PowerShell Everywhere - Unleashing the Power of Cross-Platform Scripting

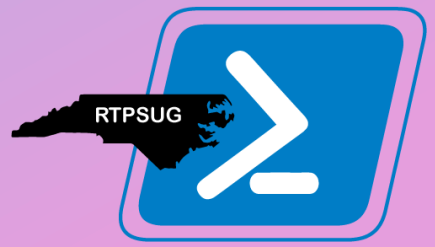
Running PowerShell in exotic systems

Frank Lesniak

X: @FrankLesniak

Danny Stutz

X: @danny_stutz



Command Prompt

```
C:\Users\flesniak>whoami
Frank Lesniak (@franklesniak)
Sr. Cybersecurity & Enterprise Technology Architect at West Monroe

Experience:
Almost 20 years in consulting. Microsoft 365 (modern work) consulting team lead.

Credentials:
MCSEs. PowerShellin' since 2006.
```

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


C:\Users\flesniak>
```

Command Prompt

```
C:\Users\dstutz>whoami
Danny Stutz (@danny_stutz)
Cybersecurity + Enterprise Technology Architect at West Monroe

Experience:
5 1/2 years working as a Microsoft 365/Entra/Azure technical lead

Credentials:
West Monroe PowerShell Interest Group lead.
```

```
**********************************************************************************************
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 security
- Carve-out and Divestiture migration work (where I also spend the majority of my consulting time)
- My music taste (send me your favorite tunes on Discord!)

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

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


C:\Users\dstutz>
```

Thank you to our sponsors!

System Frontier

TEKsystems
Own change

By attending this talk, participants will:

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.

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.

Apply PowerShell in Diverse Environments

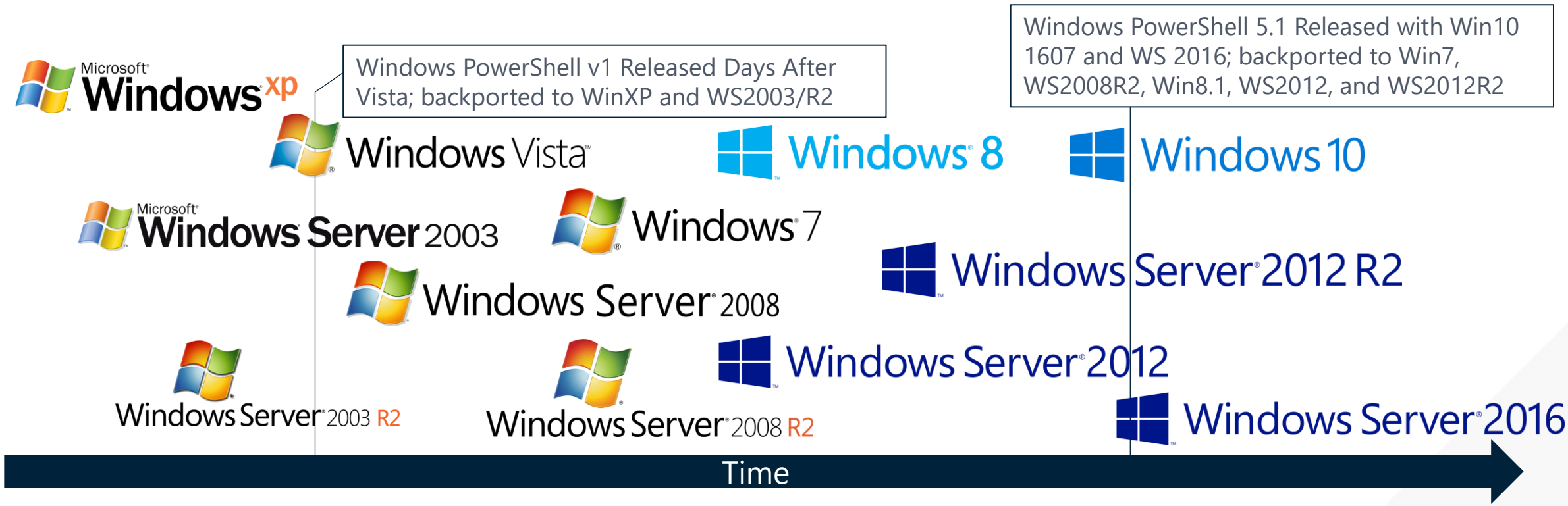
- Be inspired to utilize PowerShell's full potential in various settings, from managing cloud infrastructure and automating IoT projects to experimenting with different hardware.
- Enhance both professional and personal projects by harnessing PowerShell's versatility.

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



The year was 2018…

- confirmed the Spectre/Meltdown vulnerabilities, the first in a series that slowed down our computers by 15-25%!
- ached a record high
- ll – well, Po Windows:

After Vista; S2003/R2

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

2003
Windows 7
Win ver 2008
Windows 8
Windows Server 2012 R2
Windows Server 2012
ows Server 2008 R2
Windows 10
Windows Server 2016
Time

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: PowerShell 7.4.5 still supports a large variety of operating systems:

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 22H2 and newer + LTSC releases (side by side w/ PowerShell 5.1)
- Windows Server 2016 and newer (side by side w/ PowerShell 5.1)
- macOS 12 and newer
- Ubuntu 20.04 and newer long-term support releases; interim releases (23.10) may work but are limited to community support
- Debian 12
- Red Hat Enterprise Linux (RHEL) 8 and newer
- Alpine 3.17 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: PowerShell 7.4.5 still supports a large variety of operating systems:

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 22H2 and newer + LTSC releases (side by side w/ PowerShell 5.1)
- Windows Server 2016 and newer (side by side w/ PowerShell 5.1)
- macOS 12 and newer
- Ubuntu 20.04 and newer long-term support releases; interim releases (23.10) may work but are limited to community support
- Debian 12
- Red Hat Enterprise Linux (RHEL) 8 and newer
- Alpine 3.17 and newer
- SLES and OpenSUSE*
- Arch Linux*
- Kali Linux*
- Raspberry Pi OS*
- Other Linux distributions?

(note that CentOS and Fedora were dropped)

CentOS was dropped because Red Hat discontinued it in 2021.

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

* = Only Community-Supported (or "Experimental" Support)

In other words, if you're running one of these, you can/are running PowerShell!

What's so great about "PowerShell 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

D

Since containers are the backbone of many cloud services, since (most) containers run Linux, and since you can run PowerShell on Linux, you can run PowerShell in the cloud!

(Azure Cloud Shell)

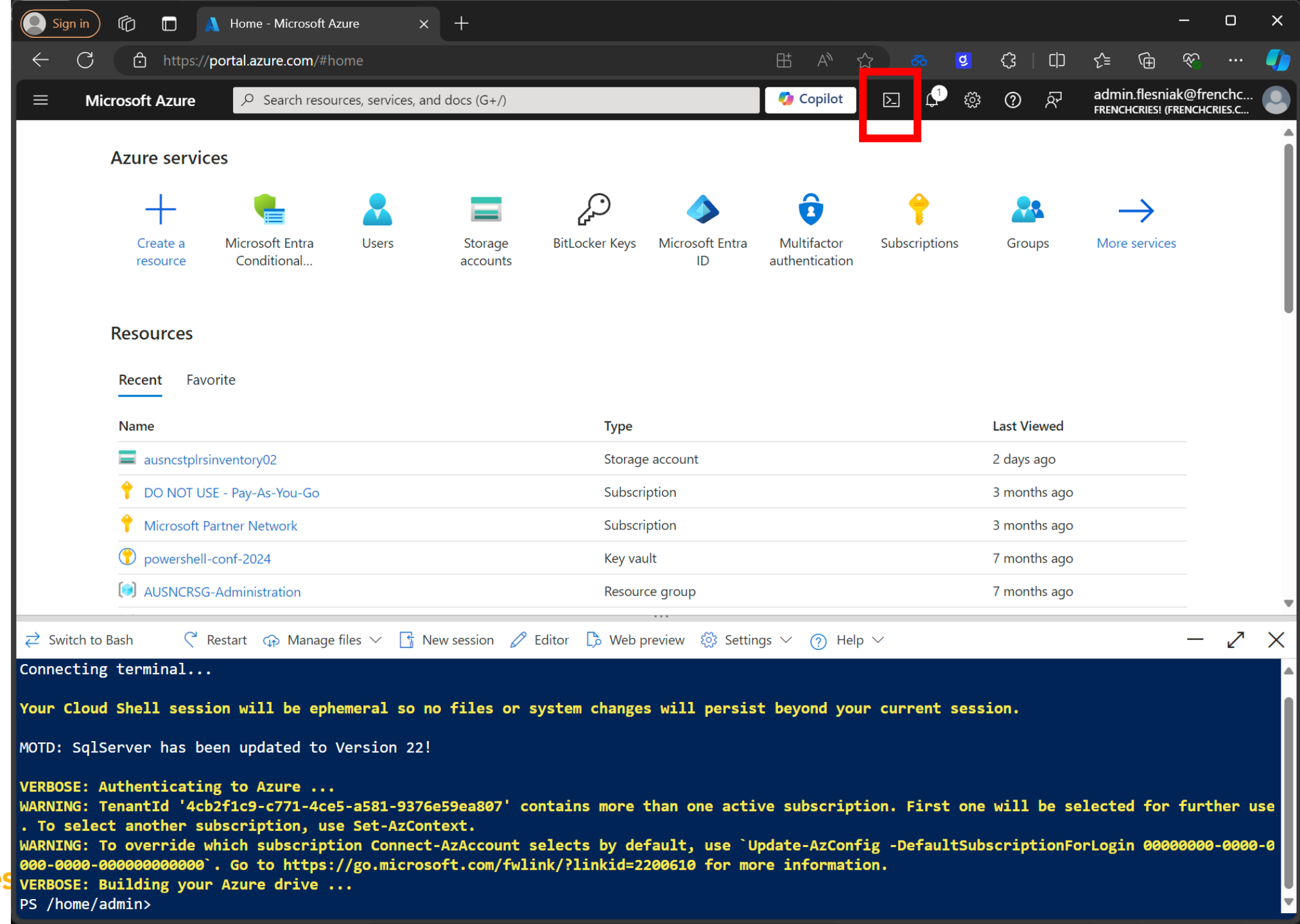
Demo – Cloud Shell

What type of OS?

What modules?

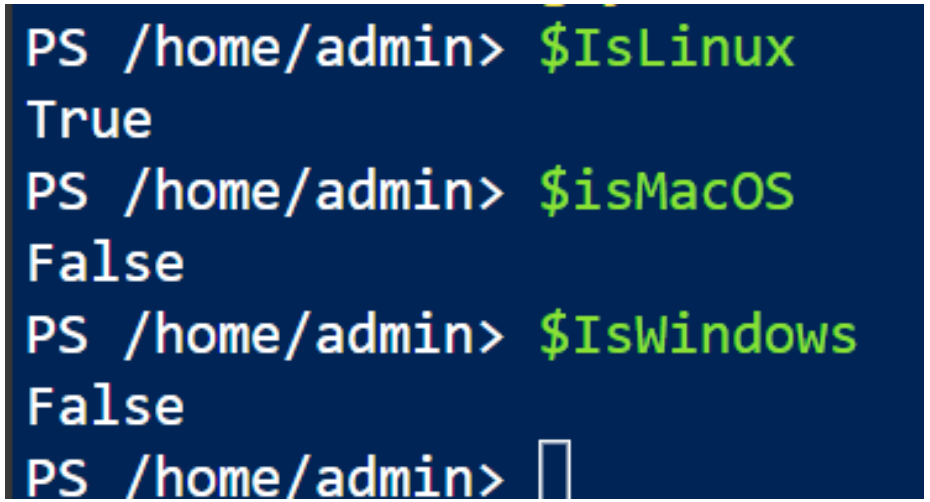
OK, but for real, what OS is this?

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


Sign in
Home - Microsoft Azure
https://portal.azure.com/#home
Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
admin.flesniak@frenchc...
FRENCHCRIES! (FRENCHCRIES.C...
Azure services
Create a resource
Microsoft Entra Conditional...
Users
Storage accounts
BitLocker Keys
Microsoft Entra ID
Multifactor authentication
Subscriptions
Groups
More services
Resources
Recent
Favorite
Name
Type
Last Viewed
ausncstplrsinventory02
Storage account
2 days ago
DO NOT USE - Pay-As-You-Go
Subscription
3 months ago
Microsoft Partner Network
Subscription
3 months ago
powershell-conf-2024
Key vault
7 months ago
AUSNCRSG-Administration
Resource group
7 months ago
Switch to Bash
Restart
Manage files
New session
Editor
Web preview
Settings
Help
Connecting terminal...
Your Cloud Shell session will be ephemeral so no files or system changes will persist beyond your current session.
MOTD: SqlServer has been updated to Version 22!
VERBOSE: Authenticating to Azure ...
WARNING: TenantId '4cb2f1c9-c771-4ce5-a581-9376e59ea807' 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 00000000-0000-0
000-0000-000000000000`. 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 '4cb2f1c9-c771-4ce5-a581-9376e59ea807' 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 00000000-0000-0
000-0000-000000000000`. 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     12.4.0                Az                                  Core,Desk
Script     3.0.4                 Az.Accounts                         Core,Desk {Disable-AzDataCollection, Disable-AzContextAutosave, Enable-A…
Script     2.0.1                 Az.Advisor                          Core,Desk {Disable-AzAdvisorRecommendation, Enable-AzAdvisorRecommendati…
Script     6.0.4                 Az.Aks                              Core,Desk {Disable-AzAksAddOn, Enable-AzAksAddOn, Get-AzAksCluster, Get-…
Script     1.1.5                 Az.AnalysisServices                 Core,Desk {Add-AzAnalysisServicesAccount, Export-AzAnalysisServicesInsta…
Script     4.0.4                 Az.ApiManagement                    Core,Desk {Add-AzApiManagementApiToGateway, Add-AzApiManagementApiToProd…
Script     1.1.0                 Az.App                              Core,Desk {Disable-AzContainerAppRevision, Enable-AzContainerAppRevision…
Script     1.3.2                 Az.AppConfiguration                 Core,Desk {Clear-AzAppConfigurationDeletedStore, Get-AzAppConfigurationD…
Script     2.2.5                 Az.ApplicationInsights              Core,Desk {Get-AzApplicationInsights, Get-AzApplicationInsightsApiKey, G…
Script     1.0.1                 Az.ArcResourceBridge                Core,Desk {Get-AzArcResourceBridge, Get-AzArcResourceBridgeAppliance Cred…
Script     2.0.2                 Az.Attestation                      Core,Desk {Add-AzAttestationPolicySigner, Get-AzAttestationPolicy, Get-A…
Script     1.0.2                 Az.Automanage                       Core,Desk {Get-AzAutomanageBestPractice, Get-AzAutomanageConfigProfile, …
Script     1.10.0                Az.Automation                       Core,Desk {Export-AzAutomationDscConfiguration, Export-AzAutomationDscNo…
Script     3.6.3                 Az.Batch                            Core,Desk {Disable-AzBatchAutoScale, Disable-AzBatchComputeNodeSchedulin…
Script     2.1.0                 Az.Billing                          Core,Desk {Get-AzBillingAccount, Get-AzBillingInvoice, Get-AzBillingPeri…
Script     3.2.2                 Az.Cdn                              Core,Desk {Clear-AzCdnEndpointContent, Clear-AzFrontDoorCdnEndpointConte…
Script     2.0.1                 Az.CloudService                     Core,Desk {Get-AzCloudService, Get-AzCloudServiceInstanceView, Get-AzClo…
Script     1.14.1                Az.CognitiveServices                Core,Desk {Get-AzCognitiveServicesAccount, Get-AzCognitiveServicesAccoun…
Script     8.4.0                 Az.Compute                          Core,Desk {Add-AzImageDataDisk, Add-AzVhd, Add-AzVMAdditionalUnattendCon…
Script     1.0.1                 Az.ConfidentialLedger               Core,Desk {Get-AzConfidentialLedger, New-AzConfidentialLedger, New-AzCon…
Script     4.0.2                 Az.ContainerInstance                Core,Desk {Add-AzContainerInstanceOutput, Get-AzContainerGroup, Get-AzCo…
Script     4.2.1                 Az.ContainerRegistry                Core,Desk {Connect-AzContainerRegistry, Get-AzContainerRegistryManifest,…
Script     1.15.0                Az.CosmosDB                         Core,Desk {Get-AzCosmosDBAccount, Get-AzCosmosDBAccountKey, Get-AzCosmos…
Script     1.1.1                 Az.DataBoxEdge                      Core,Desk {Get-AzDataBoxEdgeBandwidthSchedule, Get-AzDataBoxEdgeDevice, …
Script     1.9.0                 Az.Databricks                       Core,Desk {Get-AzDatabricksAccessConnector, Get-AzDatabricksOutboundNetw…
Script     1.18.8                Az.DataFactory                      Core,Desk {Add-AzDataFactoryV2DataFlowDebugSessionPackage, Add-AzDataFac…
Script     1.0.3                 Az.DataLakeAnalytics                Core,Desk {Get-AzDataLakeAnalyticsDataSource, New-AzDataLakeAnalyticsCat…
Script     1.3.2                 Az.DataLakeStore                    Core,Desk {Add-AzDataLakeStoreFirewallRule, Add-AzDataLakeStoreItemConte…
Script     2.4.0                 Az.DataProtection                   Core,Desk {Backup-AzDataProtectionBackupInstanceAdhoc, Edit-AzDataProtec…
Script     1.0.2                 Az.DataShare                        Core,Desk {Get-AzDataShare, Get-AzDataShareAccount, Get-AzDataShareDataS…
```

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

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

ModuleType Version    PreRelease Name                                PSEdition ExportedCommands
---------- -------    ---------- ----                                --------- ----------------
Script     12.4.0                Az                                            sable-AzContextAutosave, Enable-A…
Script     3.0.4                 Az.Accounts                                   ion, Enable-AzAdvisorRecommendati…
Script     2.0.1                 Az.Advisor                                    zAksAddOn, Get-AzAksCluster, Get-…
Script     6.0.4                 Az.Aks                                        t, Export-AzAnalysisServicesInsta…
Script     1.1.5                 Az.AnalysisServices                           way, Add-AzApiManagementApiToProd…
Script     4.0.4                 Az.ApiManagement                              on, Enable-AzContainerAppRevision…
Script     1.1.0                 Az.App                                        tedStore, Get-AzAppConfigurationD…
Script     1.3.2                 Az.AppConfiguration                           et-AzApplicationInsightsApiKey, G…
Script     2.2.5                 Az.ApplicationInsights                        -AzArcResourceBridgeApplianceCred…
Script     1.0.1                 Az.ArcResourceBridge                          r, Get-AzAttestationPolicy, Get-A…
Script     2.0.2                 Az.Attestation                                , Get-AzAutomanageConfigProfile, …
Script     1.0.2                 Az.Automanage                                 uration, Export-AzAutomationDscNo…
Script     1.10.0                Az.Automation                                 sable-AzBatchComputeNodeSchedulin…
Script     3.6.3                 Az.Batch                                      BillingInvoice, Get-AzBillingPeri…
Script     2.1.0                 Az.Billing                                    Clear-AzFrontDoorCdnEndpointConte…
Script     3.2.2                 Az.Cdn                                        oudServiceInstanceView, Get-AzClo…
Script     2.0.1                 Az.CloudService                               nt, Get-AzCognitiveServicesAccoun…
Script     1.14.1                Az.CognitiveServices                          Core,Desk {Add-AzImageDataDisk, Add-AzVhd, Add-AzVMAdditionalUnattendCon…
Script     8.4.0                 Az.Compute                                    Core,Desk {Get-AzConfidentialLedger, New-AzConfidentialLedger, New-AzCon…
Script     1.0.1                 Az.ConfidentialLedger                         Core,Desk {Add-AzContainerInstanceOutput, Get-AzContainerGroup, Get-AzCo…
Script     4.0.2                 Az.ContainerInstance                          Core,Desk {Connect-AzContainerRegistry, Get-AzContainerRegistryManifest,…
Script     4.2.1                 Az.ContainerRegistry                          Core,Desk {Get-AzCosmosDBAccount, Get-AzCosmosDBAccountKey, Get-AzCosmos…
Script     1.15.0                Az.CosmosDB                                   Core,Desk {Get-AzDataBoxEdgeBandwidthSchedule, Get-AzDataBoxEdgeDevice, …
Script     1.1.1                 Az.DataBoxEdge                                Core,Desk {Get-AzDatabricksAccessConnector, Get-AzDatabricksOutboundNetw…
Script     1.9.0                 Az.Databricks                                 Core,Desk {Add-AzDataFactoryV2DataFlowDebugSessionPackage, Add-AzDataFac…
Script     1.18.8                Az.DataFactory                                Core,Desk {Get-AzDataLakeAnalyticsDataSource, New-AzDataLakeAnalyticsCat…
Script     1.0.3                 Az.DataLakeAnalytics                          Core,Desk {Add-AzDataLakeStoreFirewallRule, Add-AzDataLakeStoreItemConte…
Script     1.3.2                 Az.DataLakeStore                              Core,Desk {Backup-AzDataProtectionBackupInstanceAdhoc, Edit-AzDataProtec…
Script     2.4.0                 Az.DataProtection                             Core,Desk {Get-AzDataShare, Get-AzDataShareAccount, Get-AzDataShareDataS…
Script     1.0.2                 Az.DataShare
```

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                                PSEdition ExportedCommands
---------- -------    ---------- ----                                --------- ----------------
Script     12.4.0                Az
Script     3.0.4                 Az.Accounts                                   llection, Disable-AzContextAutosave, Enable-A…
Script     2.0.1                 Az.Advisor                                    rRecommendation, Enable-AzAdvisorRecommendati…
Script     6.0.4                 Az.Aks                                        On, Enable-AzAksAddOn, Get-AzAksCluster, Get-…
Script     1.1.5                 Az.AnalysisServices                           rvicesAccount, Export-AzAnalysisServicesInsta…
Script     4.0.4                 Az.ApiManagement                              entApiToGateway, Add-AzApiManagementApiToProd…
Script     1.1.0                 Az.App                                        nerAppRevision, Enable-AzContainerAppRevision…
Script     1.3.2                 Az.AppConfiguration                           gurationDeletedStore, Get-AzAppConfigurationD…
Script     2.2.5                 Az.ApplicationInsights
Script     1.0.1                 Az.ArcResourceBridge                          Core,Desk {Get-Az
Script     2.0.2                 Az.Attestation                                Core,Desk {Add-AzAttest
Script     1.0.2                 Az.Automanage                                 Core,Desk {Get-AzAutoman
Script     1.10.0                Az.Automation                                 Core,Desk {Export-AzAuto
Script     3.6.3                 Az.Batch                                      Core,Desk {Disable-AzBat
Script     2.1.0                 Az.Billing                                    Core,Desk {Get-AzBilling
Script     3.2.2                 Az.Cdn                                        Core,Desk {Clear-AzCdnEr
Script     2.0.1                 Az.CloudService                               Core,Desk {Get-AzCloudSe
Script     1.14.1                Az.CognitiveServices                          Core,Desk {Get-AzCogniti
Script     8.4.0                 Az.Compute                                    Core,Desk {Add-AzImageDa
Script     1.0.1                 Az.ConfidentialLedger                         Core,Desk {Get-AzConfide
Script     4.0.2                 Az.ContainerInstance                          Core,Desk {Add-AzContair
Script     4.2.1                 Az.ContainerRegistry                          Core,Desk {Connect-AzCor
Script     1.15.0                Az.CosmosDB                                   Core,Desk {Get-AzCosmosD
Script     1.1.1                 Az.DataBoxEdge                                Core,Desk {Get-AzDataBox
Script     1.9.0                 Az.Databricks                                 Core,Desk {Get-AzDatabri
Script     1.18.8                Az.DataFactory                                Core,Desk {Add-AzDataFac
Script     1.0.3                 Az.DataLakeAnalytics                          Core,Desk {Get-AzDataLak
Script     1.3.2                 Az.DataLakeStore                              Core,Desk {Add-AzDataLak
Script     2.4.0                 Az.DataProtection                             Core,Desk {Backup-AzData
Script     1.0.2                 Az.DataShare                                  Core,Desk {Get-AzDataSha
```

Modules.

Modules as far as the eye can see!

