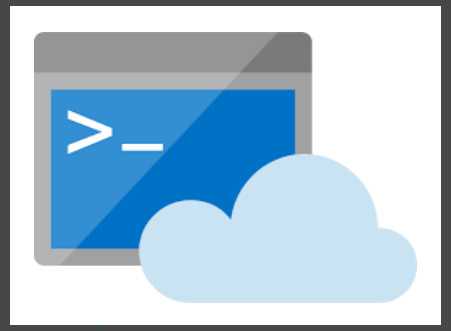
- Cloud infrastructure management – manage Azure, Microsoft 365, etc., from any operating system – or even the browser
- DevOps – cross-platform, powerful deployment pipelines
- IoT device management/use cases
- Home automation
- Hybrid environment administration – cross-OS or across cloud & on-prem
- Continued administration of legacy systems

PWSH in the Cloud

Since:

- Containers are the backbone of many cloud services
- (Most) containers run Linux
- You can run PowerShell on Linux

You can run PowerShell in the cloud!



(Azure Cloud Shell)

docker

Demo – Cloud Shell

- What type of OS?
- What modules?
- OK, but for real, what OS is this?

All code is posted to GitHub:
https://github.com/franklesniak/powershell-xplat

Sign in | Home - Microsoft Azure | https://portal.azure.com/#home | Update

Microsoft Azure | Search resources, services, and docs (G+/) | Copilot | admin.flesniak@[redacted]

Azure services

Create a resource | Microsoft Entra roles and... | Users | App registrations | Microsoft Entra ID | Storage accounts | Microsoft Purview... | Groups | Microsoft Entra Conditional... | More services

Resources

Recent | Favorite

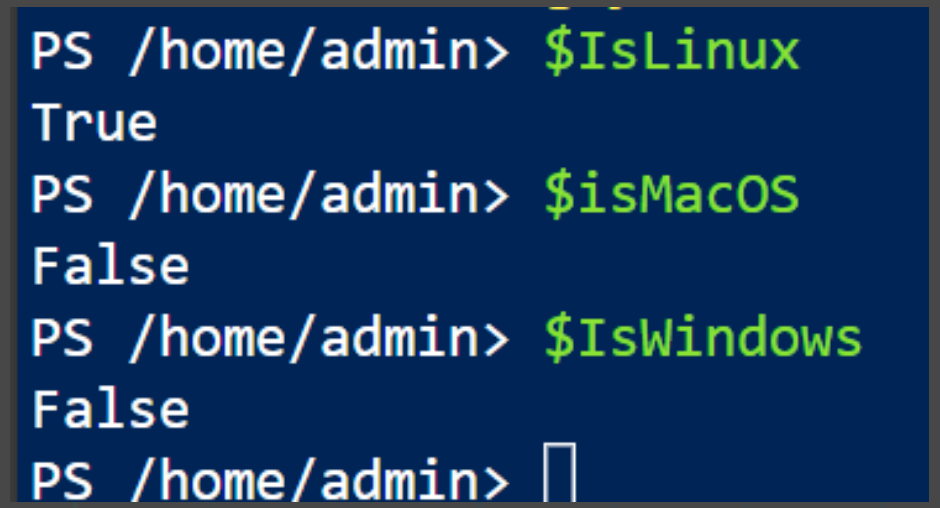
Name	Type	Last Viewed
ausncstplrsinventory02	Storage account	4 months ago
ausncstplrsadmin	Storage account	4 months ago

Switch to Bash | Restart | Manage files | New session | Editor | Web preview | Settings | Help

```
Type "help" to learn about Cloud Shell

Your Cloud Shell session will be ephemeral so no files or system changes will persist beyond your current session.

MOTD: Azure Cloud Shell now includes Predictive IntelliSense! Learn more: https://aka.ms/CloudShell/IntelliSense

VERBOSE: Authenticating to Azure ...
WARNING: TenantId [redacted]ea807' contains more than one active subscription. First one will be selected for further use. To
select another subscription, use Set-AzContext.
WARNING: To override which subscription Connect-AzAccount selects by default, use `Update-AzConfig -DefaultSubscriptionForLogin 00000000-0000-0000-0
000-000000000000`. Go to https://go.microsoft.com/fwlink/?linkid=2200610 for more information.
VERBOSE: Building your Azure drive ...
PS /home/admin>
```

Switch to Bash | Restart | Manage files | New session | Editor | Web preview | Settings | Help

```
VERBOSE: Authenticating to Azure ...
WARNING: TenantId '[redacted]ea807' contains more than one active subscription. First one will be selected for further use. To
select another subscription, use Set-AzContext.
WARNING: To override which subscription Connect-AzAccount selects by default, use `Update-AzConfig -DefaultSubscriptionForLogin 00000000-0000-0000-0
000-000000000000`. Go to https://go.microsoft.com/fwlink/?linkid=2200610 for more information.
VERBOSE: Building your Azure drive ...
PS /home/admin> $IsLinux
True
PS /home/admin> $IsMacOS
False
PS /home/admin> $IsWindows
False
PS /home/admin>
```



Based on the forward slashes in the paths, there is no real surprise here: it's Linux!

```
PS /home/admin> Get-Module -ListAvailable

    Directory: /usr/local/share/powershell/Modules

ModuleType Version    PreRelease Name                                PSEdition ExportedCommands
---------- -------    ---------- ----                                --------- ----------------
Script     13.3.0                Az                                  Core,Desk
Script     4.0.2                 Az.Accounts                         Core,Desk {Disable-AzDataCollection, Disable-AzContextAutosave, Enable-AzData…
Script     2.1.0                 Az.Advisor                          Core,Desk {Disable-AzAdvisorRecommendation, Enable-AzAdvisorRecommendation, G…
Script     6.1.0                 Az.Aks                              Core,Desk {Disable-AzAksAddOn, Enable-AzAksAddOn, Get-AzAksCluster, Get-AzAks…
Script     1.2.0                 Az.AnalysisServices                 Core,Desk {Add-AzAnalysisServicesAccount, Export-AzAnalysisServicesInstanceLo…
Script     4.1.0                 Az.ApiManagement                    Core,Desk {Add-AzApiManagementApiToGateway, Add-AzApiManagementApiToProduct, …
Script     2.0.1                 Az.App                              Core,Desk {Disable-AzContainerAppRevision, Enable-AzContainerAppRevision, Get…
Script     1.4.0                 Az.AppConfiguration                 Core,Desk {Clear-AzAppConfigurationDeletedStore, Get-AzAppConfigurationDelete…
Script     2.3.0                 Az.ApplicationInsights              Core,Desk {Get-AzApplicationInsights, Get-AzApplicationInsightsApiKey, Get-Az…
Script     1.1.0                 Az.ArcResourceBridge                Core,Desk {Get-AzArcResourceBridge, Get-AzArcResourceBridgeApplianceCredentia…
Script     2.1.0                 Az.Attestation                      Core,Desk {Add-AzAttestationPolicySigner, Get-AzAttestationPolicy, Get-AzAtte…
Script     1.1.0                 Az.Automanage                       Core,Desk {Get-AzAutomanageBestPractice, Get-AzAutomanageConfigProfile, Get-A…
Script     1.11.1                Az.Automation                       Core,Desk {Export-AzAutomationDscConfiguration, Export-AzAutomationDscNodeRep…
Script     3.7.0                 Az.Batch                            Core,Desk {Disable-AzBatchAutoScale, Disable-AzBatchComputeNodeScheduling, Di…
Script     2.2.0                 Az.Billing                          Core,Desk {Get-AzBillingAccount, Get-AzBillingInvoice, Get-AzBillingPeriod, G…
Script     3.3.0                 Az.Cdn                              Core,Desk {Clear-AzCdnEndpointContent, Clear-AzFrontDoorCdnEndpointContent, D…
Script     2.1.0                 Az.CloudService                     Core,Desk {Get-AzCloudService, Get-AzCloudServiceInstanceView, Get-AzCloudSer…
Script     1.15.0                Az.CognitiveServices                Core,Desk {Add-AzCognitiveServicesAccountNetworkRule, Get-AzCognitiveServices…
Script     9.1.0                 Az.Compute                          Core,Desk {Add-AzImageDataDisk, Add-AzVhd, Add-AzVMAdditionalUnattendContent,…
Script     1.1.0                 Az.ConfidentialLedger               Core,Desk {Get-AzConfidentialLedger, New-AzConfidentialLedger, New-AzConfiden…
Script     1.1.1                 Az.ConnectedMachine                 Core,Desk {Connect-AzConnectedMachine, Get-AzConnectedExtensionMetadata, Get-…
Script     4.1.1                 Az.ContainerInstance                Core,Desk {Add-AzContainerInstanceOutput, Get-AzContainerGroup, Get-AzContain…
Script     4.3.0                 Az.ContainerRegistry                Core,Desk {Connect-AzContainerRegistry, Get-AzContainerRegistryManifest, Get-…
Script     1.17.0                Az.CosmosDB                         Core,Desk {Get-AzCosmosDBAccount, Get-AzCosmosDBAccountKey, Get-AzCosmosDBCas…
Script     1.2.1                 Az.DataBoxEdge                      Core,Desk {Get-AzDataBoxEdgeBandwidthSchedule, Get-AzDataBoxEdgeDevice, Get-A…
Script     1.10.0                Az.Databricks                       Core,Desk {Get-AzDatabricksAccessConnector, Get-AzDatabricksOutboundNetworkDe…
Script     1.19.1                Az.DataFactory                      Core,Desk {Add-AzDataFactoryV2DataFlowDebugSessionPackage, Add-AzDataFactoryV…
```

```
PS /home/admin> Get-Module -ListAvailable

    Directory: /usr/local/share/powershell/Modules

ModuleType Version    PreRelease Name
---------- -------    ---------- ----
Script     13.3.0                Az
Script     4.0.2                 Az.Accounts
Script     2.1.0                 Az.Advisor
Script     6.1.0                 Az.Aks
Script     1.2.0                 Az.AnalysisServices
Script     4.1.0                 Az.ApiManagement
Script     2.0.1                 Az.App
Script     1.4.0                 Az.AppConfiguration
Script     2.3.0                 Az.ApplicationInsights
Script     1.1.0                 Az.ArcResourceBridge
Script     2.1.0                 Az.Attestation
Script     1.1.0                 Az.Automanage
Script     1.11.1                Az.Automation
Script     3.7.0                 Az.Batch
Script     2.2.0                 Az.Billing
Script     3.3.0                 Az.Cdn
Script     2.1.0                 Az.CloudService
Script     1.15.0                Az.CognitiveServices
Script     9.1.0                 Az.Compute
Script     1.1.0                 Az.ConfidentialLedger
Script     1.1.1                 Az.ConnectedMachine
Script     4.1.1                 Az.ContainerInstance
Script     4.3.0                 Az.ContainerRegistry
Script     1.17.0                Az.CosmosDB
Script     1.2.1                 Az.DataBoxEdge
Script     1.10.0                Az.Databricks
Script     1.19.1                Az.DataFactory
```

```
Name
----
Az
Az.Accounts
Az.Advisor
Az.Aks
Az.AnalysisServices
Az.ApiManagement
Az.App
Az.AppConfiguration
```

```
PS /home/admin> Get-Module -ListAvailable

    Directory: /usr/local/share/powershell/Modules

ModuleType Version    PreRelease Name
---------- -------    ---------- ----
Script     13.3.0                Az
Script     4.0.2                 Az.Accounts
Script     2.1.0                 Az.Advisor
Script     6.1.0                 Az.Aks
Script     1.2.0                 Az.AnalysisServices
Script     4.1.0                 Az.ApiManagement
Script     2.0.1                 Az.App
Script     1.4.0                 Az.AppConfiguration
Script     2.3.0                 Az.ApplicationInsights
Script     1.1.0                 Az.ArcResourceBridge
Script     2.1.0                 Az.Attestation
Script     1.1.0                 Az.Automanage
Script     1.11.1                Az.Automation
Script     3.7.0                 Az.Batch
Script     2.2.0                 Az.Billing
Script     3.3.0                 Az.Cdn
Script     2.1.0                 Az.CloudService
Script     1.15.0                Az.CognitiveServices
Script     9.1.0                 Az.Compute
Script     1.1.0                 Az.ConfidentialLedger
Script     1.1.1                 Az.ConnectedMachine
Script     4.1.1                 Az.ContainerInstance
Script     4.3.0                 Az.ContainerRegistry
Script     1.17.0                Az.CosmosDB
Script     1.2.1                 Az.DataBoxEdge
Script     1.10.0                Az.Databricks
Script     1.19.1                Az.DataFactory
```

Modules.

