

Get-Date – How to get and use Dates in PowerShell

 lazyadmin.nl/powershell/get-date

December 7, 2022

With Get-Date we can get the current date and time which is often used in PowerShell scripts to timestamp the output or in combination with logging. But besides timestamping, we can also use dates to compare or calculate the age of files, or in filters to select only a specific range from a result set.

Working with dates in general is also always a bit challenging. How do you compare dates, or calculate the days between two dates? And how do you get the date from x days ago?

In this article we are going to take a look at how to get dates with the Get-Date cmdlet in PowerShell, the different formatting options, calculating with dates, and comparing them.

PowerShell Get-Date

The PowerShell Get-Date cmdlet returns, by default, the current date and time from your system. Without any parameters, it will just output the date and time in long formats:

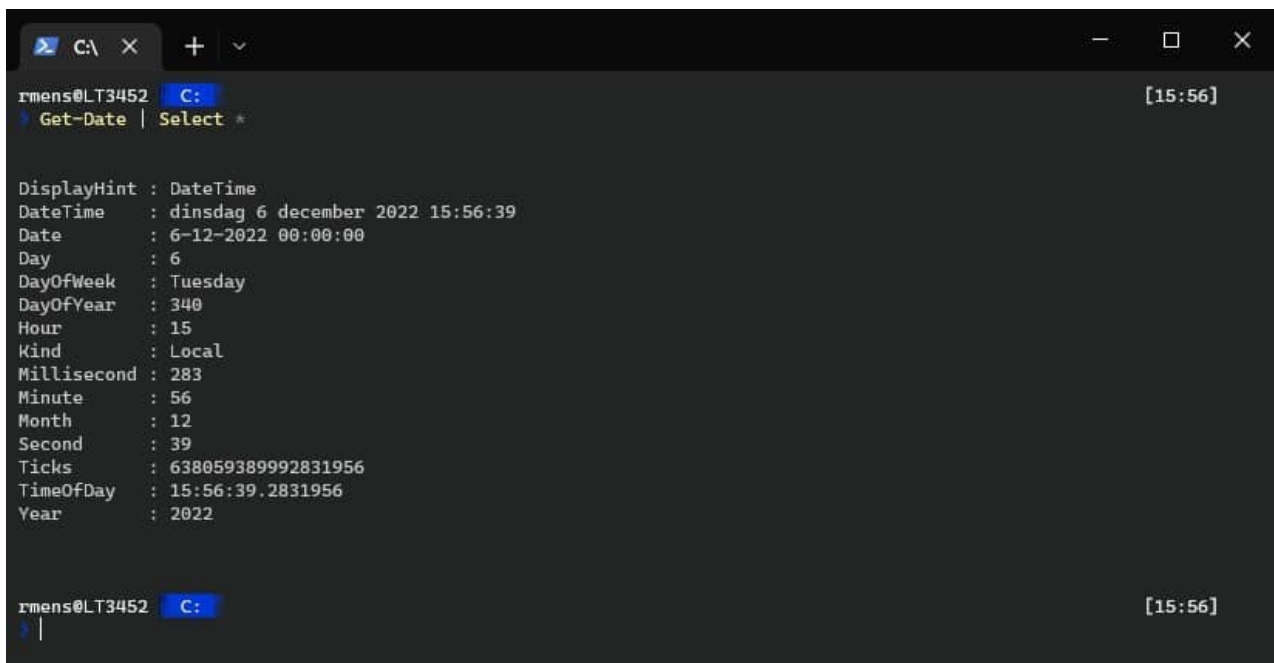
Get-Date

Result

dinsdag 6 december 2022 15:51:54

What most don't know is that the Get-Date cmdlet actually returns more than only the current date and time in a long format. If store the result into a variable or select all properties of the object, you will see that it also returns the day of the year for example:

Get-Date | Select *



```
rmens@LT3452 C:\> Get-Date | Select *

DisplayHint : DateTime
DateTime    : dinsdag 6 december 2022 15:56:39
Date        : 6-12-2022 00:00:00
Day         : 6
DayOfWeek   : Tuesday
DayOfYear   : 340
Hour        : 15
Kind        : Local
Millisecond  : 283
Minute      : 56
Month       : 12
Second      : 39
Ticks       : 638059389992831956
TimeOfDay   : 15:56:39.2831956
Year        : 2022
```

This means that if you only need the current day of the week, you can simply do:

```
Get-Date | Select DayOfWeek
```

```
# Result
```

```
DayOfWeek
```

```
-----
```

```
Tuesday
```

```
# Or to get only the value
```

```
Get-Date | Select -ExpandProperty DayOfWeek
```

```
# Result
```

```
Tuesday
```