D

```
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

```
55  #region Part 4: Load Helper Functions ##############################################
    5 references
56  function Split-StringOnLiteralString {…
169 }
170
    1 reference
171 function Test-CommandExistence {…
357 } <- #171-357 function Test-CommandExistence
358
    11 references
359 function Invoke-CommandSafely {…
559 } <- #359-559 function Invoke-CommandSafely
560 #endregion Part 4: Load Helper Functions ##############################################
561
562 #region Part 5: What OS Is This, Anyway? ##############################################
563 $hashtableOSInfo = @{}
564
565 # Create a second "bridge" hashtable
566 $hashtableBridgeOSInfo = @{}
567
568 # Adopted from: https://unix.stackexchange.com/a/6348/409368
569 # This needs to be tested more-exhaustively on the minimum OS versions that PowerShell 6.0 (original release) supported
570 if (Test-Path variable:\isLinux) {…
777 } <- #570-777 if (Test-Path variable:\isLinux)
778
```

```
PS /home/admin> # Display OS Findings:
PS /home/admin> $hashtableOSInfo

Name                           Value
----                           -----
PRETTY_NAME                    CBL-Mariner/Linux
HOME_URL                       https://aka.ms/cbl-mariner
ANSI_COLOR                     1;34
SUPPORT_URL                    https://aka.ms/cbl-mariner
VERSION                        2.0.20240829
VERSION_ID                     2.0
NAME                           Common Base Linux Mariner
ID                             mariner
BUG_REPORT_URL                 https://aka.ms/cbl-mariner
```

Demo – Docker Container

Getting the Latest Cloud Shell Container

Run it!

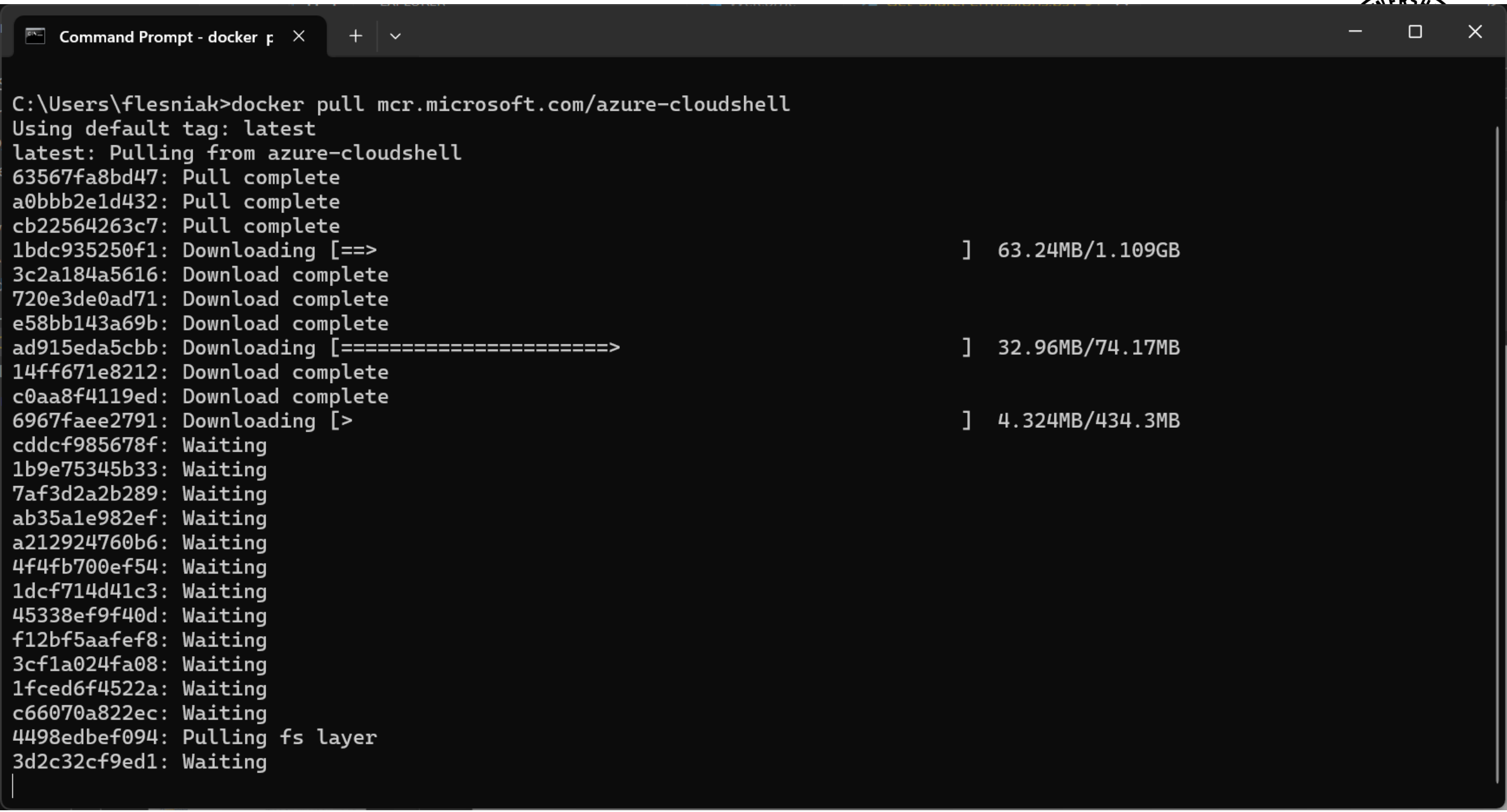
All code is posted to GitHub:
https://github.com/franklesniak/powershell-xplat

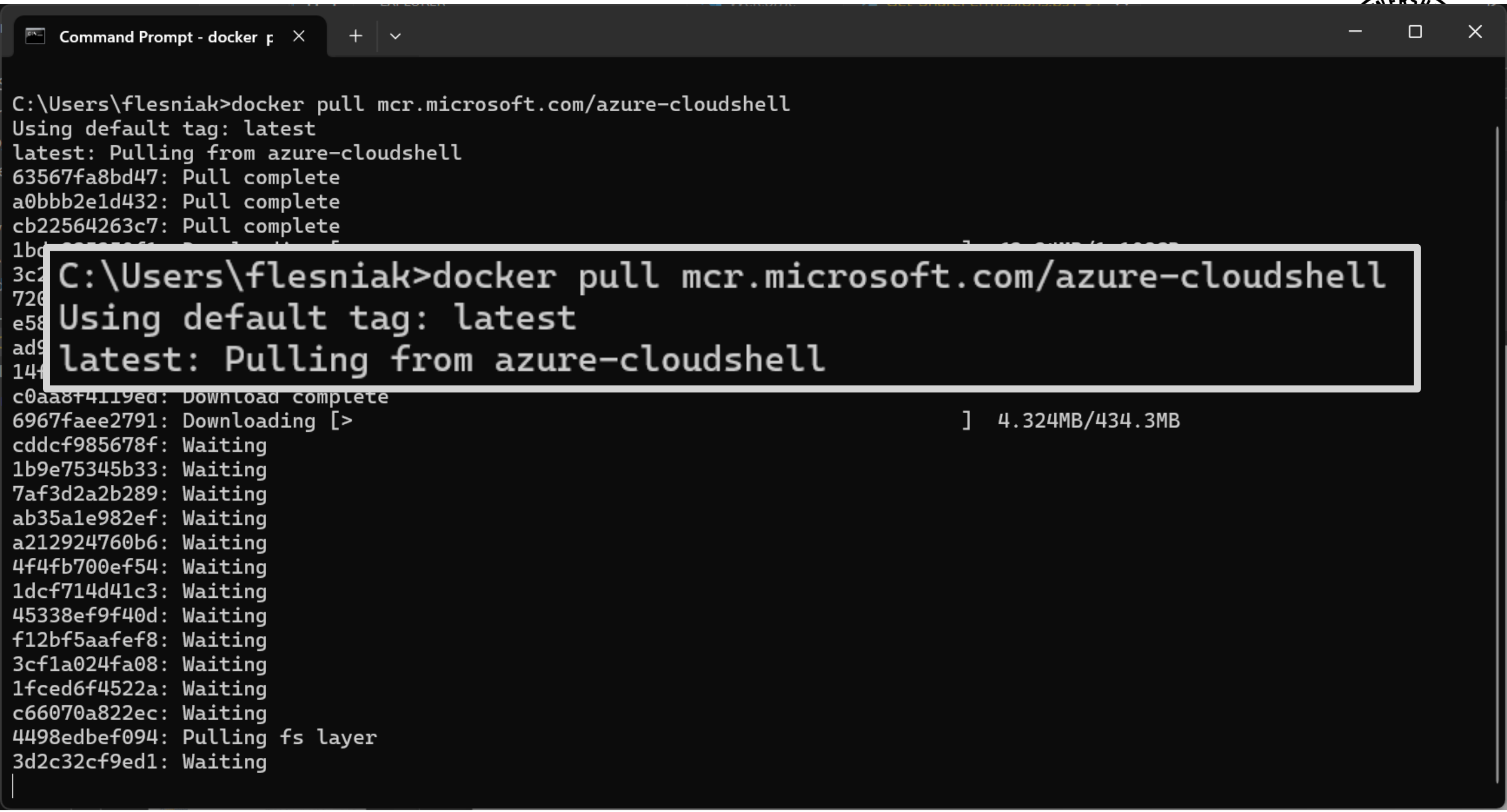
Before using containers, you first need to configure prerequisites – here they are for Windows:

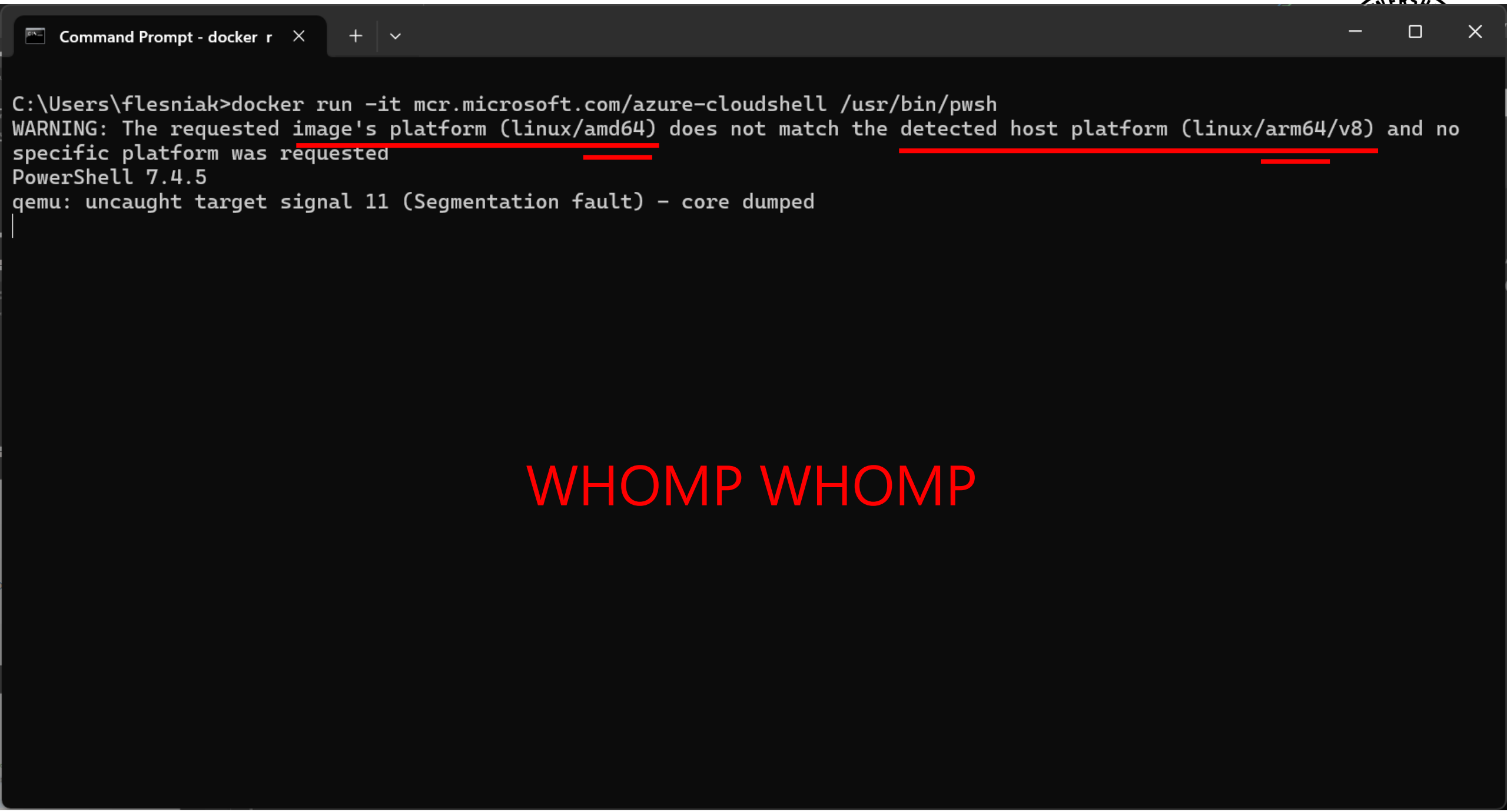
1. Install Windows Subsystem for Linux:
   ```
   wsl --install
   ```

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
63567fa8bd47: Pull complete
a0bbb2e1d432: Pull complete
cb22564263c7: Pull complete
1bdc935250f1: Downloading [==>] 63.24MB/1.109GB
3c2a184a5616: Download complete
720e3de0ad71: Download complete
e58bb143a69b: Download complete
ad915eda5cbb: Downloading [====================>] 32.96MB/74.17MB
14ff671e8212: Download complete
c0aa8f4119ed: Download complete
6967faee2791: Downloading [>] 4.324MB/434.3MB
cddcf985678f: Waiting
1b9e75345b33: Waiting
7af3d2a2b289: Waiting
ab35a1e982ef: Waiting
a212924760b6: Waiting
4f4fb700ef54: Waiting
1dcf714d41c3: Waiting
45338ef9f40d: Waiting
f12bf5aafef8: Waiting
3cf1a024fa08: Waiting
1fced6f4522a: Waiting
c66070a822ec: Waiting
4498edbef094: Pulling fs layer
3d2c32cf9ed1: Waiting


Command Prompt - docker p
C:\Users\flesniak>docker pull mcr.microsoft.com/azure-cloudshell
Using default tag: latest
latest: Pulling from azure-cloudshell
63567fa8bd47: Pull complete
a0bbb2e1d432: Pull complete
cb22564263c7: Pull complete
C:\Users\flesniak>docker pull mcr.microsoft.com/azure-cloudshell
Using default tag: latest
latest: Pulling from azure-cloudshell
c0aa8f4119ed: Download complete
6967faee2791: Downloading [>
] 4.324MB/434.3MB
cddcf985678f: Waiting
1b9e75345b33: Waiting
7af3d2a2b289: Waiting
ab35a1e982ef: Waiting
a212924760b6: Waiting
4f4fb700ef54: Waiting
1dcf714d41c3: Waiting
45338ef9f40d: Waiting
f12bf5aafef8: Waiting
3cf1a024fa08: Waiting
1fced6f4522a: Waiting
c66070a822ec: Waiting
4498edbef094: Pulling fs layer
3d2c32cf9ed1: Waiting


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 7.4.5
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 7.4.5
qemu: uncaught target signal 11 (Segmentation fault) - core dumped
```

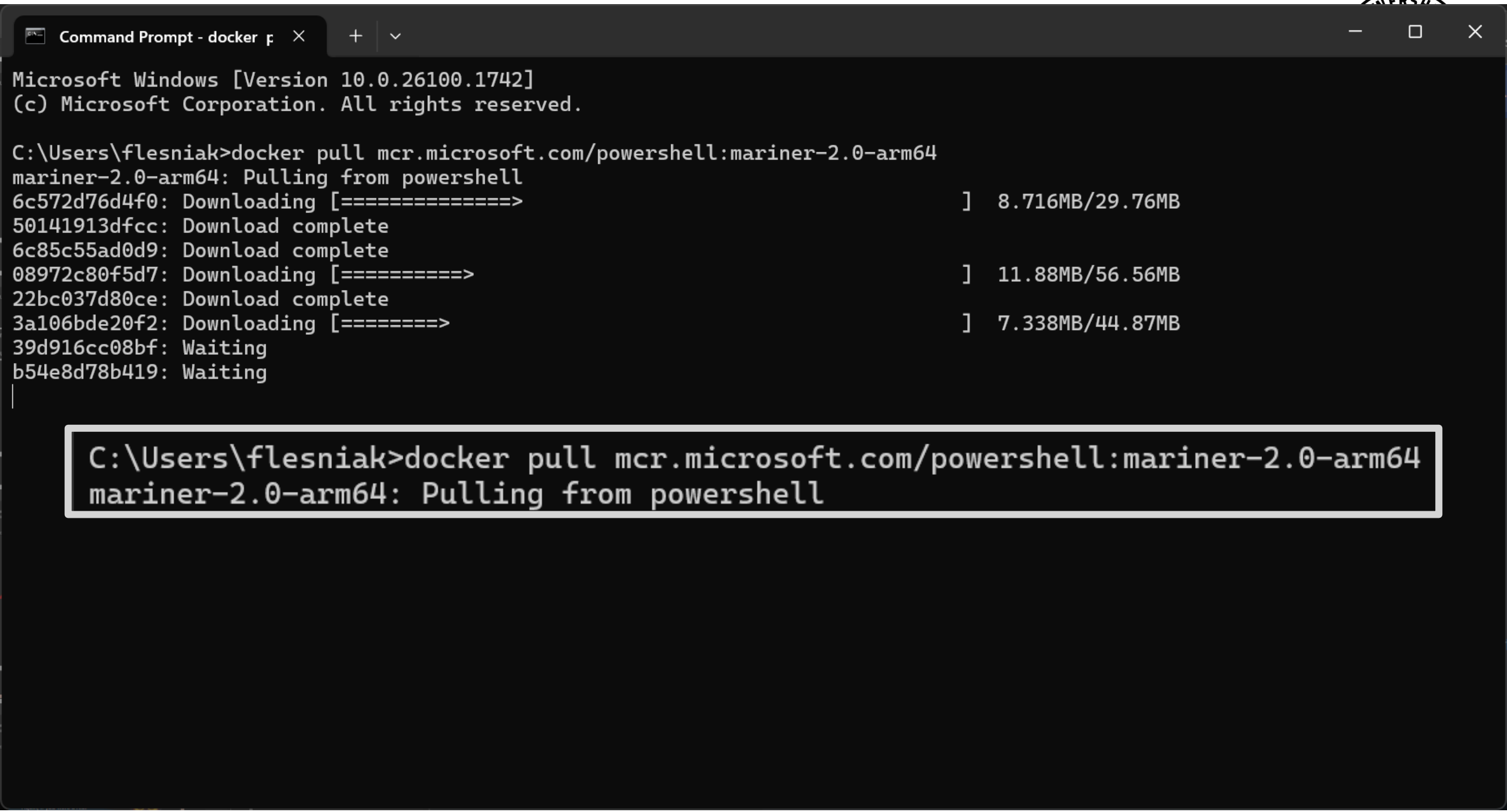
Again, if you were a *real* Linux sysadmin, you would rebuild the container from source, and then you wouldn't have these problems.

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

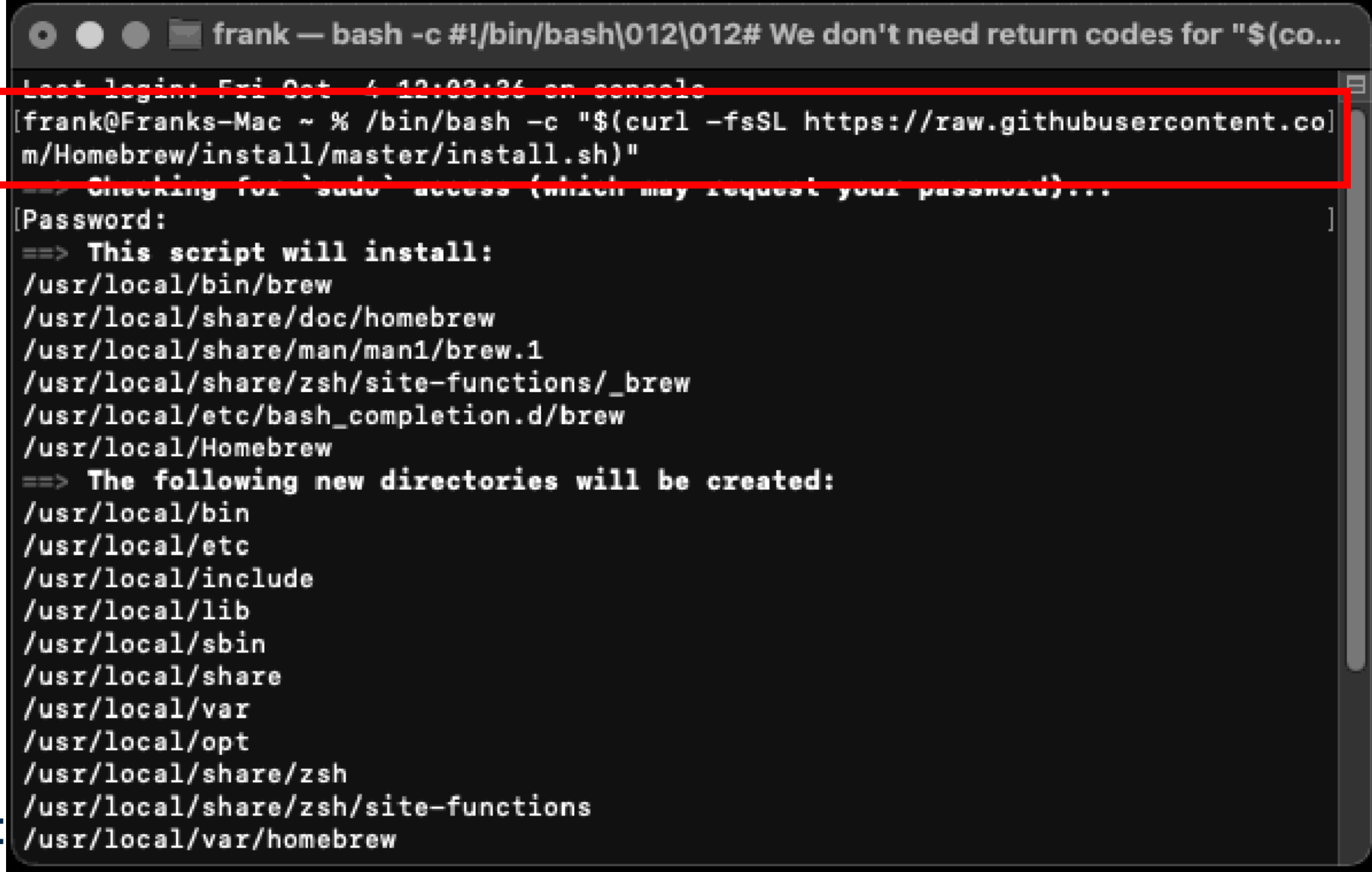

Command Prompt - docker p
Microsoft Windows [Version 10.0.26100.1742]
(c) Microsoft Corporation. All rights reserved.
C:\Users\flesniak>docker pull mcr.microsoft.com/powershell:mariner-2.0-arm64
mariner-2.0-arm64: Pulling from powershell
6c572d76d4f0: Downloading [==========>] 8.716MB/29.76MB
50141913dfcc: Download complete
6c85c55ad0d9: Download complete
08972c80f5d7: Downloading [==========>] 11.88MB/56.56MB
22bc037d80ce: Download complete
3a106bde20f2: Downloading [========>] 7.338MB/44.87MB
39d916cc08bf: Waiting
b54e8d78b419: Waiting
C:\Users\flesniak>docker pull mcr.microsoft.com/powershell:mariner-2.0-arm64
mariner-2.0-arm64: Pulling from powershell

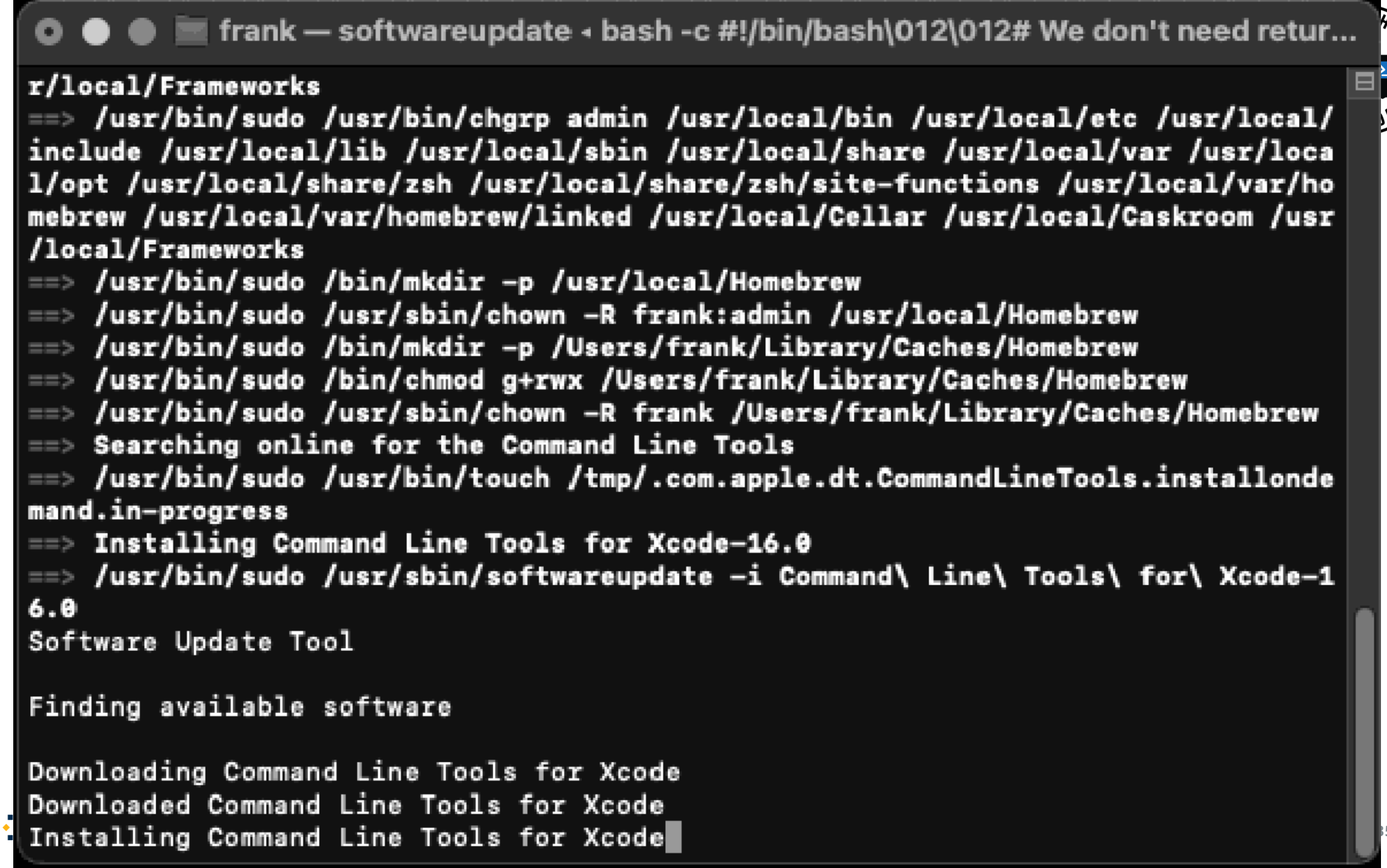

```
C:\Users\flesniak>docker run -it mcr.microsoft.com/powershell:mariner-2.0-arm64
PowerShell 7.4.5
PS /> $IsLinux
True
PS />
```