Modules as far as the eye can see!

```
PS /home/admin> Get-ComputerInfo
Get-ComputerInfo: The term 'Get-ComputerInfo' is not recognized as a name of a cmdlet, function, script file, or executable program.
Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
PS /home/admin>
```

PS /home/admin> Get-ComputerInfo
Get-ComputerInfo: The term 'Get-ComputerInfo' is not recognized

WHOMP WHOMP

```powershell
function Invoke-CrossPlatformOSInventory {
    # .SYNOPSIS
    # Inventories the currently running operating system and creates a PSCustomObject
    # containing the results.
    #
    # .DESCRIPTION
    # This function inventories the currently running operating system and creates a
    # PSCustomObject containing the results. The function is designed to be run on
    # Windows, Linux, and macOS.
    #
    # This function is a proof of concept and is not intended to be used in a
    # production environment. It is intended to be used as a starting point for
    # creating a cross-platform inventory function.
    #
    # .EXAMPLE
    # $osInfo = Invoke-CrossPlatformOSInventory
    #
    # .INPUTS
    # None. You can't pipe objects to Invoke-CrossPlatformOSInventory.
    #
    # .OUTPUTS
    # System.RuntimeType (PSObject). Invoke-CrossPlatformOSInventory returns a
    # PSObject with the following properties:
    #
```

CMDB Properties to Track Operating System Version

Here is the list of properties that I plan to have in my CMDB:

- **OSType**: The high-level category of the operating system, identifying its core family. This field provides a simple, standardized label for grouping systems by OS type, facilitating broad classification across diverse platforms.
  - Windows: "Windows" (static)
  - macOS: "macOS" (static)
  - Linux: "Linux" (static)
- **OSID**: A unique identifier for the operating system, typically extracted directly from system metadata where available. For Linux, this is sourced from the `ID` field of tools like `/etc/os-release`, providing a short, machine-readable name. For Windows and macOS, it is statically assigned to ensure consistency across platforms where no native equivalent exists.
  - Windows: "Windows" (static)
  - macOS: "macOS" (static)
  - Linux: `ID` extracted directly from operating system tools that provide OS information (e.g., `/etc/os-release`); field might contain values such as "ubuntu", "debian", "rhel", etc.
- **OSName**: The base name of the operating system, representing its core identity without additional version or edition details. This field captures the fundamental OS designation reported by native tools, providing a concise label for identification.
  - Windows: Extracted from `Win32_OperatingSystem` -> `Caption`; e.g., "Microsoft Windows 11 Pro"
  - macOS: `ProductName` from `sw_vers` (i.e., "macOS")
  - Linux: `NAME` extracted directly from operating system tools that provide OS information (e.g., `/etc/os-release`); field might contain values such as "Ubuntu", "Debian GNU/Linux", etc.

```
PS /home/admin> . ./Invoke-CrossPlatformOSInventory.ps1; Invoke-CrossPlatformOSInventory

OSType                  : Linux
OSID                    : mariner
OSName                  : Common Base Linux Mariner
OSPrettyName            : CBL-Mariner/Linux
OSVersionString         : 2.0.20250207
OSVersionMajorString    : 2
OSVersionMajor          : 2
OSVersionMinorString    : 0
OSVersionMinor          : 0
OSVersionBuild          : 20250207
OSVersionPatchString    :
OSVersionPatch          : -1
OSVersionRevisionString :
OSVersionRevision       : -1
OSBuildString           : 20250207
OSServicePack           : 0
OSEdition               :
OSSKU                   : 0
OSCodename              :
OSKernelVersion         : 6.1.124.1-microsoft-standard
OSArchitecture          : x86-64
```

Demo – Docker Container

- Getting the Latest Cloud Shell Container
- Run it!

All code is posted to GitHub:
https://github.com/franklesniak/powershell-xplat

Docker Prereqs

On Windows:

1. Install Windows Subsystem for Linux:
   `wsl --install`

2. Then, download and install Docker Desktop (when asked during installation, choose to use WSL instead of Hyper-V)

3. Finally, start Docker and sign-in

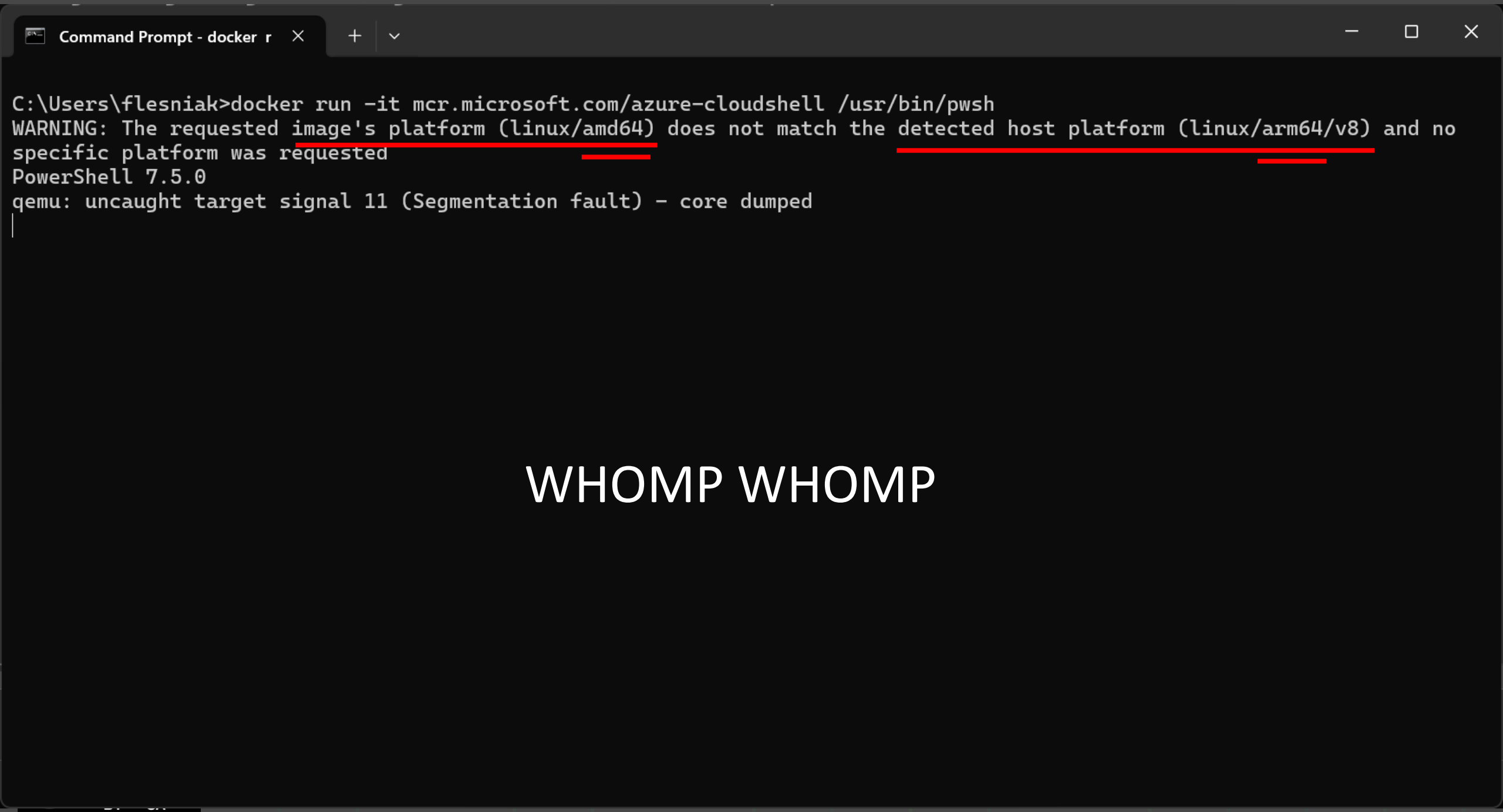
Command Prompt - docker p

```
C:\Users\flesniak>docker pull mcr.microsoft.com/azure-cloudshell
Using default tag: latest
latest: Pulling from azure-cloudshell
5cba3b109066: Downloading [=>                                         ]  981.3kB/31.49MB
f27d41e62a7e: Download complete
c4239aea5f32: Downloading [>                                          ]  538.9kB/1.263GB
b162fe1f2b76: Downloading [=>                                         ]  555.3kB/27.55MB
3901a31d84ce: Waiting
b617c733d96b: Waiting
4f3864f910ad: Waiting
55adc7b86325: Waiting
acf5fe7b5819: Waiting
b3eb6c802bfc: Waiting
74d3ae0e825e: Waiting
00096c9ce5c7: Waiting
19c1cdd2a686: Waiting
a0389b22df0b: Waiting
f23c966f4c02: Waiting
8578632d0314: Waiting
4f4fb700ef54: Waiting
62b1173b4df8: Waiting
c6c15dd9f149: Waiting
61e4805f6af0: Waiting
```

Command Prompt - docker p

```
C:\Users\flesniak>docker pull mcr.microsoft.com/azure-cloudshell
Using default tag: latest
latest: Pulling from azure-cloudshell
5cba3b109066: Downloading [=>                                         ]  981.3kB/31.49MB
f27d41e62a7e: Download complete
c4239aea5f32: Downloading [>                                          ]  538.9kB/1.263GB
b16
39
b61
4f3
55a
acf
b3eb6c802bfc: Waiting
74d3ae0e825e: Waiting
00096c9ce5c7: Waiting
19c1cdd2a686: Waiting
a0389b22df0b: Waiting
f23c966f4c02: Waiting
8578632d0314: Waiting
4f4fb700ef54: Waiting
62b1173b4df8: Waiting
c6c15dd9f149: Waiting
61e4805f6af0: Waiting
```


```
C:\Users\flesniak>docker pull mcr.microsoft.com/azure-cloudshell
Using default tag: latest
latest: Pulling from azure-cloudshell
```


Command Prompt - docker r
C:\Users\flesniak>docker run -it mcr.microsoft.com/azure-cloudshell /usr/bin/pwsh
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested
PowerShell 7.5.0
qemu: uncaught target signal 11 (Segmentation fault) - core dumped
WHOMP WHOMP

Command Prompt - docker r

```
C:\Users\flesniak>docker run -it mcr.microsoft.com/azure-cloudshell /usr/bin/pwsh
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no
specific platform was requested
PowerShell 7.5.0
qemu: uncaught target signal 11 (Segmentation fault) - core dumped
```

Again, if you were a *real* Linux sysadmin, you would rebuild the container from source, and then you wouldn't have these problems.

