How to use PowerShell Get-Content to Read a File

lazyadmin.nl/powershell/get-content

February 7, 2023

The Get-Content cmdlet can be used to retrieve the contents from a file in PowerShell. It will retrieve the contents one line at a time, storing them in an array. This allows you to use or process each line in your PowerShell script.

Get-Content comes with several parameters that allow you to select or filter the results you want to retrieve from the file. And when you need to process a lot of large files, then using the `-raw` parameter can speed up the process.

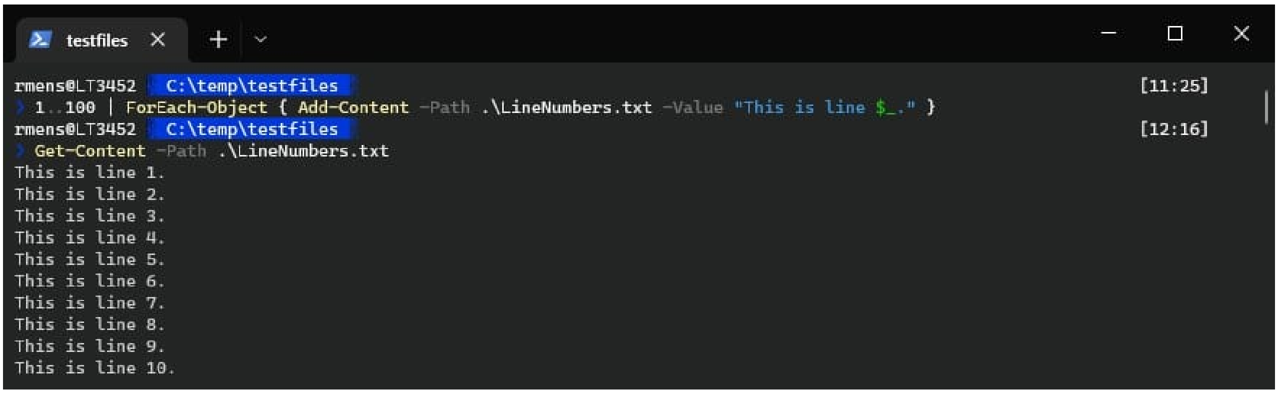
In this article, we are going to take a look at how to use the Get-Content cmdlet, and explain the different parameters that we can use.

Read File with Get-Content

We are going to start with the basics, read a file in PowerShell line-by-line using the `Get-Content` cmdlet. For the examples below I will be using a text file with 100 lines, which are numbered. This makes it easier to follow the examples:

```
1..100 | ForEach-Object { Add-Content -Path .\LineNumbers.txt -Value "This is line $_." }
```

The code above is from Microsoft and creates a text file where each line starts with the string “This is line ” followed by the line number:



```
rmens@LT3452   C:\temp\testfiles                                                  [11:25]
> 1..100 | ForEach-Object { Add-Content -Path .\LineNumbers.txt -Value "This is line $_." }
rmens@LT3452   C:\temp\testfiles                                                  [12:16]
> Get-Content -Path .\LineNumbers.txt
This is line 1.
This is line 2.
This is line 3.
This is line 4.
This is line 5.
This is line 6.
This is line 7.
This is line 8.
This is line 9.
This is line 10.
```

To view the results of the newly created file, we can use the `Get-Content` cmdlet. You only need to specify the path to the file to start reading its contents of it:

```
Get-Content -Path .\LineNumbers.txt
```

If we store the result of Get-Content in a variable, then we can see that PowerShell stores the results in an array. This means that we can use any Array function on the results, or process each line using a ForEach loop:

```
$contents = Get-Content -Path .\LineNumbers.txt
$contents.GetType()
# Result
```

