PowerShell Set Environment Variable – A Step-By-Step Guide

petri.com/powershell-set-environment-variable

December 5, 2022

Environment variables are essential for accessing command line tools and controlling program execution in operating systems. In Windows, PowerShell provides a convenient way to manage and access environment variables. You can set the environment path, edit existing paths, and handle the user profile on your computer. In this step-by-step guide, we'll explore what environment variables are, their scopes, and how to set environment variables using PowerShell.

What are environment variables in Windows?

Environmental variables often contain critical configuration information for the operating system and its modules, such as current user information, directory paths, or the location of certain core files for the operating system to function. Environment variables store string data and can be passed down to child processes, making them useful for implementing hierarchical processes.

Environment variables can be accessed and managed in multiple ways including Windows Explorer, text editors like Notepad, the command prompt, and PowerShell. PowerShell allows you to manage and access the environment variables in several supported operating systems and lets you access, change, clear, and even delete them when needed.

Different Scopes of Environment Variables

In a Windows operating system, there are three different scopes for the environment variables. These scopes follow a hierarchy (Machine -> User -> Process). And each of these scopes is capable of overwriting the parent if needed.

Machine (System) scope

Machine or System scope contains all the environment variables that are related to the system and are associated with the Windows instances. System variables can be seen and accessed by any user accessing the system. However, you need to have sufficient privileges to be able to change the machine-scoped variables.

User scope

User scope contains the environment variables linked to the user currently running the processes. User variables take priority over the machine or system variables. And with sufficient privileges, they can overwrite the system scoped variables having the same name.

Process scope

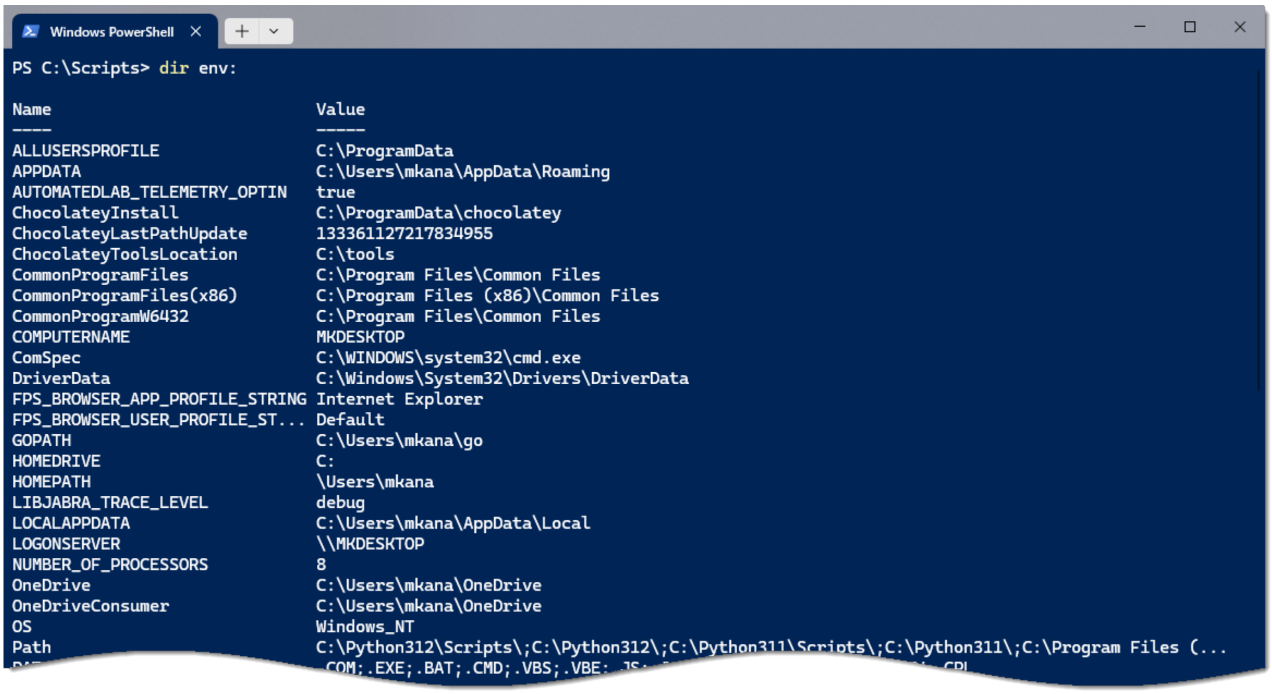
The process scope contains all the environment variables associated with the currently running process or the PowerShell session running. Environment variables under the process scope are a combination of both Machine and User scopes and they are usually inherited from the parent process along with a few Windows-created dynamic variables.