Demo – Docker Container Take 2!

- Get the Latest ARM64 PowerShell Container
- Run it!

All code is posted to GitHub:
https://github.com/franklesniak/powershell-xplat

Command Prompt - docker p

```
C:\Users\flesniak>docker pull mcr.microsoft.com/powershell:mariner-2.0-arm64
mariner-2.0-arm64: Pulling from powershell
d8f17d42cf81: Downloading [==========>                                ]  5.012MB/27.16MB
a307b6d780d7: Download complete
c5e0a084dbc4: Download complete
74317f4a9312: Downloading [=====>                                     ]  2.161MB/19.43MB
26ab0373d654: Download complete
269aeee1dc80: Downloading [====>                                      ]  2.751MB/32.84MB
e29c3d799541: Waiting
c21a5e4aae99: Waiting
4f4fb700ef54: Waiting
8e61fe07e20a: Waiting
```


```
C:\Users\flesniak>docker pull mcr.microsoft.com/powershell:mariner-2.0-arm64
mariner-2.0-arm64: Pulling from powershell
```


```
C:\Users\flesniak>docker run -it mcr.microsoft.com/powershell:mariner-2.0-arm64
PowerShell 7.5.0
PS /> $IsLinux
True
PS />
```

```
PS /> Invoke-WebRequest -Uri 'https://raw.githubusercontent.com/franklesniak/powershell-xplat/refs/heads/main/Invoke-CrossPlatformOSInventory.ps1' -OutFile ./Invoke-CrossPlatformOSInventory.ps1
PS /> . ./Invoke-CrossPlatformOSInventory.ps1; Invoke-CrossPlatformOSInventory

OSType                  : Linux
OSID                    : mariner
OSName                  : Common Base Linux Mariner
OSPrettyName            : CBL-Mariner/Linux
OSVersionString         : 2.0.20241230
OSVersionMajorString    : 2
OSVersionMajor          : 2
OSVersionMinorString    : 0
OSVersionMinor          : 0
OSVersionBuild          : 20241230
OSVersionPatchString    :
OSVersionPatch          : -1
OSVersionRevisionString :
OSVersionRevision       : -1
OSBuildString           : 20241230
OSServicePack           : 0
OSEdition               :
OSSKU                   : 0
OSCodename              :
OSKernelVersion         : 5.15.153.1-microsoft-standard-WSL2
OSArchitecture          : arm64
```

Demo – PowerShell on macOS

- Determining the version number of macOS

All code is posted to GitHub:
https://github.com/franklesniak/powershell-xplat

Setup/Prereqs

- Install Homebrew (if not already installed):

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

- Add brew to your path (if not already done):

```
echo >> ~/.zprofile
echo 'eval "$(/usr/local/bin/brew shellenv)"' >> ~/.zprofile
eval "$(/usr/local/bin/brew shellenv)"
```

- Install PowerShell:

```
brew install --cask powershell
```

- Start PowerShell:

```
pwsh
```

```
PS /Users/blake> Invoke-WebRequest -Uri 'https://raw.githubusercontent.com/franklesniak/powershell-xplat/refs/heads/main/Invoke-CrossPlatformOSInventory.ps1' -OutFile ./Invoke-CrossPlatformOSInventory.ps1
PS /Users/blake> . ./Invoke-CrossPlatformOSInventory.ps1; Invoke-CrossPlatformOSInventory

OSType                  : macOS
OSID                    : macOS
OSName                  : macOS
OSPrettyName            : macOS 15.3
OSVersionString         : 15.3
OSVersionMajorString    : 15
OSVersionMajor          : 15
OSVersionMinorString    : 3
OSVersionMinor          : 3
OSVersionBuild          : -1
OSVersionPatchString    :
OSVersionPatch          : -1
OSVersionRevisionString :
OSVersionRevision       : -1
OSBuildString           : 24D5034f
OSServicePack           : 0
OSEdition               :
OSSKU                   : 0
OSCodename              :
OSKernelVersion         : 24.3.0
OSArchitecture          : arm64
```

Demo – Windows 10 IoT Core on Raspberry Pi

- Remote In, Try Some Stuff
- “Install” PowerShell 7.x Side-By-Side; Enable Remoting
- Remote Into PowerShell 7

All code is posted to GitHub:
https://github.com/franklesniak/powershell-xplat

```
45    #region Part 0: Load required functions ##########################################
      1 reference
46  > function Get-FolderPathContainingScript { ···
190   }
      10 references
191 > function Get-PSVersion { ···
242   }
243   #endregion Part 0: Load required functions ##########################################
```

```powershell
#region Part 1: Copy scripts to the target system using remote PSSession ############
$strWin10IoTCoreDeviceIPOrHostname = "10.1.2.137"
$strWin10IoTCoreUsername = "Administrator"
$strCurrentFolder = Get-FolderPathContainingScript
$strPathToInvokeX86ExeScript = Join-Path $strCurrentFolder 'Win10_IoT_Core'
$strPathToInvokeX86ExeScript = Join-Path $strPathToInvokeX86ExeScript 'Invoke-x86Exe.ps1'
$strPathToTestPowerShellModuleInstalledScript = Join-Path $strCurrentFolder 'Win10_IoT_Core'
$strPathToTestPowerShellModuleInstalledScript = Join-Path $strPathToTestPowerShellModuleInstalledScript 'Test-
```

```
#region Part 1: Copy scripts to the target system using remote PSSession ############
$strWin10IoTCoreDeviceIPOrHostname = "10.1.2.137"
$strWin10IoTCoreUsername = "Administrator"
$strCurrentFolder = Get-FolderPathContainingScript
$strPathToInvokeX86ExeScript = Join-Path $strCurrentFolder 'Win10_IoT_Core'
$strPathToInvokeX86ExeScript = Join-Path $strPathToInvokeX86ExeScript 'Invoke-x86Exe.ps1'
$strPathToTestPowerShellModuleInstalledScript = Join-Path $strCurrentFolder 'Win10_IoT_Core'
$strPathToTestPowerShellModuleInstalledScript = Join-Path $strPathToTestPowerShellModuleInstalledScript 'Test-
```

```
#region Part 1: Copy scripts to the target system using remote PSSession ############
$strWin10IoTCoreDeviceIPOrHostname = "10.1.2.137"
$strWin10IoTCoreUsername = "Administrator"
$strCurrentFolder = Get-FolderPathContainingScript
$strPathToInvokeX86ExeScript = Join-Path $strCurrentFolder 'Win10_IoT_Core'
$strPathToInvokeX86ExeScript = Join-Path $strPathToInvokeX86ExeScript 'Invoke-x86Exe.ps1'
$strPathToTestPowerShellModuleInstalledScript = Join-Path $strCurrentFolder 'Win10_IoT_Core'
$strPathToTestPowerShellModuleInstalledScript = Join-Path $strPathToTestPowerShellModuleInstalledScript 'Test-
```

```
#region Part 1: Copy scripts to the target system using remote PSSession ############
$strWin10IoTCoreDeviceIPOrHostname = "10.1.2.137"
$strWin10IoTCoreUsername = "Administrator"
$strCurrentFolder = Get-FolderPathContainingScript
$strPathToInvokeX86ExeScript = Join-Path $strCurrentFolder 'Win10_IoT_Core'
$strPathToInvokeX86ExeScript = Join-Path $strPathToInvokeX86ExeScript 'Invoke-x86Exe.ps1'
$strPathToTestPowerShellModuleInstalledScript = Join-Path $strCurrentFolder 'Win10_IoT_Core'
$strPathToTestPowerShellModuleInstalledScript = Join-Path $strPathToTestPowerShellModuleInstalledScript 'Test-
```

PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> $credential = Get-Credential $strWin10IoTCoreUsername

PowerShell credential request
Enter your credentials.
Password for user Administrator: **************

PS C:\Users\flesniak\Github\powershell-xplat> $PSSession = New-PSSession -ComputerName $strWin10IoTCoreDeviceIPOrHostnam
e -Credential $credential
PS C:\Users\flesniak\Github\powershell-xplat>
```


```
PS C:\Users\flesniak\Github\powershell-xplat> $credential = Get-Credential $strWin10IoTCoreUsername

PowerShell credential request
Enter your credentials.
Password for user Administrator: **************

PS C:\Users\flesniak\Github\powershell-xplat> $PSSession = New-PSSession -ComputerName $strWin10IoTCoreDeviceIPOrHostnam
e -Credential $credential
PS C:\Users\flesniak\Github\powershell-xplat>
```

Pow

PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> $credential = Get-Credential $strWin10IoTCoreUsername

PowerShell credential request
Enter your credentials.
Password for user Administrator: **************

PS C:\Users\flesniak\Github\powershell-xplat> $PSSession = New-PSSession -ComputerName $strWin10IoTCoreDeviceIPOrHostnam
e -Credential $credential
PS C:\Users\flesniak\Github\powershell-xplat> $strWin10IoTCoreDownloadPathRoot = ("C:\Data\Users\" + $strWin10IoTCoreUse
rname + "\Downloads")
PS C:\Users\flesniak\Github\powershell-xplat> Copy-Item $strPathToInvokeX86ExeScript -Destination $strWin10IoTCoreDownlo
adPathRoot -ToSession $PSSession -Force
PS C:\Users\flesniak\Github\powershell-xplat> Copy-Item $strPathToTestPowerShellModuleInstalledScript -Destination $strW
in10IoTCoreDownloadPathRoot -ToSession $PSSession -Force
PS C:\Users\flesniak\Github\powershell-xplat>
```


```
PS C:\Users\flesniak\Github\powershell-xplat> $strWin10IoTCoreDownloadPathRoot = ("C:\Data\Users\" + $strWin10IoTCoreUse
rname + "\Downloads")
```

PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> $credential = Get-Credential $strWin10IoTCoreUsername

PowerShell credential request
Enter your credentials.
Password for user Administrator: **************

PS C:\Users\flesniak\Github\powershell-xplat> $PSSession = New-PSSession -ComputerName $strWin10IoTCoreDeviceIPOrHostnam
e -Credential $credential
PS C:\Users\flesniak\Github\powershell-xplat> $strWin10IoTCoreDownloadPathRoot = ("C:\Data\Users\" + $strWin10IoTCoreUse
rname + "\Downloads")
PS C:\Users\flesniak\Github\powershell-xplat> Copy-Item $strPathToInvokeX86ExeScript -Destination $strWin10IoTCoreDownlo
adPathRoot -ToSession $PSSession -Force
PS C:\Users\flesniak\Github\powershell-xplat> Copy-Item $strPathToTestPowerShellModuleInstalledScript -Destination $strW
in10IoTCoreDownloadPathRoot -ToSession $PSSession -Force
PS C:\Users\flesniak\Github\powershell-xplat>
```


