Building a Ping Sweep Tool with PowerShell

Posted on February 13, 2015 by **Jeff Hicks** in PowerShell with 1 Comment

I think one of the best ways to learn PowerShell is by using it. I've often found the best way t
on. With that in mind, I'm going to start us on a little PowerShell project. Although the end r
journey is more important than the destination.

I'm hoping that as you read the articles in this short series, you'll learn new PowerShell com
Honestly, I'm not sure where we'll end up. I know where I want to start and what I want to sl
course of working on this project, I'll come up with another idea or concept to add on.

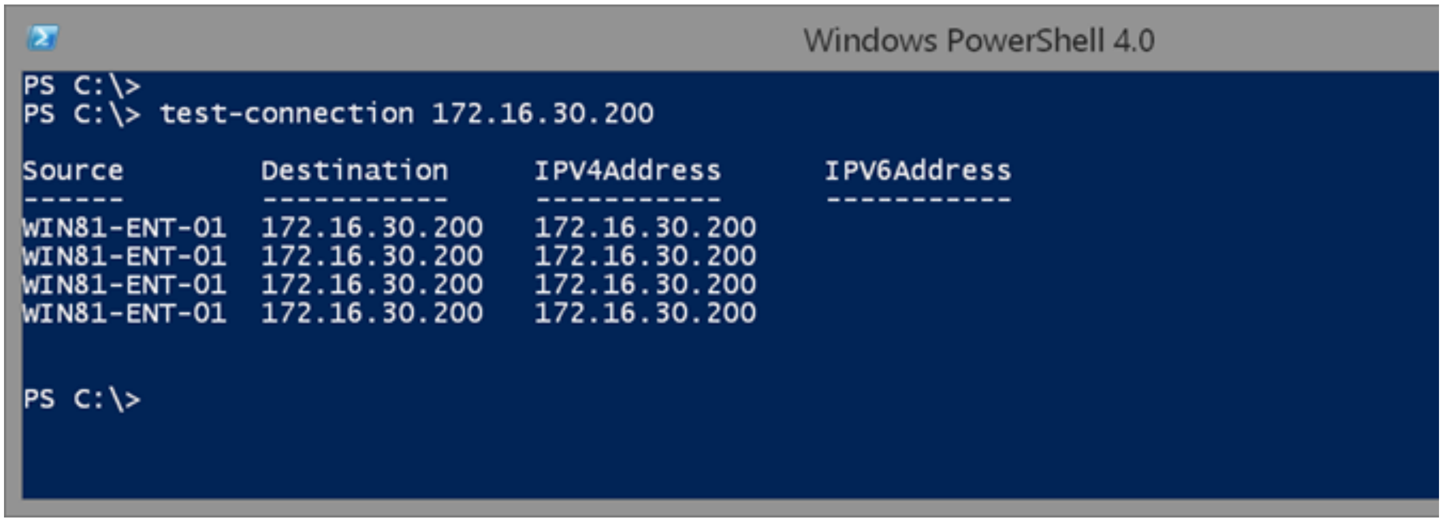
This is the way I work: start with a core concept and get it working, then slowly add addition
advance everything that I want to include, I rarely create a tool with everything from the ver
the project, then the more likely the need to debug and troubleshoot. I find it easier to take

step to work before adding the next. But enough philosophy, let's get scripting.

PowerShell Ping Sweep Tool Article Series

The task is to create a PowerShell tool that can be used to ping a range of IP addresses. Eve
function that we can run just like a cmdlet. But we don't need to do that just yet. First, I alwa
work at an interactive console prompt. Because there is no difference between commands
script, let's get the basics sorted out first without the distractions of trying to create a functi

I know that the PowerShell command for the actual pinging will be Test-Connection. I'll trust
at full help and examples for this command. The cmdlet will take a computername or IP add