Checking environment variable with PowerShell

All the environment variables in PowerShell are stored in the PS drive and are visible in ‘Env: ‘.

There are multiple ways you can retrieve the environment variables along with their values in your operating system using PowerShell. Some of them are:

```
dir env:
```



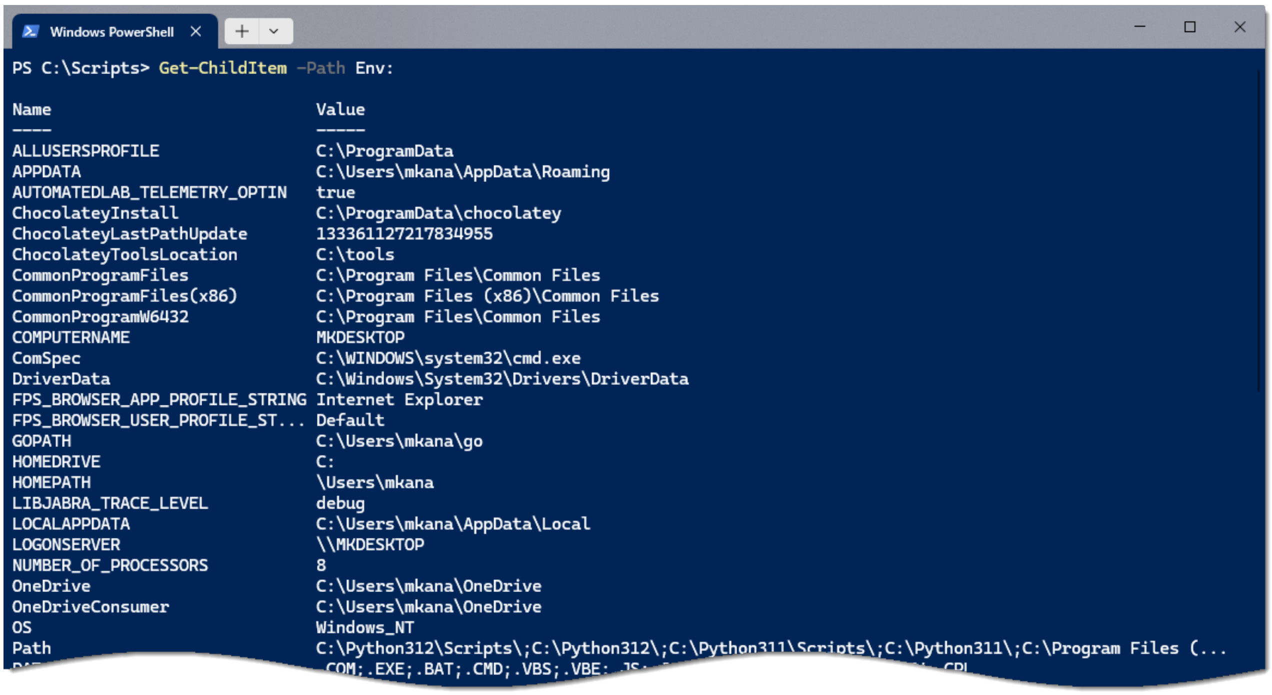
```
PS C:\Scripts> dir env:

Name                           Value
----                           -----
ALLUSERSPROFILE                C:\ProgramData
APPDATA                        C:\Users\mkana\AppData\Roaming
AUTOMATEDLAB_TELEMETRY_OPTIN   true
ChocolateyInstall              C:\ProgramData\chocolatey
ChocolateyLastPathUpdate       133361127217834955
ChocolateyToolsLocation        C:\tools
CommonProgramFiles             C:\Program Files\Common Files
CommonProgramFiles(x86)        C:\Program Files (x86)\Common Files
CommonProgramW6432             C:\Program Files\Common Files
COMPUTERNAME                   MKDESKTOP
ComSpec                        C:\WINDOWS\system32\cmd.exe
DriverData                     C:\Windows\System32\Drivers\DriverData
FPS_BROWSER_APP_PROFILE_STRING Internet Explorer
FPS_BROWSER_USER_PROFILE_ST... Default
GOPATH                         C:\Users\mkana\go
HOMEDRIVE                      C:
HOMEPATH                       \Users\mkana
LIBJABRA_TRACE_LEVEL           debug
LOCALAPPDATA                   C:\Users\mkana\AppData\Local
LOGONSERVER                    \\MKDESKTOP
NUMBER_OF_PROCESSORS           8
OneDrive                       C:\Users\mkana\OneDrive
OneDriveConsumer               C:\Users\mkana\OneDrive
OS                             Windows_NT
Path                           C:\Python312\Scripts\;C:\Python312\;C:\Python311\Scripts\;C:\Python311\;C:\Program Files (...
                                .COM;.EXE;.BAT;.CMD;.VBS;.VBE;
```