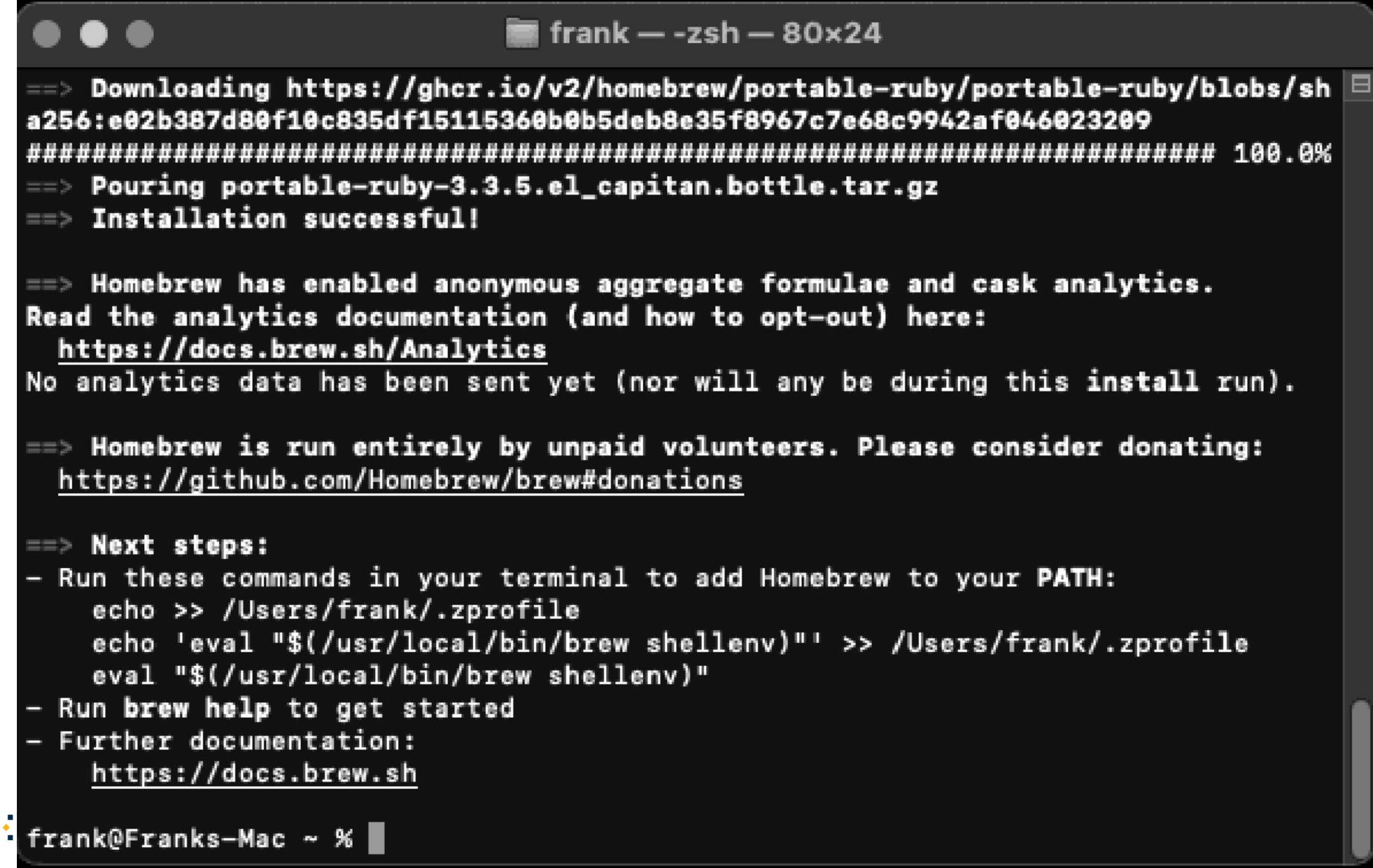
PowerShell on macOS

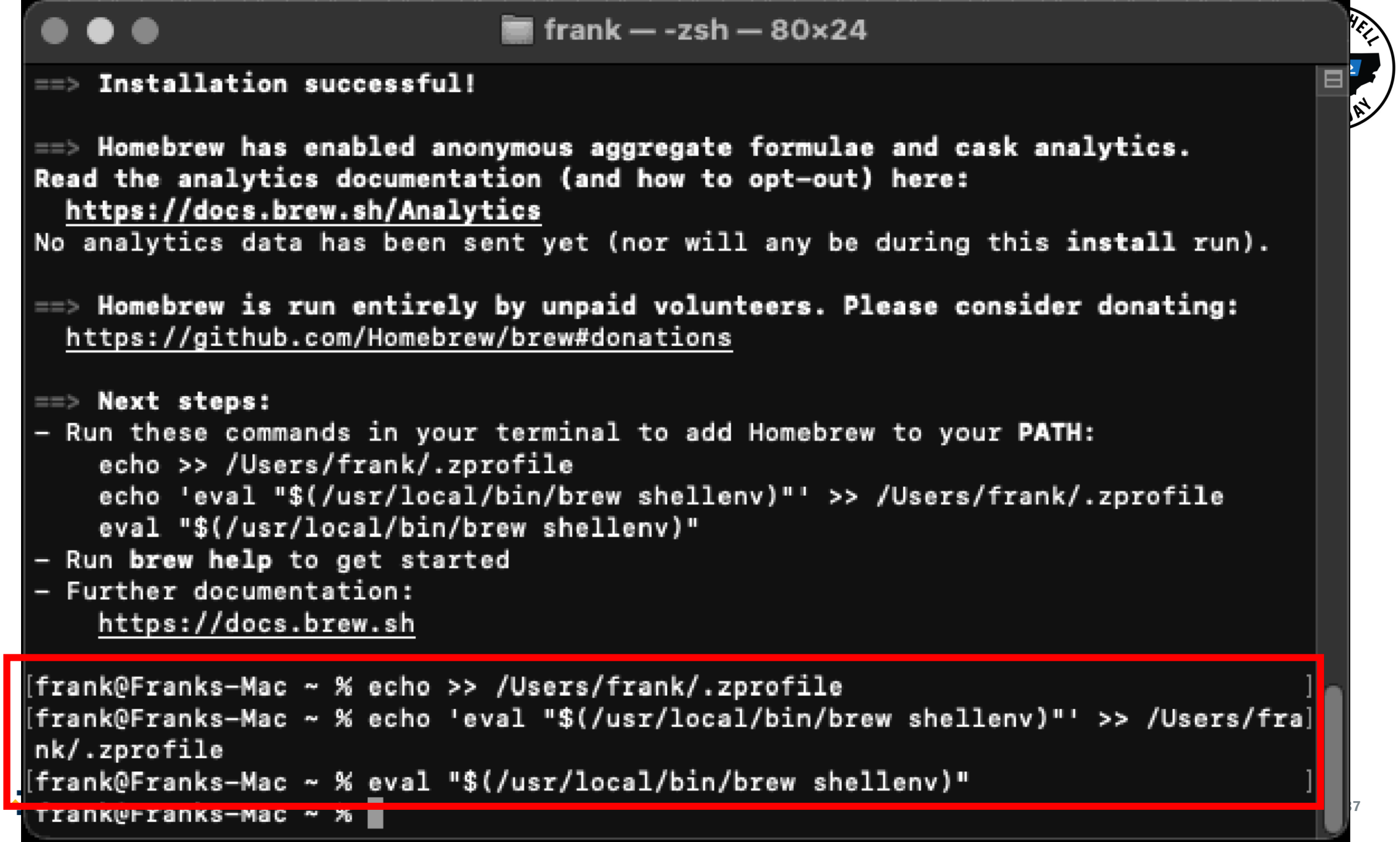
Installation

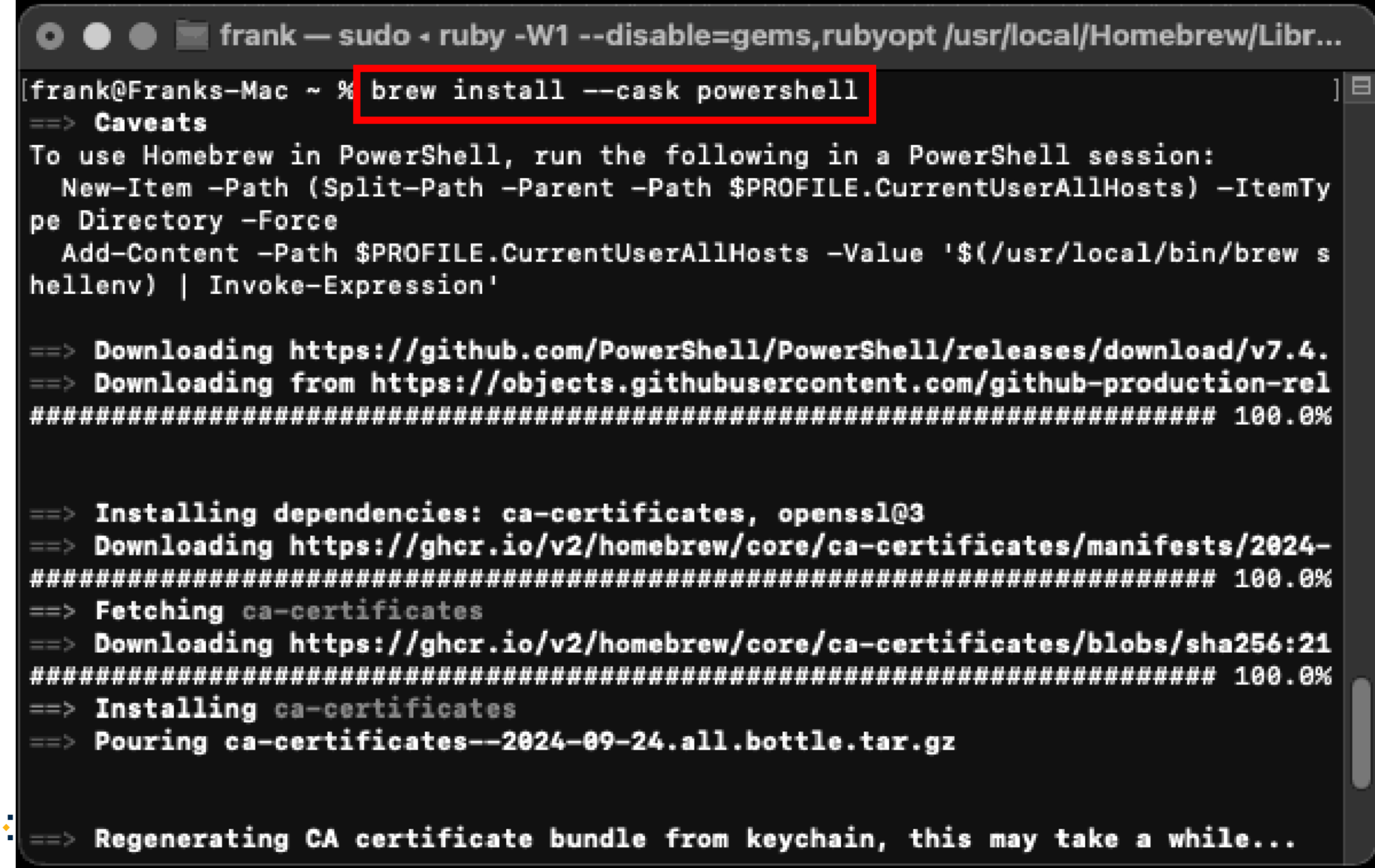
All code is posted to GitHub:
https://github.com/franklesniak/powershell-xplat

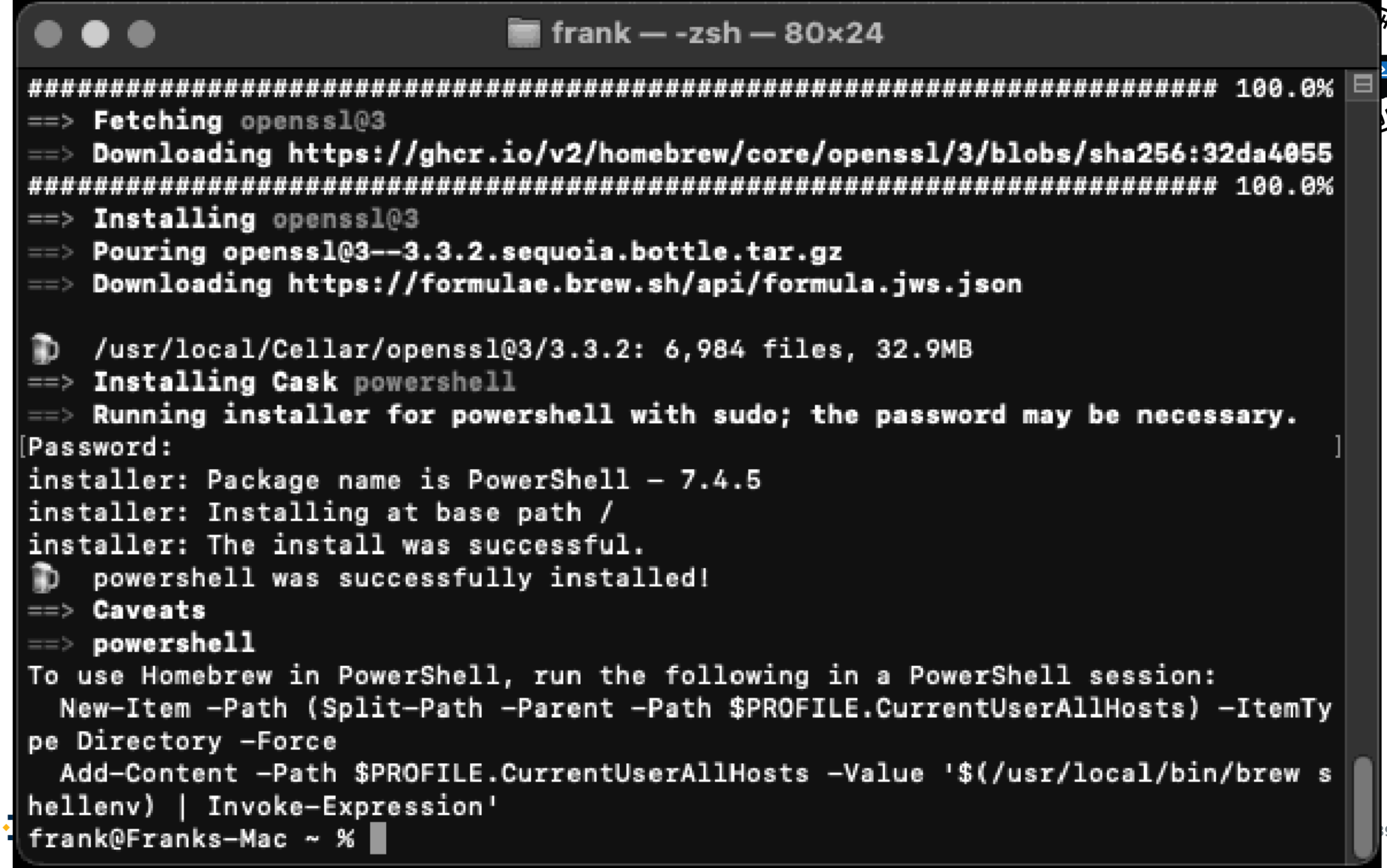

frank — bash -c #!/bin/bash\012\012# We don't need return codes for "$(co...
Last login: Fri Oct 4 12:03:36 on console
frank@Franks-Mac ~ % /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
==> Checking for `sudo` access (which may request your password)...
Password:
==> This script will install:
/usr/local/bin/brew
/usr/local/share/doc/homebrew
/usr/local/share/man/man1/brew.1
/usr/local/share/zsh/site-functions/_brew
/usr/local/etc/bash_completion.d/brew
/usr/local/Homebrew
==> The following new directories will be created:
/usr/local/bin
/usr/local/etc
/usr/local/include
/usr/local/lib
/usr/local/sbin
/usr/local/share
/usr/local/var
/usr/local/opt
/usr/local/share/zsh
/usr/local/share/zsh/site-functions
/usr/local/var/homebrew

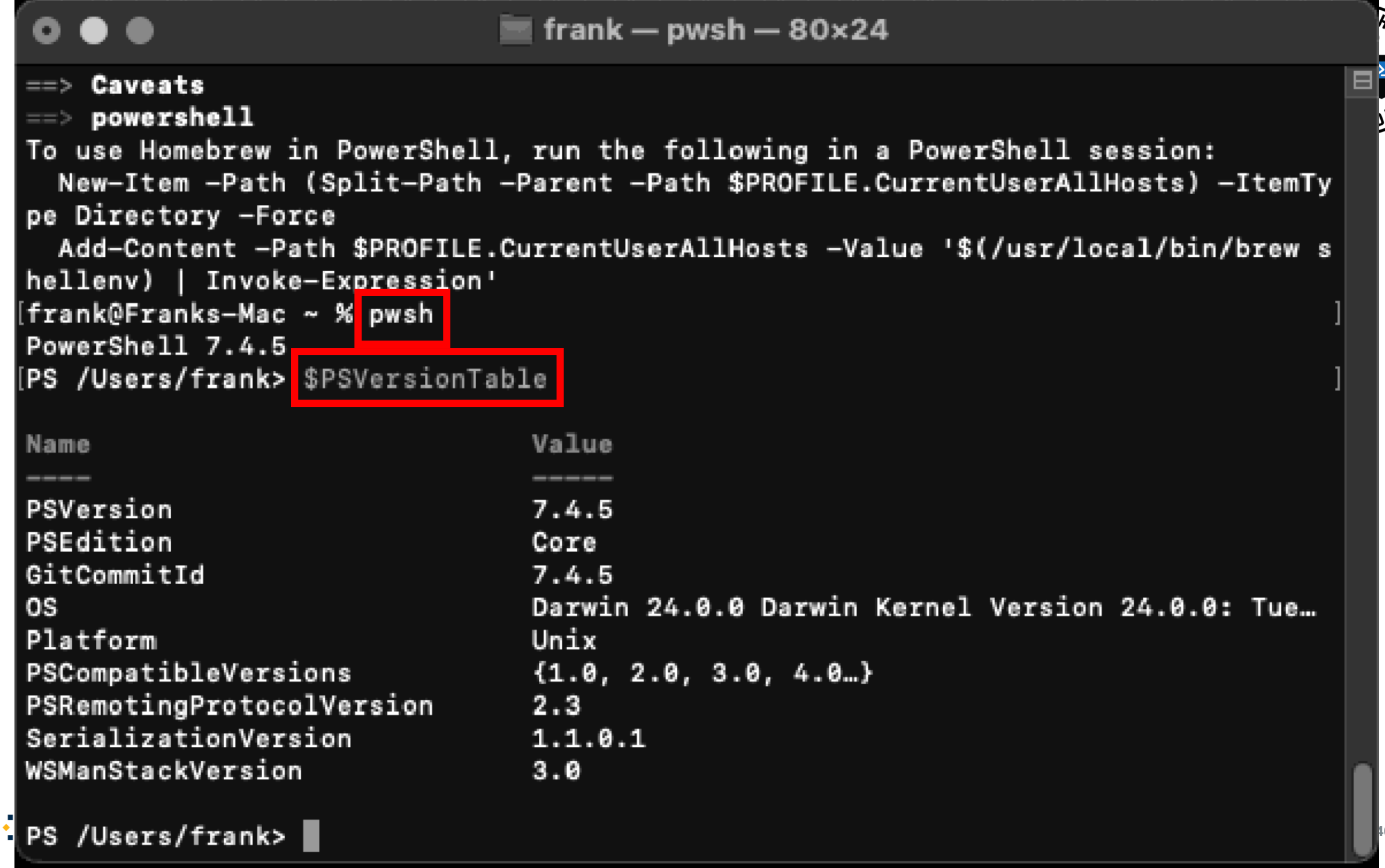

frank — softwareupdate ◂ bash -c #!/bin/bash\012\012# We don't need retur...
r/local/Frameworks
==> /usr/bin/sudo /usr/bin/chgrp admin /usr/local/bin /usr/local/etc /usr/local/include /usr/local/lib /usr/local/sbin /usr/local/share /usr/local/var /usr/local/opt /usr/local/share/zsh /usr/local/share/zsh/site-functions /usr/local/var/homebrew /usr/local/var/homebrew/linked /usr/local/Cellar /usr/local/Caskroom /usr/local/Frameworks
==> /usr/bin/sudo /bin/mkdir -p /usr/local/Homebrew
==> /usr/bin/sudo /usr/sbin/chown -R frank:admin /usr/local/Homebrew
==> /usr/bin/sudo /bin/mkdir -p /Users/frank/Library/Caches/Homebrew
==> /usr/bin/sudo /bin/chmod g+rwx /Users/frank/Library/Caches/Homebrew
==> /usr/bin/sudo /usr/sbin/chown -R frank /Users/frank/Library/Caches/Homebrew
==> Searching online for the Command Line Tools
==> /usr/bin/sudo /usr/bin/touch /tmp/.com.apple.dt.CommandLineTools.installondemand.in-progress
==> Installing Command Line Tools for Xcode-16.0
==> /usr/bin/sudo /usr/sbin/softwareupdate -i Command\ Line\ Tools\ for\ Xcode-16.0
Software Update Tool
Finding available software
Downloading Command Line Tools for Xcode
Downloaded Command Line Tools for Xcode
Installing Command Line Tools for Xcode

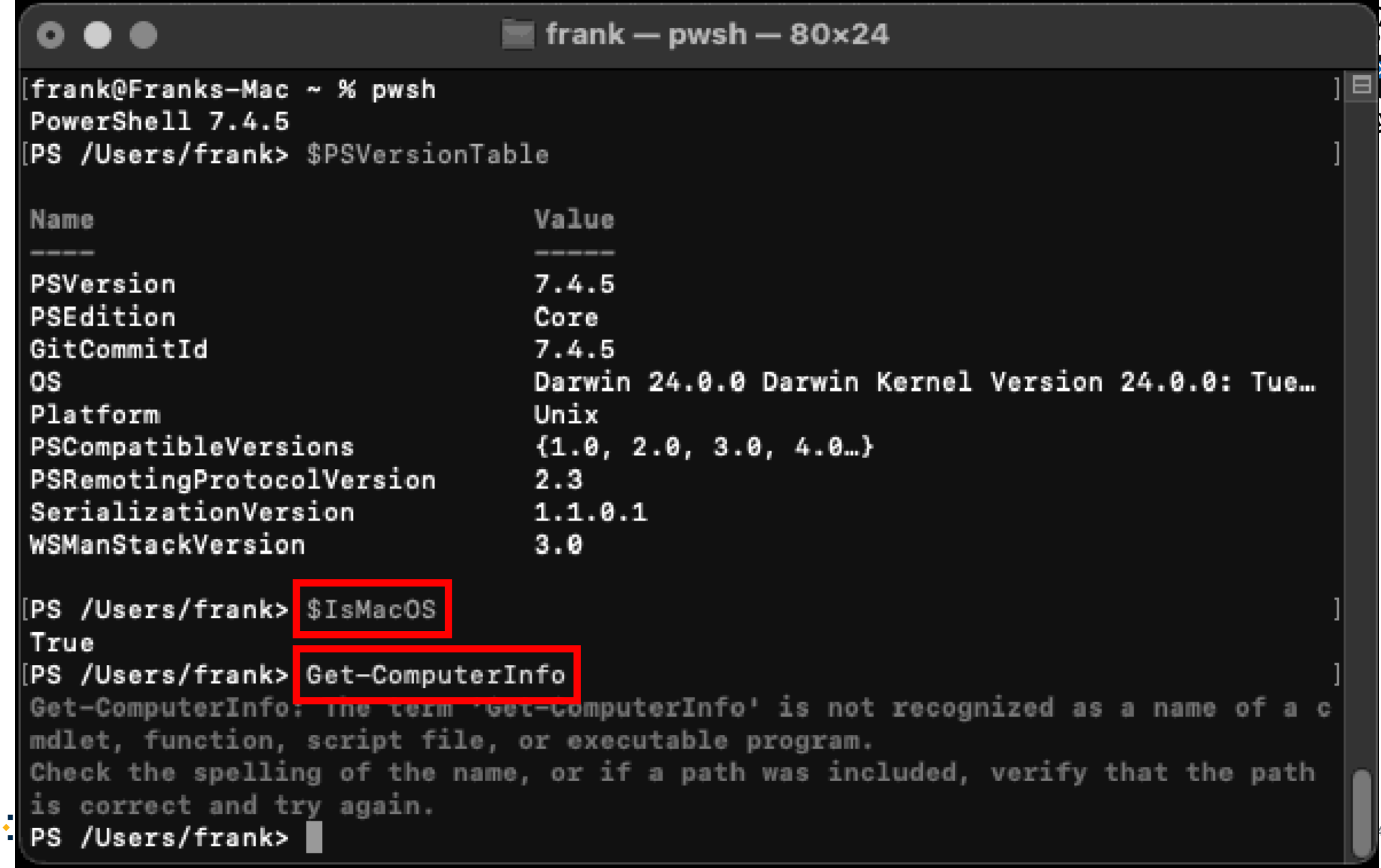

frank — -zsh — 80×24
==> Downloading https://ghcr.io/v2/homebrew/portable-ruby/portable-ruby/blobs/sha256:e02b387d80f10c835df15115360b0b5deb8e35f8967c7e68c9942af046023209
100.0%
==> Pouring portable-ruby-3.3.5.el_capitan.bottle.tar.gz
==> Installation successful!
==> Homebrew has enabled anonymous aggregate formulae and cask analytics.
Read the analytics documentation (and how to opt-out) here:
https://docs.brew.sh/Analytics
No analytics data has been sent yet (nor will any be during this install run).
==> Homebrew is run entirely by unpaid volunteers. Please consider donating:
https://github.com/Homebrew/brew#donations
==> Next steps:
- Run these commands in your terminal to add Homebrew to your PATH:
echo >> /Users/frank/.zprofile
echo 'eval "$(/usr/local/bin/brew shellenv)"' >> /Users/frank/.zprofile
eval "$(/usr/local/bin/brew shellenv)"
- Run brew help to get started
- Further documentation:
https://docs.brew.sh
frank@Franks-Mac ~ %

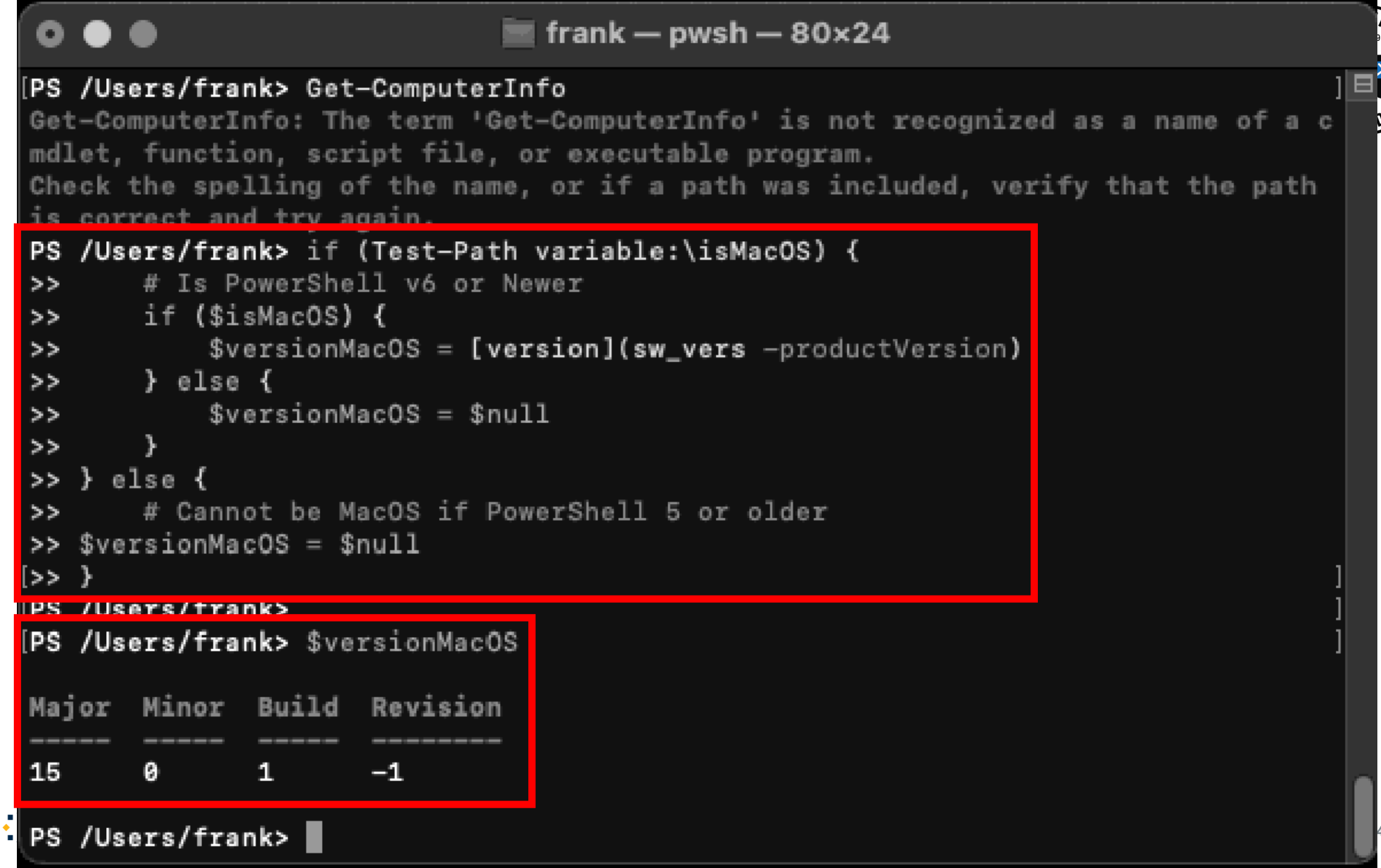

frank — -zsh — 80×24
==> Installation successful!
==> Homebrew has enabled anonymous aggregate formulae and cask analytics.
Read the analytics documentation (and how to opt-out) here:
https://docs.brew.sh/Analytics
No analytics data has been sent yet (nor will any be during this install run).
==> Homebrew is run entirely by unpaid volunteers. Please consider donating:
https://github.com/Homebrew/brew#donations
==> Next steps:
- Run these commands in your terminal to add Homebrew to your PATH:
echo >> /Users/frank/.zprofile
echo 'eval "$(/usr/local/bin/brew shellenv)"' >> /Users/frank/.zprofile
eval "$(/usr/local/bin/brew shellenv)"
- Run brew help to get started
- Further documentation:
https://docs.brew.sh
[frank@Franks-Mac ~ % echo >> /Users/frank/.zprofile
[frank@Franks-Mac ~ % echo 'eval "$(/usr/local/bin/brew shellenv)"' >> /Users/fra
nk/.zprofile
[frank@Franks-Mac ~ % eval "$(/usr/local/bin/brew shellenv)"
frank@Franks-Mac ~ %


frank — sudo ◂ ruby -W1 --disable=gems,rubyopt /usr/local/Homebrew/Libr...
[frank@Franks-Mac ~ % brew install --cask powershell
==> Caveats
To use Homebrew in PowerShell, run the following in a PowerShell session:
New-Item -Path (Split-Path -Parent -Path $PROFILE.CurrentUserAllHosts) -ItemType Directory -Force
Add-Content -Path $PROFILE.CurrentUserAllHosts -Value '$(/usr/local/bin/brew shellenv) | Invoke-Expression'
==> Downloading https://github.com/PowerShell/PowerShell/releases/download/v7.4.
==> Downloading from https://objects.githubusercontent.com/github-production-rel
100.0%
==> Installing dependencies: ca-certificates, openssl@3
==> Downloading https://ghcr.io/v2/homebrew/core/ca-certificates/manifests/2024-
100.0%
==> Fetching ca-certificates
==> Downloading https://ghcr.io/v2/homebrew/core/ca-certificates/blobs/sha256:21
100.0%
==> Installing ca-certificates
==> Pouring ca-certificates--2024-09-24.all.bottle.tar.gz
==> Regenerating CA certificate bundle from keychain, this may take a while...