If you only want to return the current date or current time, then we can use the parameter `DisplayHint`. This determines which element is displayed:

```
Get-Date -DisplayHint date
```

```
# Result
```

```
dinsdag 6 december 2022
```

```
# Or time
```

```
Get-Date -DisplayHint time
```

```
15:51:54
```

We can also get the date information from dates in the future or past. With the parameters `-day`, `-month`, `-year`, we can specify the data object that we want to create. So for example, if you want to know on what day 6 Dec is in 2023, we change the date:

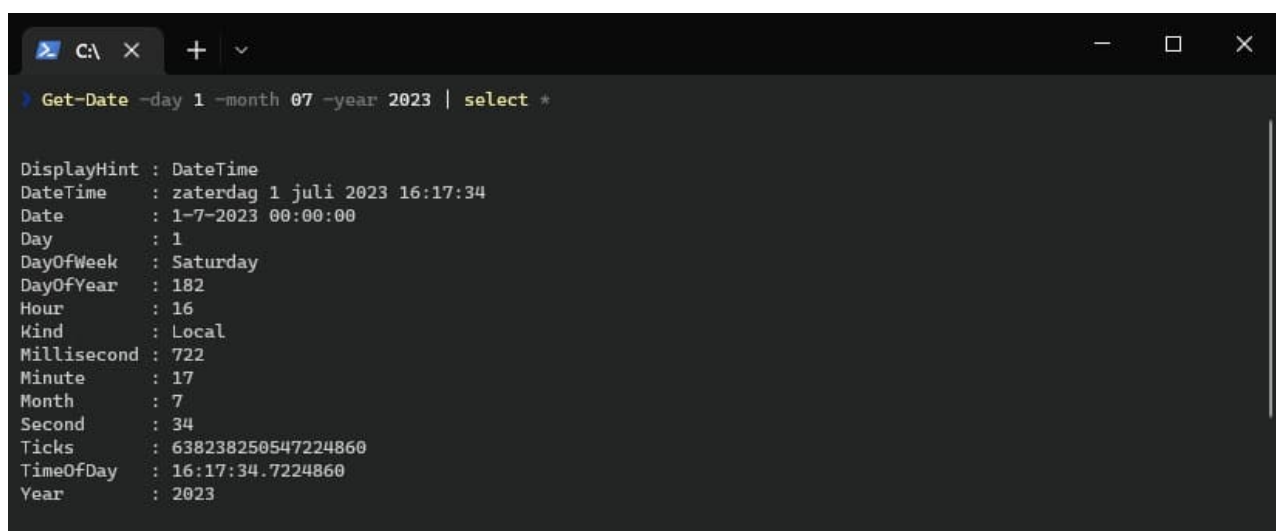
```
Get-Date -year 2023
```

```
# Result (woensdag = wednesday 🗓️ )
```

```
woensdag 6 december 2023 16:11:58
```

As you can see, I have only specified the year. So for the other values, the current date is used. But you can also specify the complete date:

```
Get-Date -day 1 -month 07 -year 2023 | select *
```



```
Get-Date -day 1 -month 07 -year 2023 | select *

DisplayHint : DateTime
DateTime    : zaterdag 1 juli 2023 16:17:34
Date        : 1-7-2023 00:00:00
Day         : 1
DayOfWeek   : Saturday
DayOfYear   : 182
Hour        : 16
Kind        : Local
Millisecond : 722
Minute      : 17
Month       : 7
Second      : 34
Ticks       : 638238250547224860
TimeOfDay   : 16:17:34.7224860
Year        : 2023
```

Format Dates

To format dates in PowerShell we can use Get-Date with either the **.Net format** or the **UFormat** specifier. The .Net format specifier uses the more commonly known letters to specify the format, whereas UFormat uses % followed by a letter. In both cases, the output is a string object and not a date object. So you only get the date formatted as requested.

Let's start with the .Net format specified to format the date in PowerShell. To format the date we can use for example the following specifiers:

Specifier	Description
dd	2 digits Day of the month
MM	2 digits Month number
yy	2 digits Year number
dddd	Full name of the day
MMMM	Full name of the month
yyyy	4 digits Year number

PowerShell Date Formats

You can find the complete list of specifiers [here](#) in the documentation.

So to format the date, we can specify the parameter **-Format** followed by a string with the date format that we want. For example:

```
Get-Date -Format "dd-MM-yyyy"
```

```
06-12-2022
```

UFormat uses the same principle, only a different specifier of course.

Note

Good to know though is that specifiers of UFormat are changed since PowerShell 6.2. So keep that in mind when running your script on different PowerShell versions.