```
PS C:\Users\flesniak\Github\powershell-xplat> Copy-Item $strPathToInvokeX86ExeScript -Destination $strWin10IoTCoreDownlo
adPathRoot -ToSession $PSSession -Force
PS C:\Users\flesniak\Github\powershell-xplat> Copy-Item $strPathToTestPowerShellModuleInstalledScript -Destination $strW
in10IoTCoreDownloadPathRoot -ToSession $PSSession -Force
```

PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> $credential = Get-Credential $strWin10IoTCoreUsername

PowerShell credential request
Enter your credentials.
Password for user Administrator: **************

PS C:\Users\flesniak\Github\powershell-xplat> $PSSession = New-PSSession -ComputerName $strWin10IoTCoreDeviceIPOrHostnam
e -Credential $credential
PS C:\Users\flesniak\Github\powershell-xplat> $strWin10IoTCoreDownloadPathRoot = ("C:\Data\Users\" + $strWin10IoTCoreUse
rname + "\Downloads")
PS C:\Users\flesniak\Github\powershell-xplat> Copy-Item $strPathToInvokeX86ExeScript -Destination $strWin10IoTCoreDownlo
adPathRoot -ToSession $PSSession -Force
PS C:\Users\flesniak\Github\powershell-xplat> Copy-Item $strPathToTestPowerShellModuleInstalledScript -Destination $strW
in10IoTCoreDownloadPathRoot -ToSession $PSSession -Force
PS C:\Users\flesniak\Github\powershell-xplat> Enter-PSSession -Session $PSSession
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
```


```
PS C:\Users\flesniak\Github\powershell-xplat> Enter-PSSession -Session $PSSession
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
```

```
275  > function Get-DownloadFolder { ···
294    }
```

```
#region Part 3: Try some things on Windows 10 IoT Core ##############################
$strDownloadFolder = Get-DownloadFolder
$strPathToInvokeX86ExeScript = Join-Path $strDownloadFolder 'Invoke-x86Exe.ps1'
$strPathToTestPowerShellModuleInstalledScript = Join-Path $strDownloadFolder 'Test-PowerShellModuleInstalled.p
```

```
275  > function Get-DownloadFolder { ···
294    }
```

```
#region Part 3: Try some things on Windows 10 IoT Core ################################
$strDownloadFolder = Get-DownloadFolder
$strPathToInvokeX86ExeScript = Join-Path $strDownloadFolder 'Invoke-x86Exe.ps1'
$strPathToTestPowerShellModuleInstalledScript = Join-Path $strDownloadFolder 'Test-PowerShellModuleInstalled.p
```

PowerShell

```
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> # This script downloads Streams for Sysinternals and attempts to
 run it:
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> & $strPathToInvokeX86ExeScript
Program 'streams.exe' failed to run: The specified executable is not a valid application for this OS platform.At
C:\Data\Users\Administrator\Downloads\Invoke-x86Exe.ps1:156 char:1
+ & '.\Streams.exe' -d -s -q
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~.
At C:\Data\Users\Administrator\Downloads\Invoke-x86Exe.ps1:156 char:1
+ & '.\Streams.exe' -d -s -q
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ResourceUnavailable: (:) [], ApplicationFailedException
    + FullyQualifiedErrorId : NativeCommandFailed

[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams>
```


```
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> # This script downloads Streams for Sysinternals and attempts to
 run it:
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> & $strPathToInvokeX86ExeScript
Program 'streams.exe' failed to run: The specified executable is not a valid application for this OS platform.At
C:\Data\Users\Administrator\Downloads\Invoke-x86Exe.ps1:156 char:1
+ & '.\Streams.exe' -d -s -q
```

PowerShell

```
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> # This script downloads Streams for Sysinternals and attempts to
 run it:
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> & $strPathToInvokeX86ExeScript
Program 'streams.exe' failed to run: The specified executable is not a valid application for this OS platform.At
C:\Data\Users\Administrator\Downloads\Invoke-x86Exe.ps1:156 char:1
+ & '.\Streams.exe' -d -s -q
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~.
At C:\Data\Users\Administrator\Downloads\Invoke-x86Exe.ps1:156 char:1
+ & '.\Streams.exe' -d -s -q
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ResourceUnavailable: (:) [], ApplicationFailedException
    + FullyQualifiedErrorId : NativeCommandFailed

[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> # Hint:
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> $env:PROCESSOR_ARCHITECTURE
ARM
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams>
```


```
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> # Hint:
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> $env:PROCESSOR_ARCHITECTURE
ARM
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams>
```

Pow

PowerShell

```
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> # This script checks if the PSCompression module is inst
alled:
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> & $strPathToTestPowerShellModuleInstalledScript
WARNING: PSCompression module not found. Please install it and then try again.
You can install the PSCompression PowerShell module from the PowerShell Gallery by running the following command:
```


```
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> # This script checks if the PSCompression module is inst
alled:
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> & $strPathToTestPowerShellModuleInstalledScript
WARNING: PSCompression module not found. Please install it and then try again.
You can install the PSCompression PowerShell module from the PowerShell Gallery by running the following command:
```


```
Install-Module PSCompression;

If the installation command fails, you may need to upgrade the version of PowerShellGet. To do so, run the following com
mands, then restart PowerShell:
Set-ExecutionPolicy Bypass -Scope Process -Force;
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12;
Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force;
Install-Module PowerShellGet -MinimumVersion 2.2.4 -SkipPublisherCheck -Force -AllowClobber;


[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams>
```

PowerShell

```
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> # This script checks if the PSCompression module is inst
alled:
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> & $strPathToTestPowerShellModuleInstalledScript
WARNING: PSCompression module not found. Please install it and then try again.
You can install the PSCompression PowerShell module from the PowerShell Gallery by running the following command:
Install-Module PSCompression;

If the installation command fails, you may need to upgrade the version of PowerShellGet. To do so, run the following com
mands, then restart PowerShell:
Set-ExecutionPolicy Bypass -Scope Process -Force;
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12;
Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force;
Install-Module PowerShellGet -MinimumVersion 2.2.4 -SkipPublisherCheck -Force -AllowClobber;
```

```
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> Install-Module PSCompression;
The term 'Install-Module' is not recognized as the name of a cmdlet, function, script file, or operable program. Check
the spelling of the name, or if a path was included, verify that the path is correct and try again.
    + CategoryInfo          : ObjectNotFound: (Install-Module:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException
```

PowerShell

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12;
Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force;
Install-Module PowerShellGet -MinimumVersion 2.2.4 -SkipPublisherCheck -Force -AllowClobber;


[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> Install-Module PSCompression;
The term 'Install-Module' is not recognized as the name of a cmdlet, function, script file, or operable program. Check
the spelling of the name, or if a path was included, verify that the path is correct and try again.
    + CategoryInfo          : ObjectNotFound: (Install-Module:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> # Get more information about the installed version of Po
werShell:
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> $PSVersionTable

Name                           Value
----                           -----
BuildVersion                   10.0.17763.107
WSManStackVersion              3.0
PSVersion                      5.1.17763.107
SerializationVersion           1.1.0.1
PSRemotingProtocolVersion      2.3
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
CLRVersion
PSEdition                      Core
```

PowerShell

```
Install-Module PowerShellGet -MinimumVersion 2.2.4 -SkipPublisherCheck -Force -AllowClobber;


[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> Install-Module PSCompression;
The term 'Install-Module' is not recognized as the name of a cmdlet, function, script file, or operable program. Check
the spelling of the name, or if a path was included, verify that the path is correct and try again.
    + CategoryInfo          : ObjectNotFound: (Install-Module:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> # Get more information about the installed version of Po
werShell:
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> $PSVersionTable

Name                           Value
----                           -----
BuildVersion                   10.0.17763.107
WSManStackVersion              3.0
PSVersion                      5.1.17763.107
SerializationVersion           1.1.0.1
PSRemotingProtocolVersion      2.3
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
CLRVersion
PSEdition                      Core


[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> # Note the PSVersion
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> #     and note the PSEdition!
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> Exit-PSSession
PS C:\Users\flesniak\Github\powershell-xplat>
```

Pow

PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> $strPathToInvokePowerShellDownloadScript = Join-Path $strCurrentFolder 'In
voke-PowerShellDownload.ps1'
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> $results = & $strPathToInvokePowerShellDownloadScript -PreferZIP -Windows
-ARM
```

PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> $strPathToInvokePowerShellDownloadScript = Join-Path $strCurrentFolder 'Invoke-PowerShellDownload.ps1'
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> $results = & $strPathToInvokePowerShellDownloadScript -PreferZIP -Windows -ARM
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> # Display the $results variable:
```

```
PS C:\Users\flesniak\Github\powershell-xplat> $results

DownloadSuccess                 : True
DownloadedFilePath              : C:\Users\flesniak\Downloads\PowerShell-7.3.12-win-arm32.zip
DownloadedVersion               : 7.3.12
FolderPathContainingDownload    : C:\Users\flesniak\Downloads
DownloadedFileName              : PowerShell-7.3.12-win-arm32.zip
DownloadedFileNameWithoutExtension : PowerShell-7.3.12-win-arm32
DownloadedFileType              : ZIP
```

PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> $strPathToInvokePowerShellDownloadScript = Join-Path $strCurrentFolder 'In
voke-PowerShellDownload.ps1'
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> $results = & $strPathToInvokePowerShellDownloadScript -PreferZIP -Windows
-ARM
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> # Display the $results variable:
PS C:\Users\flesniak\Github\powershell-xplat> $results

DownloadSuccess                 : True
DownloadedFilePath              : C:\Users\flesniak\Downloads\PowerShell-7.3.12-win-arm32.zip
DownloadedVersion               : 7.3.12
FolderPathContainingDownload    : C:\Users\flesniak\Downloads
DownloadedFileName              : PowerShell-7.3.12-win-arm32.zip
DownloadedFileNameWithoutExtension : PowerShell-7.3.12-win-arm32
DownloadedFileType              : ZIP

PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
```

```
PS C:\Users\flesniak\Github\powershell-xplat> # Transfer the downloaded file
PS C:\Users\flesniak\Github\powershell-xplat> Copy-Item ($results.DownloadedFilePath) -Destination $strWin10IoTCoreDownl
oadPathRoot -ToSession $PSSession -Force
Copying C:\Users\flesniak\Downloads\PowerShell-7.3.12-win-ar. [From localhost to 10.1.2.137                          ]
```

PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> # Before we transfer it, let's update the paths to reflect the correct one
s on the
PS C:\Users\flesniak\Github\powershell-xplat> # remote host
PS C:\Users\flesniak\Github\powershell-xplat> $strHostFolderPath = $results.FolderPathContainingDownload
PS C:\Users\flesniak\Github\powershell-xplat> $results.FolderPathContainingDownload = $strWin10IoTCoreDownloadPathRoot
PS C:\Users\flesniak\Github\powershell-xplat> $results.DownloadedFilePath = Join-Path $strWin10IoTCoreDownloadPathRoot (
$results.DownloadedFileName)
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> # Convert it to JSON:
PS C:\Users\flesniak\Github\powershell-xplat> $strPathToDownloadResultsJSONFile = Join-Path $strCurrentFolder 'PowerShel
lDownloadResults.json'
PS C:\Users\flesniak\Github\powershell-xplat> $results | ConvertTo-Json | Out-File -FilePath $strPathToDownloadResultsJS
ONFile -Force
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> # Transfer it
PS C:\Users\flesniak\Github\powershell-xplat> Copy-Item $strPathToDownloadResultsJSONFile -Destination $strWin10IoTCoreD
ownloadPathRoot -ToSession $PSSession -Force
PS C:\Users\flesniak\Github\powershell-xplat>
```

Pow

PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> # Before we transfer it, let's update the paths to reflect the correct one
s on the
PS C:\Users\flesniak\Github\powershell-xplat> # remote host
PS C:\Users\flesniak\Github\powershell-xplat> $strHostFolderPath = $results.FolderPathContainingDownload
PS C:\Users\flesniak\Github\powershell-xplat> $results.FolderPathContainingDownload = $strWin10IoTCoreDownloadPathRoot
PS C:\Users\flesniak\Github\powershell-xplat> $results.DownloadedFilePath = Join-Path $strWin10IoTCoreDownloadPathRoot (
$results.DownloadedFileName)
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> # Convert it to JSON:
PS C:\Users\flesniak\Github\powershell-xplat> $strPathToDownloadResultsJSONFile = Join-Path $strCurrentFolder 'PowerShel
lDownloadResults.json'
PS C:\Users\flesniak\Github\powershell-xplat> $results | ConvertTo-Json | Out-File -FilePath $strPathToDownloadResultsJS
ONFile -Force
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> # Transfer it
PS C:\Users\flesniak\Github\powershell-xplat> Copy-Item $strPathToDownloadResultsJSONFile -Destination $strWin10IoTCoreD
ownloadPathRoot -ToSession $PSSession -Force
PS C:\Users\flesniak\Github\powershell-xplat>
```