Retrieve environment variables using PowerShell (Image Credit: Mike Kanakos/Petri)

or:

```
Get-ChildItem -Path Env:
```



```
PS C:\Scripts> Get-ChildItem -Path Env:

Name                           Value
----                           -----
ALLUSERSPROFILE                C:\ProgramData
APPDATA                        C:\Users\mkana\AppData\Roaming
AUTOMATEDLAB_TELEMETRY_OPTIN   true
ChocolateyInstall              C:\ProgramData\chocolatey
ChocolateyLastPathUpdate       133361127217834955
ChocolateyToolsLocation        C:\tools
CommonProgramFiles             C:\Program Files\Common Files
CommonProgramFiles(x86)        C:\Program Files (x86)\Common Files
CommonProgramW6432             C:\Program Files\Common Files
COMPUTERNAME                   MKDESKTOP
ComSpec                        C:\WINDOWS\system32\cmd.exe
DriverData                     C:\Windows\System32\Drivers\DriverData
FPS_BROWSER_APP_PROFILE_STRING Internet Explorer
FPS_BROWSER_USER_PROFILE_ST... Default
GOPATH                         C:\Users\mkana\go
HOMEDRIVE                      C:
HOMEPATH                       \Users\mkana
LIBJABRA_TRACE_LEVEL           debug
LOCALAPPDATA                   C:\Users\mkana\AppData\Local
LOGONSERVER                    \\MKDESKTOP
NUMBER_OF_PROCESSORS           8
OneDrive                       C:\Users\mkana\OneDrive
OneDriveConsumer               C:\Users\mkana\OneDrive
OS                             Windows_NT
Path                           C:\Python312\Scripts\;C:\Python312\;C:\Python311\Scripts\;C:\Python311\;C:\Program Files (...
                               .COM;.EXE;.BAT;.CMD;.VBS;.VBE;
```

Retrieve environment variables using PowerShell using Get-ChildItem (Image Credit: Mike Kanakos/Petri)

You can also retrieve a specific environment variable by appending the variable name after the aforementioned commands.

Example: To retrieve the HomeDrive (an example var), you can use the command below:

```
Get-ChildItem -Path Env:\HomeDrive
```

Setting Environment Variables with PowerShell

There are a few different ways that you can follow to set the environment variables using PowerShell.

