

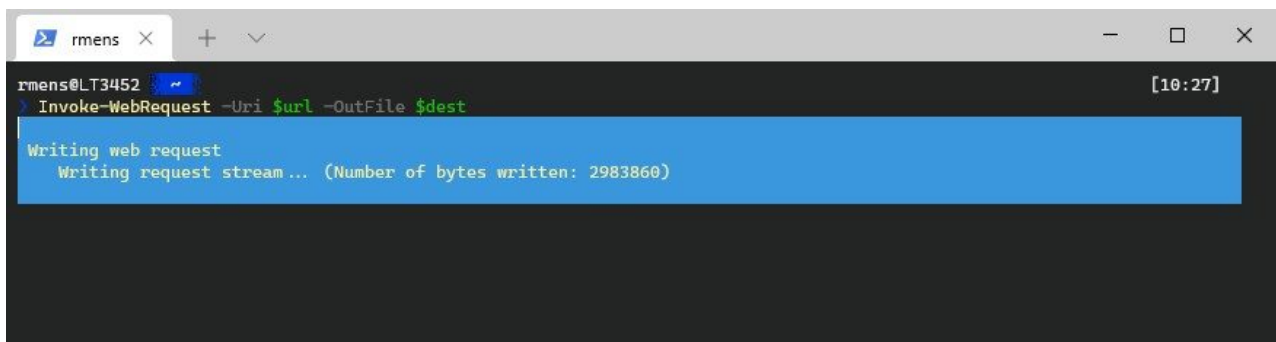
How to Download a File with PowerShell

 lazyadmin.nl/powershell/download-file-powershell

September 14, 2021

PowerShell is a great scripting language to write all kinds of scripts. But did you know that you can also download a file with PowerShell? You can use PowerShell to download single or multiple files from the internet.

There are a couple of methods when it comes to downloading files with PowerShell. We can download files from any URL with PowerShell, local network shares, and from behind credential protected websites.



```
rmens@LT3452 ~  
PS C:\> Invoke-WebRequest -Uri $url -OutFile $dest  
  
Writing web request  
Writing request stream... (Number of bytes written: 2983860)
```

In this article, we are going to start with the most straightforward method to download a single file and we are also going to take a look at other (faster) methods to download a file with PowerShell.

Powershell Download File from URL

We are going to start with the most common way to download a file from an URL with PowerShell. For this, we will be using the `Invoke-WebRequest` cmdlet. To download a file we need to know the source URL and give up a destination for the file that we want to download.

The parameter `-OutFile` is required. You don't need to enter the full path, but a file name is required.

Source URL

```
$url = "http://speed.transip.nl/10mb.bin"
```

Destination file

```
$dest = "c:\temp\testfiles.bin"
```

Download the file

```
Invoke-WebRequest -Uri $url -OutFile $dest
```

Invoke-WebRequest will overwrite the local file if it already exists without any warning

Authentication with Invoke-WebRequest

Some online resources require you to log in before you can access/download the files. With the Invoke-WebRequest cmdlet, we can provide the credentials that are needed for downloading the files.

If you are creating a script that will need to run automatically, then you will need to store the credentials in the script itself. I recommend creating a secure string password and store it in a text file on the computer that is running the script. It still won't be super secure, but it's a little bit better than using a plaintext password in your script.

```
# URL and Destination
$url = "http://speed.transip.nl/10mb.bin"
$dest = "c:\temp\testfiles"
# Define username and password
$username = 'LazyUsrName'
$password = 'StrongPlainTextPasswd'
# Convert to SecureString
$secPassword = ConvertTo-SecureString $password -AsPlainText -Force
# Create Credential Object
$credObject = New-Object System.Management.Automation.PSCredential ($username,
$secPassword)
# Download file
Invoke-WebRequest -Uri $url -OutFile $dest -Credential $credObject
```

Download files faster with Start-BitsTransfer in PowerShell

The Invoke-WebRequest method is available in all PowerShell versions and can also be used on Linux machines. But the downside is that it's a bit slow. With Invoke-WebRequest, the file is buffered in the memory first, before it's written to the disk.

A faster and better way is to use the Start-BitsTransfer cmdlet in PowerShell. This cmdlet allows you to queue files, set priority (useful for bandwidth limitation), can run in the background and download multiple files asynchronous.

This is the most basic method of downloading a file with BitsTransfer, you only need a source and destination.

```
# URL and Destination
$url = "http://speed.transip.nl/10mb.bin"
$dest = "c:\temp\testfiles"
# Download file
Start-BitsTransfer -Source $url -Destination $dest
By default, the download jobs run in the foreground consuming the maximum bandwidth
available. You can change this by setting the priority of the job:
```

- Foreground – Default
- High

- Normal
- Low

Only the idle network bandwidth is used when you set the priority to high, normal, or low.