Specifier	Description
%d	2 digits Day of the month
%m	2 digits Month number
%y	2 digits Year number
%A	Full name of the day
%B	Full name of the month
%Y	4 digits Year number

PowerShell Date Formats

You can find the complete list of UFormat specifiers [here](#) in the documentation.

```
Get-Date -UFormat "%A %d %b %Y"
```

Result

```
dinsdag 06 dec 2022
```

Datetime Format

Just like with the date format, we can choose between .Net format and UFormat specifiers to format the DateTime result of **Get-Date** in PowerShell. Both return a formatted string and not a date object.

To format the time part of the date with the commonly used .Net format specifier you will need to use the **-format** parameter followed by a string with the time or DateTime format that you want to be returned.

Specifier	Description
HH:mm	Time in 24-hour format
HH:mm:ss	Time in 24-hour format with seconds
K	Time zone offset of UTC
hh:mm	Time in 12-hour format
hh:mm – tt	Time in 12-hour format with AM/PM

PowerShell format Time

```
Get-Date -Format "HH:mm K"
```

Result

```
16:09 +01:00
```

To format the DateTime in PowerShell using the UFormat specifier, you will need to use the parameter `-UFormat` followed by a string with the format. You can find all options for UFormat [here](#) in the documentation.

For example, to format the DateTime in PowerShell we can do the following:

```
Get-Date -UFormat "%A %d %B %Y %R %Z"  
# Result  
dinsdag 06 december 2022 16:14 +01
```

Get-Date Minus 1 day

Adding or subtracting days from dates in PowerShell can be done by using the AddDays method. We can use a positive number to add days and a negative number to subtract days. So if we want to get the date from yesterday, we will first get the current date with Get-Date minus 1 day:

```
(Get-Date).AddDays(-1)  
# Result  
maandag 5 december 2022 16:24:48
```

As you can see, we need to wrap the Get-Date cmdlet in parentheses. This way the Get-Date cmdlet is executed first, and then the other commands or methods.

When you need to calculate multiple dates, for example, when we need to [restore files between two dates](#) in SharePoint, then you can also store the current date in a variable. And then use the AddDays method to add or subtract days from it:

```
$today = Get-Date # dinsdag 6 december 2022 16:28:51  
$tomorrow = $today.AddDays(1) # woensdag 7 december 2022 16:28:51  
$yesterday = $today.AddDays(-1) # maandag 5 december 2022 16:28:51
```

AddDays is not the only method that we can use to calculate dates. We can also add or subtract months, years, or hours for example. All using the same principle as the AddDays method:

- AddTicks
- AddMilliseconds
- AddSeconds
- AddMinutes
- AddHours
- AddDays
- AddMonths
- AddYears

And the funny thing is, that we can combine them all if we want:

```
$today = Get-Date # dinsdag 6 december 2022 16:28:51  
$today.AddYears(2).AddMonths(3).AddDays(1).AddHours(2).AddMinutes(10)
```

Result:

vrijdag 7 maart 2025 18:38:51

Compare Dates in PowerShell

Comparing dates in PowerShell is actually easier than you might think. The `DateTime` objects can be compared directly with each other using the normal PowerShell comparison operators `-gt` and `-lt` for example.

So to compare two dates in PowerShell we can do the following:

```
$today = Get-Date # dinsdag 6 december 2022 16:44:51
```

```
$secondDate = Get-Date -Date "20-10-2022" # donderdag 20 oktober 2022
```

```
# Check if secondDate is before or after Today
```

```
$today -gt $secondDate # True
```

```
$today -lt $secondDate # False
```

We can use this method also to check if a file is created before or after a date, using the `Get-ChildItem` cmdlet:

```
$secondDate = Get-Date -Date "20-10-2022" # donderdag 20 oktober 2022
```

```
$fileDate = get-childitem updates.txt | Select -ExpandProperty CreationTime
```

```
# FileDate is donderdag 24 februari 2022 13:11:41
```

```
if ($secondDate -gt $fileDate) {
```

```
Write-Host "File is older then $secondDate"
```

```
}
```

Convert String to Date with PowerShell

Before you can work with dates in PowerShell, you will need to make sure that the dates are `DateTime` objects. For example, when you have a CSV file with dates in it, that you would like to use. Then you will need to convert the string to date first.