frank — -zsh — 80×24
100.0%
==> Fetching openssl@3
==> Downloading https://ghcr.io/v2/homebrew/core/openssl/3/blobs/sha256:32da4055
100.0%
==> Installing openssl@3
==> Pouring openssl@3--3.3.2.sequoia.bottle.tar.gz
==> Downloading https://formulae.brew.sh/api/formula.jws.json
/usr/local/Cellar/openssl@3/3.3.2: 6,984 files, 32.9MB
==> Installing Cask powershell
==> Running installer for powershell with sudo; the password may be necessary.
Password:
installer: Package name is PowerShell - 7.4.5
installer: Installing at base path /
installer: The install was successful.
powershell was successfully installed!
==> Caveats
==> powershell
To use Homebrew in PowerShell, run the following in a PowerShell session:
New-Item -Path (Split-Path -Parent -Path $PROFILE.CurrentUserAllHosts) -ItemType Directory -Force
Add-Content -Path $PROFILE.CurrentUserAllHosts -Value '$(/usr/local/bin/brew shellenv) | Invoke-Expression'
frank@Franks-Mac ~ %


frank — pwsh — 80×24
==> Caveats
==> powershell
To use Homebrew in PowerShell, run the following in a PowerShell session:
New-Item -Path (Split-Path -Parent -Path $PROFILE.CurrentUserAllHosts) -ItemType Directory -Force
Add-Content -Path $PROFILE.CurrentUserAllHosts -Value '$(/usr/local/bin/brew shellenv) | Invoke-Expression'
frank@Franks-Mac ~ % pwsh
PowerShell 7.4.5
PS /Users/frank> $PSVersionTable
Name Value
PSVersion 7.4.5
PSEdition Core
GitCommitId 7.4.5
OS Darwin 24.0.0 Darwin Kernel Version 24.0.0: Tue…
Platform Unix
PSCompatibleVersions {1.0, 2.0, 3.0, 4.0…}
PSRemotingProtocolVersion 2.3
SerializationVersion 1.1.0.1
WSManStackVersion 3.0
PS /Users/frank>


frank — pwsh — 80×24
[frank@Franks-Mac ~ % pwsh
PowerShell 7.4.5
[PS /Users/frank> $PSVersionTable
Name Value
PSVersion 7.4.5
PSEdition Core
GitCommitId 7.4.5
OS Darwin 24.0.0 Darwin Kernel Version 24.0.0: Tue…
Platform Unix
PSCompatibleVersions {1.0, 2.0, 3.0, 4.0…}
PSRemotingProtocolVersion 2.3
SerializationVersion 1.1.0.1
WSManStackVersion 3.0
[PS /Users/frank> $IsMacOS
True
[PS /Users/frank> Get-ComputerInfo
Get-ComputerInfo: The term 'Get-ComputerInfo' is not recognized as a name of a c
mdlet, 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 /Users/frank>


frank — pwsh — 80×24
PS /Users/frank> 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 /Users/frank> if (Test-Path variable:\isMacOS) {
>>   # Is PowerShell v6 or Newer
>>   if ($isMacOS) {
>> $versionMacOS = [version](sw_vers -productVersion)
>>   } else {
>> $versionMacOS = $null
>>   }
>> } else {
>>   # Cannot be MacOS if PowerShell 5 or older
>> $versionMacOS = $null
>> }
PS /Users/frank> $versionMacOS
Major Minor Build Revision
----- ----- ----- --------
15 0 1 -1
PS /Users/frank>

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

westMONROE

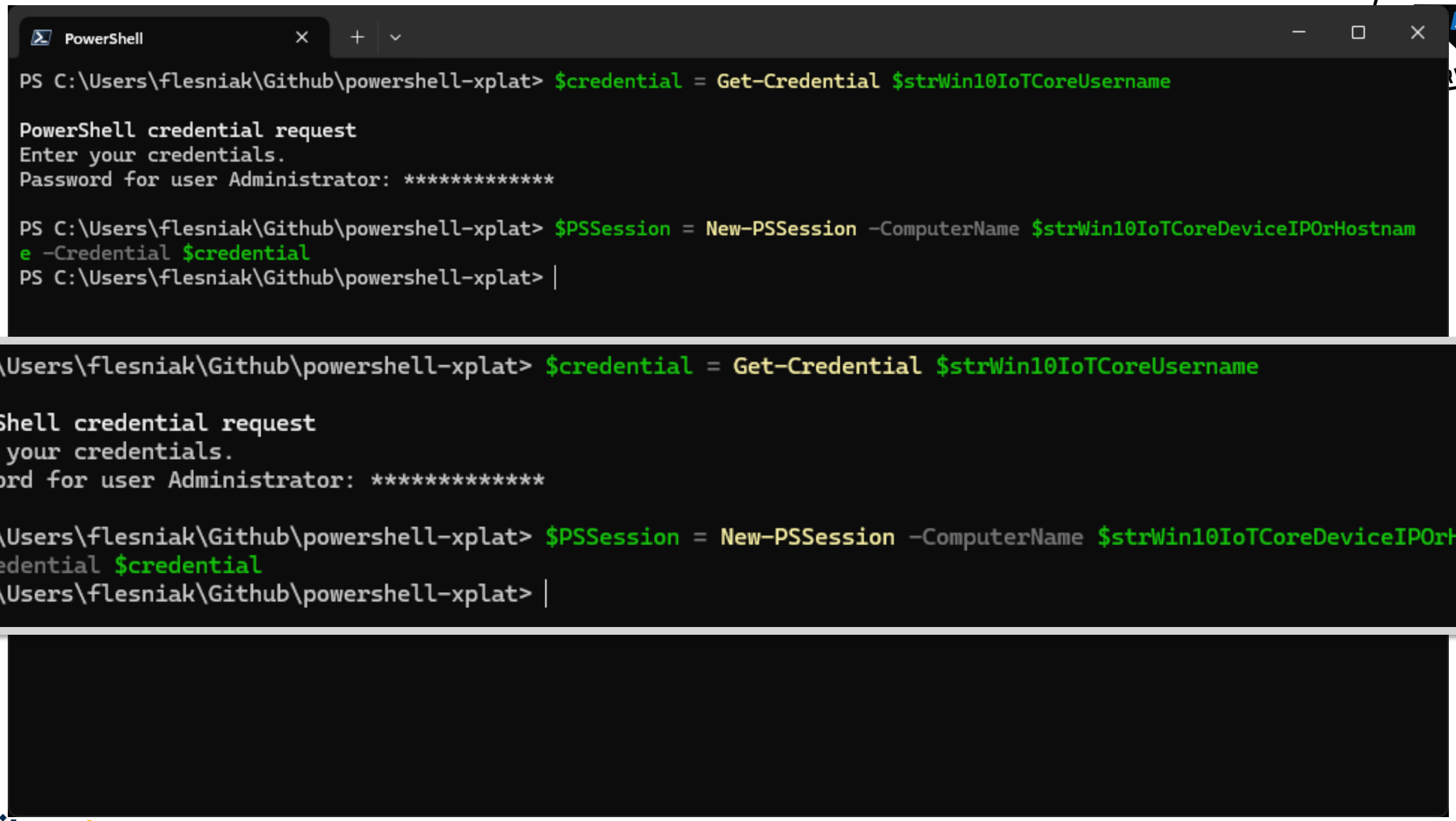

```
45    #region Part 0: Load required functions ##########################################
      1 reference
46  > function Get-FolderPathContainingScript { ···
190   }
      10 references
191 > function Get-PSVersion { ···
242   }
243   #endregion Part 0: Load required functions ##########################################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


```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-
```


```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-
```


```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-
```

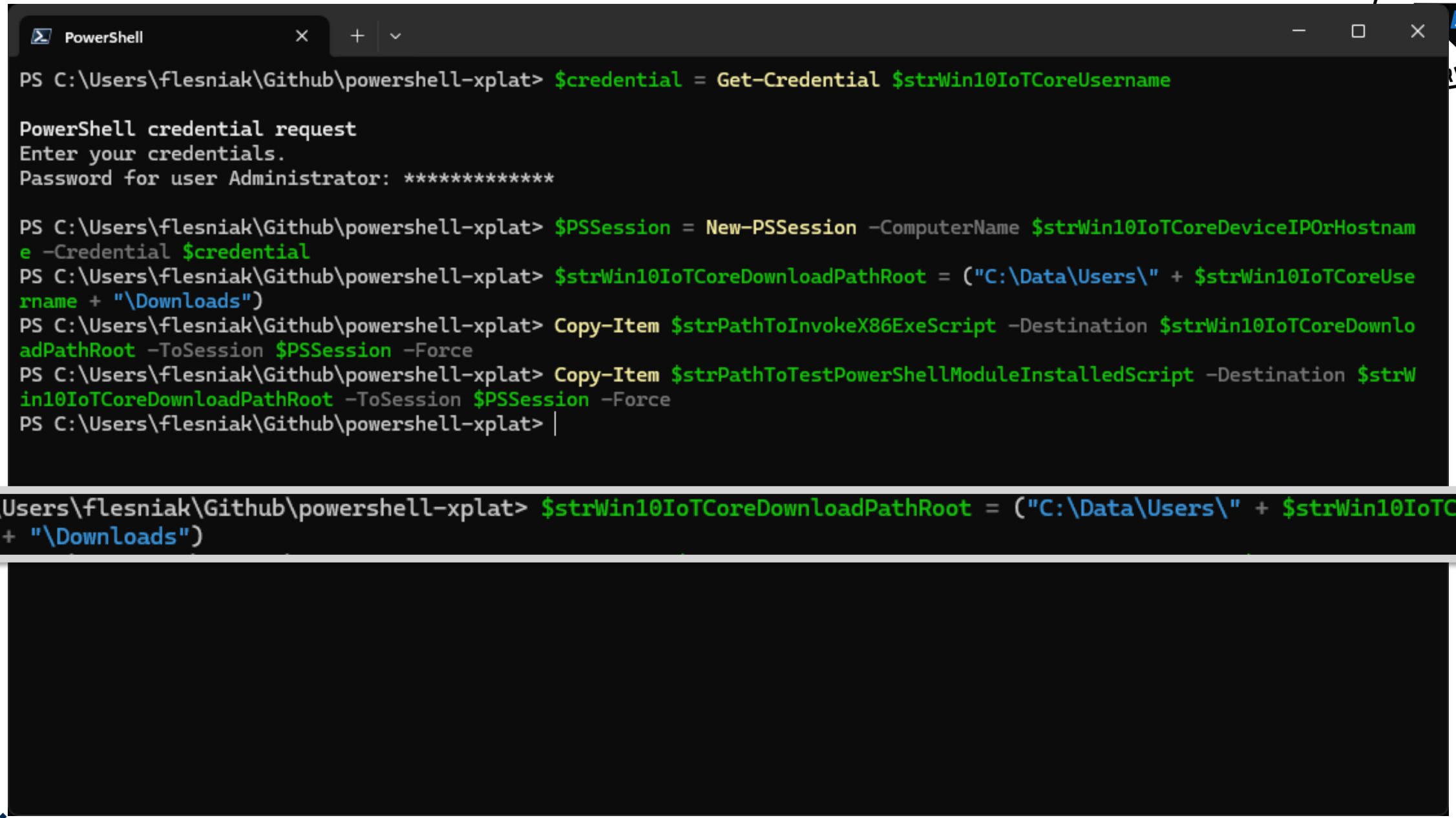

```powershell
#region Part 1: Copy scripts to the target system using remote PSSession ############
$strWin10IoTCoreDeviceIPOrHostname = "10.1.2.137"
$strWin10IoTCoreUsername = "Administrator"
$strCurrentFolder = Get-FolderPathContainingScript
$strPathToInvokeX86ExeScript = Join-Path $strCurrentFolder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



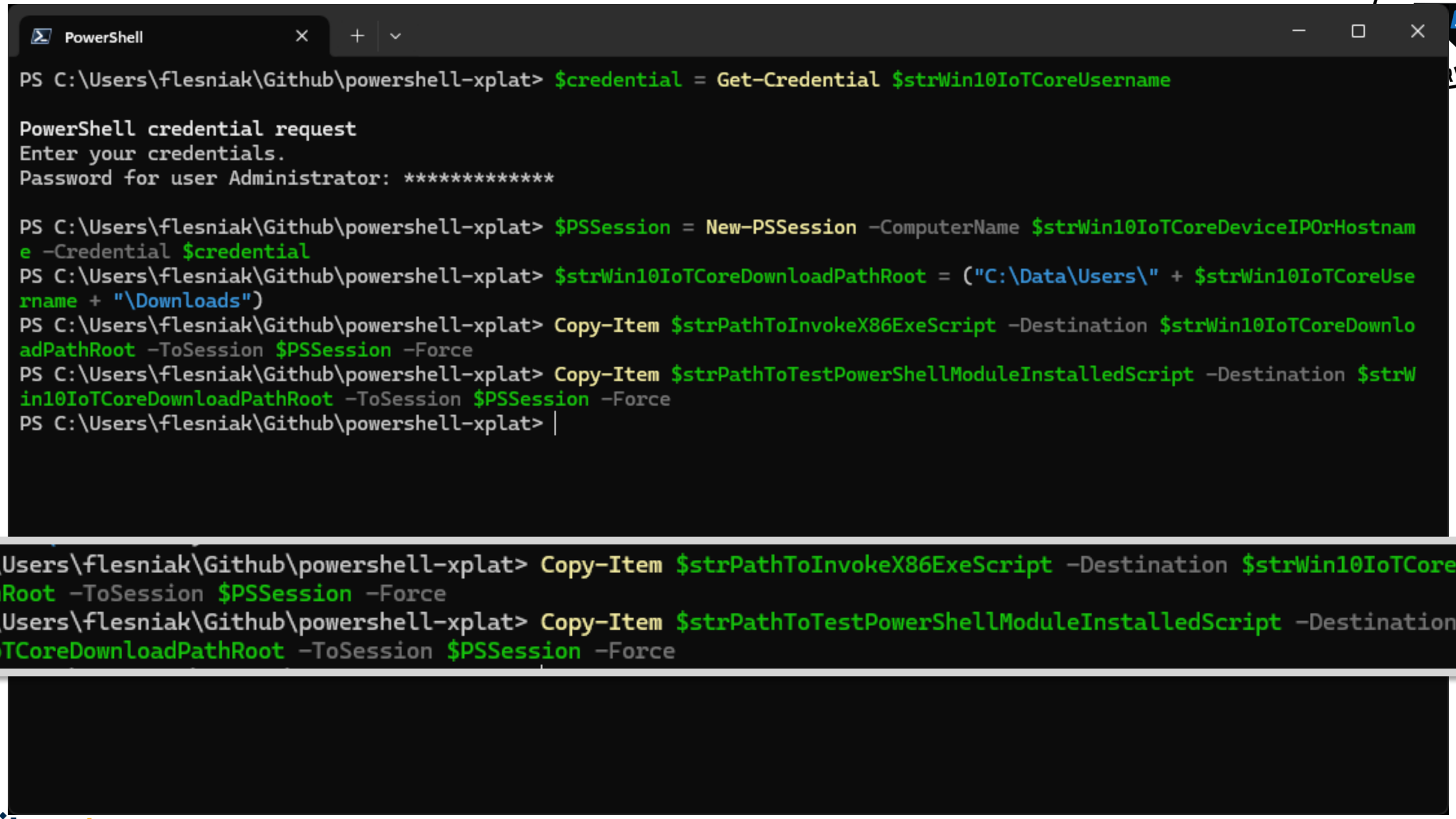
PS C:\Users\flesniak\Github\powershell-xplat> $credential = Get-Credential $strWin10IoTCoreUsername

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

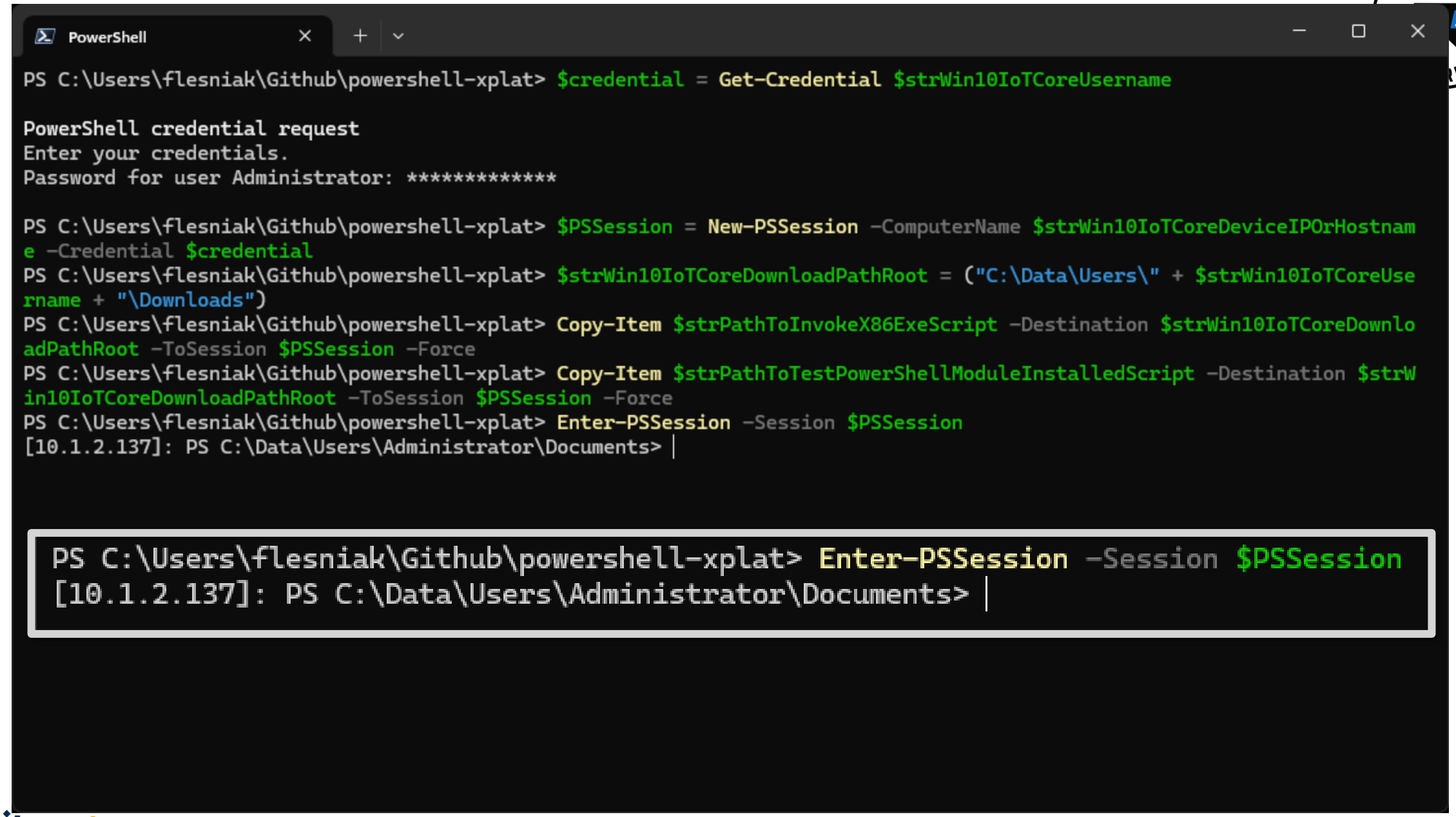
PS C:\Users\flesniak\Github\powershell-xplat> $PSSession = New-PSSession -ComputerName $strWin10IoTCoreDeviceIPOrHostname -Credential $credential
PS C:\Users\flesniak\Github\powershell-xplat>



PS C:\Users\flesniak\Github\powershell-xplat> $strWin10IoTCoreDownloadPathRoot = ("C:\Data\Users\" + $strWin10IoTCoreUsername + "\Downloads")



PS C:\Users\flesniak\Github\powershell-xplat> Copy-Item $strPathToInvokeX86ExeScript -Destination $strWin10IoTCoreDownloadPathRoot -ToSession $PSSession -Force
PS C:\Users\flesniak\Github\powershell-xplat> Copy-Item $strPathToTestPowerShellModuleInstalledScript -Destination $strWin10IoTCoreDownloadPathRoot -ToSession $PSSession -Force



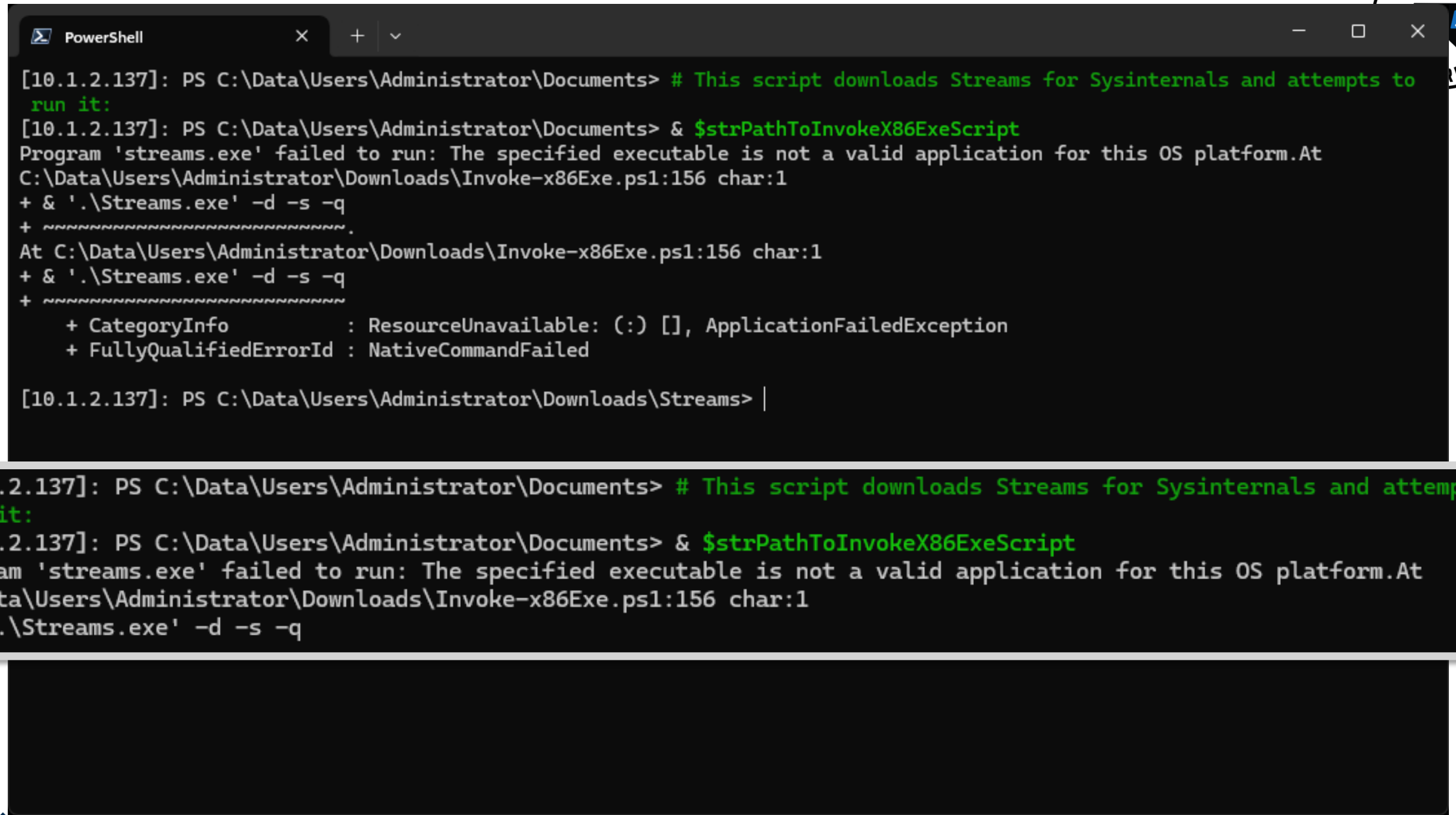

```
PS C:\Users\flesniak\Github\powershell-xplat> Enter-PSSession -Session $PSSession
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents>
```


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

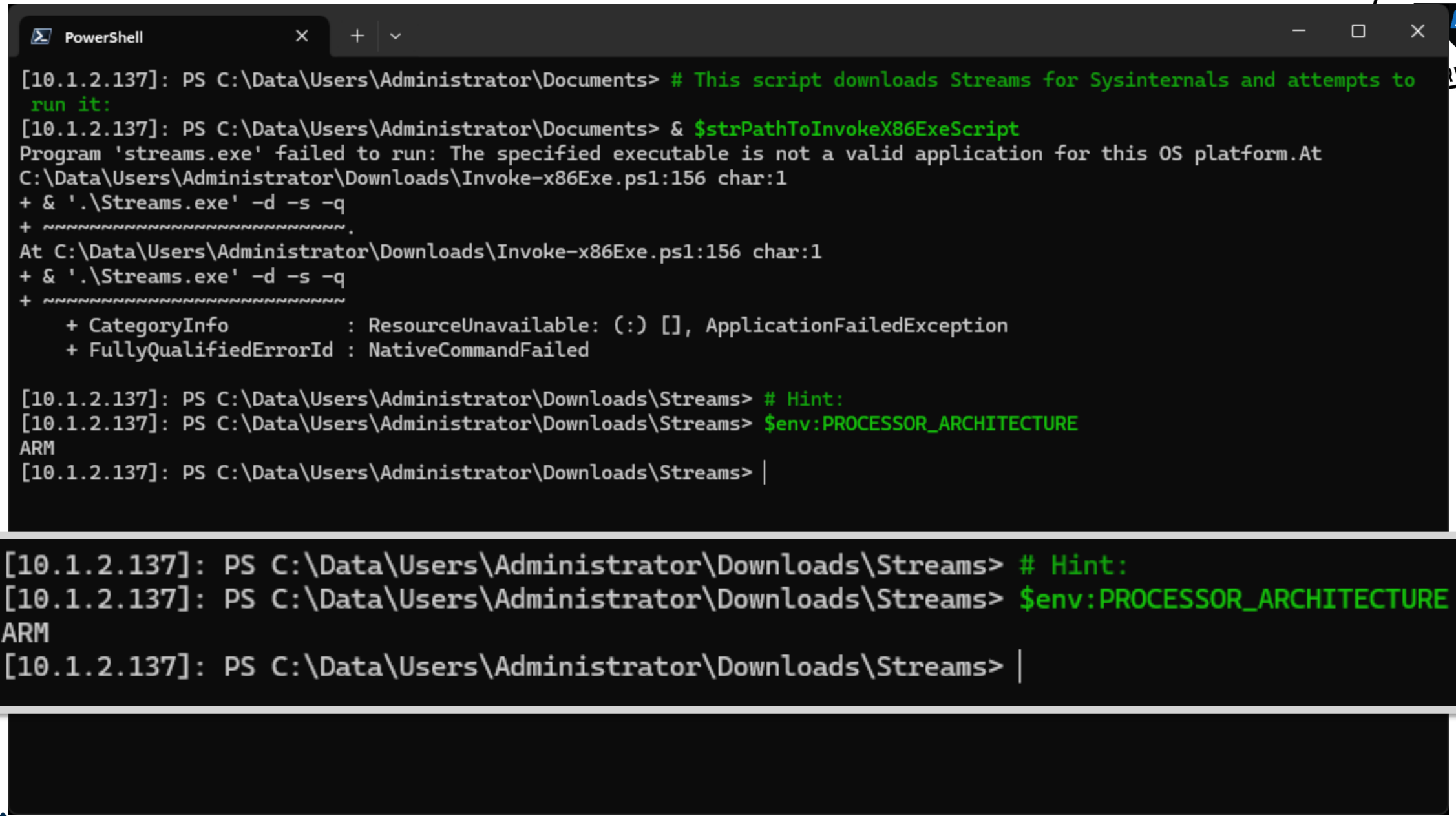
```
#region Part 3: Try some things on Windows 10 IoT Core ##############################
$strDownloadFolder = Get-DownloadFolder
$strPathToInvokeX86ExeScript = Join-Path $strDownloadFolder 'Invoke-x86Exe.ps1'
$strPathToTestPowerShellModuleInstalledScript = Join-Path $strDownloadFolder 'Test-PowerShellModuleInstalled.p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


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

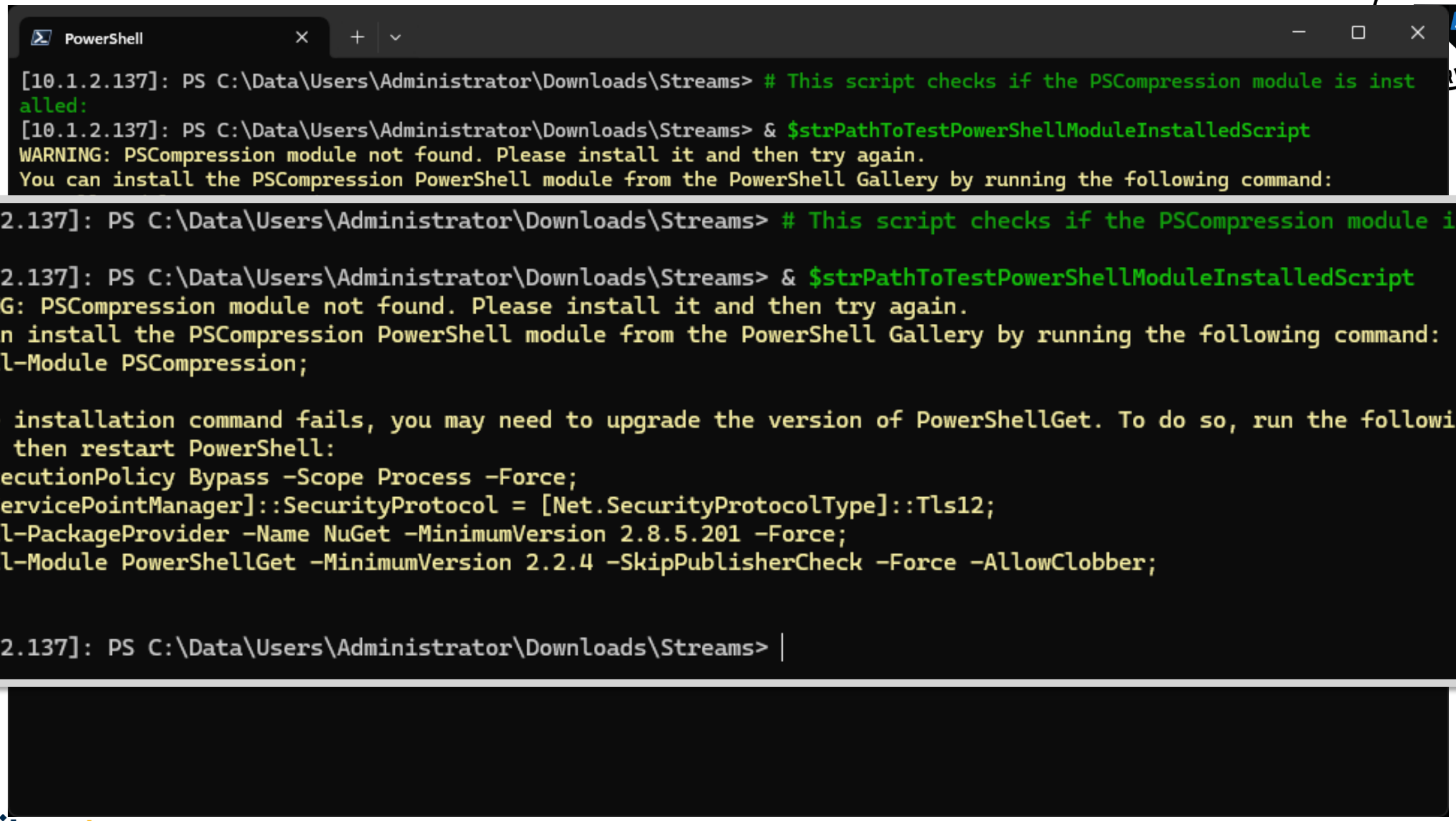
```
#region Part 3: Try some things on Windows 10 IoT Core ################################
$strDownloadFolder = Get-DownloadFolder
$strPathToInvokeX86ExeScript = Join-Path $strDownloadFolder 'Invoke-x86Exe.ps1'
$strPathToTestPowerShellModuleInstalledScript = Join-Path $strDownloadFolder 'Test-PowerShellModuleInstalled.p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