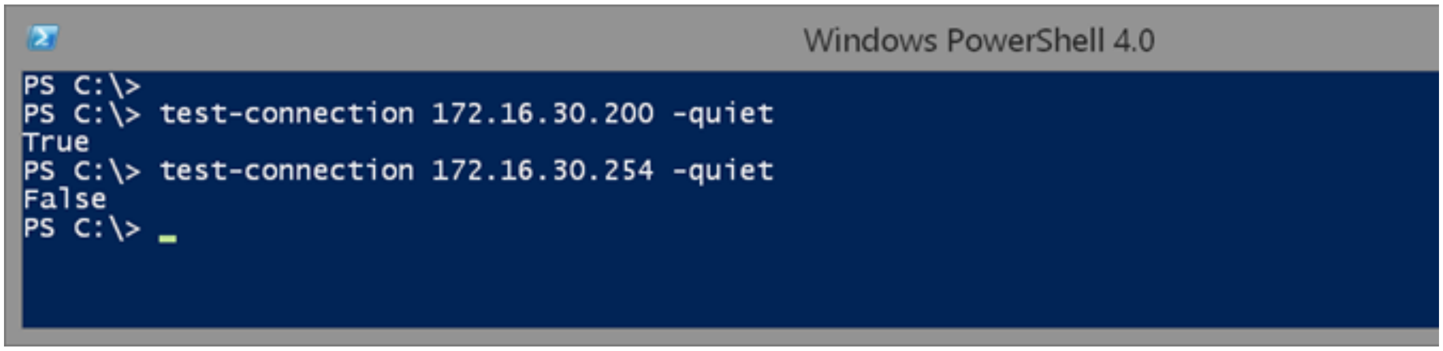


Windows PowerShell 4.0

```
PS C:\>
PS C:\> test-connection 172.16.30.200

Source        Destination     IPV4Address      IPV6Address
------        -----------     -----------      -----------
WIN81-ENT-01  172.16.30.200   172.16.30.200
WIN81-ENT-01  172.16.30.200   172.16.30.200
WIN81-ENT-01  172.16.30.200   172.16.30.200
WIN81-ENT-01  172.16.30.200   172.16.30.200


PS C:\>
```

The test-connection cmdlet in Windows PowerShell. (Image Credit: Je

In looking at the help, I see a –Quiet parameter. When used, the cmdlet will return either Tr
there was a response or not.


Windows PowerShell 4.0

```
PS C:\>
PS C:\> test-connection 172.16.30.200 -quiet
True
PS C:\> test-connection 172.16.30.254 -quiet
False
PS C:\> _
```

Using the -quiet parameter with the test-connection cmdlet in Windows PowerShell.

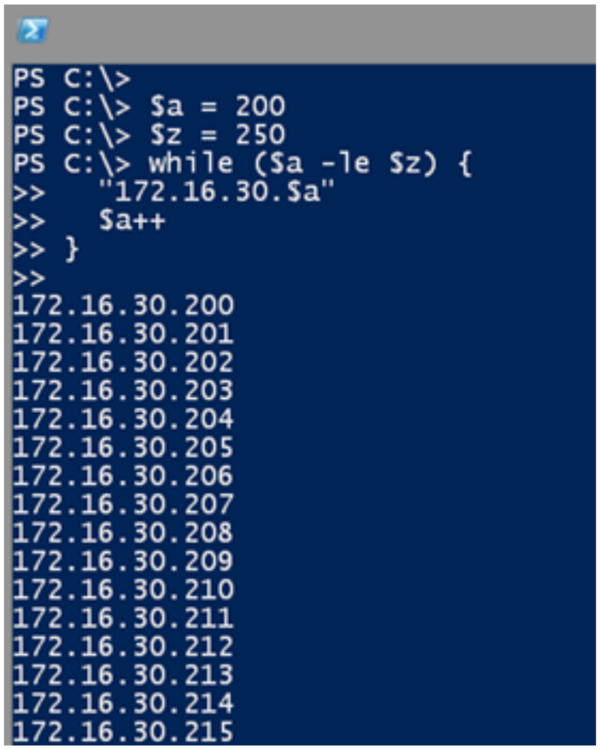
By default, Test-Connection sends four pings. Since I'm testing a local subnet, if the comput
it probably isn't going to respond to any, so I can improve performance by using the –Count

```
Test-connection 172.16.30.200 –count 1 -quiet
```

Next, I need to figure out how to ping a range of addresses such as 172.16.30.200 to 172.16
type all those addresses. There are a few ways to solve this. Here's the first:

```
$a = 200
$z = 250
while ($a -le $z) {
  "172.16.30.$a"
  $a++
}
```