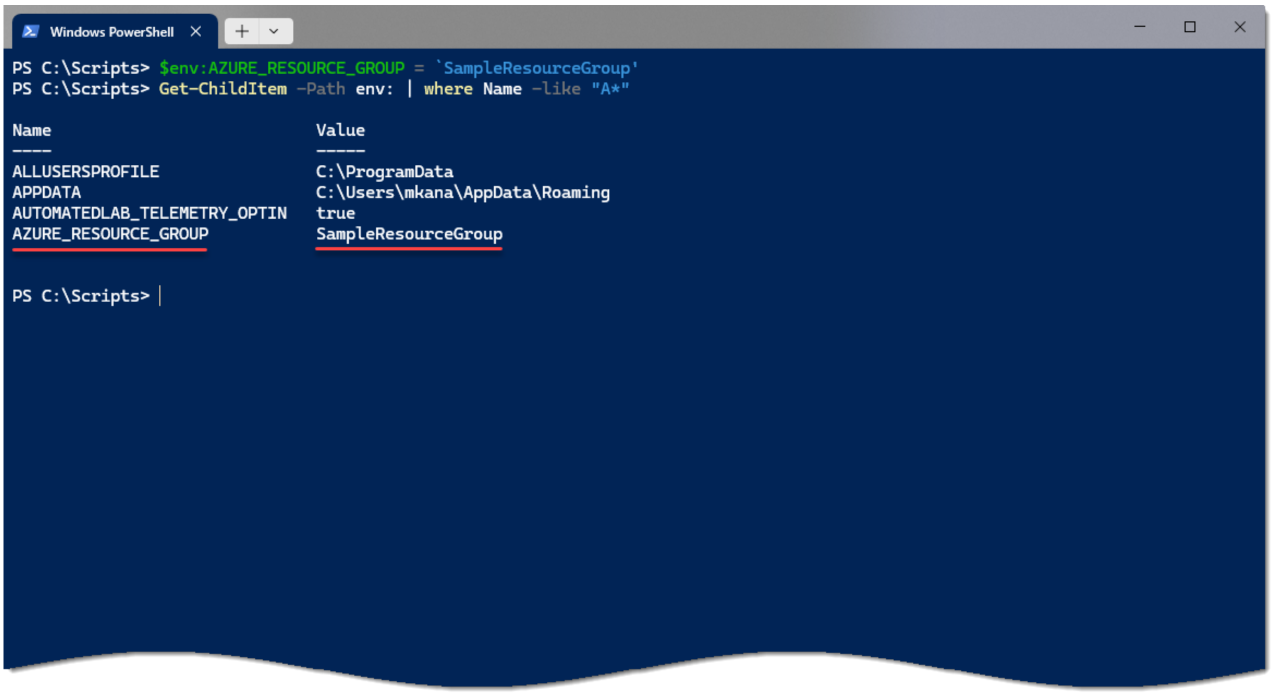
Adding/appending Environment Variables

One of the easiest ways to add or append an environment variable using PowerShell is to use $Env to set an **environment variable** using the assignment operator (=) and to append or create a new environment variable using the (+=) operator.

For example, if you wish to create the AZURE_RESOURCE_GROUP environment variable in your system that doesn't exist by default, you can do it using the below command:

```
$env:AZURE_RESOURCE_GROUP = 'SampleResourceGroup'
```

Note: If the AZURE_RESOURCE_GROUP already existed in your system, it will be replaced with the value you pass in the above command.



```
PS C:\Scripts> $env:AZURE_RESOURCE_GROUP = `SampleResourceGroup'
PS C:\Scripts> Get-ChildItem -Path env: | where Name -like "A*"

Name                           Value
----                           -----
ALLUSERSPROFILE                C:\ProgramData
APPDATA                        C:\Users\mkana\AppData\Roaming
AUTOMATEDLAB_TELEMETRY_OPTIN   true
AZURE_RESOURCE_GROUP           SampleResourceGroup


PS C:\Scripts> |
```