Another option is to run the download job **asynchronous**, allowing you to start multiple download jobs at the same time. When you use this method, **make sure that you complete** the download job when it's finish.

Start-BitsTransfer -Source \$url -Destination \$dest -Asynchronous -Priority normal

```

rmens@LT3452 [20:16]
> Start-BitsTransfer -Source $url -Destination $dest -Asynchronous -Priority normal

JobId                DisplayName  TransferType JobState  OwnerAccount
-----
540efa12-e4bb-4593-87cf-11ca7f68cf3c BITS Transfer Download Connecting THUNNISSEN\rmens

rmens@LT3452 [20:18]
> Get-ChildItem $dest -Hidden

Directory: C:\temp\testfiles

Mode                LastWriteTime         Length Name
----
-a-h--           14-9-2021   20:18         10485760 BIT809E.tmp

rmens@LT3452 [20:18]
> |

```

As you can see I have downloaded the same bin file as before. But if we look in the destination folder we only see a .tmp file.

You will need to run Complete-BitsTransfer to finish the download job.

Get-BitsTransfer | Complete-BitsTransfer

Downloading Multiple Files with PowerShell

To download multiple files with PowerShell we first need to know which files are available. We can use the Invoke-WebRequest cmdlet first to get the content from the webpage.

If you take a look at the content of <http://speed.transip.nl> then you will see a list of binary files that we can download.

First, we are going to scrape the website

```
$content = Invoke-WebRequest -URI "http://speed.transip.nl"
```

This will return not only the content of the webpage but also other properties, like Links and InputFields.

```

rmen x + -
StatusCode : 200
StatusDescription : OK
Content : <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
        <html>
        <head>
        <title>Index of /</title>
        </head>
        <body>
        <ul><li><a href="1000mb.bin"> 1000mb.bin</a></li>
        <li><a href="1 ...
RawContent : HTTP/1.1 200 OK
        Keep-Alive: timeout=5, max=100
        Connection: Keep-Alive
        Content-Length: 655
        Content-Type: text/html; charset=ISO-8859-1
        Date: Tue, 14 Sep 2021 11:01:05 GMT
        Server: Apache

        <!DOCTY ...
Forms : {}
Headers : {[Keep-Alive, timeout=5, max=100], [Connection, Keep-Alive], [Content-Length, 655], [Content-Type,
        text/html; charset=ISO-8859-1] ... }
Images : {}
InputFields : {}
Links : {@{innerHTML=1000mb.bin; innerText=1000mb.bin; outerHTML=<A href="1000mb.bin">1000mb.bin</A>; outer
        Text=1000mb.bin; tagName=A; href=1000mb.bin}, @{innerHTML=100mb.bin; innerText=100mb.bin; outerHTML
        =<A href="100mb.bin">100mb.bin</A>; outerText=100mb.bin; tagName=A; href=100mb.bin}, @{innerHTML=10
        mb.bin; innerText=10mb.bin; outerHTML=<A href="10mb.bin">10mb.bin</A>; outerText=10mb.bin; tagName=
        A; href=10mb.bin}, @{innerHTML=1gb.bin; innerText=1gb.bin; outerHTML=<A href="1gb.bin">1gb.bin</A>;
        outerText=1gb.bin; tagName=A; href=1gb.bin} ... }
ParsedHtml : mshtml.HTMLDocumentClass

```

This is a pretty simple webpage, but let's say we only want the files that start with the name "random". We can filter the links with a simple like query and select only the href property from each link.

```
$randomBinFiles = $content.links | where {$_.innerHTML -like 'random*'} | select href
```

So we now have the links for all random binary files. All we need to do is download each one of them.

When you need to download multiple files it's better to use the **Start-BitsTransfer** cmdlet. It allows you to download multiple files simultaneously in the background with the parameter **-Asynchronous**

Other advantages of the BitsTransfer cmdlet is it can handle connection interruptions and is aware of your network bandwidth usage.

```

$url = "http://speed.transip.nl"
# Create full links for each entry
$randomBinFiles.foreach( { $_.href = $url + "/" + $_.href })
# Download each file in the background
$randomBinFiles.foreach({
Start-BitsTransfer ($url + "/" + $_.href) -Asynchronous
})
# Complete the BitsTransfer
Get-BitsTransfer | Complete-BitsTransfer