In this example, I'm using a looping construct that says, "While the value of $a is less than o
string that looks like an IP address using the current value of $a and then increment $a by 1

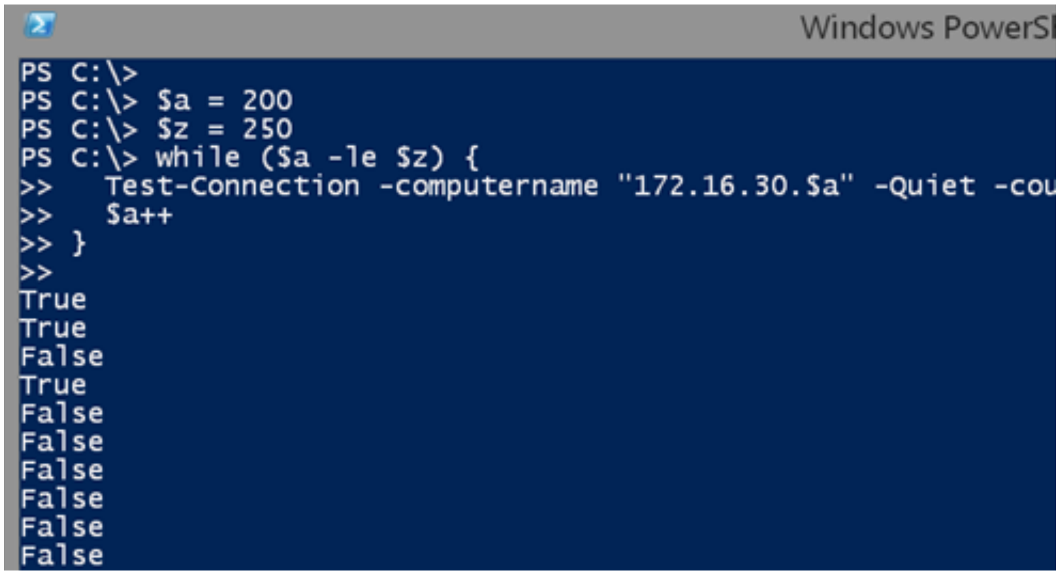


```
PS C:\>
PS C:\> $a = 200
PS C:\> $z = 250
PS C:\> while ($a -le $z) {
>>   "172.16.30.$a"
>>   $a++
>> }
>>
172.16.30.200
172.16.30.201
172.16.30.202
172.16.30.203
172.16.30.204
172.16.30.205
172.16.30.206
172.16.30.207
172.16.30.208
172.16.30.209
172.16.30.210
172.16.30.211
172.16.30.212
172.16.30.213
172.16.30.214
172.16.30.215
```

My loop construct in Windows PowerShell. (Image Credit: Jeff Hicks)

All I have to do is insert my Test-Connection command:

```
$a = 200
$z = 250
while ($a -le $z) {
Test-Connection -computername "172.16.30.$a" -Quiet -count 1
$a++
}
```



```
PS C:\>
PS C:\> $a = 200
PS C:\> $z = 250
PS C:\> while ($a -le $z) {
>>    Test-Connection -computername "172.16.30.$a" -Quiet -cou
>>    $a++
>> }
>>
True
True
False
True
False
False
False
False
False
False
```

My loop construct with the test-connection command. (Image Credit: J

And here is an alternative:

```
$a = 200
$z = 250
$a..$z | foreach {
  $ip = "172.16.30.0" -replace "0$",$_
  Test-Connection -ComputerName $IP -Count 1 -Quiet}
```