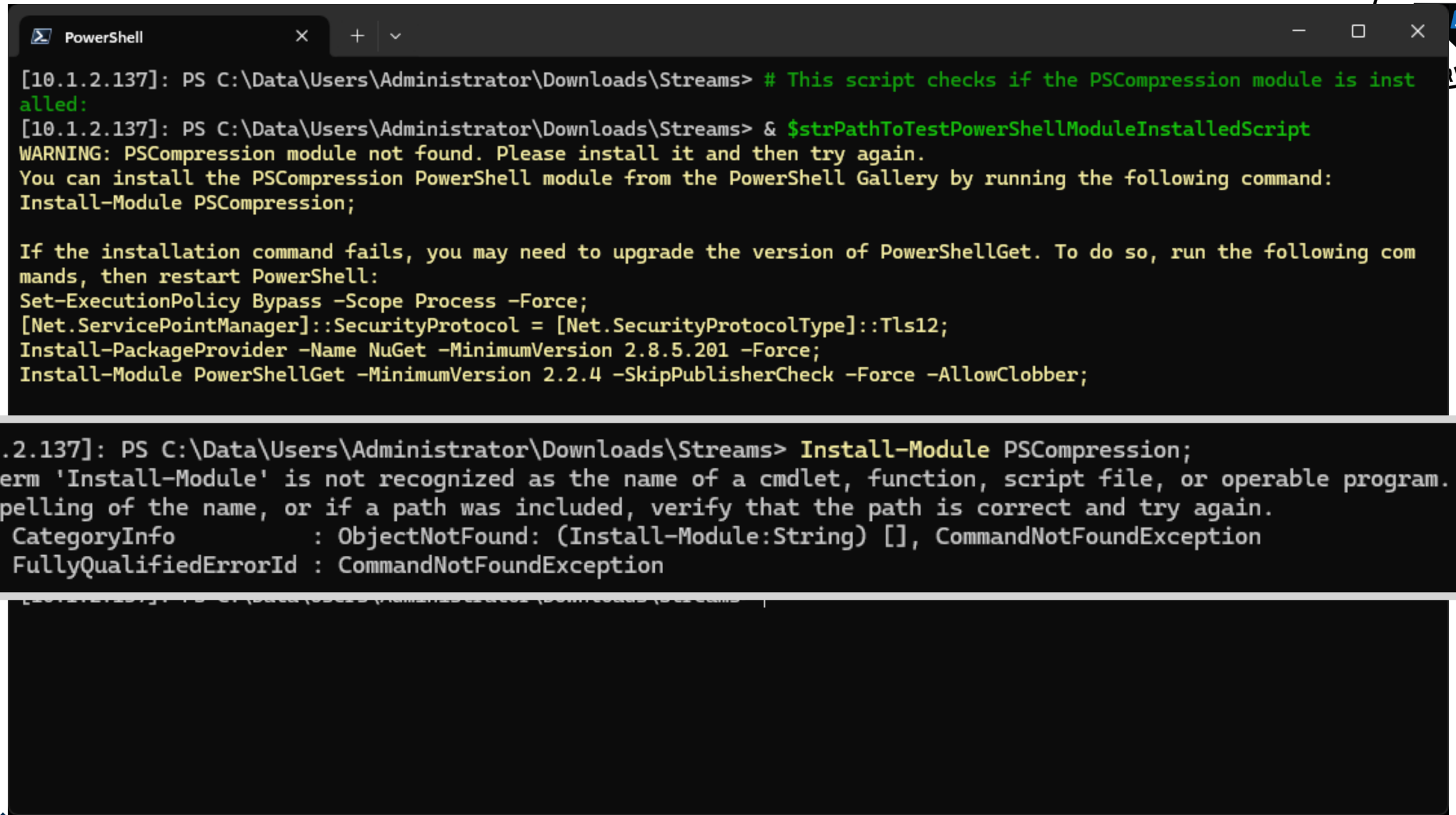


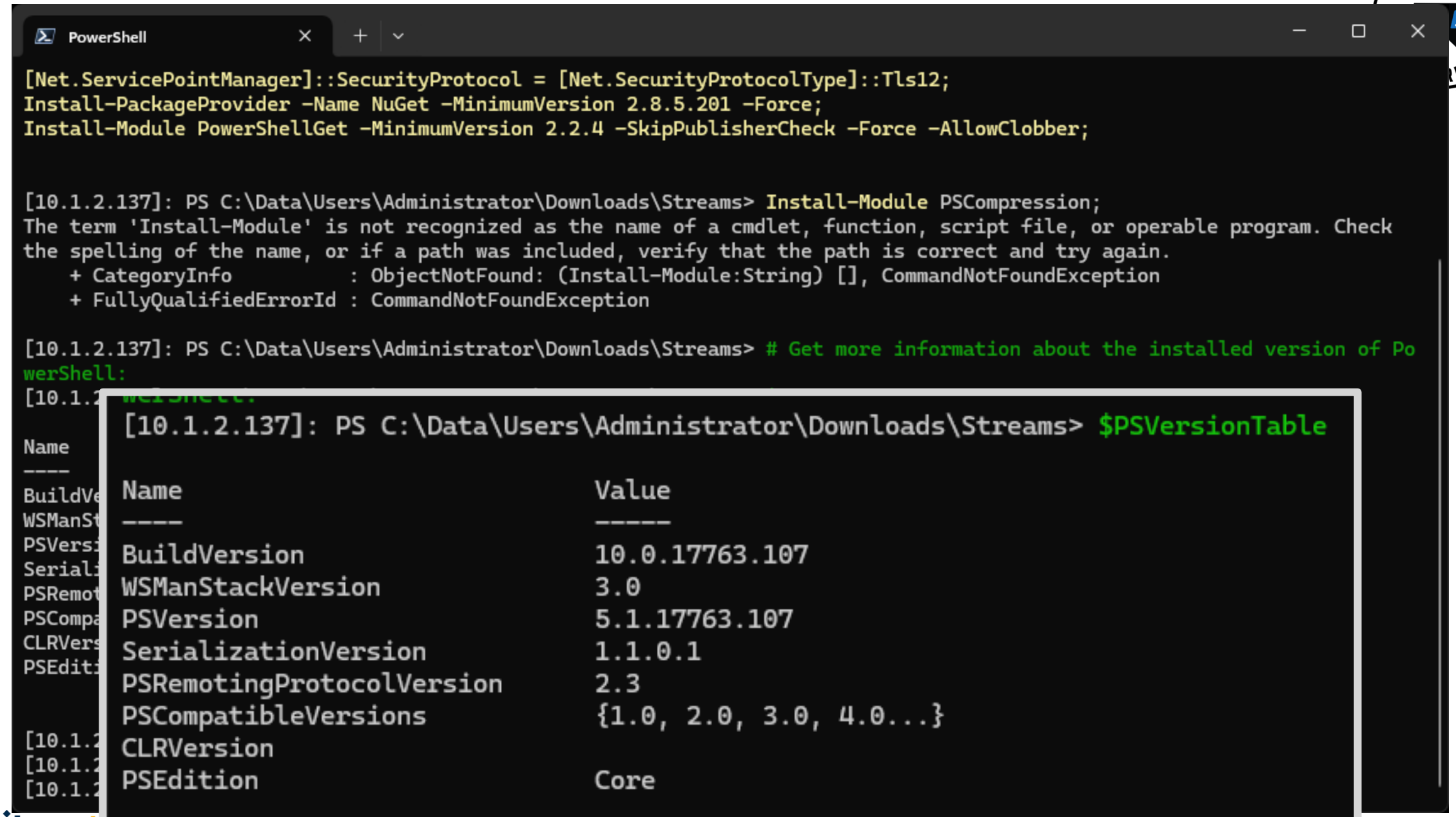
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> # This script downloads Streams for Sysinternals and attempts to run it:
[10.1.2.137]: PS C:\Data\Users\Administrator\Documents> & $strPathToInvokeX86ExeScript
Program 'streams.exe' failed to run: The specified executable is not a valid application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

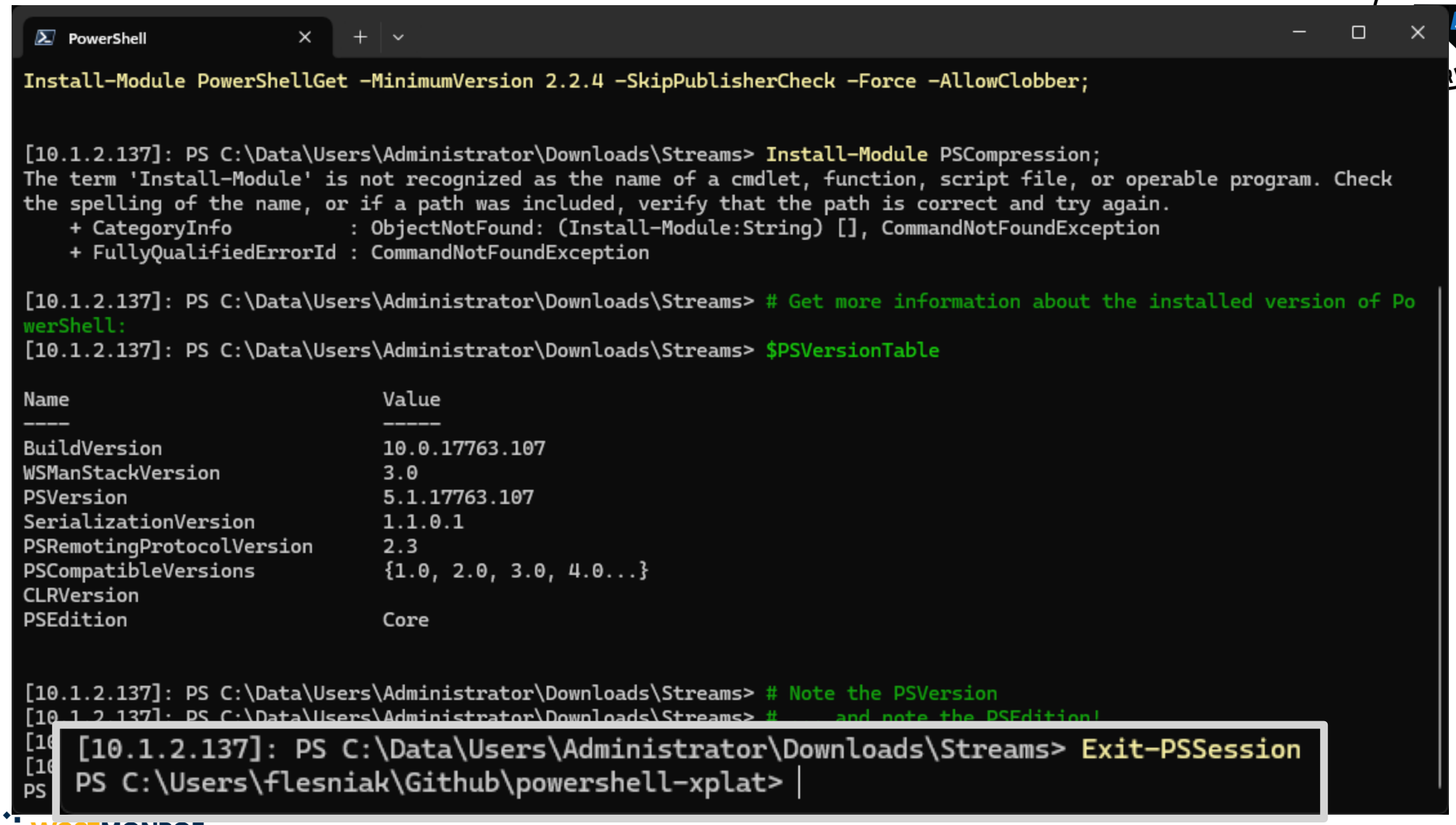


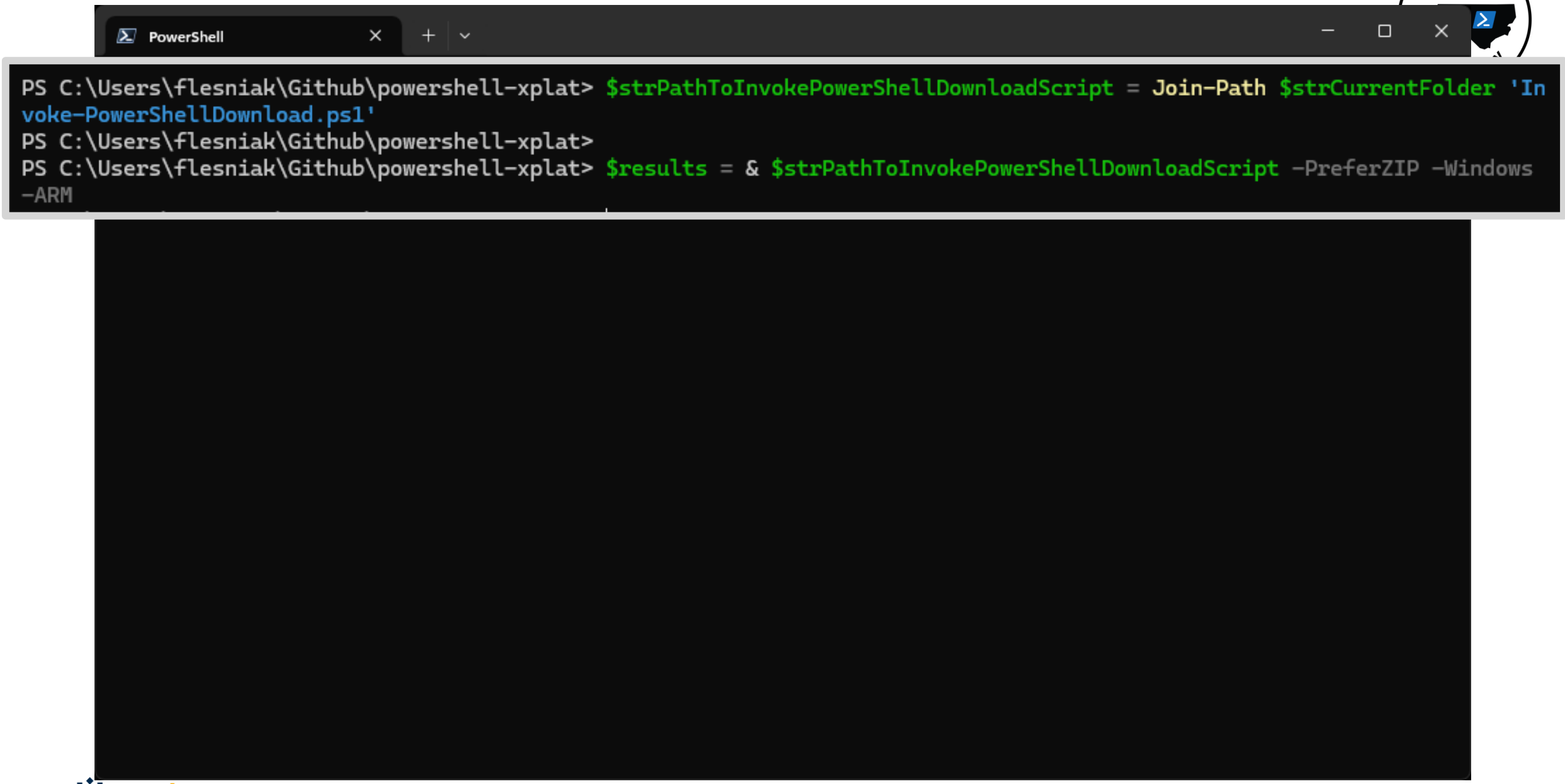
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> # Hint:
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams> $env:PROCESSOR_ARCHITECTURE
ARM
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads\Streams>

