

# How to Check PowerShell Version

 lazyadmin.nl/powershell/get-powershell-version

September 21, 2021

The PowerShell version determines which features you can use. So before you can run a specific script or cmdlet, you might need to check which PowerShell version you are using.

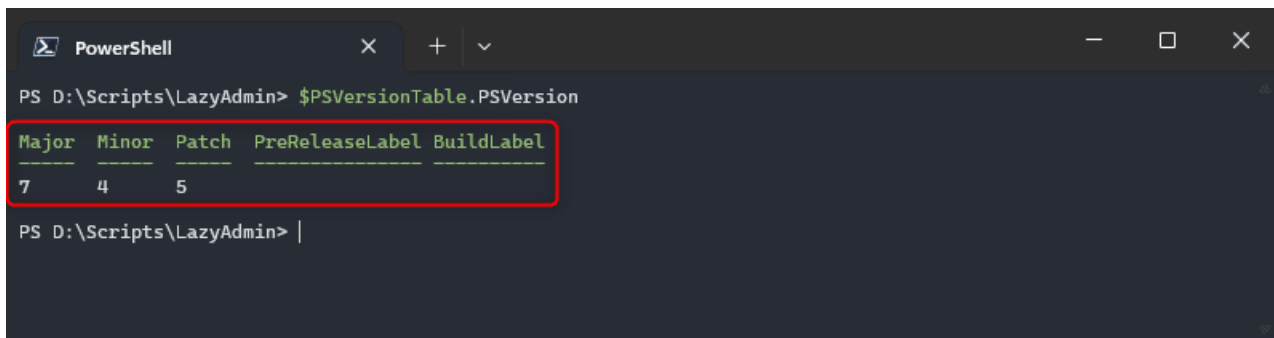
You can check the PowerShell version with a single command. If you are using PowerShell 7 then it's even displayed automatically when you open a new PowerShell session.

In this article, I will explain how you can quickly check the PowerShell version, and how to use this method inside your scripts or on a remote machine.

## Check PowerShell Version

To check which version of PowerShell you are using, you can use a single command in PowerShell to find the exact version number:

```
$PSVersionTable.PSVersion
```

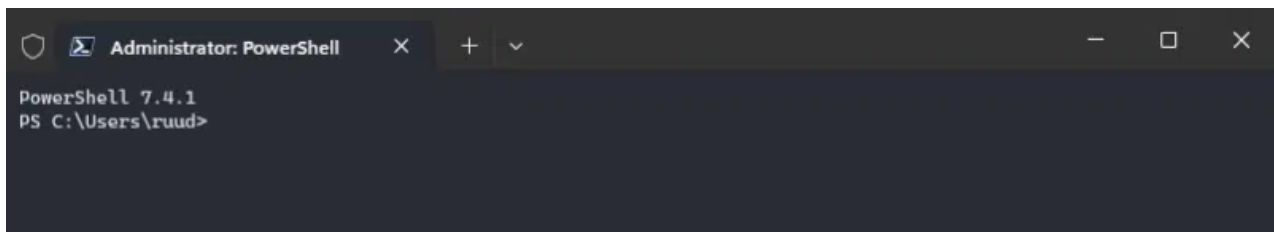


```
PS D:\Scripts\LazyAdmin> $PSVersionTable.PSVersion
```

Major	Minor	Patch	PreReleaseLabel	BuildLabel
7	4	5		

```
PS D:\Scripts\LazyAdmin> |
```

The PSVersion value is the PowerShell version that you are using. In the example above, we are using PowerShell 7.4.5. As mentioned in the beginning, when you open PowerShell 7, you will also see the version number printed on the first line:



```
PowerShell 7.4.1
PS C:\Users\ruud>
```

## Get the PowerShell version in Script

When you are creating a PowerShell script, that uses specific cmdlets that only work in PowerShell 7 or 5.1 for example, then you will need to build in a check for the version.

Instead of getting all the version information, we can get the major version number (5 or 7 for example) by directly selecting the major version number:

```
$PSVersionTable.PSVersion.Major
```

```
# Returns
```

```
7
```

## Using the \$host Variable

Another commonly mentioned option is to use the `$host` variable to check the PowerShell version. The `$host` variable returns information about the host that is running PowerShell. This works great in your local PowerShell console:

