How-to use Test-NetConnection in PowerShell

lazyadmin.nl/powershell/test-netconnection

March 1, 2021

Did you know that the ping command, with its 38 years, is older than the command prompt? You probably already have replaced the command prompt from your daily toolset with PowerShell. So it's now time to replace the ping command with **PowerShell Test-NetConnection**.

The advantage of ping is that it's a really short command to type, but the downside it's that it only tests networking connecting over ICMP. It doesn't do any tracing or port testing. Most of the time we need to combine the ping command with tracert and nslookup to get all the details we need.

Test-NetConnection is the successor of the PowerShell Test-Connection cmdlet. The latter was the first step in replacing the ping command, with some advantages. Test-NetConnection is a lot more powerful and should be your go-to tool when it comes to troubleshooting network problems.

PowerShell Test-Connection

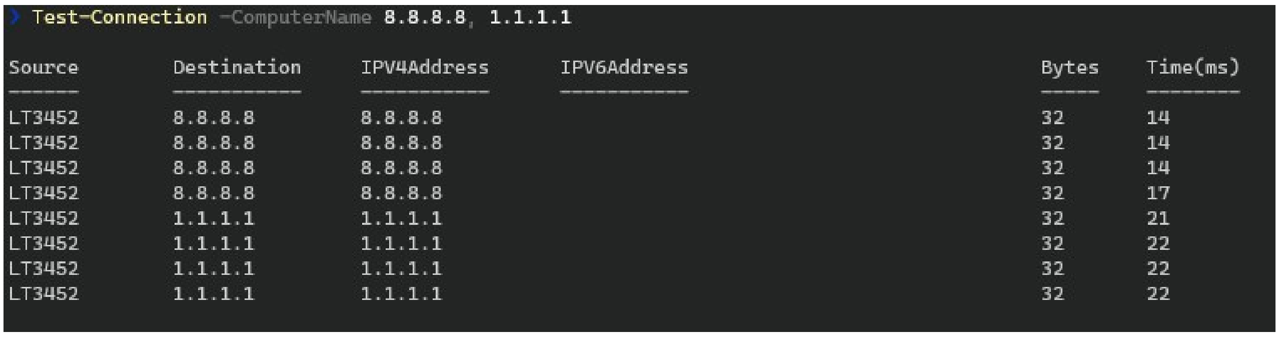
Before we take a look at how to use the Test-NetConnection cmdlet in PowerShell lets first take a quick look at `Test-Connection`.

Just like ping, uses Test-Connection also the ICMP protocol to test the network connectivity of a network device. In the simplest form, you can type `Test-Connection <computername>` or `<ip-address>` to do a quick connectivity test.

```
# Check connection on IP Address
Test-Connection 8.8.8.8
Source Destination IPV4Address Bytes Time(ms)
------ ----------- ----------- ------- ----
lab01 8.8.8.8 8.8.8.8 32 12
lab01 8.8.8.8 8.8.8.8 32 13
lab01 8.8.8.8 8.8.8.8 32 12
lab01 8.8.8.8 8.8.8.8 32 12
# Check connection on DNS Name
Test-Connection google.com
```

But we can of course a lot more. For example, we can test multiple destinations with one command:

```
Test-Connection -ComputerName 8.8.8.8, 1.1.1.1
```


```
❯ Test-Connection -ComputerName 8.8.8.8, 1.1.1.1

Source        Destination     IPV4Address      IPV6Address                                    Bytes    Time(ms)
------        -----------     -----------      -----------                                    -----    --------
LT3452        8.8.8.8         8.8.8.8                                                         32       14
LT3452        8.8.8.8         8.8.8.8                                                         32       14
LT3452        8.8.8.8         8.8.8.8                                                         32       14
LT3452        8.8.8.8         8.8.8.8                                                         32       17
LT3452        1.1.1.1         1.1.1.1                                                         32       21
LT3452        1.1.1.1         1.1.1.1                                                         32       22
LT3452        1.1.1.1         1.1.1.1                                                         32       22
LT3452        1.1.1.1         1.1.1.1                                                         32       22
```

Or specify parameters like the number of hops, buffer size or even add a delay between the pings:

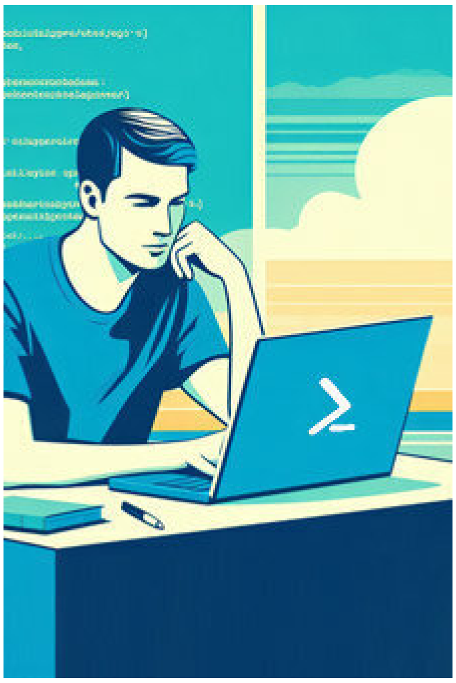
Test-Connection -ComputerName 8.8.8.8 -Count 3 -Delay 2 -MaxHops 255 -BufferSize 256

Testing a remote computer

We can also test the network connection in PowerShell of a remote computer with Test-Connection. The only requirement is that your account has access to the remote computer.