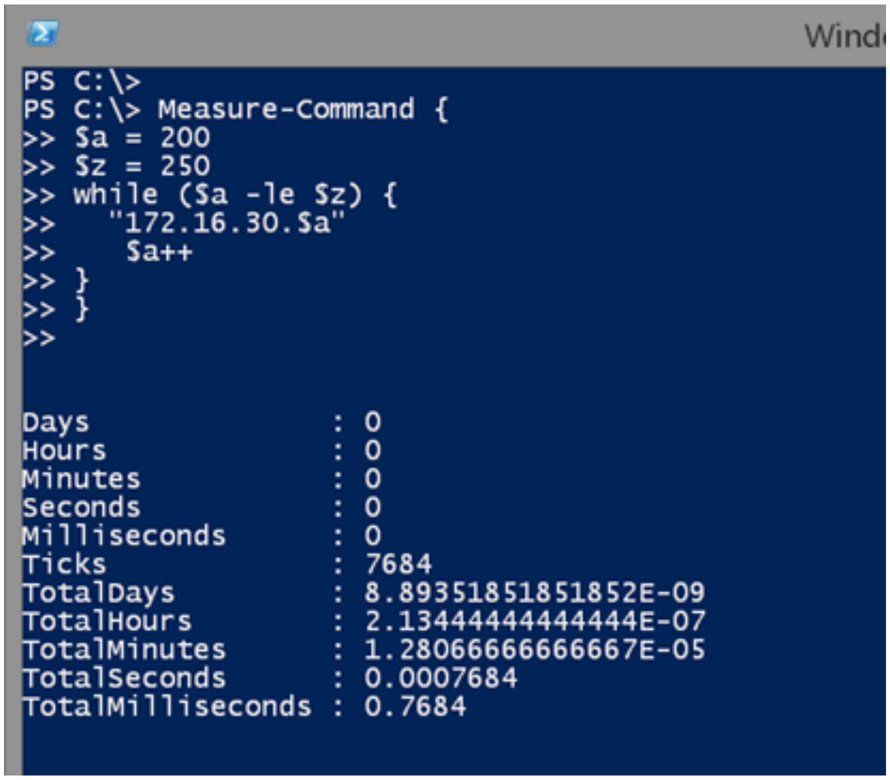
In this example, I am getting all of the numbers between $a and $z using the Range (..) oper ForEach-Object, where I am constructing the IP address by using the –Replace operator. The regular expression pattern to match and the value to replace on that match.

The pattern is looking for an ending in 0, and it will be replaced with some number betweer one hand, I would say whichever you find easier to understand. But perhaps some hard dat Connection out of the picture, which technique is faster at creating an IP address? To figure Command.

To use this cmdlet, simply wrap the commands to be tested in a set of curly braces to create

```
Measure-Command {
```

```
$a = 200
$z = 250
while ($a -le $z) {
  "172.16.30.$a"
    $a++
}
}
```

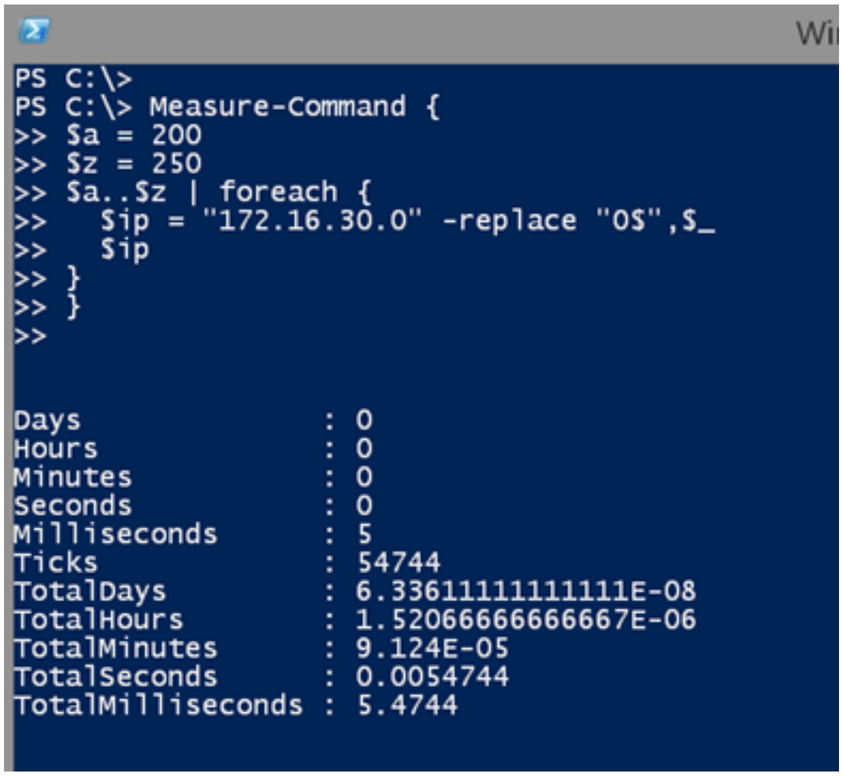


```
PS C:\>
PS C:\> Measure-Command {
>> $a = 200
>> $z = 250
>> while ($a -le $z) {
>>    "172.16.30.$a"
>>     $a++
>> }
>> }
>>


Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 0
Milliseconds      : 0
Ticks             : 7684
TotalDays         : 8.89351851851852E-09
TotalHours        : 2.13444444444444E-07
TotalMinutes      : 1.28066666666667E-05
TotalSeconds      : 0.0007684
TotalMilliseconds : 0.7684
```