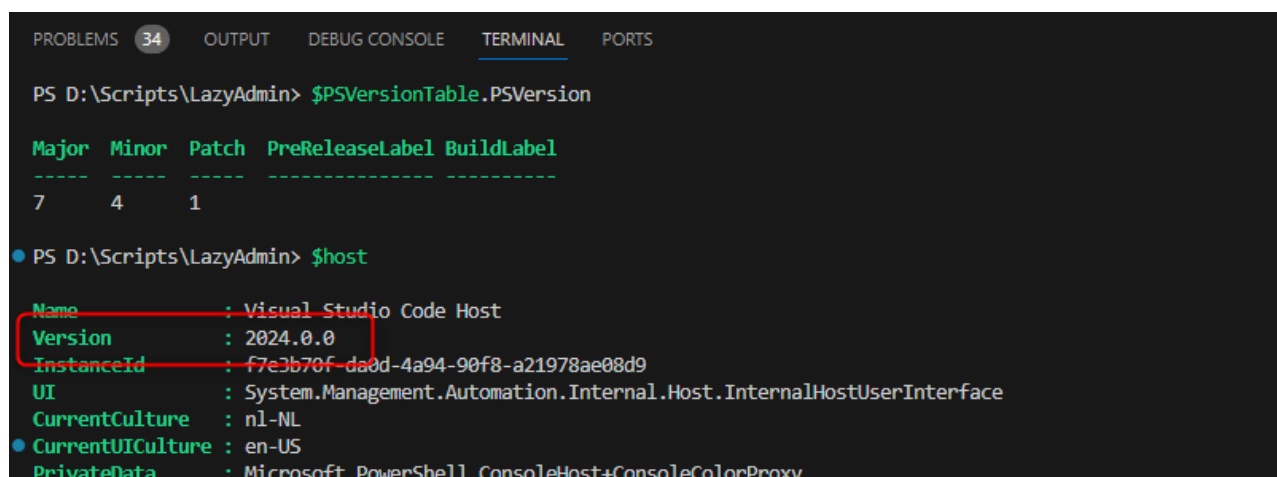
```
$host
```

```
# Or directly the version number
```

```
$host.version
```

But using the `$host` variable or the `Get-Host` cmdlet will cause problems when using it on a remote machine or for example in Visual Studio code. The problem is, that it returns information about the host, and not about the PowerShell engine.

Take the example below, if we look at the `$PSVersionTable` variable in Visual Studio Code, then it returns PowerShell version 7.4.1. But the `$host` variable returns the version of Visual Studio Code instead of the PowerShell engine.



```
PROBLEMS 34 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\Scripts\LazyAdmin> $PSVersionTable.PSVersion

Major Minor Patch PreReleaseLabel BuildLabel
-----
7      4      1

PS D:\Scripts\LazyAdmin> $host

Name       : Visual Studio Code Host
Version    : 2024.0.0
InstanceId  : f7e3b70f-da0d-4a94-90f8-a21978ae08d9
UI          : System.Management.Automation.Internal.Host.InternalHostUserInterface
CurrentCulture : nl-NL
CurrentUICulture : en-US
PrivateData : Microsoft.PowerShell.ConsoleHost+ConsoleColorProxy
```

## Using the Registry

The last option that you can use to check the PowerShell version is to use the registry. Now this may not be the most convenient method, but this option can be useful when you need to inventory multiple computers.

There is only one challenge with the registry method, the version number for PowerShell 7 is stored in a different location compared to the older version.

So to check the version numbers we can use the following two registry paths:

```
# Get the older PowerShell versions
```

```
Get-ItemPropertyValue -Path  
HKLM:\SOFTWARE\Microsoft\PowerShell\3\PowerShellEngine -Name  
'PowerShellVersion'  
# Get the PowerShell 7 Version  
Get-ItemPropertyValue -Path  
HKLM:\SOFTWARE\Microsoft\PowerShellCore\InstalledVersions\31ab5147-9a97-4452-  
8443-d9709f0516e1 -Name 'SemanticVersion'
```

## Get PowerShell Version from Remote Host

---

Before you can run a script on a remote computer you sometimes first need to check which version is installed. Some servers may still have only version 2 or 3 installed, which may limit the cmdlets that you can use on your script.

To get the version from a remote host we are going to use the [invoke-command](#) cmdlet. This allows us to run a script block on a remote computer and return the results to use:

```
Invoke-Command -Computename lazy-lab11 -Scriptblock {$PSVersionTable.PSVersion}
```

## Updating your PowerShell

---

PowerShell is a built-in tool in Windows, which means that it is updated with Windows Updates. In Windows 10 and Windows 11, you get PowerShell 5.1 by default. To update to PowerShell 7 you will need to manually install it.

Read more about installing PowerShell 7 and updating it in [this article](#).

## Wrapping Up

---

Even though the \$host variable is easier to remember, it is recommended to use the \$PSVersionTable instead. This is more reliable than the host variable, especially in scripts or when you need to work with remote computers.

I hope you found this article useful, if you have any questions just drop a comment below.

Did you **Liked** this **Article**?

Get the latest articles like this **in your mailbox**  
or share this article

I hate spam to, so you can unsubscribe at any time.