IsPublic IsSerial Name BaseType

-------- -------- ---- --------

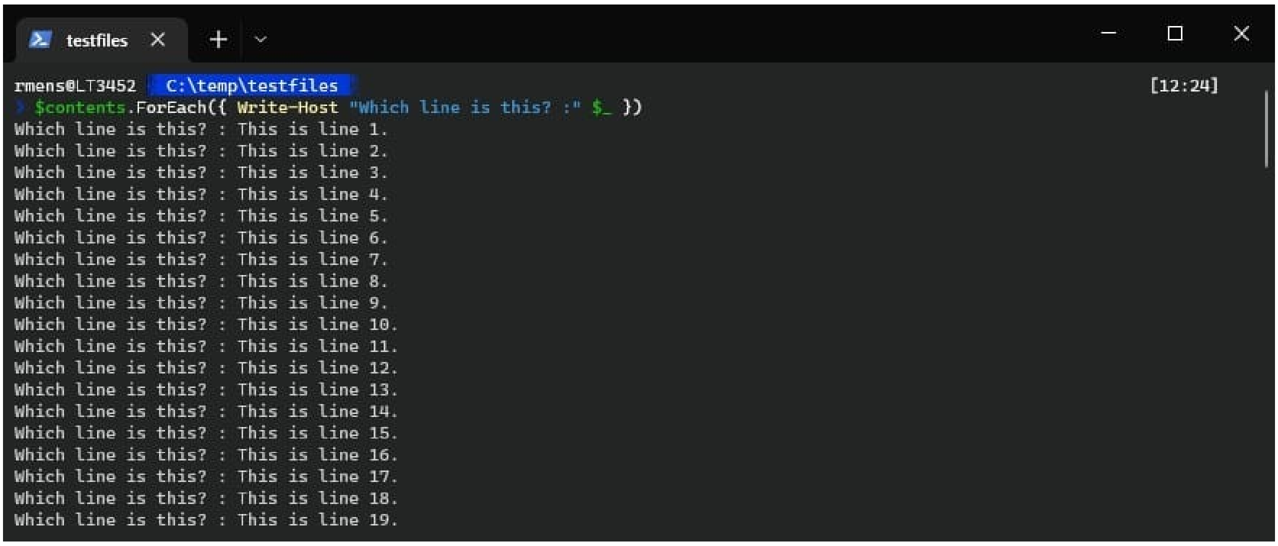
True True Object[] System.Array

# Return the 6th line (the index of an array start at 0 😉)

$contents[5]

# Result

This is line 6



```
rmens@LT3452   C:\temp\testfiles                                                    [12:24]
> $contents.ForEach({ Write-Host "Which line is this? :" $_ })
Which line is this? : This is line 1.
Which line is this? : This is line 2.
Which line is this? : This is line 3.
Which line is this? : This is line 4.
Which line is this? : This is line 5.
Which line is this? : This is line 6.
Which line is this? : This is line 7.
Which line is this? : This is line 8.
Which line is this? : This is line 9.
Which line is this? : This is line 10.
Which line is this? : This is line 11.
Which line is this? : This is line 12.
Which line is this? : This is line 13.
Which line is this? : This is line 14.
Which line is this? : This is line 15.
Which line is this? : This is line 16.
Which line is this? : This is line 17.
Which line is this? : This is line 18.
Which line is this? : This is line 19.
```

Using a ForEach loop on the Get-Contents result

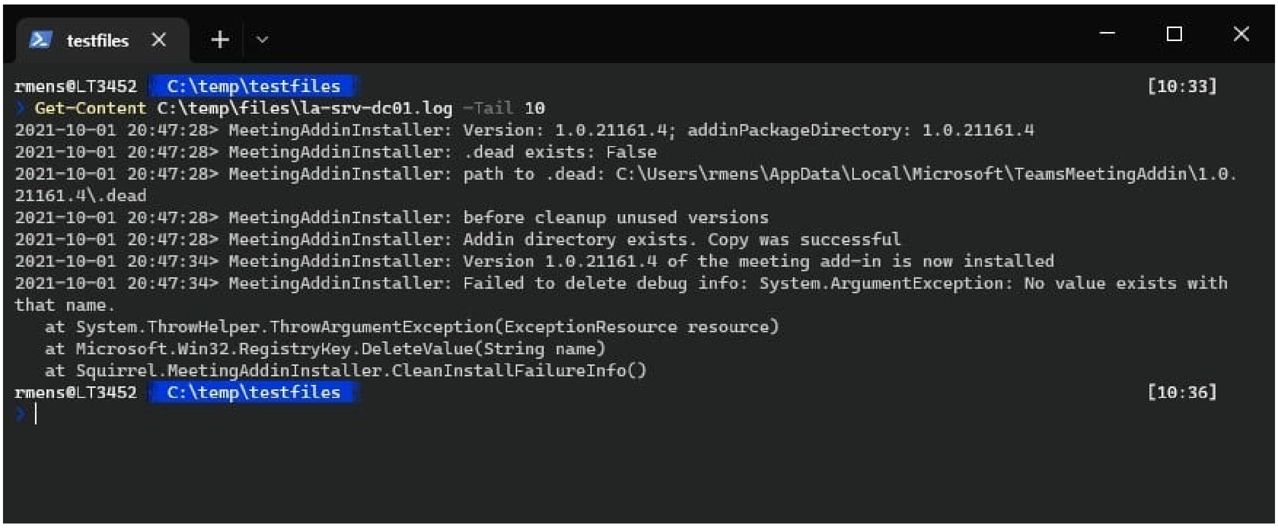
Returning the First or Last x Lines

By default, the Get-Content cmdlet will return all the contents of a file in PowerShell. But in some cases, you only want to view the first or last x lines from a file. For example, when debugging an issue, you often only need the last couple of lines from a log file.