Pow

PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> # Before we transfer it, let's update the paths to reflect the correct one
s on the
PS C:\Users\flesniak\Github\powershell-xplat> # remote host
PS C:\Users\flesniak\Github\powershell-xplat> $strHostFolderPath = $results.FolderPathContainingDownload
PS C:\Users\flesniak\Github\powershell-xplat> $results.FolderPathContainingDownload = $strWin10IoTCoreDownloadPathRoot
PS C:\Users\flesniak\Github\powershell-xplat> $results.DownloadedFilePath = Join-Path $strWin10IoTCoreDownloadPathRoot (
$results.DownloadedFileName)
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> # Convert it to JSON:
PS C:\Users\flesniak\Github\powershell-xplat> $strPathToDownloadResultsJSONFile = Join-Path $strCurrentFolder 'PowerShel
lDownloadResults.json'
PS C:\Users\flesniak\Github\powershell-xplat> $results | ConvertTo-Json | Out-File -FilePath $strPathToDownloadResultsJS
ONFile -Force
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
```

```
PS C:\Users\flesniak\Github\powershell-xplat> # Transfer it
PS C:\Users\flesniak\Github\powershell-xplat> Copy-Item $strPathToDownloadResultsJSONFile -Destination $strWin10IoTCoreD
ownloadPathRoot -ToSession $PSSession -Force
```

PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> # Before we transfer it, let's update the paths to reflect the correct one
s on the
PS C:\Users\flesniak\Github\powershell-xplat> # remote host
PS C:\Users\flesniak\Github\powershell-xplat> $strHostFolderPath = $results.FolderPathContainingDownload
PS C:\Users\flesniak\Github\powershell-xplat> $results.FolderPathContainingDownload = $strWin10IoTCoreDownloadPathRoot
PS C:\Users\flesniak\Github\powershell-xplat> $results.DownloadedFilePath = Join-Path $strWin10IoTCoreDownloadPathRoot (
$results.DownloadedFileName)
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> # Convert it to JSON:
PS C:\Users\flesniak\Github\powershell-xplat> $strPathToDownloadResultsJSONFile = Join-Path $strCurrentFolder 'PowerShel
lDownloadResults.json'
PS C:\Users\flesniak\Github\powershell-xplat> $results | ConvertTo-Json | Out-File -FilePath $strPathToDownloadResultsJS
ONFile -Force
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> # Transfer it
PS C:\Users\flesniak\Github\powershell-xplat> Copy-Item $strPathToDownloadResultsJSONFile -Destination $strWin10IoTCoreD
ownloadPathRoot -ToSession $PSSession -Force
PS C:\Users\flesniak\Github\powershell-xplat>
```

```
PS C:\Users\flesniak\Github\powershell-xplat> # Put the downloaded file info back on the host just in case
PS C:\Users\flesniak\Github\powershell-xplat> $results.FolderPathContainingDownload = $strHostFolderPath
PS C:\Users\flesniak\Github\powershell-xplat> $results.DownloadedFilePath = Join-Path $strHostFolderPath ($results.Downl
oadedFileName)
```

PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> # Enter the PSSession for the IoT Core device
PS C:\Users\flesniak\Github\powershell-xplat> Enter-PSSession $PSSession
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> $strDownloadFolder = Get-DownloadFolder
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> Set-Location $strDownloadFolder
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> # Load the JSON file
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $strJSONFileName = 'PowerShellDownloadResults.json'
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $results = Get-Content (Join-Path $strDownloadFolder $strJSONFil
eName) | ConvertFrom-Json
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> # Look! We have our object!
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $results


DownloadSuccess                 : True
DownloadedFilePath              : C:\Data\Users\Administrator\Downloads\PowerShell-7.3.12-win-arm32.zip
DownloadedVersion               : 7.3.12
FolderPathContainingDownload    : C:\Data\Users\Administrator\Downloads
DownloadedFileName              : PowerShell-7.3.12-win-arm32.zip
DownloadedFileNameWithoutExtension : PowerShell-7.3.12-win-arm32
DownloadedFileType              : ZIP



[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
```

PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> # Enter the PSSession for the IoT Core device
PS C:\Users\flesniak\Github\powershell-xplat> Enter-PSSession $PSSession
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> $strDownloadFolder = Get-DownloadFolder
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> Set-Location $strDownloadFolder
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> # Load the JSON file
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $strJSONFileName = 'PowerShellDownloadResults.json'
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $results = Get-Content (Join-Path $strDownloadFolder $strJSONFil
eName) | ConvertFrom-Json
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> # Look! We have our object!
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $results


DownloadSuccess                 : True
DownloadedFilePath              : C:\Data\Users\Administrator\Downloads\PowerShell-7.3.12-win-arm32.zip
DownloadedVersion               : 7.3.12
FolderPathContainingDownload    : C:\Data\Users\Administrator\Downloads
DownloadedFileName              : PowerShell-7.3.12-win-arm32.zip
DownloadedFileNameWithoutExtension : PowerShell-7.3.12-win-arm32
DownloadedFileType              : ZIP



[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
```

PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> # Enter the PSSession for the IoT Core device
PS C:\Users\flesniak\Github\powershell-xplat> Enter-PSSession $PSSession
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> $strDownloadFolder = Get-DownloadFolder
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> Set-Location $strDownloadFolder
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> # Load the JSON file
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $strJSONFileName = 'PowerShellDownloadResults.json'
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $results = Get-Content (Join-Path $strDownloadFolder $strJSONFil
eName) | ConvertFrom-Json
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> # Look! We have our object!
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $results


DownloadSuccess                 : True
DownloadedFilePath              : C:\Data\Users\Administrator\Downloads\PowerShell-7.3.12-win-arm32.zip
DownloadedVersion               : 7.3.12
FolderPathContainingDownload    : C:\Data\Users\Administrator\Downloads
DownloadedFileName              : PowerShell-7.3.12-win-arm32.zip
DownloadedFileNameWithoutExtension : PowerShell-7.3.12-win-arm32
DownloadedFileType              : ZIP



[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
```

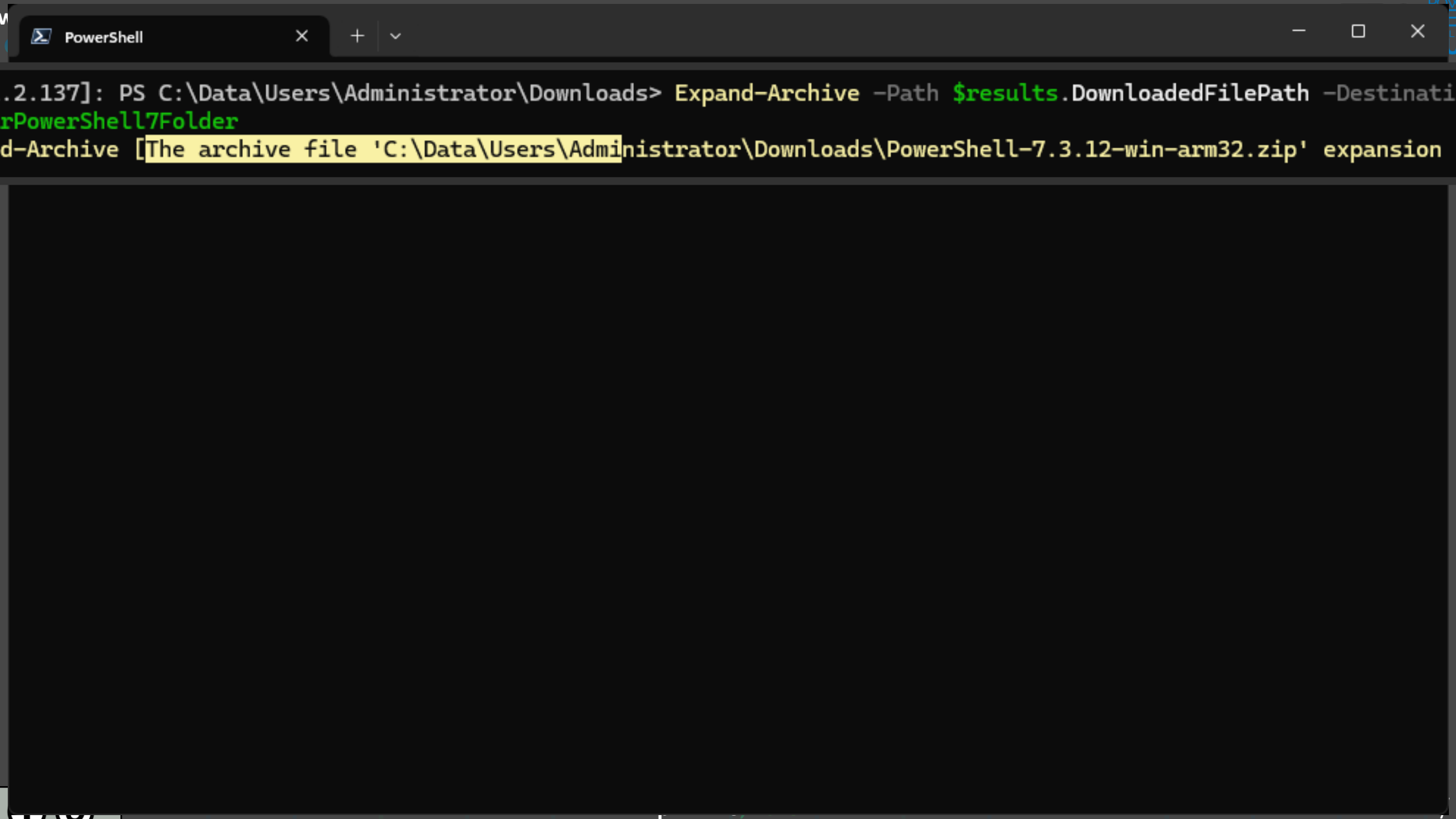
PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> # Enter the PSSession for the IoT Core device
PS C:\Users\flesniak\Github\powershell-xplat> Enter-PSSession $PSSession
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> $strDownloadFolder = Get-DownloadFolder
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> Set-Location $strDownloadFolder
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> # Load the JSON file
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $strJSONFileName = 'PowerShellDownloadResults.json'
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $results = Get-Content (Join-Path $strDownloadFolder $strJSONFil
eName) | ConvertFrom-Json
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> # Look! We have our object!
```

```
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $results


DownloadSuccess                 : True
DownloadedFilePath              : C:\Data\Users\Administrator\Downloads\PowerShell-7.3.12-win-arm32.zip
DownloadedVersion               : 7.3.12
FolderPathContainingDownload    : C:\Data\Users\Administrator\Downloads
DownloadedFileName              : PowerShell-7.3.12-win-arm32.zip
DownloadedFileNameWithoutExtension : PowerShell-7.3.12-win-arm32
DownloadedFileType              : ZIP
```