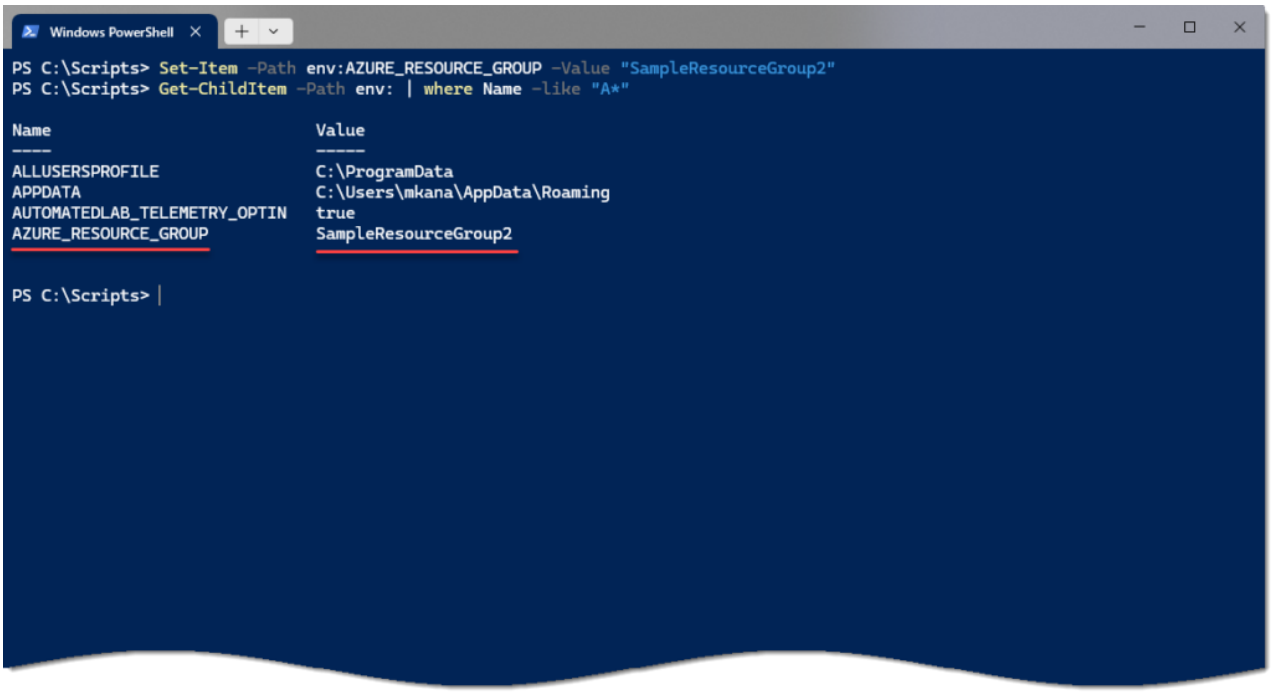
Create an environment variable (Image Credit: Mike Kanakos/Petri)

Using the Set-Item cmdlet

The Set-Item and Get-Item cmdlets are used to change and retrieve the value of the environment variables or the registry values or keys. You can also use the Set-Item cmdlet to set or create an environment variable.

The below cmdlet will set the environment variable AZURE_RESOURCE_GROUP to ‘SampleResourceGroup2’

```
Set-Item -Path env:AZURE_RESOURCE_GROUP -Value “SampleResourceGroup2"
```



```
PS C:\Scripts> Set-Item -Path env:AZURE_RESOURCE_GROUP -Value "SampleResourceGroup2"
PS C:\Scripts> Get-ChildItem -Path env: | where Name -like "A*"

Name                           Value
----                           -----
ALLUSERSPROFILE                C:\ProgramData
APPDATA                        C:\Users\mkana\AppData\Roaming
AUTOMATEDLAB_TELEMETRY_OPTIN   true
AZURE_RESOURCE_GROUP           SampleResourceGroup2


PS C:\Scripts> |
```

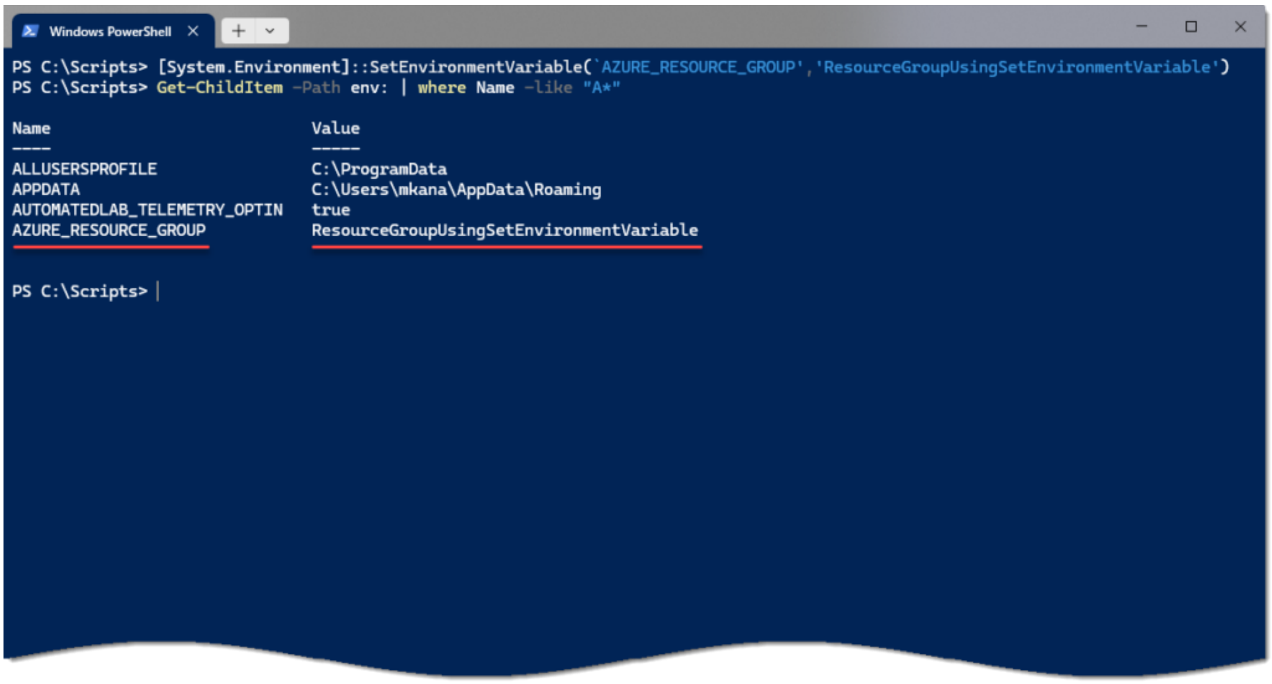
PowerShell set environment variable using Set-Item (Image Credit: Mike Kanakos/Petri)