The Measure-Command in Windows PowerShell. (Image Credit: Jeff Hicks)

Pretty darn fast. Let's check the other technique.

```
Measure-Command {
$a = 200
$z = 250
$a..$z | foreach {
  $ip = "172.16.30.0" -replace "0$",$_
  $ip
}
}
```



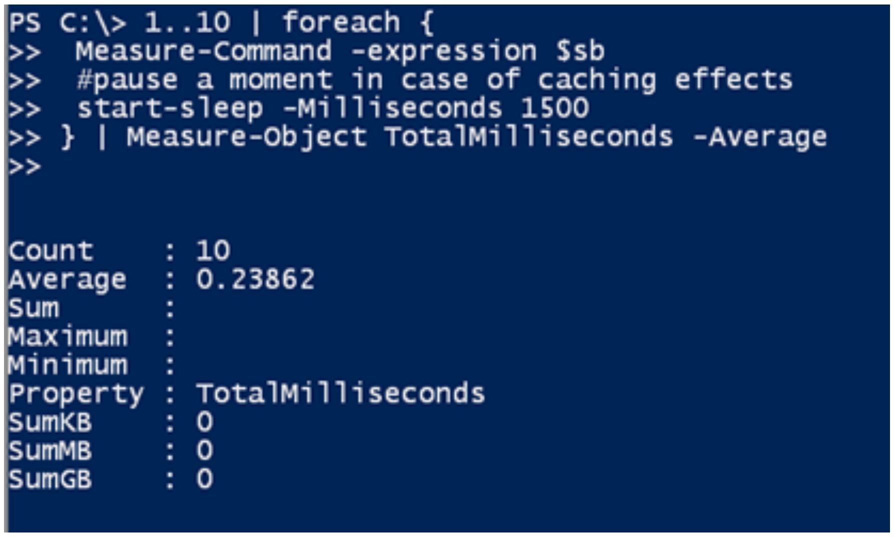
```
PS C:\>
PS C:\> Measure-Command {
>> $a = 200
>> $z = 250
>> $a..$z | foreach {
>>    $ip = "172.16.30.0" -replace "0$",$_
>>    $ip
>> }
>> }
>>


Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 0
Milliseconds      : 5
Ticks             : 54744
TotalDays         : 6.33611111111111E-08
TotalHours        : 1.52066666666667E-06
TotalMinutes      : 9.124E-05
TotalSeconds      : 0.0054744
TotalMilliseconds : 5.4744
```

Another approach to using Measure-Command. (Image Credit: Jeff Hicks)

When using Measure-Command, first make sure the scriptblock runs successfully. As you ca
significantly faster at least from the computer's perspective. Personally, there's not much hu
But you may want to run a few tests and average the results to be sure.

```
$sb = {
$a = 200
$z = 250
while ($a -le $z) {
  "172.16.30.$a"
   $a++
 }
}
1..10 | foreach {
 Measure-Command -expression $sb
 #pause a moment in case of caching effects
 start-sleep -Milliseconds 1500
} | Measure-Object TotalMilliseconds -Average
```