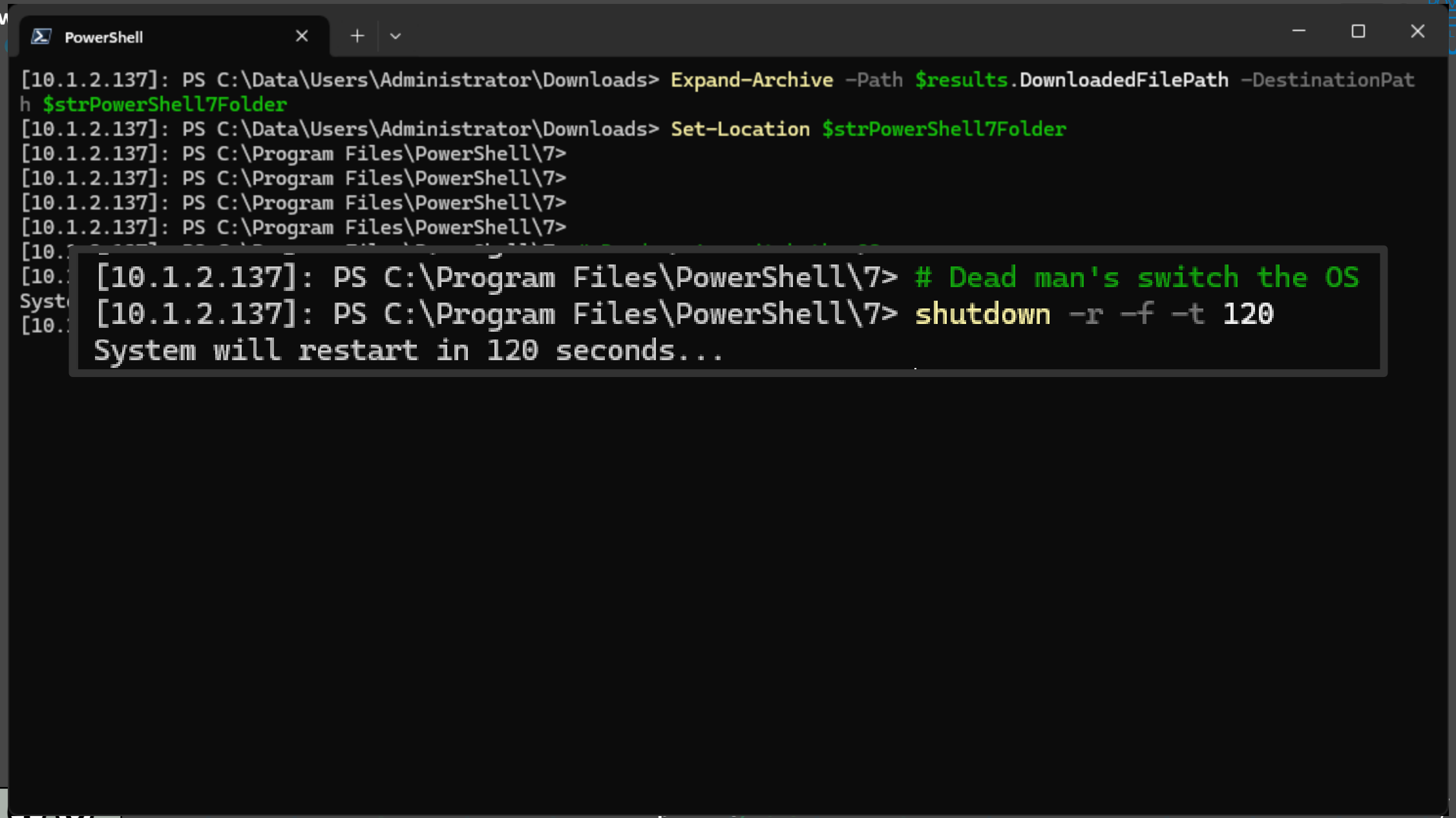
PowerShell

```
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> # Get the Program Files path - note: this technique is not fully
 backward compatible!
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $strProgramFilesFolderPath = $env:ProgramW6432
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> if ([string]::IsNullOrEmpty($strProgramFilesFolderPath)) {
>>     $strProgramFilesFolderPath = $env:ProgramFiles
>>     if ([string]::IsNullOrEmpty($strProgramFilesFolderPath)) {
>>         Write-Warning 'Uh, we couldn''t find a Program Files folder?'
>>     }
>> }
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> # Determine the future folder for PowerShell 7
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $strMajorVersion = [string](([version]$results.DownloadedVersion
).Major)
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $strPowerShell7Folder = Join-Path $strProgramFilesFolderPath 'Po
werShell'
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $strPowerShell7Folder = Join-Path $strPowerShell7Folder $strMajo
rVersion
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads>
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $strPowerShell7Folder
C:\Program Files\PowerShell\7
```


PowerShell
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> Expand-Archive -Path $results.DownloadedFilePath -DestinationPath $strPowerShell7Folder
Expand-Archive [The archive file 'C:\Data\Users\Administrator\Downloads\PowerShell-7.3.12-win-arm32.zip' expansion i.]

PowerShell

```
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> Expand-Archive -Path $results.DownloadedFilePath -DestinationPath $strPowerShell7Folder
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> Set-Location $strPowerShell7Folder
[10.1.2.137]: PS C:\Program Files\PowerShell\7>
[10.1.2.137]: PS C:\Program Files\PowerShell\7>
[10.1.2.137]: PS C:\Program Files\PowerShell\7> # Dead man's switch the OS
[10.1.2.137]: PS C:\Program Files\PowerShell\7> shutdown -r -f -t 120
System will restart in 120 seconds...
[10.1.2.137]: PS C:\Program Files\PowerShell\7>
```


PowerShell
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> Expand-Archive -Path $results.DownloadedFilePath -DestinationPath $strPowerShell7Folder
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> Set-Location $strPowerShell7Folder
[10.1.2.137]: PS C:\Program Files\PowerShell\7>
[10.1.2.137]: PS C:\Program Files\PowerShell\7>
[10.1.2.137]: PS C:\Program Files\PowerShell\7>
[10.1.2.137]: PS C:\Program Files\PowerShell\7>
[10.1.2.137]: PS C:\Program Files\PowerShell\7> # Dead man's switch the OS
[10.1.2.137]: PS C:\Program Files\PowerShell\7> shutdown -r -f -t 120
System will restart in 120 seconds...

PowerShell

```
[10.1.2.137]: PS C:\Program Files\PowerShell\7> # Enable PowerShell 7 remoting:
[10.1.2.137]: PS C:\Program Files\PowerShell\7> .\Install-PowerShellRemoting.ps1 -PowerShellHome .
VERBOSE: PowerShellHome: .
VERBOSE: Using PowerShell Version: 7.3.12
VERBOSE: Performing the operation "Copy File" on target "Item: C:\Program Files\PowerShell\7\pwrshplugin.dll Destination
: C:\Windows\System32\PowerShell\7.3.12\pwrshplugin.dll".
VERBOSE: Created Plugin Config File: C:\Windows\System32\PowerShell\7.3.12\RemotePowerShellConfig.txt

Get-PSSessionConfiguration PowerShell.7.3.12


Name          : PowerShell.7.3.12
PSVersion     : 7.0
StartupScript :
RunAsUser     :
Permission    : BUILTIN\Administrators AccessAllowed

VERBOSE: PowerShellHome: .
VERBOSE: Using PowerShell Version: 7
VERBOSE: Performing the operation "Copy File" on target "Item: C:\Program Files\PowerShell\7\pwrshplugin.dll Destination
: C:\Windows\System32\PowerShell\7\pwrshplugin.dll".
VERBOSE: Created Plugin Config File: C:\Windows\System32\PowerShell\7\RemotePowerShellConfig.txt

Get-PSSessionConfiguration PowerShell.7
Processing data from remote server 10.1.2.137 failed with the following error message: The I/O operation has been aborte
d because of either a thread exit or an application request. For more information, see the about_Remote_Troubleshooting
Help topic.
[10.1.2.137]: PS>
```

PowerShell

```
[10.1.2.137]: PS C:\Program Files\PowerShell\7>
[10.1.2.137]: PS C:\Program Files\PowerShell\7>
[10.1.2.137]: PS C:\Program Files\PowerShell\7> # Enable PowerShell 7 remoting:
[10.1.2.137]: PS C:\Program Files\PowerShell\7> .\Install-PowerShellRemoting.ps1 -PowerShellHome .
VERBOSE: PowerShellHome: .
VERBOSE: Using PowerShell Version: 7.3.12
VERBOSE: Performing the operation "Copy File" on target "Item: C:\Program Files\PowerShell\7\pwrshplugin.dll Destination
: C:\Windows\System32\PowerShell\7.3.12\pwrshplugin.dll".
VERBOSE: Created Plugin Config File: C:\Windows\System32\PowerShell\7.3.12\RemotePowerShellConfig.txt

Get-PSSessionConfiguration PowerShell.7.3.12


Name          : PowerShell.7.3.12
PSVersion     : 7.0
StartupScript :
RunAsUser     :
Permission    : BUILTIN\Administrators AccessAllowed

VERBOSE: PowerShellHome: .
VERBOSE: Using PowerShell Version: 7
VERBOSE: Performing the operation "Copy File" on target "Item: C:\Program Files\PowerShell\7\pwrshplugin.dll Destination
: C:\Windows\System32\PowerShell\7\pwrshplugin.dll".
VERBOSE: Created Plugin Config File: C:\Windows\System32\PowerShell\7\RemotePowerShellConfig.txt

Get-PSSessionConfiguration PowerShell.7
```

```
Processing data from remote server 10.1.2.137 failed with the following error message: The I/O operation has been aborte
d because of either a thread exit or an application request. For more information, see the about_Remote_Troubleshooting
```

```
[10.1.2.137]: PS>
```

Pow

PowerShell

```
$strMajorVersion = [string](([version]$results.DownloadedVersion).Major)
$strConfigurationName = 'PowerShell.' + $strMajorVersion
```

```
PS C:\U                                                                    or)
PS C:\U
PS C:\U
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> $PSSession = New-PSSession -ComputerName $strWin10IoTCoreDeviceIPOrHostnam
e -Credential $credential -ConfigurationName $strConfigurationName
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> Enter-PSSession $PSSession
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
```

PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> $strMajorVersion = [string](([version]$results.DownloadedVersion).Major)
PS C:\Users\flesniak\Github\powershell-xplat> $strConfigurationName = 'PowerShell.' + $strMajorVersion
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> $PSSession = New-PSSession -ComputerName $strWin10IoTCoreDeviceIPOrHostname -Credential $credential -ConfigurationName $strConfigurationName
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> Enter-PSSession $PSSession
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
```

PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> $strMajorVersion = [string](([version]$results.DownloadedVersion).Major)
PS C:\Users\flesniak\Github\powershell-xplat> $strConfigurationName = 'PowerShell.' + $strMajorVersion
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> $PSSession = New-PSSession -ComputerName $strWin10IoTCoreDeviceIPOrHostnam
e -Credential $credential -ConfigurationName $strConfigurationName
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\
[10.1.2.137]
```

```
PS C:\Users\flesniak\Github\powershell-xplat> Enter-PSSession $PSSession
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
```