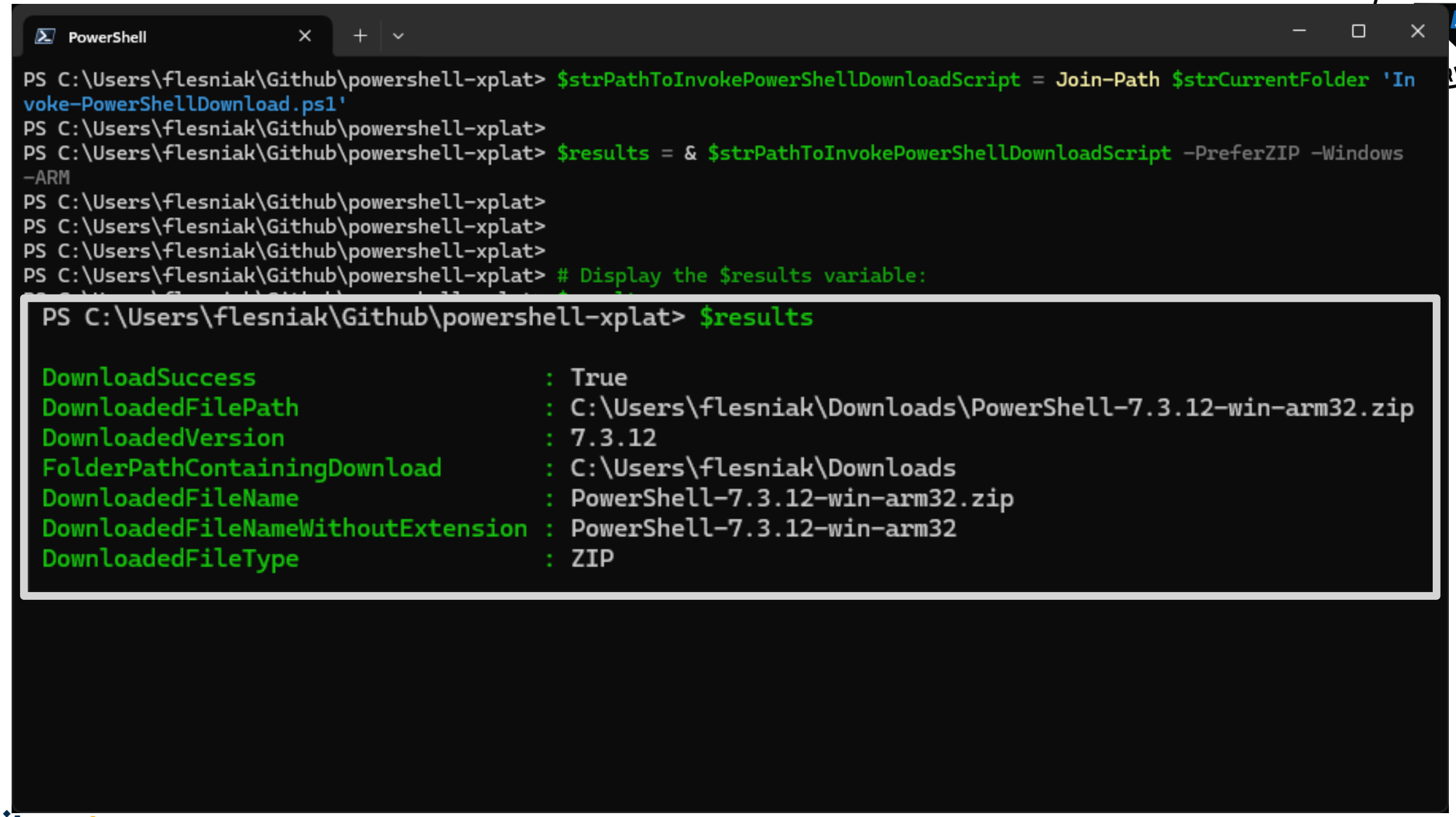


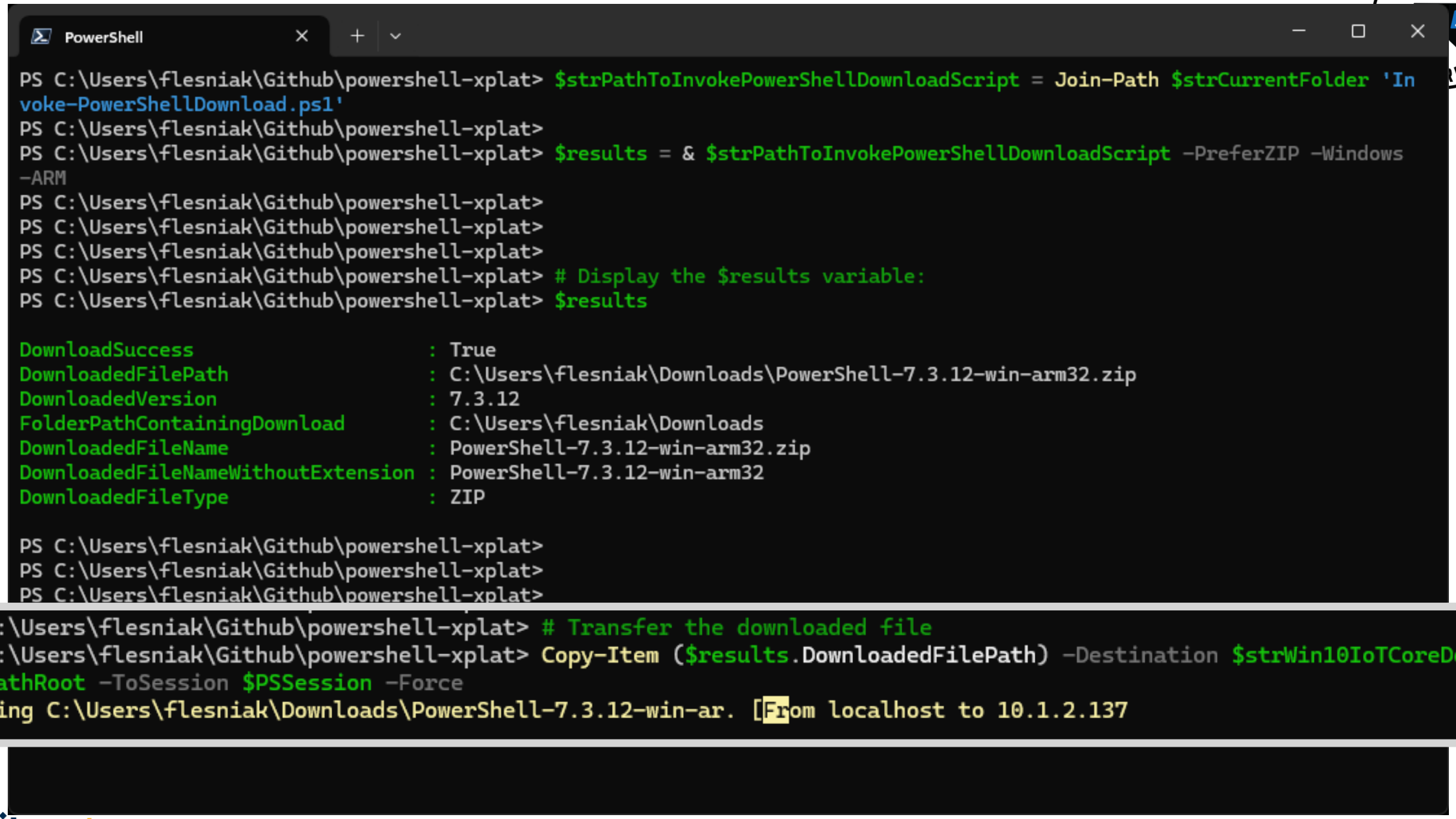






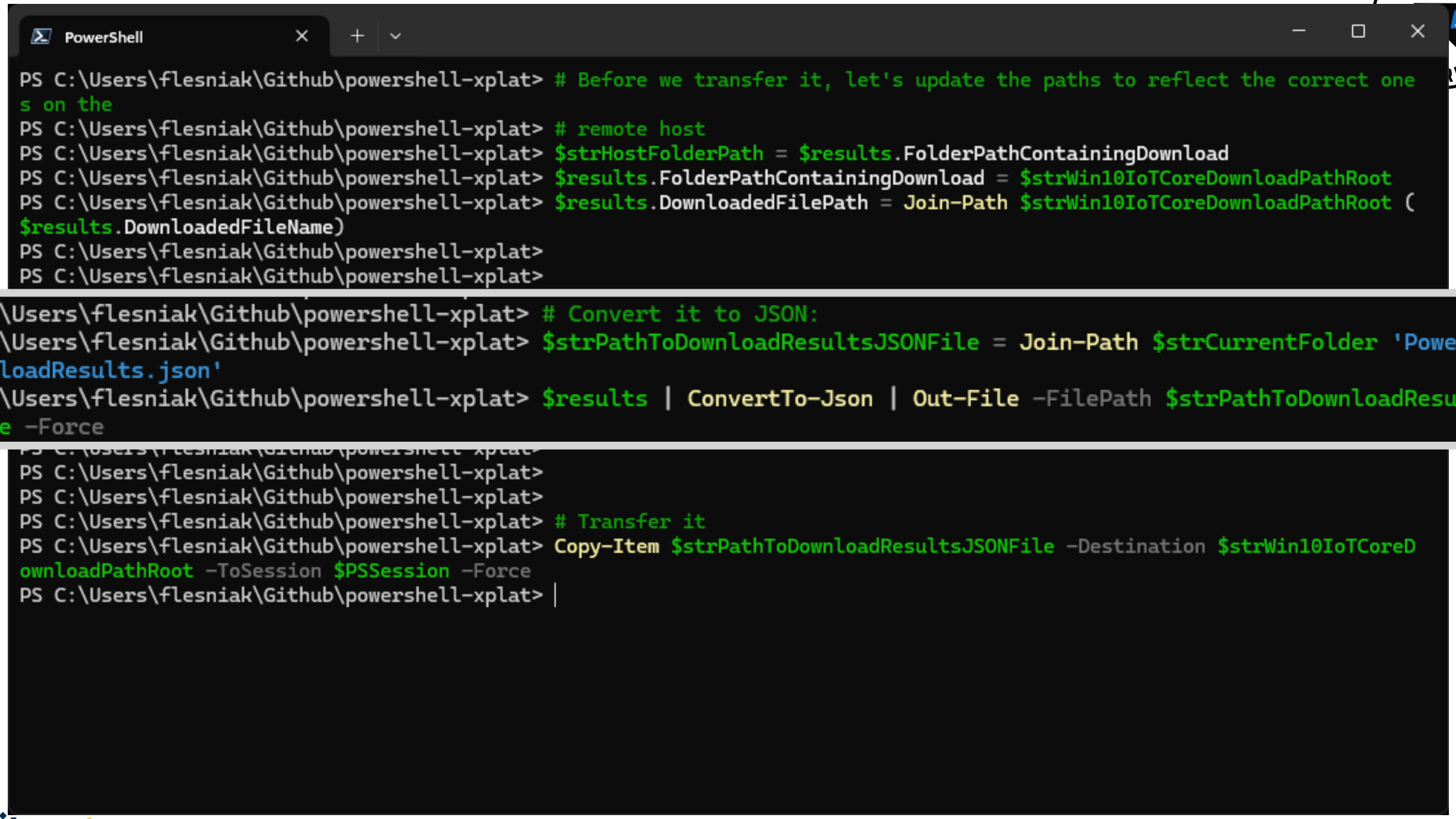


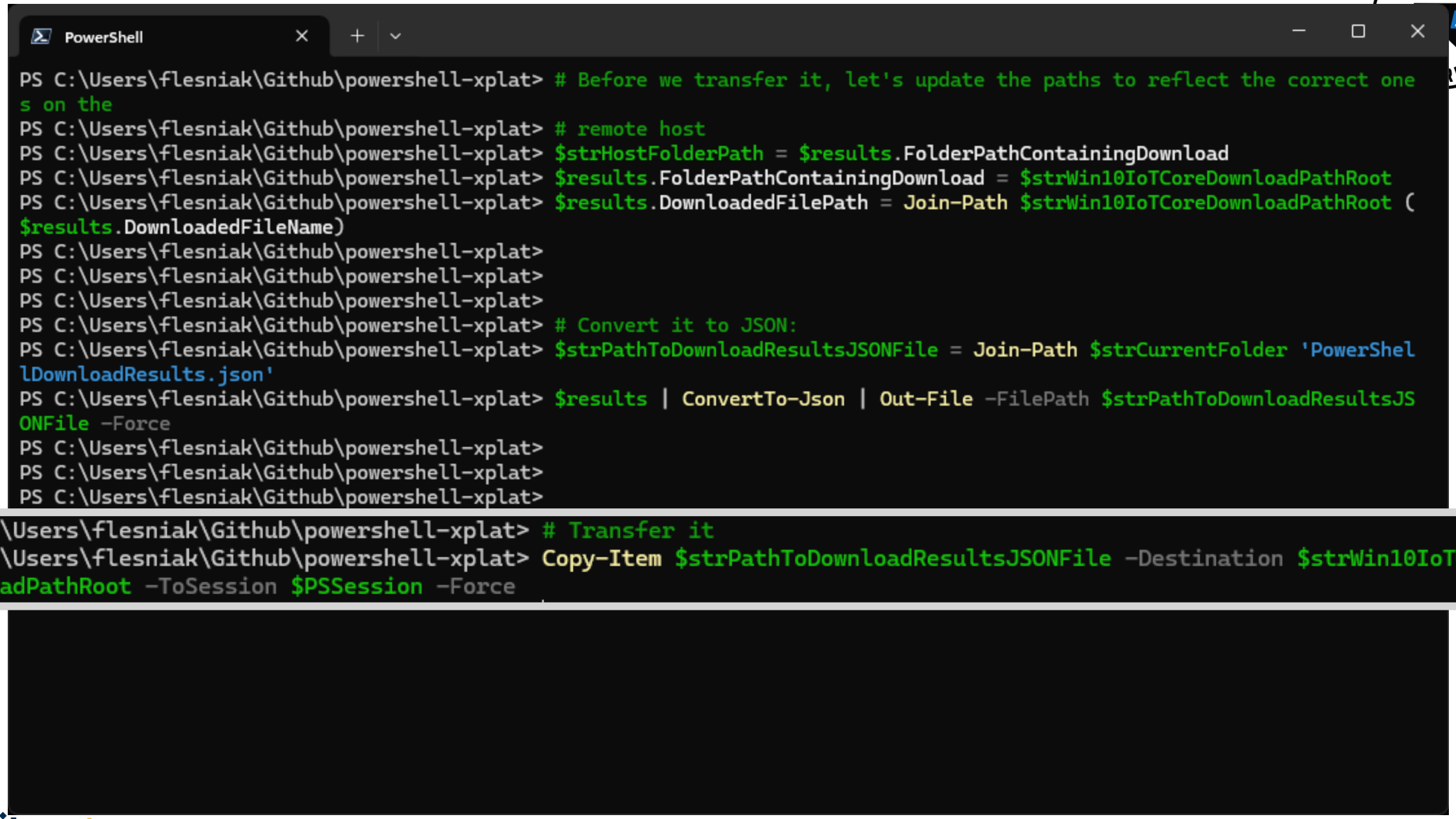


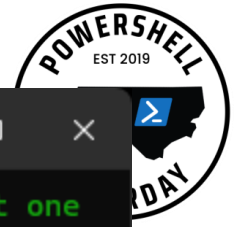


PowerShell

```
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>
```



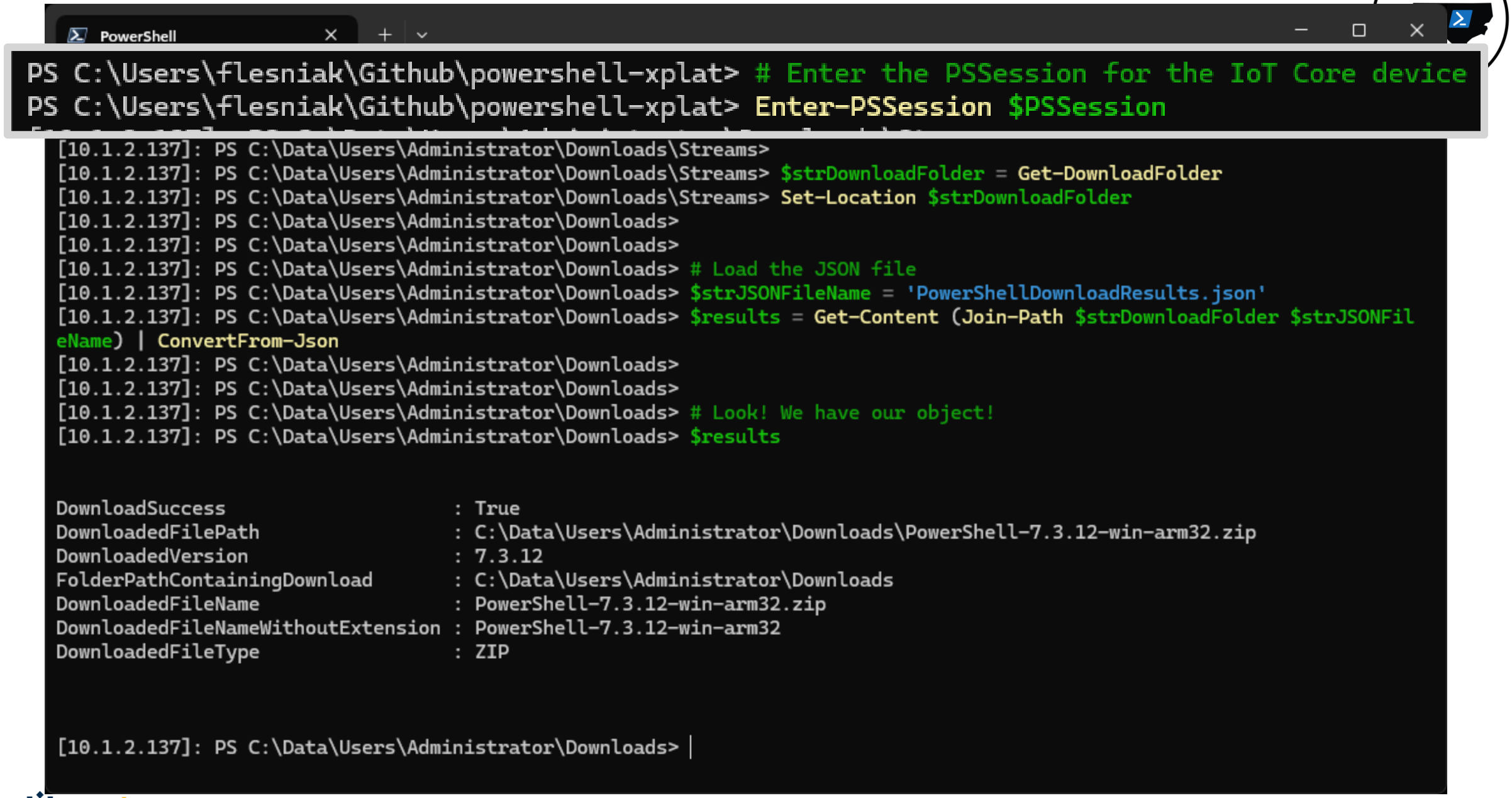


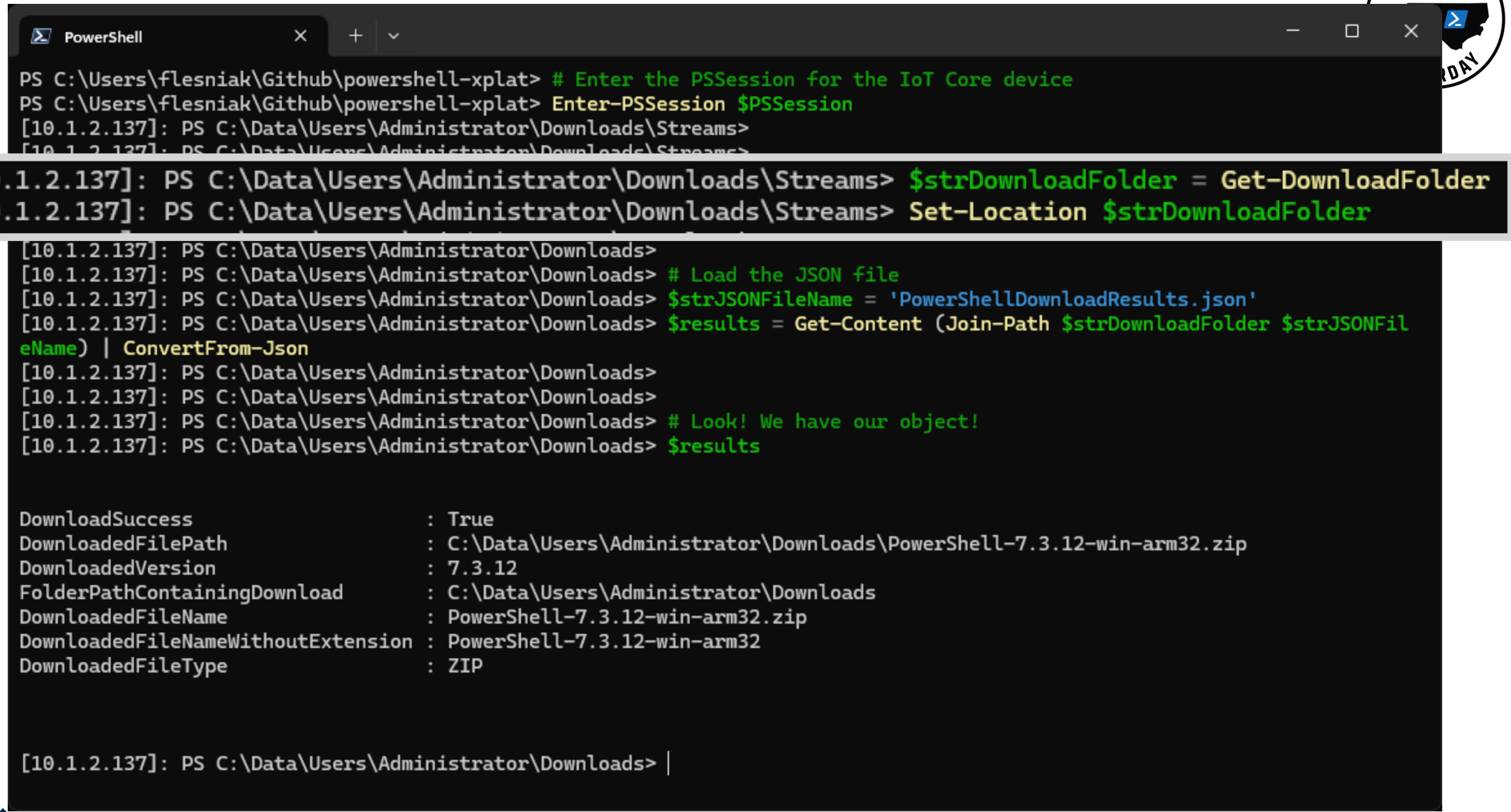


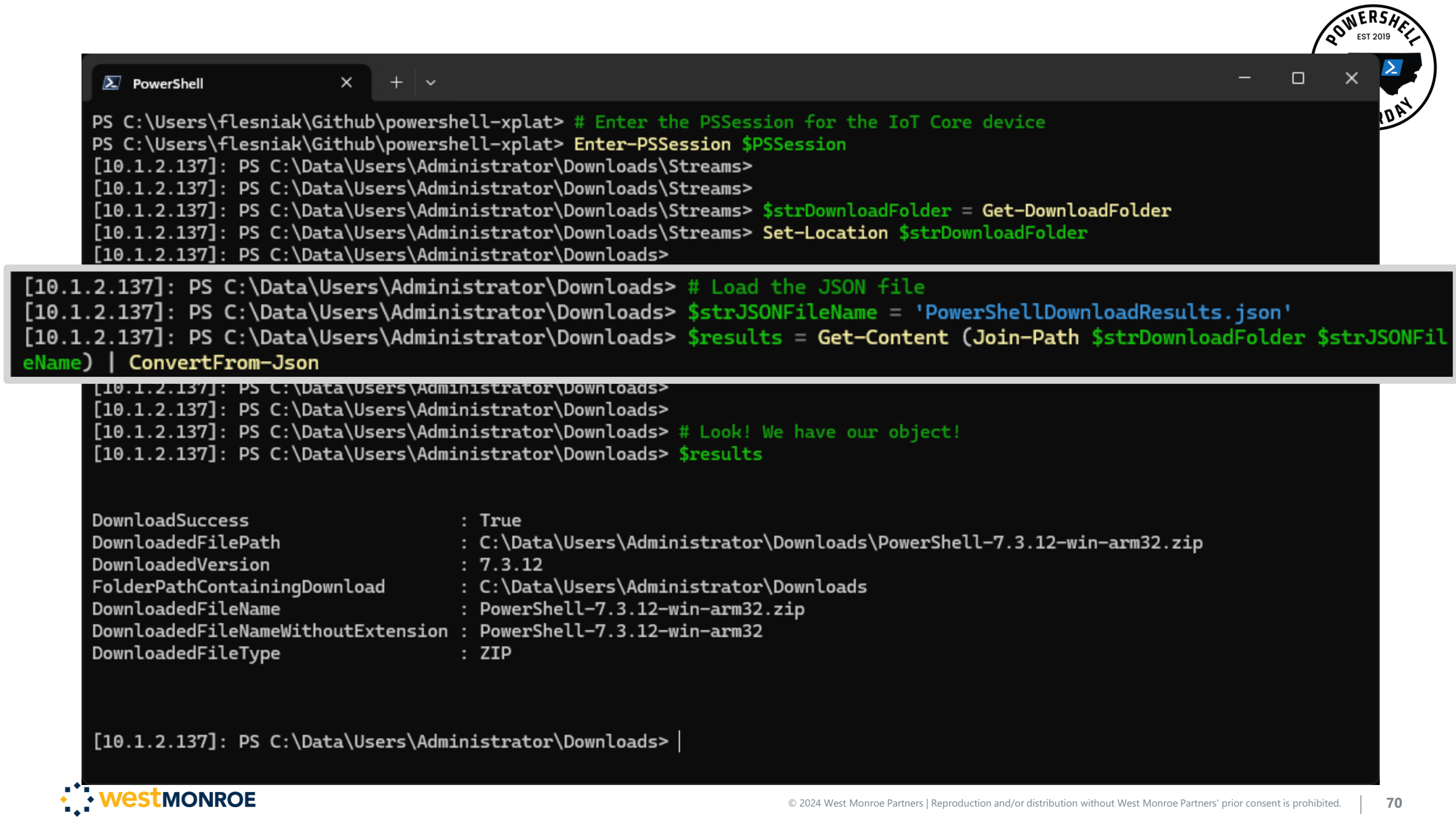
PowerShell

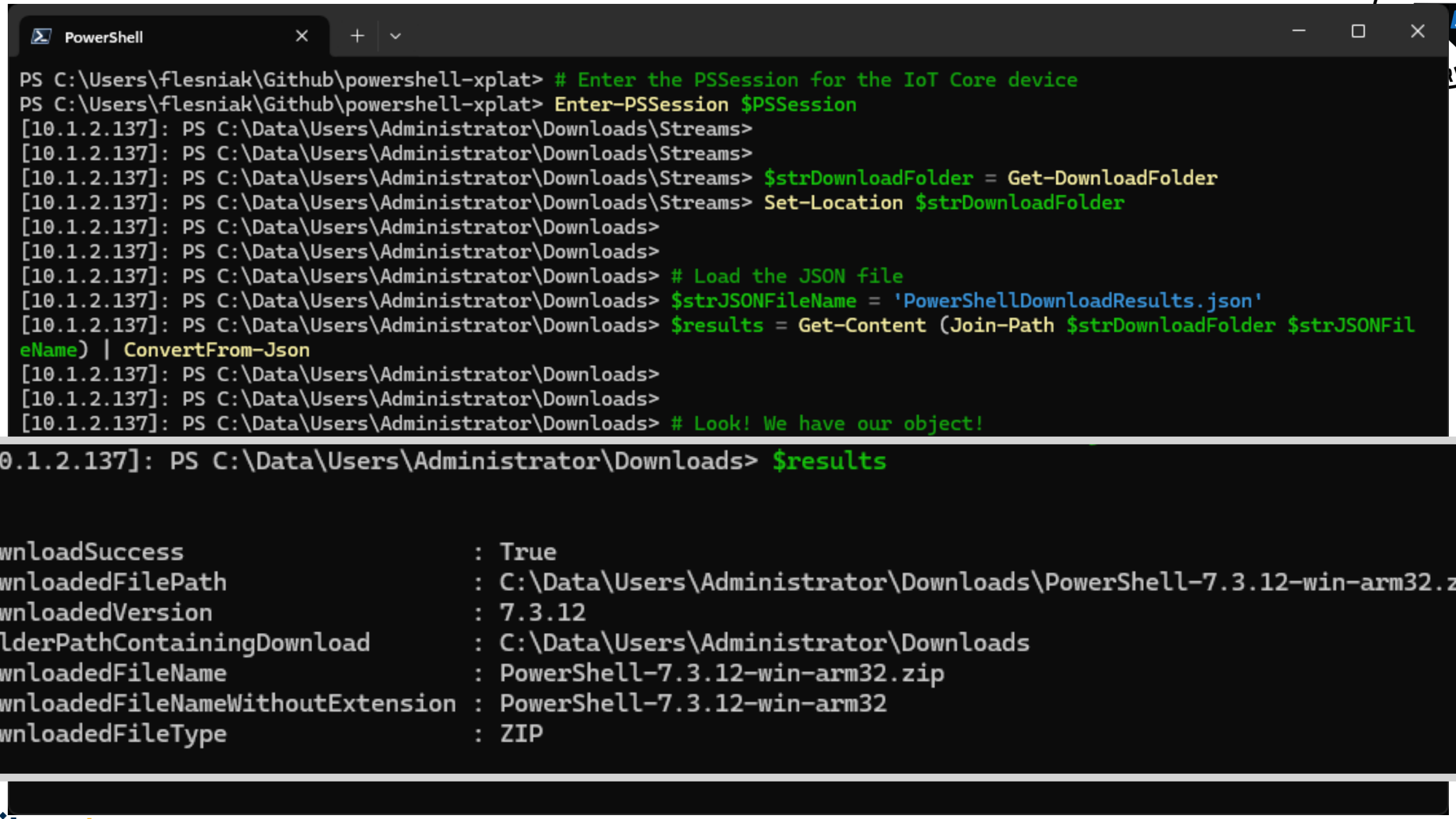
```
PS C:\Users\flesniak\Github\powershell-xplat> # Before we transfer it, let's update the paths to reflect the correct one
s on the
PS C:\Users\flesniak\Github\powershell-xplat> # remote host
PS C:\Users\flesniak\Github\powershell-xplat> $strHostFolderPath = $results.FolderPathContainingDownload
PS 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)
```