For this we have two options, we can use the parameter `-TotalCount`, which returns the first x items of the content. Or we can use the parameter `-Tail`, which returns the last x items of a file.

So let’s go back to our log file example. We want to view the last 10 lines from the log. To do this, we need to specify the path and add `-Tail` 10 to return only the last 10 lines:

Get-Content C:\temp\files\la-srv-dc01.log -Tail 10

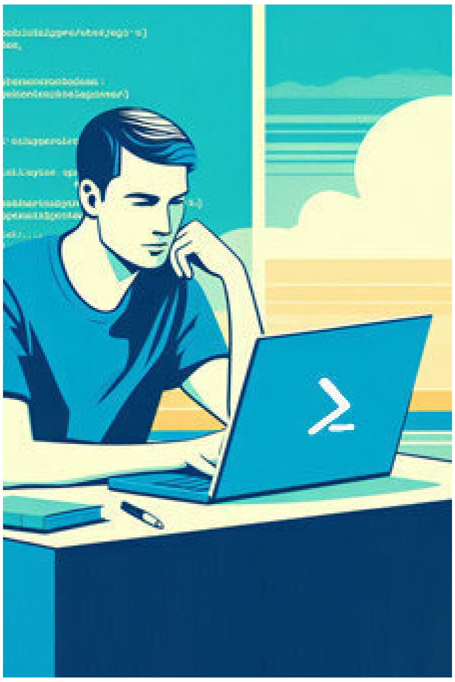


```
rmens@LT3452   C:\temp\testfiles                                                                          [10:33]
❯ Get-Content C:\temp\files\la-srv-dc01.log -Tail 10
2021-10-01 20:47:28> MeetingAddinInstaller: Version: 1.0.21161.4; addinPackageDirectory: 1.0.21161.4
2021-10-01 20:47:28> MeetingAddinInstaller: .dead exists: False
2021-10-01 20:47:28> MeetingAddinInstaller: path to .dead: C:\Users\rmens\AppData\Local\Microsoft\TeamsMeetingAddin\1.0.
21161.4\.dead
2021-10-01 20:47:28> MeetingAddinInstaller: before cleanup unused versions
2021-10-01 20:47:28> MeetingAddinInstaller: Addin directory exists. Copy was successful
2021-10-01 20:47:34> MeetingAddinInstaller: Version 1.0.21161.4 of the meeting add-in is now installed
2021-10-01 20:47:34> MeetingAddinInstaller: Failed to delete debug info: System.ArgumentException: No value exists with
that name.
   at System.ThrowHelper.ThrowArgumentException(ExceptionResource resource)
   at Microsoft.Win32.RegistryKey.DeleteValue(String name)
   at Squirrel.MeetingAddinInstaller.CleanInstallFailureInfo()
rmens@LT3452   C:\temp\testfiles                                                                          [10:36]
❯ |
```

Get-Content return the last x lines

To read the first x item from a file we will need to use the parameter `-TotalCount`:

Get-Content -Path .\LineNumbers.txt -TotalCount 5



Return a single line from a file

If you know which specific line you need from a text file, then it’s possible to select only a single line from the text file. As we know, the results are stored in an array. So we can use the array index to select a line number that we need.

Keep in mind that the array index starts at 0, so if we want to return line 16, we need to specify index number 15. We wrap the `Get-Content` command in parentheses `( )` so that PowerShell first completes the Get-Content command before it continues.

```
(Get-Content -Path .\LineNumbers.txt)[15]
# Result
This is line 16.
```

If you have a large text file then you might not want to read it completely to only return a single line. What you can do is select only the first or last x lines from the file, and then select the line number using the array index.