```

We can start all the download jobs by using the parameter **-Asynchronous**. Without it, the BitsTransfer cmdlet downloads the first file completely before starting the next download while putting your script on hold in the meantime.

```
rmens@LT3452 [13:32]
> $randomBinFiles.foreach({
>>     Start-BitsTransfer ($url + "/" + $_.href) -Asynchronous
>> })

JobId                               DisplayName      TransferType JobState      OwnerAccount
-----
0c44b8ed-cfb2-40b3-a0b5-6a43b54686d9 BITS Transfer Download Connecting
f808c2c1-7cad-4fe0-a7f3-14f68fc1be8c BITS Transfer Download Connecting
98d5fb98-7c21-4f3b-b04f-674777163493 BITS Transfer Download Connecting
056cbbfb-3c6c-4366-8ff8-a38fea54f941 BITS Transfer Download Connecting
990cc36c-bcbe-494d-948f-1c3c07b4d779 BITS Transfer Download Connecting

rmens@LT3452 [13:32]
> Get-BitsTransfer

JobId                               DisplayName      TransferType JobState      OwnerAccount
-----
0c44b8ed-cfb2-40b3-a0b5-6a43b54686d9 BITS Transfer Download Transferring
f808c2c1-7cad-4fe0-a7f3-14f68fc1be8c BITS Transfer Download Transferring
98d5fb98-7c21-4f3b-b04f-674777163493 BITS Transfer Download Transferred
056cbbfb-3c6c-4366-8ff8-a38fea54f941 BITS Transfer Download Transferring
990cc36c-bcbe-494d-948f-1c3c07b4d779 BITS Transfer Download TransientError

rmens@LT3452 [13:32]
> |
```

You can use the Get-BitsTransfer cmdlet to show the progress of the download. If you want to stop the download job then use the Remove-BitsTransfer cmdlet. You can stop a single job based on its JobId or all jobs with:

```
Get-BitsTransfer | % {Remove-BitsTransfer $_.JobId}
```

Complete the BitsTransfer download

It's important to complete the transferred file when using BitsTransfer in combination with the **Asynchronous** parameter. When using Asynchronous it creates a temp file during the download process. But to actually use the file you will need to run the following cmdlet:

```
Get-BitsTransfer | Complete-BitsTransfer
```

PowerShell Download file from Server

We won't be using the Invoke-WebRequest to download files from a local network source, like a server or NAS, with PowerShell. Instead, we can simply use the Copy-Item cmd to download a file from a server.

The Copy-Items cmdlet takes a source and destination, just like the Invoke-WebRequest cmdlet.

```
# Set Source and destination
$source = "\\LA-WIN10-LAB01\speedtest\"
$destination = "c:\temp\testfiles"
# Copy all items in the source folder
Copy-Item -path $source -Destination $destination -Recurse
```

If you want to know more about the Copy-Item cmdlet, then you should read this article where I explain more about the cmdlet and alternatives.

Powershell Download Zip File

The method to download zip files is pretty much the same as a normal file. But I wanted to show you how that downloads and extracts the zip file. This way you can immediately process the files inside the zip file without manual interaction.

I am going to use [this sample csv on GitHub](#) which we can download in a zip file. We have to set a destination for the zip file itself and a path where we want to extract the files to.

The Invoke-WebRequest downloads the zip file just like any other file.

```
# URL and Destination
```

```
$url = "https://github.com/datapackage-examples/sample-csv/archive/refs/heads/master.zip"
```

```
# Create temp destination for the zip and get zipfile name from source URL
```

```
$zipFile = "c:\temp\zipfiles" + $(Split-Path -Path $url -Leaf)
```

```
# Extract path
```

```
$extractPath = "c:\temp\zipfiles"
```

```
# Download file
```

```
Invoke-WebRequest -Uri $url -OutFile $zipFile
```

The next step is to extract the zip file automatically in the desired location. For this we are going to use a COM object. With the COM object we can extract the zip file and copy the content to the desired location.

```
# Create instance of COM Object
```

```
$objShell = New-Object -ComObject Shell.Application
```

```
# Extract the Files
```

```
$extractedFiles = $objShell.Namespace($zipFile).Items()
```

```
# Copy the extracted files to the destination folder
```

```
$objShell.Namespace($extractPath).CopyHere($extractedFiles)
```

Wrapping Up

Downloading files with PowerShell is pretty easy when you have the exact URL of the source file. When you need to scrape a website first then it can be a little bit more work to set up properly.

Try to use the Start-BitsTransfer cmdlet for downloading files and set the priority to normal when using it in an autonouse script. BitsTransfer has more option when it comes to retries, resuming and bandwidth control then Invoke-WebRequest.

If you have any questions about how you can download a file with PowerShell, then 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.