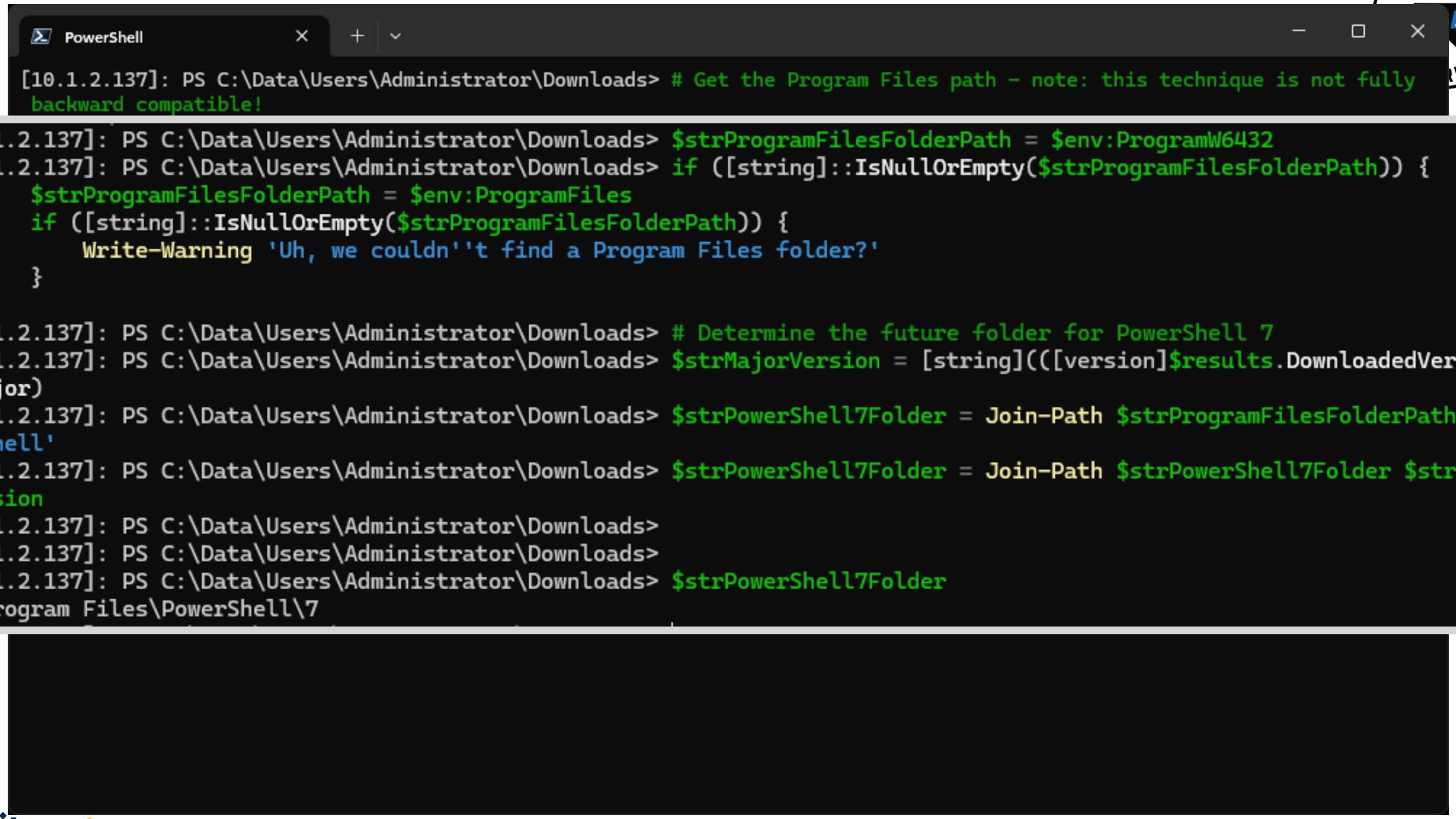


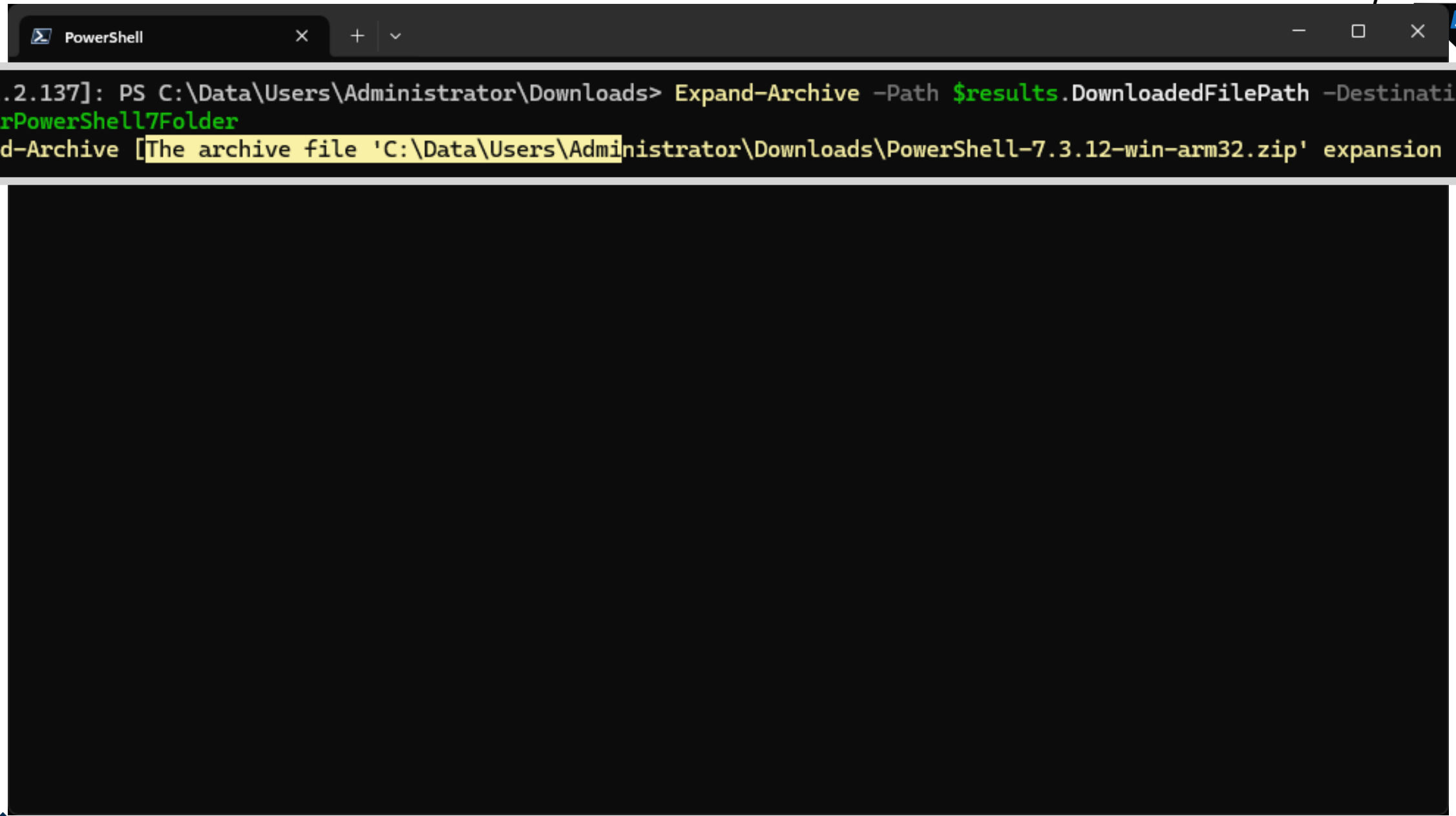


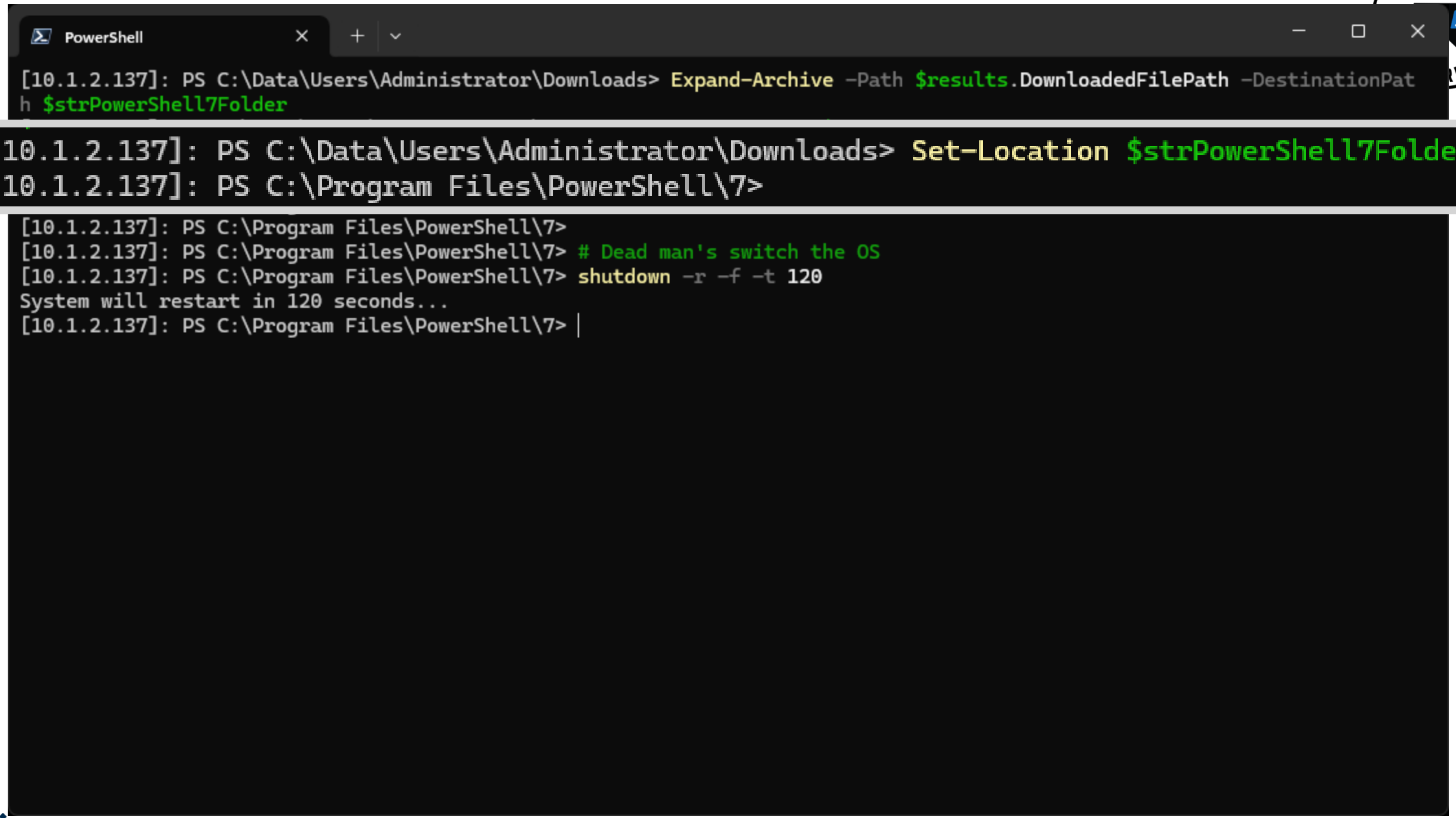


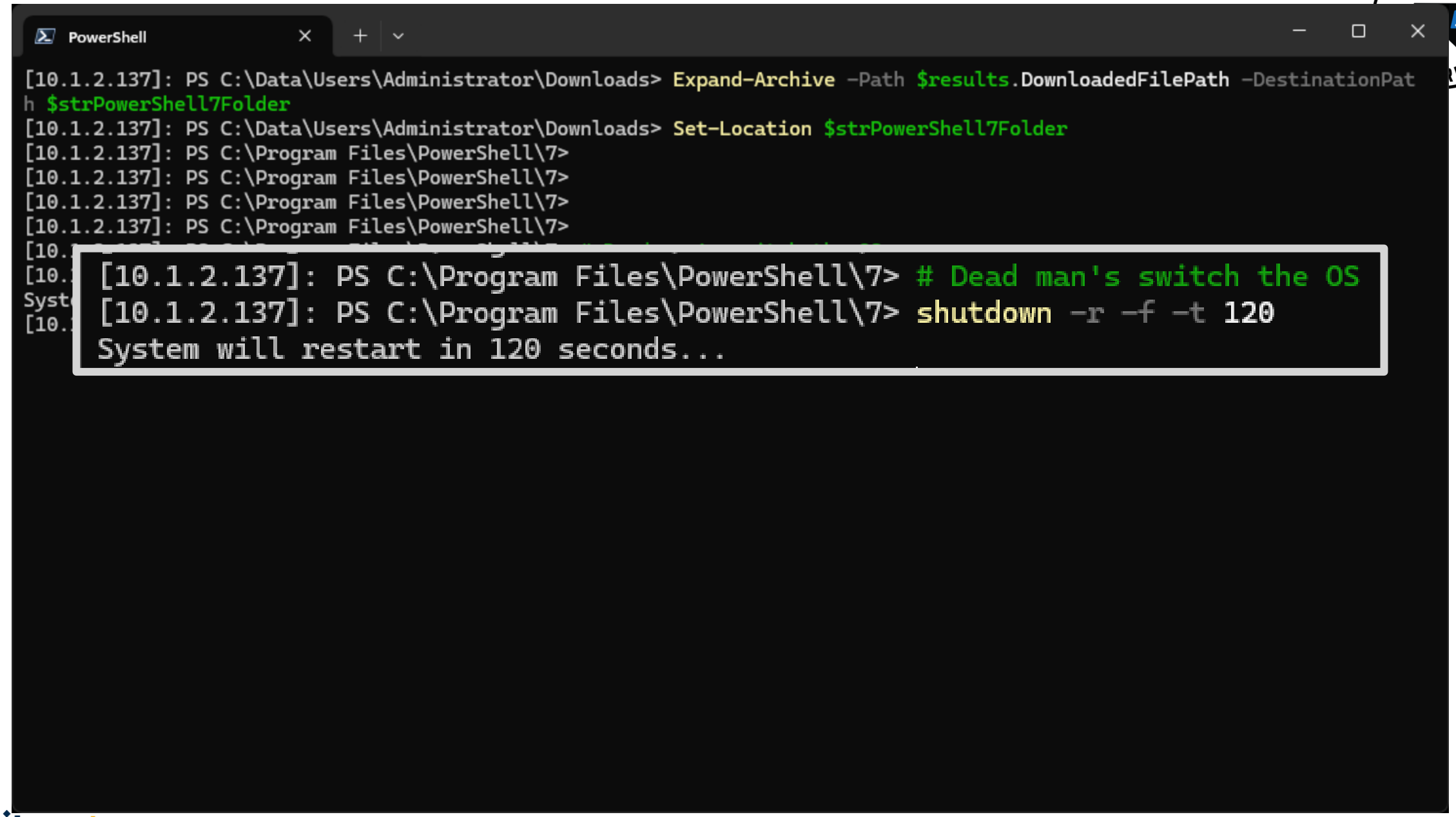


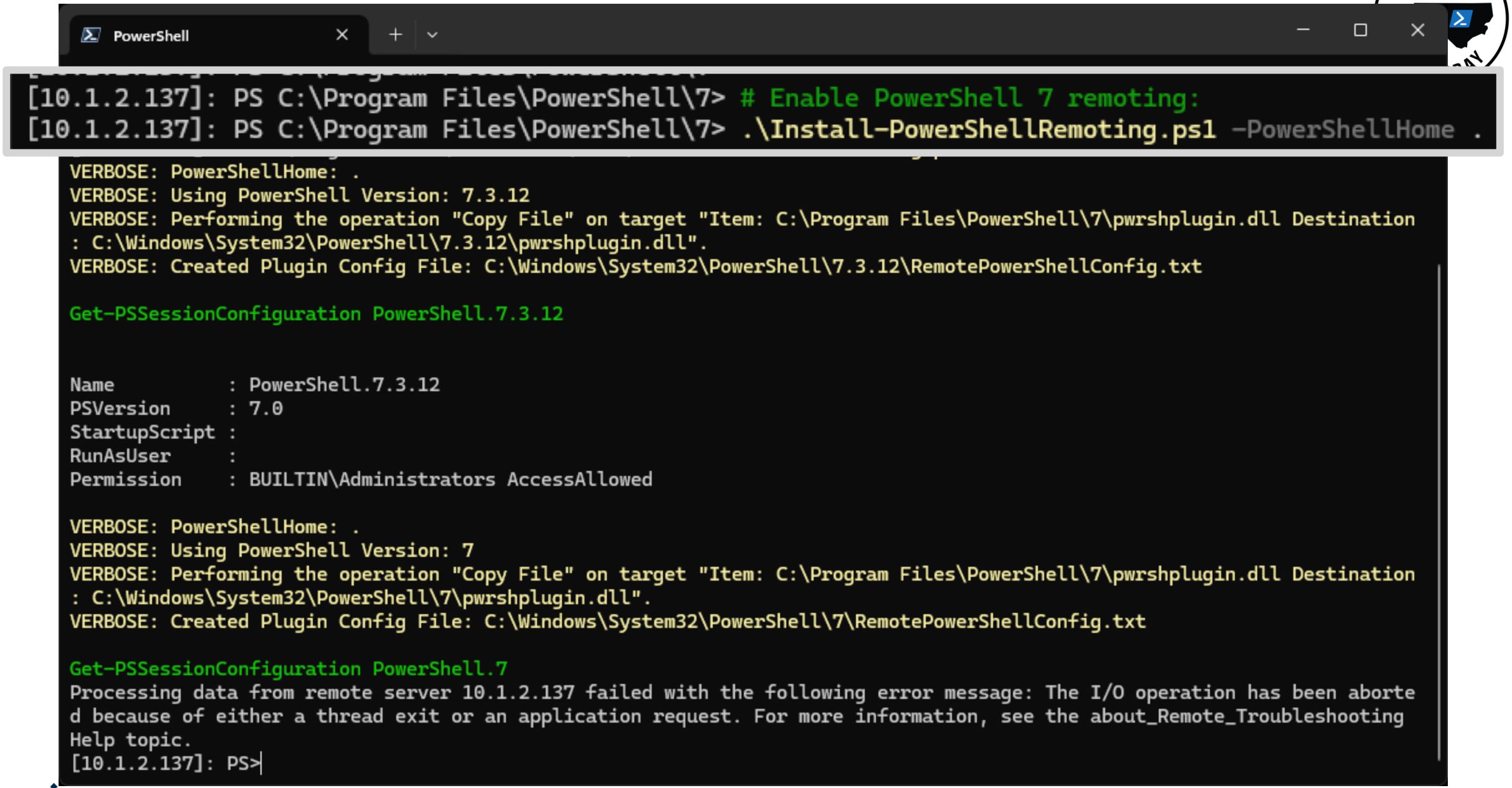

```
[10.1.2.137]: PS C:\Data\Users\Administrator\Downloads> $results


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
```