To do this we can simply cast a string to a `DateTime` object:

```
[DateTime] "2022-11-24"
```

```
# Result
```

```
maandag 24 november 2022 00:00:00
```

We can also format the results immediately if needed, by placing the `dateTime` casting between parentheses and then converting it to a formatted string:

```
([DateTime] "2022-11-24").ToString('yyyy-MM-dd')
```

If the string that you want to convert to date has a divergent format, then you might need to specify the format of the date string using `ParseExact`. Important here is that you specify the input format, not the output!

```
$dateString = '14-nov-22'
```

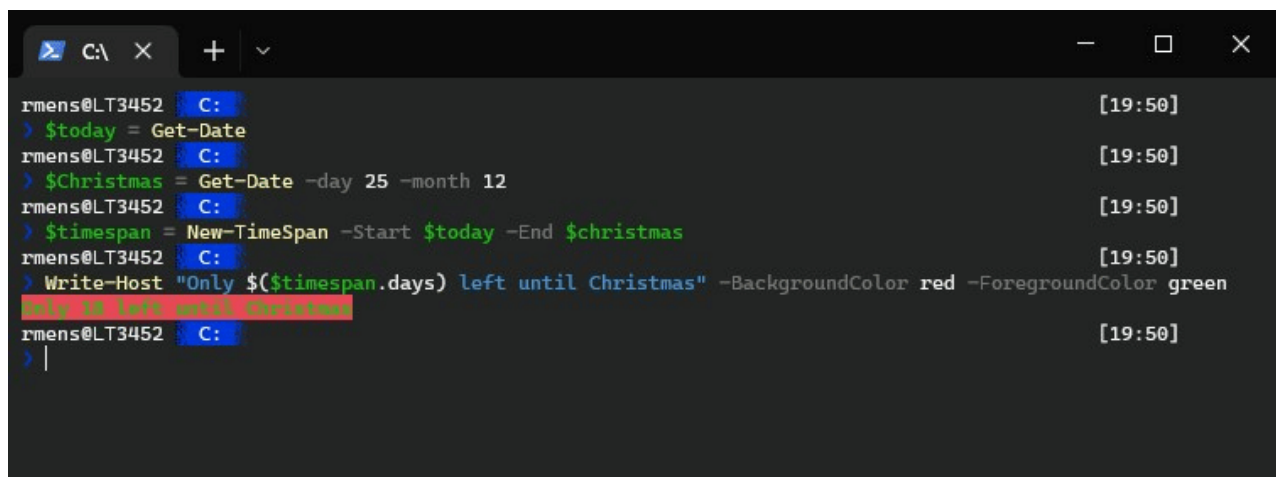
```
$date = [datetime]::ParseExact($dateString, "dd-MMM-yy", $null)
```

Get Days Between Dates

We can compare dates, add, or subtract days from dates, but did you know that we can also calculate the days (or time) between dates in PowerShell? To do this we are going to use the `New-TimeSpan` cmdlet with two date objects.

Let's say that we want to know the days until Christmas. For this, we take the current date (start date) with `Get-Date` and create a new date object for the 25th of Dec (end-date):

```
$today = Get-Date
$Christmas = Get-Date -day 25 -month 12
New-TimeSpan -Start $today -End $Christmas
# Result
Days : 18
Hours : 0
Minutes : 5
Seconds : 22
Milliseconds : 241
Ticks : 15555222411487
TotalDays : 18,0037296429248
TotalHours : 432,089511430194
TotalMinutes : 25925,3706858117
TotalSeconds : 1555522,2411487
TotalMilliseconds : 1555522241,1487
```

A screenshot of a PowerShell terminal window. The window title bar shows 'C:\' and standard window controls. The terminal content shows a series of commands and their outputs. The commands are: '\$today = Get-Date', '\$Christmas = Get-Date -day 25 -month 12', '\$timespan = New-TimeSpan -Start \$today -End \$Christmas', and 'Write-Host "Only \$(\$timespan.days) left until Christmas" -BackgroundColor red -ForegroundColor green'. The output of the last command is 'Only 18 left until Christmas', which is highlighted with a red background and green text. The prompt 'rmens@LT3452' and the directory 'C:' are visible at the start of each line.

```
rmens@LT3452 C: [19:50]
> $today = Get-Date
rmens@LT3452 C: [19:50]
> $Christmas = Get-Date -day 25 -month 12
rmens@LT3452 C: [19:50]
> $timespan = New-TimeSpan -Start $today -End $Christmas
rmens@LT3452 C: [19:50]
> Write-Host "Only $($timespan.days) left until Christmas" -BackgroundColor red -ForegroundColor green
Only 18 left until Christmas
rmens@LT3452 C: [19:50]
> |
```

Get Days Between Dates

Wrapping Up

Working with dates may seem difficult at first, but I have to say, PowerShell made it pretty easy. As you have seen we can easily compare dates, add or subtract date and time from dates, and calculate the date and time between two dates.

I hope you found this article helpful, 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.