Using the [System.Environment] .NET class method

Microsoft's .NET framework class library is a powerful means to use and execute PowerShell scripts. The .NET class [System.Environment] offers methods to set and get the environment variables. To set the environment variables using the .NET class method, you need to use the SetEnvironmentVariable() method for a given scope or to create a new one if it doesn't already exist.

For example, you can use SetEnvironmentVariable() to set the AZURE_RESOURCE_GROUP environment variable using the below commands.

```
[System.Environment]::SetEnvironmentVariable(‘AZURE_RESOURCE_GROUP’,'ResourceGroup
UsingSetEnvironmentVariable')
```



```
PS C:\Scripts> [System.Environment]::SetEnvironmentVariable(`AZURE_RESOURCE_GROUP','ResourceGroupUsingSetEnvironmentVariable')
PS C:\Scripts> Get-ChildItem -Path env: | where Name -like "A*"

Name                           Value
----                           -----
ALLUSERSPROFILE                C:\ProgramData
APPDATA                        C:\Users\mkana\AppData\Roaming
AUTOMATEDLAB_TELEMETRY_OPTIN   true
AZURE_RESOURCE_GROUP           ResourceGroupUsingSetEnvironmentVariable


PS C:\Scripts>
```

Use .NET class method to set an environment variable (Image Credit: Mike Kanakos/Petri)

Are environment variable changes permanent?

The default scope for environment variables created by PowerShell is the process scope. This means that the environment variables are accessible only within the current PowerShell session or process. They are not available to other processes or sessions.

When you close your PowerShell window / prompt, the environment variables created in that session will be removed. You can make environment variables persistent by creating them in either the machine or user scopes. The aformented scopes both persist outside of the current process, allowing you to save a new or changed environment variable.

To change values in the Machine or User scopes, you must use the methods of the System.Environment class. To create environment variables in the machine scope, you can use the `SetEnvironmentVariable()` method of the .NET class. Here's an example:

```
[Environment]::SetEnvironmentVariable("MACHINE_VARIABLE", "value", "somevalue")
```

This will create a machine-wide environment variable named “MACHINE_VARIABLE” with the value “somevalue”. To create environment variables in the user scope, you can use the $Env variable. Here’s an example:

```
$Env:USER_VARIABLE = "someothervalue"
```

This will create a user-specific environment variable named “USER_VARIABLE” with the value “someothervalue”.

How to remove an environment variable with PowerShell

There are multiple techniques to remove or delete an environment variable in PowerShell. One way is by utilizing the $Env variable or the SetEnvironmentVariable() method from the .NET class.