PowerShell

```
PS C:\Users\flesniak\Github\powershell-xplat> $strMajorVersion = [string](([version]$results.DownloadedVersion).Major)
PS C:\Users\flesniak\Github\powershell-xplat> $strConfigurationName = 'PowerShell.' + $strMajorVersion
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> $PSSession = New-PSSession -ComputerName $strWin10IoTCoreDeviceIPOrHostnam
e -Credential $credential -ConfigurationName $strConfigurationName
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat>
PS C:\Users\flesniak\Github\powershell-xplat> Enter-PSSession $PSSession
[10.1.2.137]:

Name
----
PSVersion
PSEdition
GitCommitId
OS
Platform
PSCompatibleV
PSRemotingPro
Serialization
WSManStackVer

[10.1.2.137]:
```

```
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> $PSVersionTable

Name                           Value
----                           -----
PSVersion                      7.3.12
PSEdition                      Core
GitCommitId                    7.3.12
OS                             Microsoft Windows 10.0.17763
Platform                       Win32NT
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0…}
PSRemotingProtocolVersion
SerializationVersion           1.1.0.1
WSManStackVersion              3.0
```

```
> function Get-DownloadFolder {…
  }
```

Pow

PowerShell

```
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> $strDownloadFolder = Get-DownloadFolder
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> $strPathToTestPowerShellModuleInstalledScript = Join-Path $strDownloadFolder 'Test-PowerShellModuleInstalled.ps1'
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> # This script checks if the PSCompression module is installed:
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> & $strPathToTestPowerShellModuleInstalledScript
WARNING: PSCompression module not found. Please install it and then try again.
You can install the PSCompression PowerShell module from the PowerShell Gallery by running the following command:
Install-Module PSCompression;

If the installation command fails, you may need to upgrade the version of PowerShellGet. To do so, run the following commands, then restart PowerShell:
Set-ExecutionPolicy Bypass -Scope Process -Force;
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12;
Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force;
Install-Module PowerShellGet -MinimumVersion 2.2.4 -SkipPublisherCheck -Force -AllowClobber;


[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
```

PowerShell

```
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> $strDownloadFolder = Get-DownloadFolder
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> $strPathToTestPowerShellModuleInstalledScript = Join-Path $strDo
wnloadFolder 'Test-PowerShellModuleInstalled.ps1'
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> # This script checks if the PSCompression module is installed:
```

```
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> & $strPathToTestPowerShellModuleInstalledScript
WARNING: PSCompression module not found. Please install it and then try again.
You can install the PSCompression PowerShell module from the PowerShell Gallery by running the following command:
Install-Module PSCompression;

If the installation command fails, you may need to upgrade the version of PowerShellGet. To do so, run the following com
mands, then restart PowerShell:
Set-ExecutionPolicy Bypass -Scope Process -Force;
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12;
Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force;
Install-Module PowerShellGet -MinimumVersion 2.2.4 -SkipPublisherCheck -Force -AllowClobber;
```

Pow

PowerShell

```
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> $strDownloadFolder = Get-DownloadFolder
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> $strPathToTestPowerShellModuleInstalledScript = Join-Path $strDo
wnloadFolder 'Test-PowerShellModuleInstalled.ps1'
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> # This script checks if the PSCompression module is installed:
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> & $strPathToTestPowerShellModuleInstalledScript
WARNING: PSCompression module not found. Please install it and then try again.
You can install the PSCompression PowerShell module from the PowerShell Gallery by running the following command:
Install-Module PSCompression;

If the installation command fails, you may need to upgrade the version of PowerShellGet. To do so, run the following com
mands, then restart PowerShell:
Set-ExecutionPolicy Bypass -Scope Process -Force;
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12;
Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force;
Install-Module PowerShellGet -MinimumVersion 2.2.4 -SkipPublisherCheck -Force -AllowClobber;
```

```
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> Install-Module PSCompression;

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository, change its
InstallationPolicy value by running the Set-PSRepository cmdlet. Are you sure you want to install the modules from
'PSGallery'?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [?] Help (default is "N"): y
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
```

PowerShell

```
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> Get-ComputerInfo

WindowsBuildLabEx                           : 17763.107.armfre.rs5_release_svc_prod2.181026-1406
WindowsCurrentVersion                       : 6.3
WindowsEditionId                            : IoTUAP
WindowsInstallationType                     : IoTCore
WindowsInstallDateFromRegistry              : 1/1/1970 12:00:00 AM
WindowsProductId                            :
WindowsProductName                          : IoTUAP
WindowsRegisteredOrganization               :
WindowsRegisteredOwner                      :
WindowsSystemRoot                           : C:\windows
WindowsVersion                              : 1511
WindowsUBR                                  : 107
BiosCharacteristics                         : {11, 12, 16, 32…}
BiosBIOSVersion                             : {BC2836 - 1, 0.1, Pi2 Board EFI Apr 14 2017 14:13:26 - 0}
BiosBuildNumber                             :
BiosCaption                                 : 0.1
BiosCodeSet                                 :
BiosCurrentLanguage                         :
BiosDescription                             : 0.1
BiosEmbeddedControllerMajorVersion          : 255
BiosEmbeddedControllerMinorVersion          : 255
BiosFirmwareType                            : Uefi
BiosIdentificationCode                      :
BiosInstallableLanguages                    :
BiosInstallDate                             :
BiosLanguageEdition                         :
BiosListOfLanguages                         :
BiosManufacturer                            : Microsoft Corp.
```

PowerShell

```
CsCurrentTimeZone                       : -420
CsDaylightInEffect                      : True
CsDescription                           : ARM processor family
CsDNSHostName                           : minwinpc
CsDomain                                : WORKGROUP
CsDomainRole                            : StandaloneWorkstation
CsEnableDaylightSavingsTime             : True
CsFrontPanelResetStatus                 : Unknown
CsHypervisorPresent                     : False
CsInfraredSupported                     : False
CsInitialLoadInfo                       :
CsInstallDate                           :
CsManufacturer                          : Raspberry Pi
CsModel                                 : Raspberry Pi 2 Model B
CsName                                  : MINWINPC
CsNetworkAdapters                       : {}
CsNetworkServerModeEnabled              : True
CsNumberOfLogicalProcessors             : 4
CsNumberOfProcessors                    : 4
CsProcessors                            : {ARM Family 7 Model C07 Revision   5, ARM Family 7 Model C07
                                          Revision   5, ARM Family 7 Model C07 Revision   5, ARM
                                          Family 7 Model C07 Revision   5}
CsOEMStringArray                        :
CsPartOfDomain                          : False
CsPauseAfterReset                       : -1
CsPCSystemType                          : Mobile
CsPCSystemTypeEx                        : Slate
CsPowerManagementCapabilities           :
```

PowerShell

```
PS C:\Users\flesniak> Enter-PSSession -Session $PSSession
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> . .\Invoke-CrossPlatformOSInventory.ps1; Invoke-CrossPlatformOSI
nventory


OSType                  : Windows
OSID                    : Windows
OSName                  : Windows Core System
OSPrettyName            : Windows Core System
OSVersionString         : 10.0.17763.1577
OSVersionMajorString    : 10
OSVersionMajor          : 10
OSVersionMinorString    : 0
OSVersionMinor          : 0
OSVersionBuild          : 17763
OSVersionPatchString    : 0
OSVersionPatch          : 0
OSVersionRevisionString : 1577
OSVersionRevision       : 1577
OSBuildString           : 17763
OSServicePack           : 0
OSEdition               :
OSSKU                   : 123
OSCodename              :
OSKernelVersion         : 10.0.17763.1577
OSArchitecture          : ARM



[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
```

Architectures

- In a roundabout way, we've already hit on "alternative processor architectures" a bit…
- PowerShell runs on:
  - Intel/AMD x86
  - Intel/AMD x86-64 (also known as AMD64 or x64)
  - Intel Itanium (IA64)
  - ARM (32-bit)
  - ARM (64-bit) (also known as ARM64)

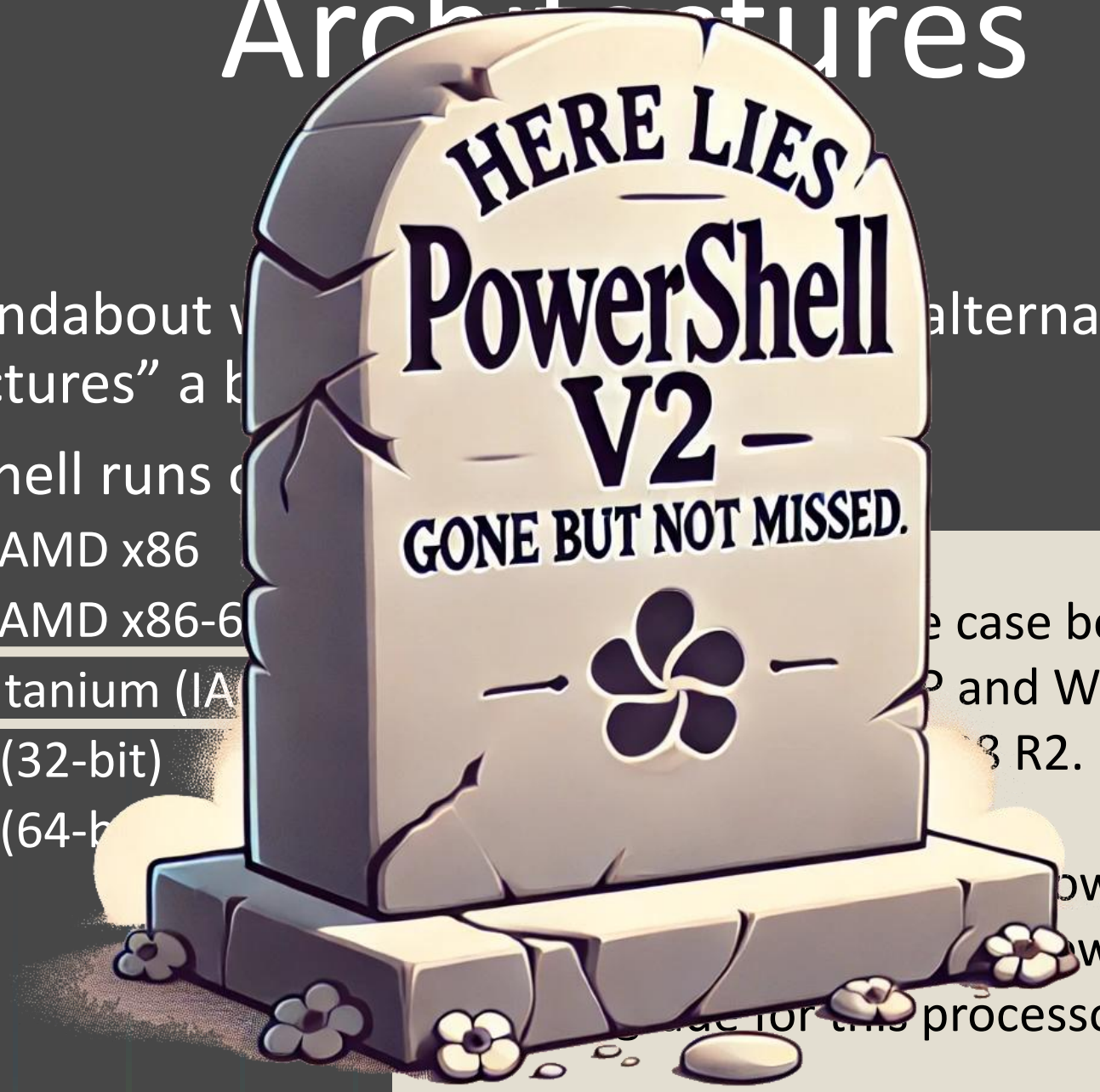
Architectures

- In a roundabout way, we've already hit on "alternative processor architectures" a bit…
- PowerShell runs on:
  - Intel/AMD x86
  - Intel/AMD x86-64 (also
  - Intel Itanium (IA64)
  - ARM (32-bit)
  - ARM (64-bit) (also knov

Itanium is a fun edge case because it was only around for Windows XP and Windows Server 2003-2008 R2.