Test-Connection -Source srv-lab02 -ComputerName 8.8.8.8

> You can also type tnc instead of Test-NetConnecting.
>
> *So you can type:* `tnc 8.8.8.8` *to ping Google for example.*



PowerShell Test Connection to Server

Another useful feature of PowerShell Test-Connection is that it can return `$true` or `$false`. This allows you to check if a computer is available before connecting to it. Test-Connect will return `$true` if any of the 4 pings are successful.

```
if (Test-Connection -TargetName srv-lab02 -Quiet) { New-PSSession -ComputerName srv-lab02}
```

PowerShell Test-NetConnection

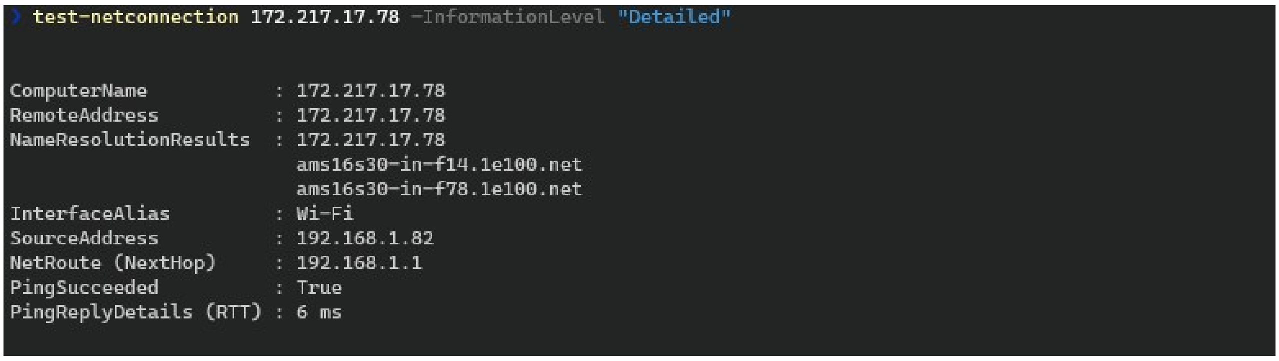
The **Test-NetConnection** cmdlet is the successor of Test-Connection, so the basic functions are all the same, only the output is more detailed. We can still do a quick ping to Google:

```
# Test the network connection to Google
Test-NetConnection 8.8.8.8
ComputerName : 8.8.8.8
RemoteAddress : 8.8.8.8
InterfaceAlias : Wi-Fi
SourceAddress : 192.168.1.82
PingSucceeded : True
PingReplyDetails (RTT) : 9 ms
```

As you can see the results are more detailed. It will show you the interface that is used, your source IP Address, and if the ping succeeded. We can even expand this by adding the `informationLevel "Detailed"` switch.

With the `informationLevel` set to detailed it will do basically a nslookup on the destination address and it will add the first hop in the lookup.

test-netconnection Google.com -InformationLevel "Detailed" ComputerName : Google.com RemoteAddress : 172.217.17.78 NameResolutionResults : 172.217.17.78 InterfaceAlias : Wi-Fi SourceAddress : 192.168.1.82 NetRoute (NextHop) : 192.168.1.1 PingSucceeded : True PingReplyDetails (RTT) : 6 ms



```
❯ test-netconnection 172.217.17.78 -InformationLevel "Detailed"


ComputerName           : 172.217.17.78
RemoteAddress          : 172.217.17.78
NameResolutionResults  : 172.217.17.78
                         ams16s30-in-f14.1e100.net
                         ams16s30-in-f78.1e100.net
InterfaceAlias         : Wi-Fi
SourceAddress          : 192.168.1.82
NetRoute (NextHop)     : 192.168.1.1
PingSucceeded          : True
PingReplyDetails (RTT) : 6 ms
```