For example, let’s say we need the fifth line from a file. We set the `TotalCount` to 5, reading only the first 5 lines, and return the last item from the array:

```
(Get-Content -Path .\LineNumbers.txt -TotalCount 5)[-1]
# Result
This is line 5.
```

Reading Multiple Files with Get-Content

It’s also possible to read the contents of multiple files at once with Get-Content in PowerShell. To do this you only need to specify the path where the files are located followed by an asterisk `*`.

```
Get-Content -Path c:\temp\files\*
```

Keep in mind though that there won’t be any reference in the results to the source files. So you won’t know from which file line x comes.

We can also filter which files we want to read. So for example, if you only want to get the contents of all the .log files then you can add the parameter `-Filter` to it:

```
Get-Content -Path c:\temp\files\* -Filter '*.log'
```

Using -Raw

The Get-Content cmdlet reads each line of the file into the array, which is great when you need to process each line independently or you are looking for a single line. But array a terrible if you need to do a search and replace on the contents of a file.

For example, when you have an HTML file that you are using as a template for an email that you are about the send with PowerShell. In the HTML file, we have placeholders that we need to replace, like this:

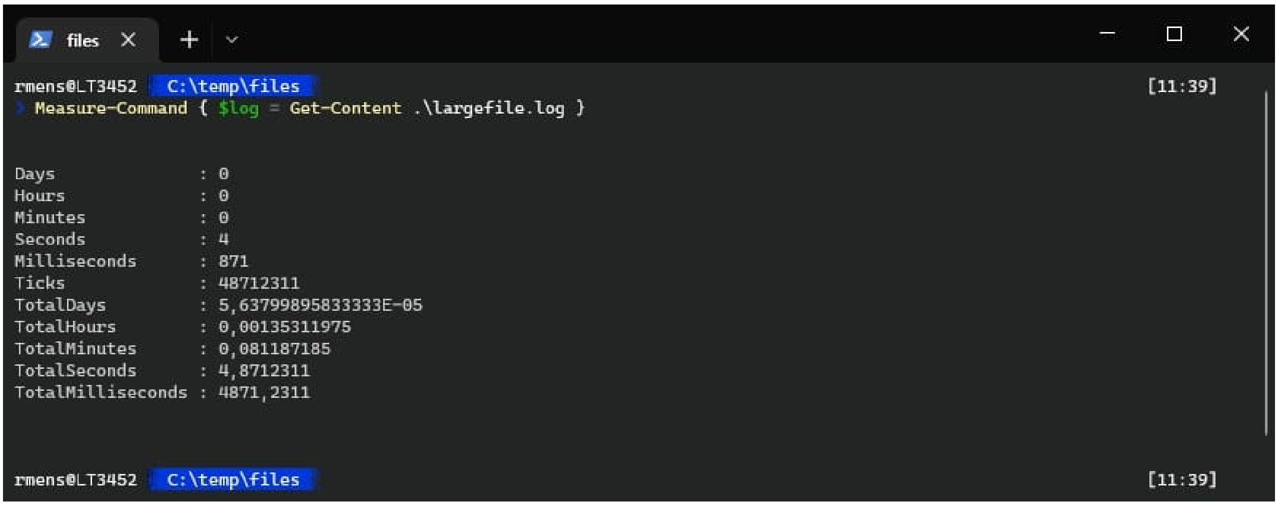
<p>Hi {{manager.firstname}},</p>
<p>
The account for {{user.fullname}} is ready. Below you will find the login details for {{user.firstname}}. We have set a temporary password which must be
changed after the first login.
</p>
<p>
In the attachment you will find a starting guide for new users
</p>

Instead of looping through each line to find the placeholders and replace them, we can use the `-Raw` parameter to read the whole file as a single string and use the -Replace operator to replace the placeholders:

(Get-Content -Path c:\temp\template.html -Raw) -replace '{{user.fullname}}', $user.fullName -replace '{{user.firstname}}', $user.givenName

Another advantage of using the -Raw parameter is that it's a lot faster. Get-Content works great with small text files, but when you need to process large text files then you will probably notice that the command is a bit sluggish.