So, the "in-box" version of PowerShell is v1 or v2 – Microsoft did not offer a PowerShell v3 or later upgrade for this processor architecture

Architectures



- In a roundabout v alternative processor architectures" a b
- PowerShell runs o
  - Intel/AMD x86
  - Intel/AMD x86-6
  - Intel Itanium (IA
  - ARM (32-bit)
  - ARM (64-b

e case because it was only P and Windows Server 2003- 3 R2.

owerShell is v1 or v2 – owerShell v3 or later processor architecture

Architectures

- In a roundabout way, we've already hit on "alternative processor architectures" a bit…
- PowerShell runs on:
  - Intel/AMD x86
  - Intel/AMD x86-64 (also
  - Intel Itanium (IA64)
  - ARM (32-bit)
  - ARM (64-bit) (also knov

32-bit ARM is also interesting because Microsoft launched the Surface and Surface 2 (RT) devices in the Windows 8 / Windows 8.1 era and didn't offer an upgrade to Windows 10.

These devices have PowerShell, but many cmdlets are neutered because of the "locked" nature of the OS

Architectures

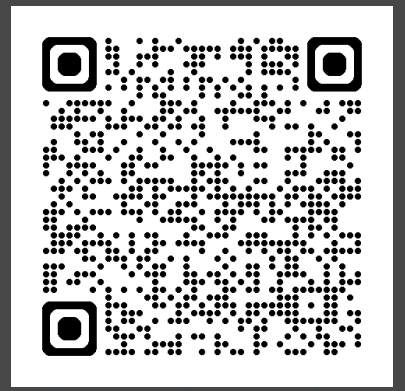
- In a roundabout way, we've already hit on "alternative processor architectures" a bit…
- PowerShell runs on:
  - Intel/AMD x86
  - Intel/AMD x86-64 (also known as AMD64 or x64)
  - Intel Itanium (IA64)
  - ARM (32-bit)
  - ARM (64-bit) (also known as ARM64)

Avoiding Pitfalls

- We know that we could encounter a range of PowerShell versions, and we know there are differences across platforms. How do we avoid these pitfalls?
  - VSCode
  - The PowerShell extension
  - …plus some special configuration

Code for Compatibility

- Write, place, and configure a PSScriptAnalyzer file, and VSCode will tell you if you're being naughty!
- We have one pre-written that highlights all known, available backward-compatibility and cross-platform compatibility issues: https://github.com/franklesniak/PowerShell_Resources/blob/master/PSScriptAnalyzerSettings.psd1



PowerShell + DevOps
Global Summit

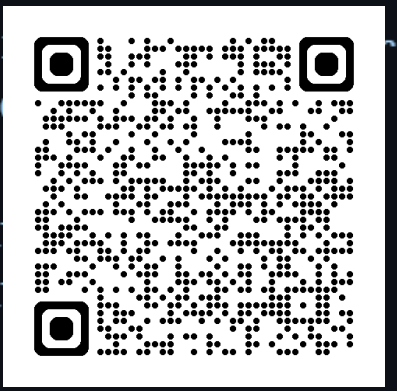
```
PSUseCompatibleCmdlets = @{
    compatibility = @(
        'desktop-2.0-windows',
        'desktop-3.0-windows',
        'desktop-4.0-windows',
        'desktop-5.1.14393.206-windows',
        'core-6.1.0-windows',
        'core-6.1.0-linux',
        'core-6.1.0-linux-arm',
        'core-6.1.0-macos'
    )
}
```

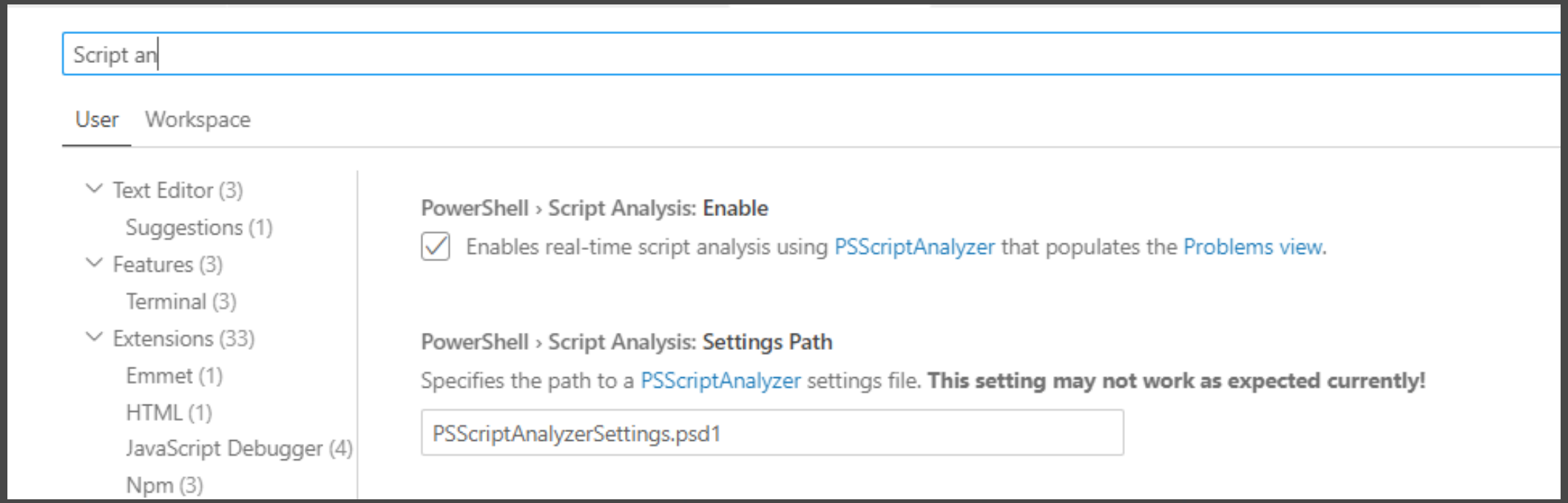
```
PSUseCompatibleCommands = @{
    # Turn the rule on
    Enable = $true

    # List the PowerShell platforms with which we want to check compatibility
    TargetProfiles = @(
        'ubuntu_x64_18.04_6.2.4_x64_4.0.30319.42000_core',
        'ubuntu_x64_18.04_7.0.0_x64_3.1.2_core',
        'win-48_x64_10.0.17763.0_5.1.17763.316_x64_4.0.30319.42000_framework',
        'win-4_x64_10.0.18362.0_6.2.4_x64_4.0.30319.42000_core',
        'win-4_x64_10.0.18362.0_7.0.0_x64_3.1.2_core',
        'win-8_x64_10.0.14393.0_5.1.14393.2791_x64_4.0.30319.42000_framework',
        'win-8_x64_10.0.14393.0_6.2.4_x64_4.0.30319.42000_core',
        'win-8_x64_10.0.14393.0_7.0.0_x64_3.1.2_core',
        'win-8_x64_10.0.17763.0_5.1.17763.316_x64_4.0.30
        'win-8_x64_10.0.17763.0_6.2.4_x64_4.0.30319.420
        'win-8_x64_10.0.17763.0_7.0.0_x64_3.1.2_core',
        'win-8_x64_6.2.9200.0_3.0_x64_4.0.30319.42000_fr
        'win-8_x64_6.3.9600.0_4.0_x64_4.0.30319.42000_fr
    )
}
```

PowerShell + DevOps
Global Summit

```
PSUseCompatibleSyntax = @{
    # Turn the rule on
    Enable = $true

    # List the targeted versions of PowerShell
    TargetVersions = @(
        '3.0',
        '4.0',
        '5.0',
        '6.0',
        '7.0'
    )
}
```

```
UseCompatibleTypes = @{
    # Turn the rule on
    Enable = $true

    # List the PowerShell platforms with which we want to check compatibility
    TargetProfiles = @(
        'ubuntu_x64_18.04_6.2.4_x64_4.0.30319.42000_core',
        'ubuntu_x64_18.04_7.0.0_x64_3.1.2_core',
        'win-48_x64_10.0.17763.0_5.1.17763.316_x64_4.0.30319.42000_framework',
        'win-4_x64_10.0.18362.0_6.2.4_x64_4.0.30319.42000_core',
        'win-4_x64_10.0.18362.0_7.0.0_x64_3.1.2_core',
        'win-8_x64_10.0.14393.0_5.1.14393.2791_x64_4.0.30319.42000_framework',
        'win-8_x64_10.0.14393.0_6.2.4_x64_4.0.30319.42000_core',
        'win-8_x64_10.0.14393.0_7.0.0_x64_3.1.2_core',
        'win-8_x64_10.0.17763.0_5.1.17763.316_x64_4.0.
        'win-8_x64_10.0.17763.0_6.2.4_x64_4.0.30319.42
        'win-8_x64_10.0.17763.0_7.0.0_x64_3.1.2_core',
        'win-8_x64_6.2.9200.0_3.0_x64_4.0.30319.42000_
        'win-8_x64_6.3.9600.0_4.0_x64_4.0.30319.42000_
    )
}
```




Script an
User
Workspace
Text Editor (3)
Suggestions (1)
Features (3)
Terminal (3)
Extensions (33)
Emmet (1)
HTML (1)
JavaScript Debugger (4)
Npm (3)
PowerShell › Script Analysis: Enable
Enables real-time script analysis using PSScriptAnalyzer that populates the Problems view.
PowerShell › Script Analysis: Settings Path
Specifies the path to a PSScriptAnalyzer settings file. This setting may not work as expected currently!
PSScriptAnalyzerSettings.psd1

```
The command 'Get-ComputerInfo' is not available by default in PowerShell version '6.2.4' on platform 'Ubuntu 18.04.4
LTS' PSScriptAnalyzer(PSUseCompatibleCommands)

The command 'Get-ComputerInfo' is not available by default in PowerShell version '7.0.0' on platform 'Ubuntu 18.04.4
LTS' PSScriptAnalyzer(PSUseCompatibleCommands)

The command 'Get-ComputerInfo' is not available by default in PowerShell version '3.0' on platform 'Microsoft Windows Server 2012
Datacenter' PSScriptAnalyzer(PSUseCompatibleCommands)

The command 'Get-ComputerInfo' is not available by default in PowerShell version '4.0' on platform 'Microsoft Windows Server 2012 R2
Datacenter' PSScriptAnalyzer(PSUseCompatibleCommands)

Get-ComputerInfo

Gets a consolidated object of system and operating system properties.

Get-ComputerInfo
```

Recap: Objectives

1. Understand PowerShell's Cross-Platform Capabilities:

✓Learn how PowerShell operates across various operating systems and hardware platforms, including Windows, macOS, Linux distributions, Azure Cloud Shell, Raspberry Pi, ARM devices, and historical platforms like Itanium.

Recap: Objectives

(continued)

2. Gain Practical Skills for Cross-Platform Scripting:

✓Identify key differences in PowerShell's behavior on different platforms.

✓Acquire practical tips, tricks, and best practices for writing effective cross-platform scripts.

Recap: Objectives

(continued)

3. Apply PowerShell in Diverse Environments:

✓Be inspired to utilize PowerShell's full potential in various settings, from managing cloud infrastructure and automating IoT projects to experimenting with different hardware.

✓Enhance both professional and personal projects by harnessing PowerShell's versatility.

Questions + Audience Discussion

THANK YOU

Feedback is a gift. Please review this session