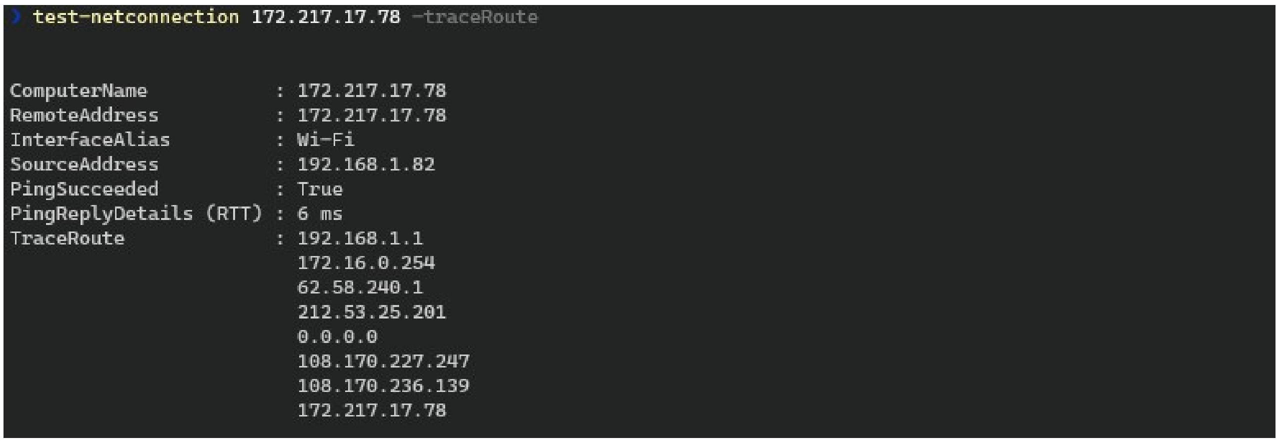
You can do a quick network connection test in PowerShell when you run Test-NetConnection without any parameters. This way we can check if we are connected to the local network, have access to internet and are able to resolve DNS names.

❯ Test-NetConnection
ComputerName : internetbeacon.msedge.net
RemoteAddress : 13.107.4.52
InterfaceAlias : Wi-Fi
SourceAddress : 192.168.1.82
PingSucceeded : True
PingReplyDetails (RTT) : 10 ms

## PowerShell TraceRoute with Test-NetConnection

With the Test-NetConnection cmdlet in PowerShell we can also do a **traceroute**. You only need to add the `traceroute` parameter to the cmdlet.

# PowerShell TraceRoute with Test-NetConnection
Test-NetConnection 172.217.17.87 -traceRoute



```
❯ test-netconnection 172.217.17.78 -traceRoute


ComputerName           : 172.217.17.78
RemoteAddress          : 172.217.17.78
InterfaceAlias         : Wi-Fi
SourceAddress          : 192.168.1.82
PingSucceeded          : True
PingReplyDetails (RTT) : 6 ms
TraceRoute             : 192.168.1.1
                         172.16.0.254
                         62.58.240.1
                         212.53.25.201
                         0.0.0.0
                         108.170.227.247
                         108.170.236.139
                         172.217.17.78
```

With the Traceroute parameter added to the PowerShell Test-NetConnection cmdlet, you will get a list of every hop that is passed during the ping. You can limit the amount the hops to trace with the `hops` parameter.

To test the latency of each hop, you can select the TraceRoute object, and test the connection of each hop:

```
Test-NetConnection 172.217.17.78 -traceRoute -Hops 3 | select-object TraceRoute |
foreach-object {test-connection $_.TraceRoute -count 1}
```

Powershell Port Scan

In Windows, we can’t really ping a port with the ping cmdlet. We could use telnet to test if a port responds, but with Test-NetConnection we can scan a port in PowerShell much easier.

We can define any TCP port that we want to test, or use one of the common ports HTTP, RDP, SMB, or WINRM.

```
# Test if HTTP port is open
Test-NetConnection google.com -CommonTCPPort "Http"
# Or define a port number
Test-NetConnection google.com -Port 80
```

With this, we can create a simple PowerShell Port Scan script that can check the open ports on a server. The script below is to give you an idea of how you can scan multiple ports with PowerShell.

```
$ports = 22,53,80,445
$ports | ForEach-Object {$port = $_; if (Test-NetConnection -ComputerName 8.8.8.8 -Port
$port -InformationLevel Quiet -WarningAction SilentlyContinue) {"Port $port is open" }
else {"Port $port is closed"} }
```

Wrapping Up

As you have seen is the PowerShell Test-NetConnection is a really powerful and useful cmdlet to use in your daily work. At the beginning of the article, I mentioned that ping is shorter to type. You can solve that by creating an Alias in your PowerShell profile or use the built-in alias `tnc`.

```
# Create an alias "nettest" or maybe just "tn"
Set-Alias nettest Test-NetConnection
```

This way you can simply type “nettest” instead of Test-NetConnection. Read more about setting up your PowerShell Profile in this article.

If you have any questions, then just drop a comment below.

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