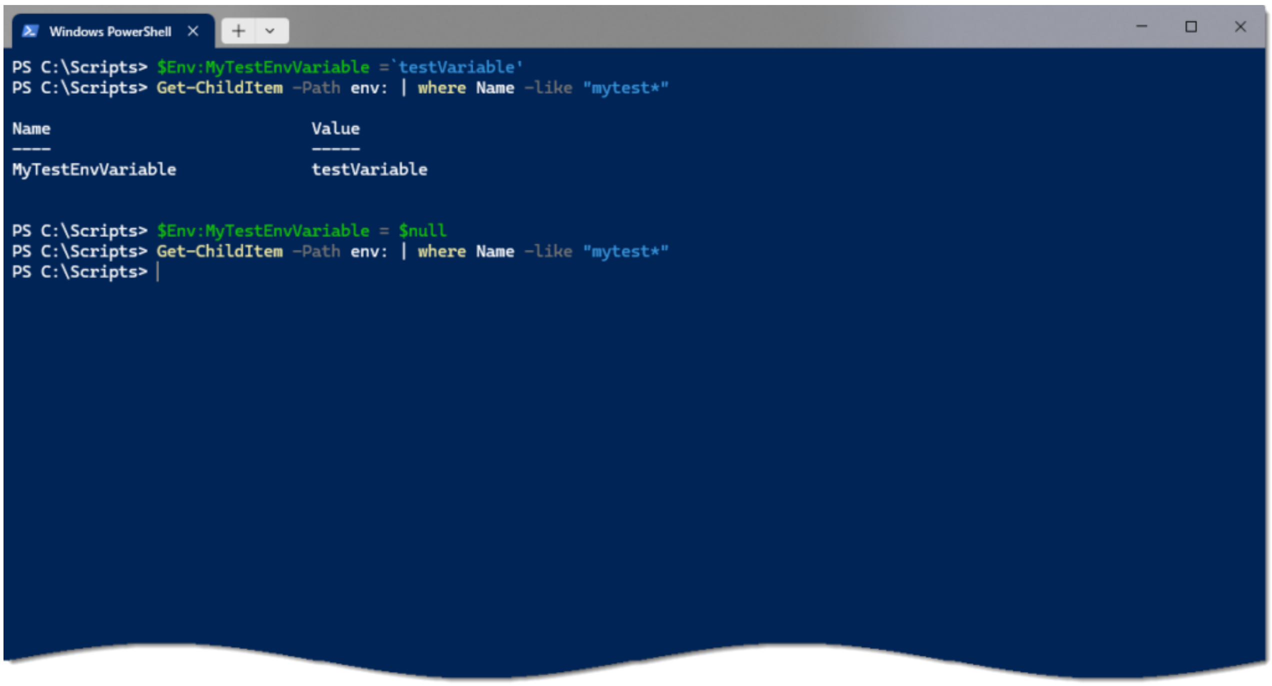
Let’s consider an example where we create a test environment variable named ‘MyTestEnvVariable’ using the following command:

```
$Env:MyTestEnvVariable = 'testVariable'
```

To remove this system-wide environment variable, you can employ the following command that utilizes the SetEnvironmentVariable() method:

```
[Environment]::SetEnvironmentVariable("MyTestVariable", $null, "Machine")
```

Alternatively, you can also use the $Env variable to clear and delete the environment variable by executing the following command:$Env:MyTestEnvVariable = $null



```
PS C:\Scripts> $Env:MyTestEnvVariable = 'testVariable'
PS C:\Scripts> Get-ChildItem -Path env: | where Name -like "mytest*"

Name                           Value
----                           -----
MyTestEnvVariable              testVariable


PS C:\Scripts> $Env:MyTestEnvVariable = $null
PS C:\Scripts> Get-ChildItem -Path env: | where Name -like "mytest*"
PS C:\Scripts> |
```

Remove environment variable using PowerShell (Image Credit: Mike Kanakos/Petri)

Conclusion

This guide has given you a comprehensive understanding of how to set environment variables using PowerShell. You've learned why environment variables are important for accessing command line tools and controlling program execution. You've also explored the different scopes of environment variables in Windows and discovered various methods for managing them with PowerShell.

By using $Env, Set-Item cmdlet, and the [System.Environment] .NET class method, you can easily set and manage environment variables. Remember, changes made to variables in different scopes have different permanence levels. And when you need to remove or delete variables, PowerShell offers simple techniques. Overall, this guide equips you with the knowledge and skills to effectively manage environment variables in a casual and straightforward manner.

Table of contents

Sort by

18 April, 2023

In the instructions on how to set a variabelle you do not state if this would be user or process scope? Is it possible to set a USER scope variabele that persists using Powershell?