For example, a text file of 50Mb takes almost 5 seconds to read as you can see in the screenshot below:



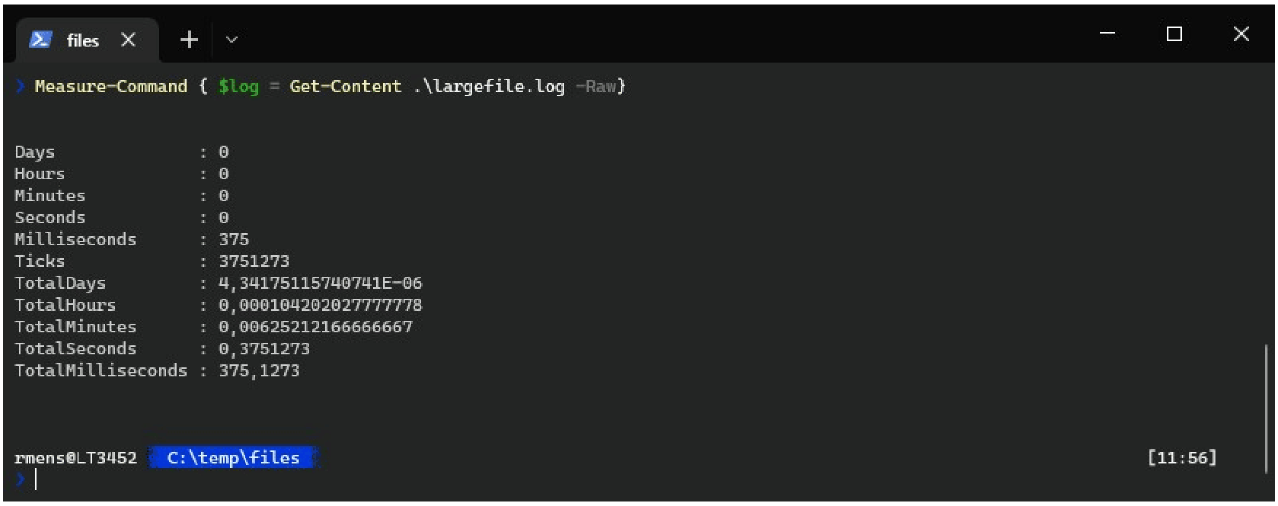
```
rmens@LT3452  C:\temp\files                                          [11:39]
› Measure-Command { $log = Get-Content .\largefile.log }


Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 4
Milliseconds      : 871
Ticks             : 48712311
TotalDays         : 5,63799895833333E-05
TotalHours        : 0,00135311975
TotalMinutes      : 0,081187185
TotalSeconds      : 4,8712311
TotalMilliseconds : 4871,2311



rmens@LT3452  C:\temp\files                                          [11:39]
```

But when using -Raw it took less than half a second:



```
> Measure-Command { $log = Get-Content .\largefile.log -Raw}


Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 0
Milliseconds      : 375
Ticks             : 3751273
TotalDays         : 4,34175115740741E-06
TotalHours        : 0,000104202027777778
TotalMinutes      : 0,00625212166666667
TotalSeconds      : 0,3751273
TotalMilliseconds : 375,1273



rmens@LT3452  C:\temp\files                                          [11:56]
> |
```

If you still need the results in an array then we can use `-Raw` combined with the `-split` string operator to create an array of each line of the file. This method is still 5 times faster than using Get-Content without `-Raw`.

$log = (Get-Content .\largefile.log -Raw) -split "`r`n"

Reading files faster with ReadAllLines

When processing very large files, 100Mb and large, then there is an even faster method to read the contents of the file with PowerShell. The .Net method `System.IO.File.ReadAllLines()`. will read all the lines into the memory in half of the time compared to using the -Raw method.

The results are, just like with Get-Content, stored in an array. So you can process that data in the same way.

$log = [System.IO.File]::ReadAllLines('c:\temp\files\largefile.log')
You can also use the method `[System.IO.File]::ReadLines`, which is even faster. But this will immediately stream the contents to the console, instead of storing it in the memory. The problem with this method is that you will need to close the file when done, otherwise, it will be locked by the stream.

Watch files with -Wait

The last interesting function of Get-Contents is that we can monitor or watch the contents of a file using the `-Wait` parameter. If you, for example, want to monitor a log file. Then instead of rerunning the Get-Content cmdlet every time, you can add the `-wait` parameter.

As soon as the contents of the file are updated (and saved), it will update the output of Get-Content.

Get-Content -Path c:\temp\testfiles\LineNumbers.txt -Tail 5 -Wait
# Result
This is line 96.

```
This is line 97.
This is line 98.
This is line 99.
This is line 100.
```

If we add new lines to the end of the file, then these will show up in the results.

Wrapping Up

The Get-Content cmdlet in PowerShell is great when you need to read a file quickly. But when working with large files you really need to use the -Raw parameter at least to speed up the process. Also, give the .Net method a try, you will be amazed at how fast it is.

I hope you found this article useful, if you have any questions, then 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.