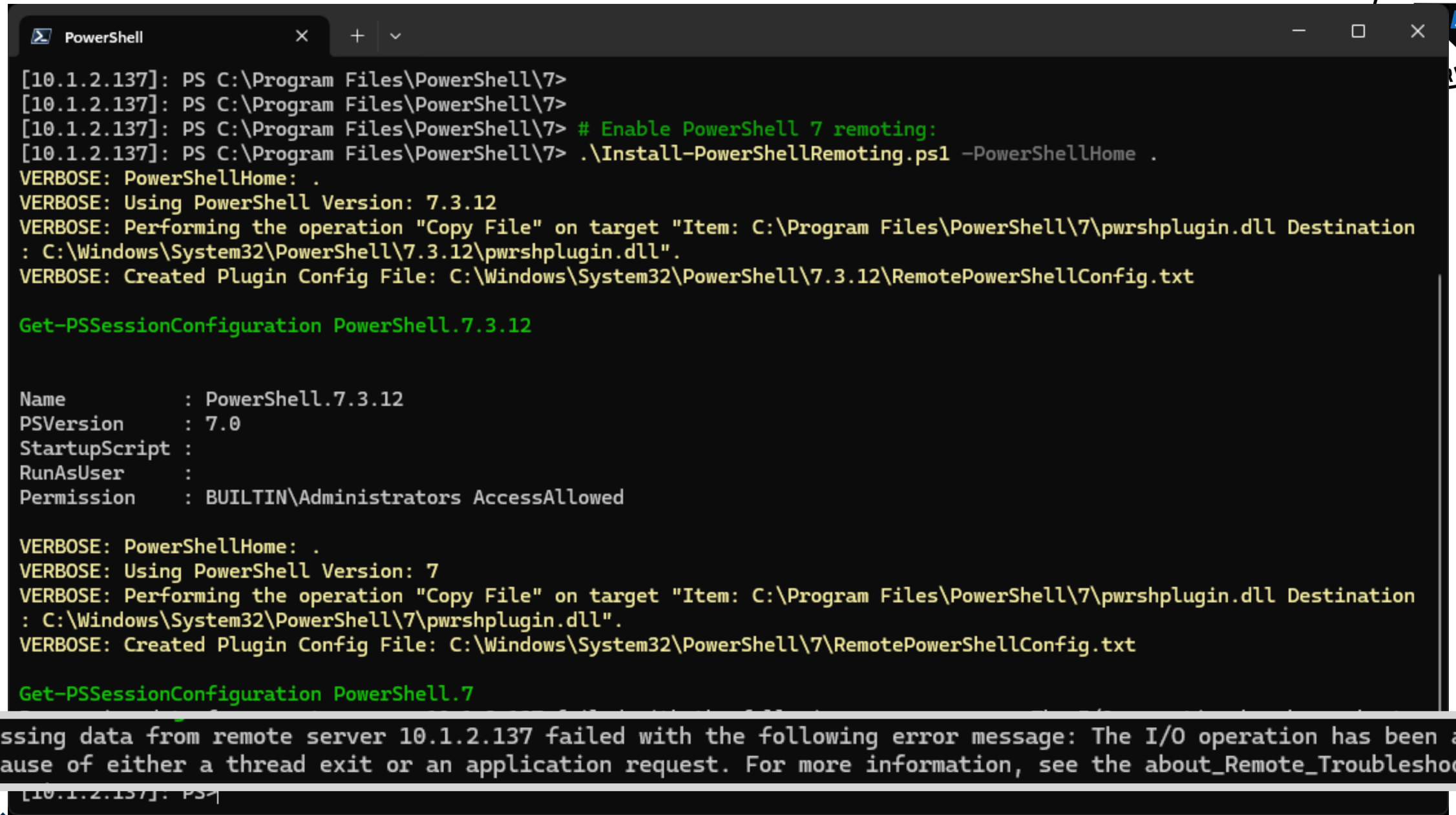


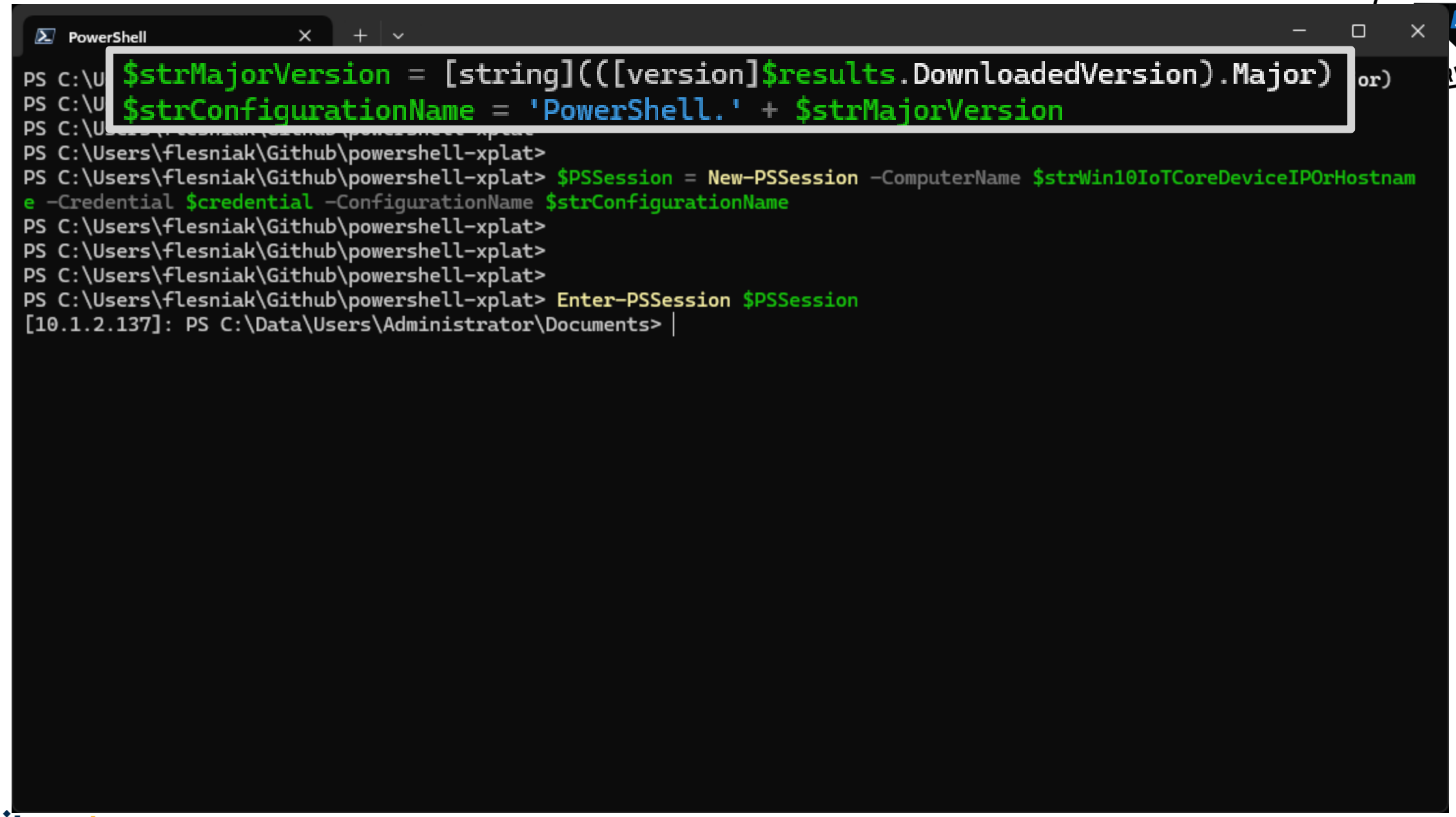


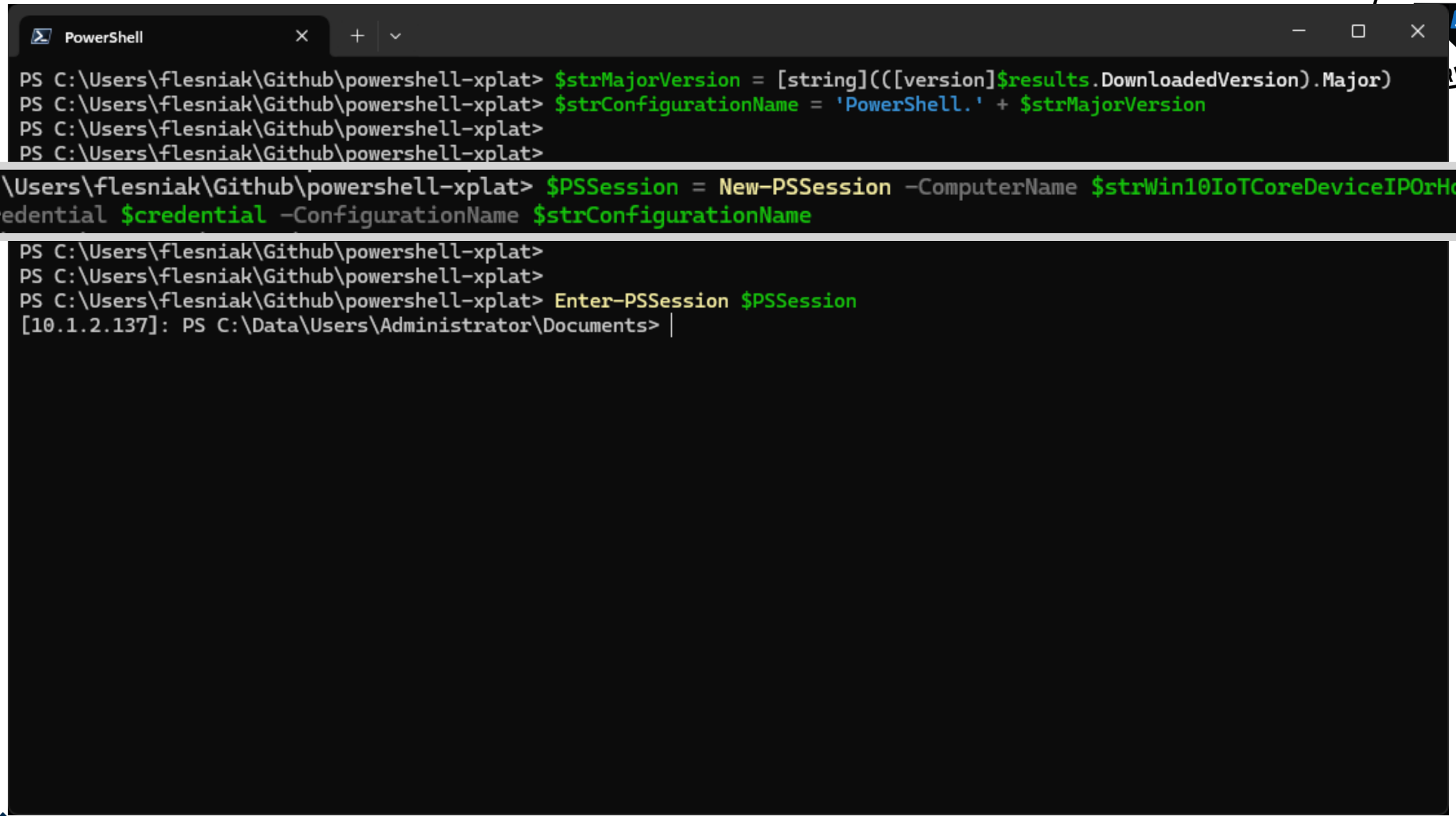


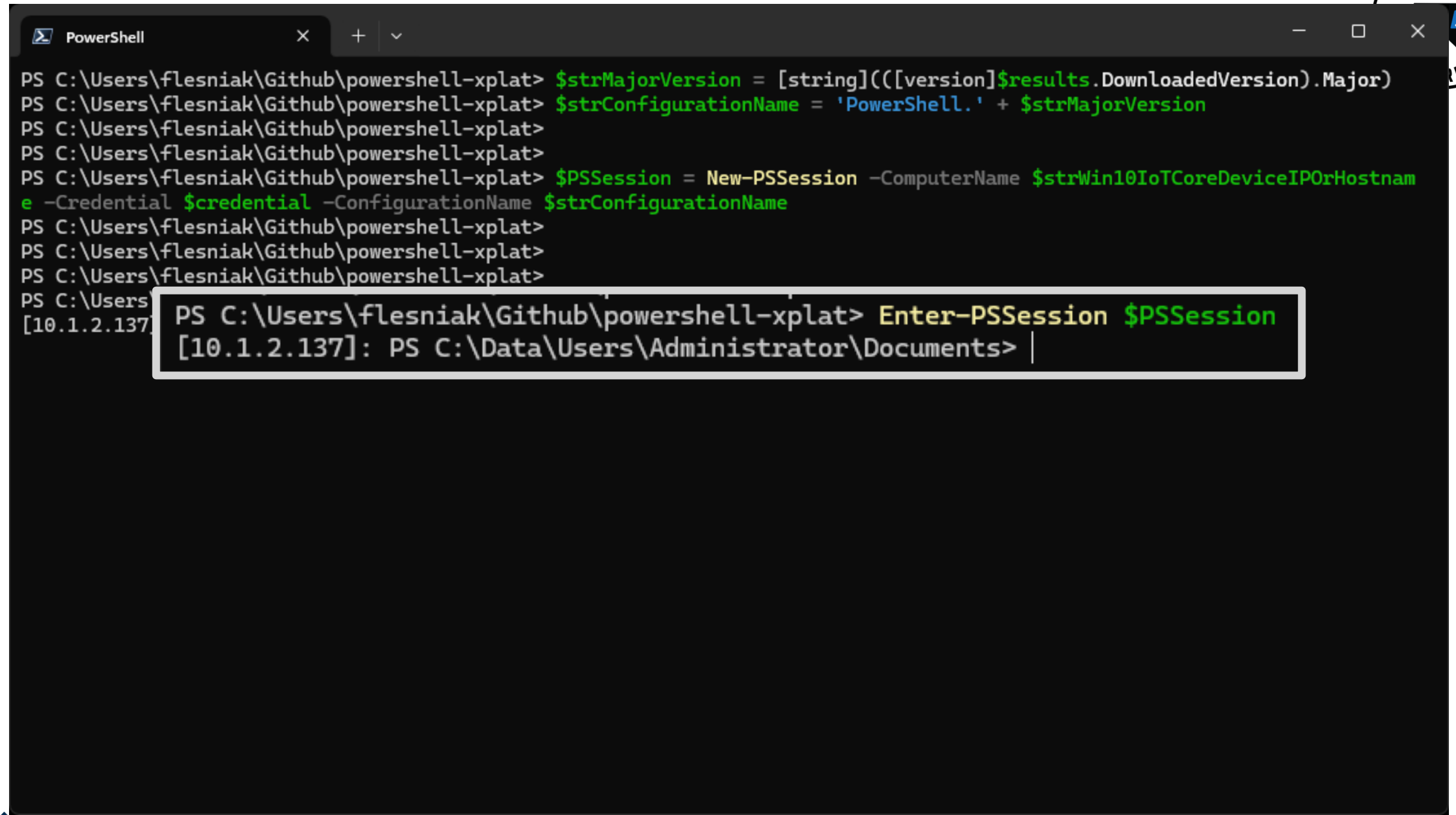


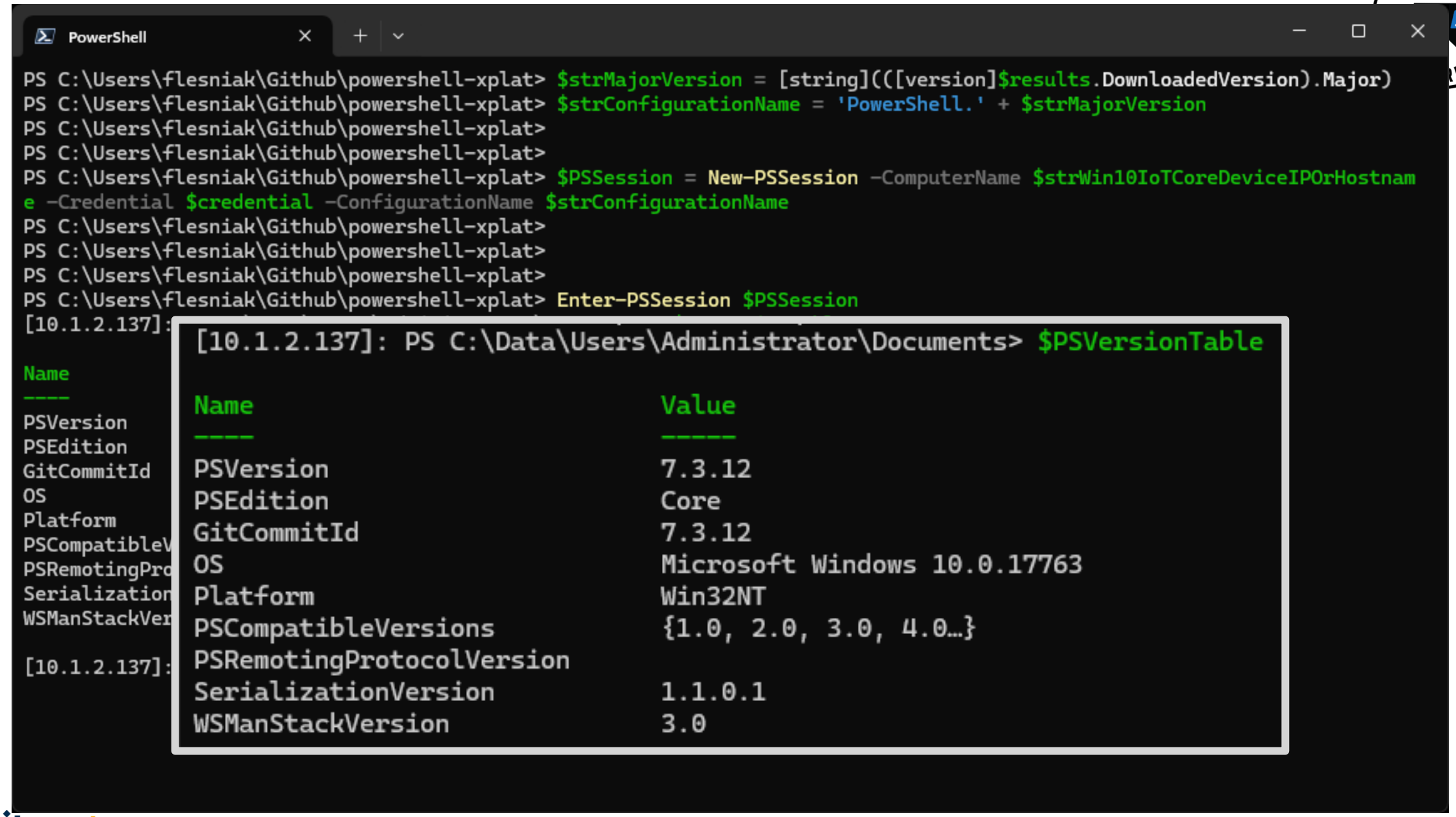




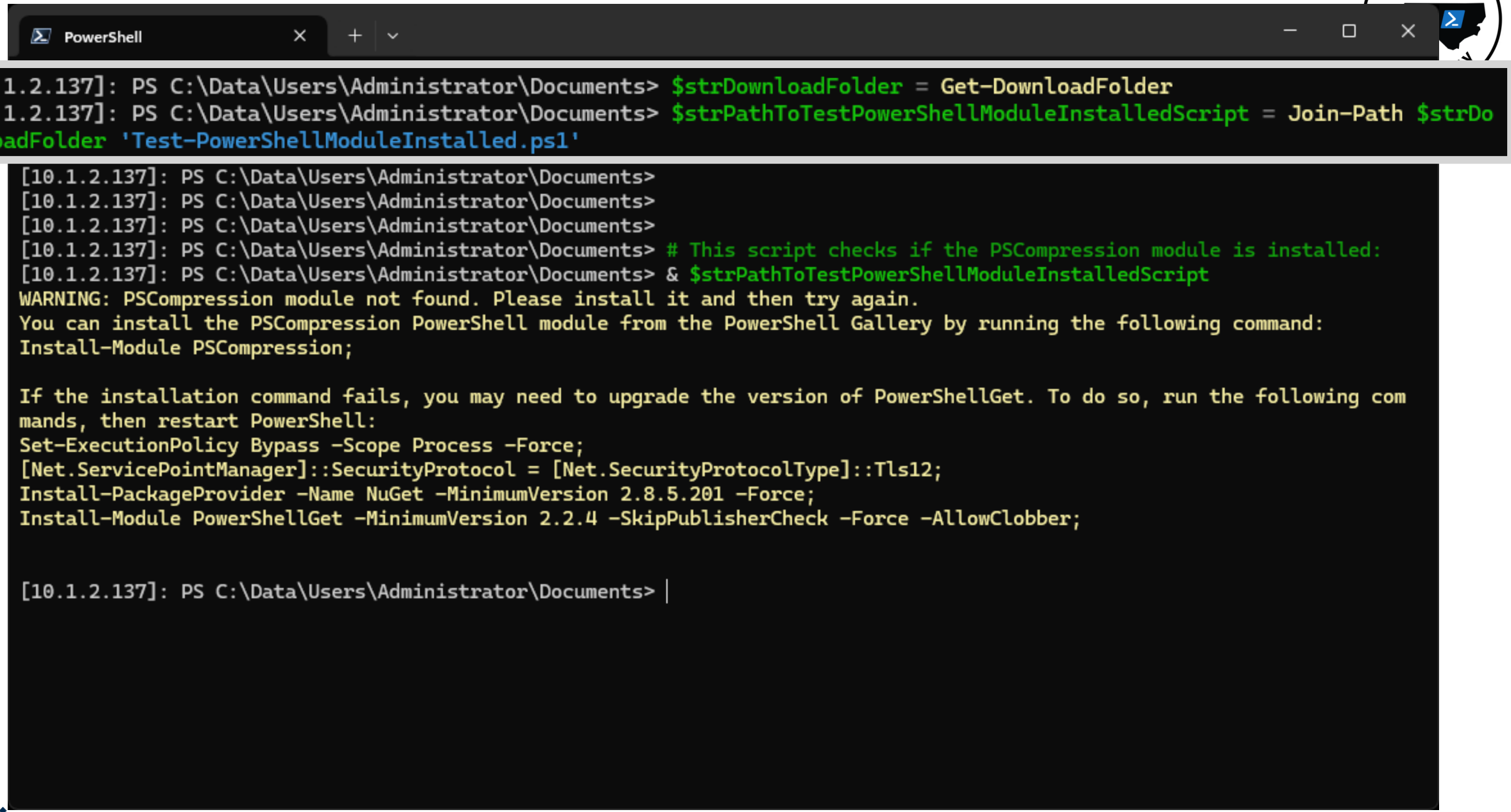


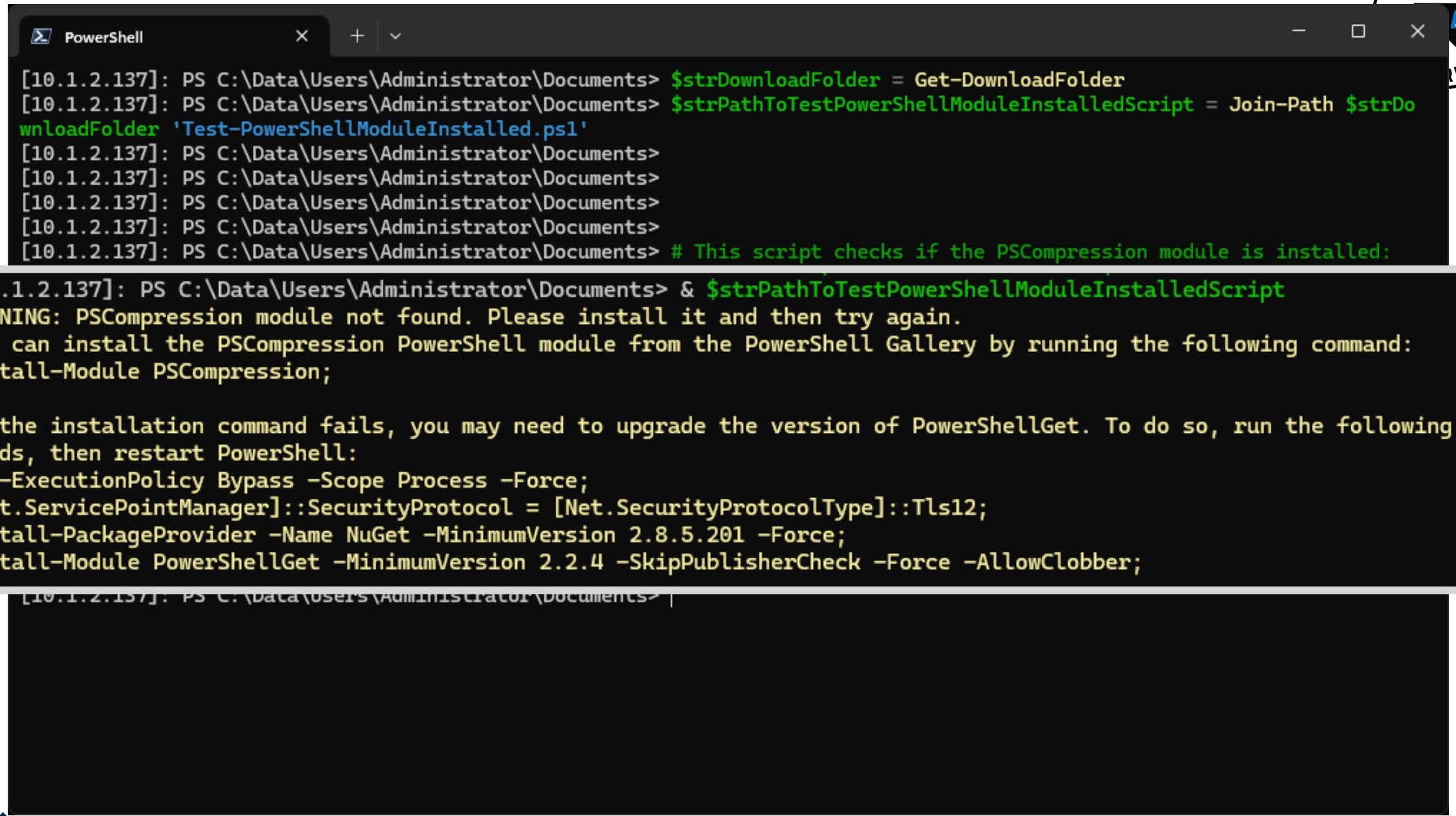


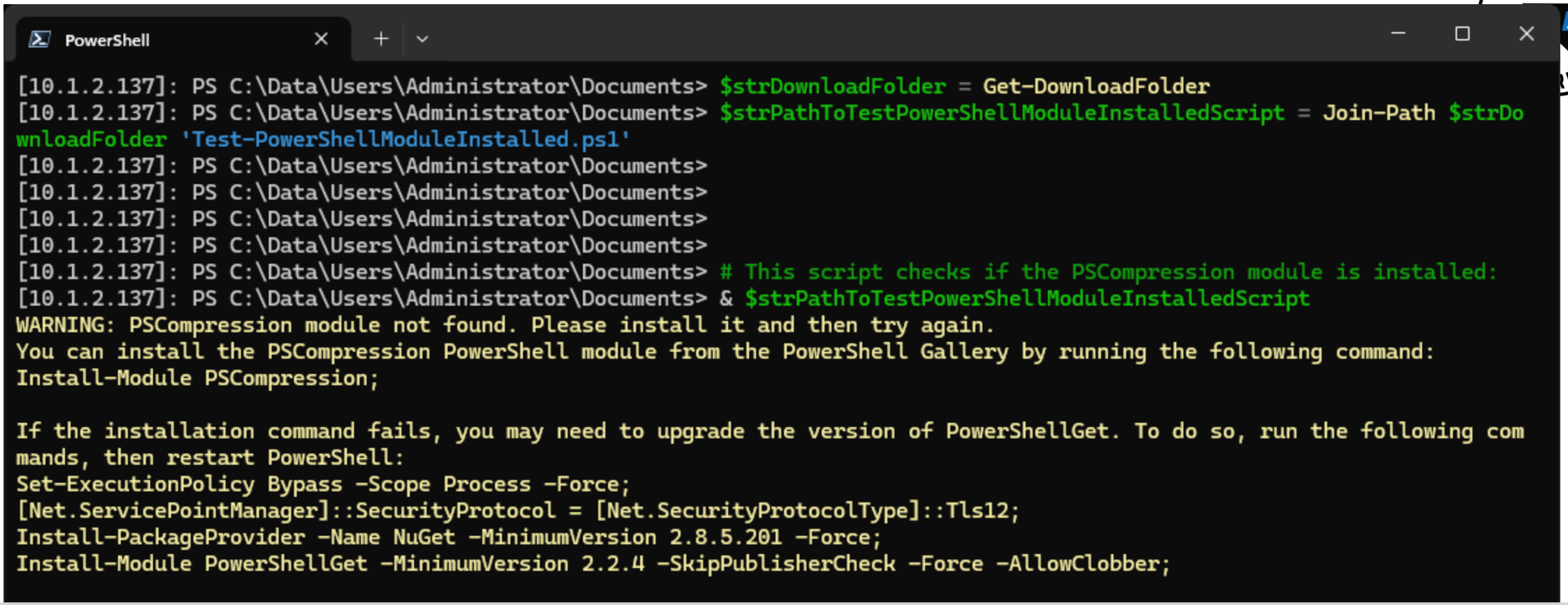


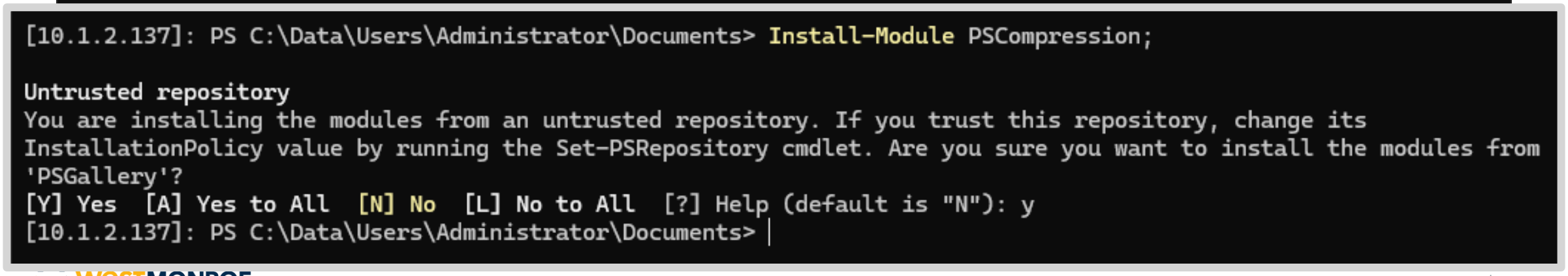


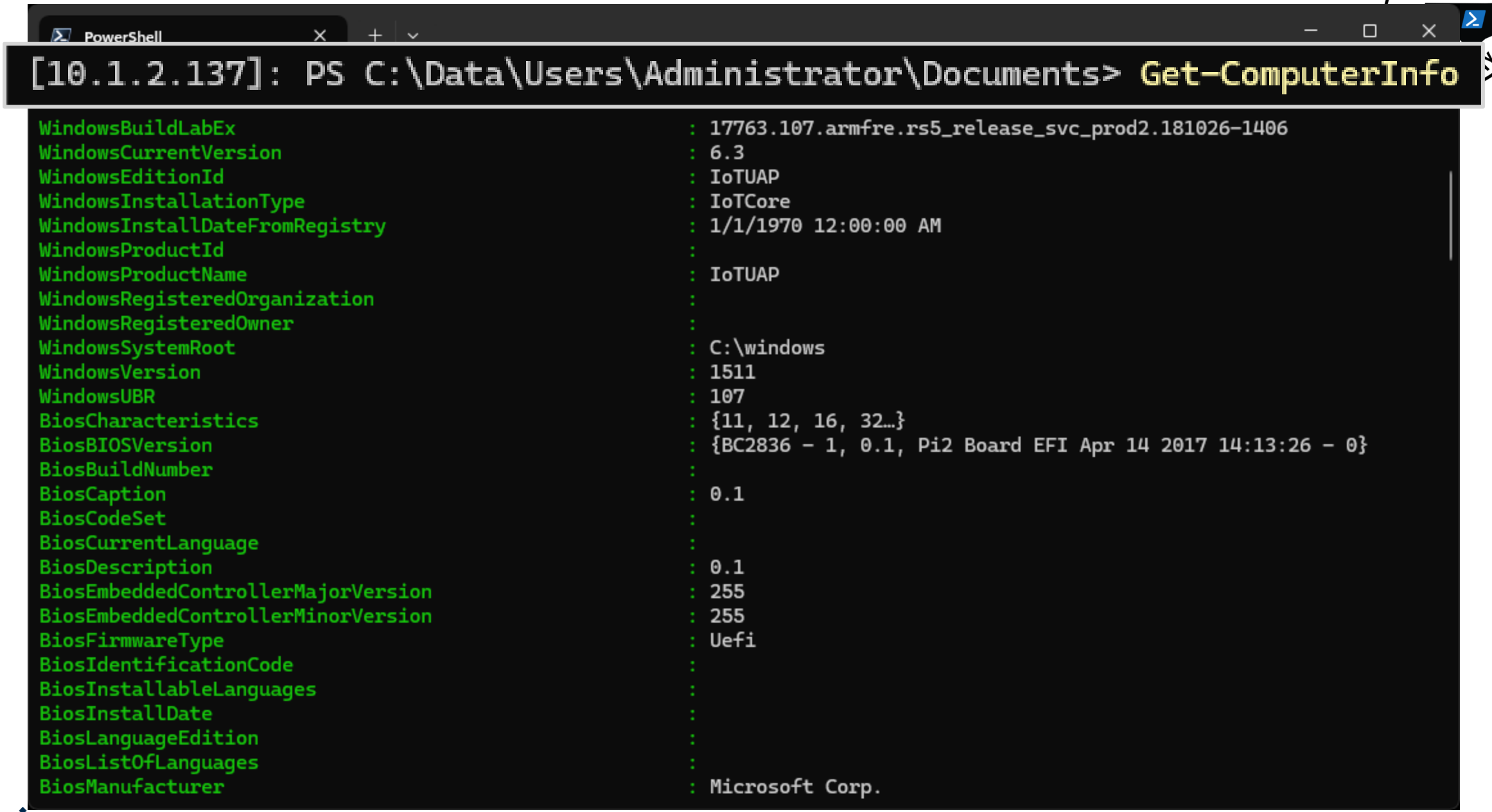

```
> function Get-DownloadFolder { ···
  }
```

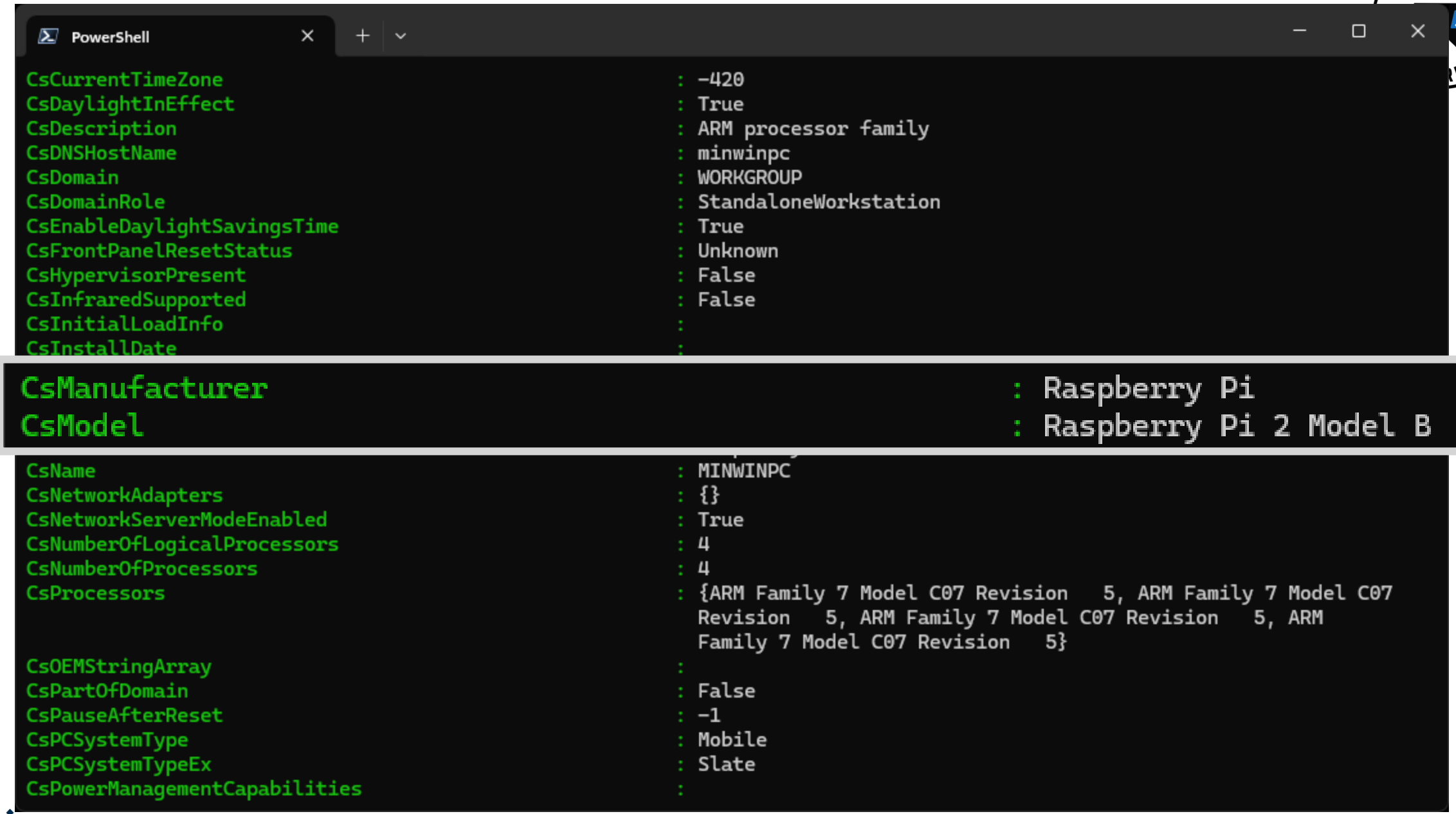












In a roundabout way, we've already hit on "alternative processor architectures" a bit…

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)

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)

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

In a roundabout way, we've already r architectures" a bit…

PowerShell runs on:

- Intel/AMD x86
- Intel/AMD x86-64 (also known as AMD
- Intel Itanium (IA64)
- ARM (32-bit)
- ARM (64-bit) (also known as ARM64)

it was only around
ver 2003-2008 R2.

Shell is v1 or v2 –
Shell v3 or later
itecture

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)

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

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)

TIP: In the repo, we have some PowerShell code for detecting the processor architecture, including on Linux/macOS

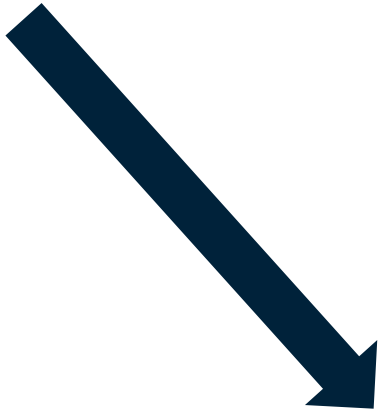
We know that we could encounter a range of PowerShell versions, and we know there are differences across platforms. How do we avoid these pitfalls?

- VSCode
- The PowerShell extension
- …plus some special configuration

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

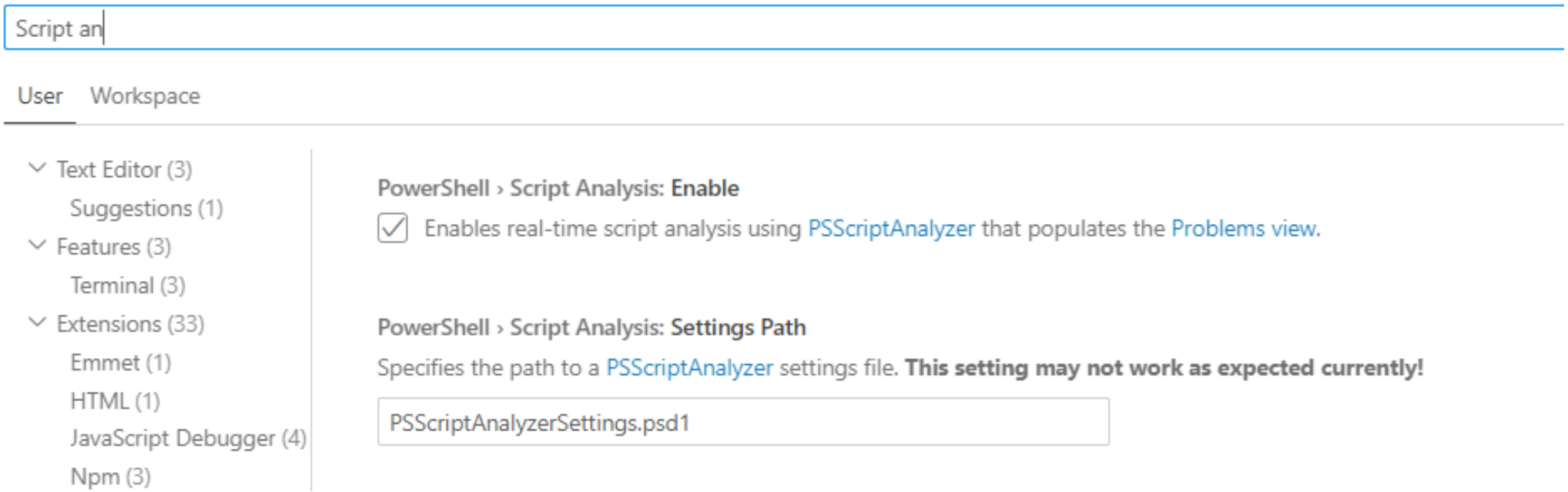


```
PSUseCompatibleCmdlets = @{
    compatibility = @(
        'desktop-2.0-windows',
        'desktop-3.0-windows',
        'desktop-4.0-windows',
        'desktop-5.1.14393.206-windows',
        'core-6.1.0-windows',
        'core-6.1.0-linux',
        'core-6.1.0-linux-arm',
        'core-6.1.0-macos'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

```
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_4.0.30319.42000_framework',
        'win-8_x64_10.0.17763.0_6.2.4_x64_4.0.30319.42000_core',
        '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_framework',
        'win-8_x64_6.3.9600.0_4.0_x64_4.0.30319.42000_framework'
    )
}
```

```
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_4.0.30319.42000_framework',
        'win-8_x64_10.0.17763.0_6.2.4_x64_4.0.30319.42000_core',
        '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_framework',
        'win-8_x64_6.3.9600.0_4.0_x64_4.0.30319.42000_framework'
    )
}
```




```
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: We hope you agree that we've achieved these learning objectives!

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.

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.

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.