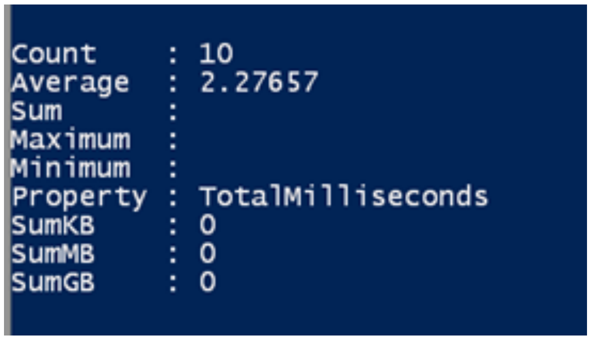
This command is looping 10 times and measuring the scriptblock each time through. The re
Object to calculate the average.



```
PS C:\> 1..10 | foreach {
>>  Measure-Command -expression $sb
>>  #pause a moment in case of caching effects
>>  start-sleep -Milliseconds 1500
>> } | Measure-Object TotalMilliseconds -Average
>>


Count    : 10
Average  : 0.23862
Sum      :
Maximum  :
Minimum  :
Property : TotalMilliseconds
SumKB    : 0
SumMB    : 0
SumGB    : 0
```

The Measure-Command results are being piped to Measure-Object. (Image Credit: Jeff Hicks)

Changing the scriptblock to use the other code I get this result.



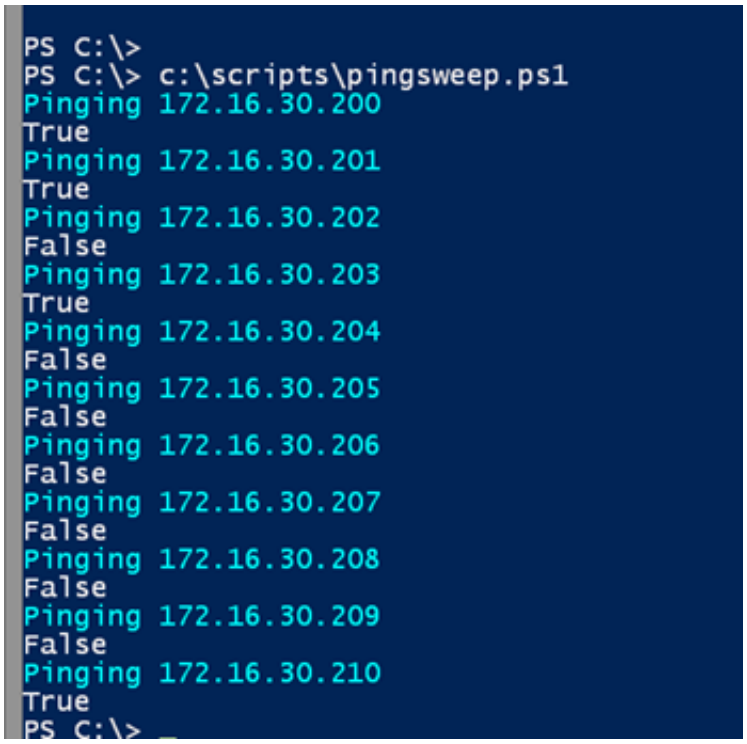
```
Count    : 10
Average  : 2.27657
Sum      :
Maximum  :
Minimum  :
Property : TotalMilliseconds
SumKB    : 0
SumMB    : 0
SumGB    : 0
```

Ok. I'll go with the first option. At this point I can start scripting something to test my logic ar
advanced function.

```
$subnet = "172.16.30.0"
$start = 200
$end = 210
$ping = 1
while ($start -le $end) {
$IP = "172.16.30.$start"
Write-Host "Pinging $IP" -ForegroundColor Cyan
Test-Connection -ComputerName $IP -count 1 -Quiet
$start++
}
```

Sponsored

I've added a Write-Host line to see what IP address is being tested and to verify my code tha
script from the console:



```
PS C:\>
PS C:\> c:\scripts\pingsweep.ps1
Pinging 172.16.30.200
True
Pinging 172.16.30.201
True
Pinging 172.16.30.202
False
Pinging 172.16.30.203
True
Pinging 172.16.30.204
False
Pinging 172.16.30.205
False
Pinging 172.16.30.206
False
Pinging 172.16.30.207
False
Pinging 172.16.30.208
False
Pinging 172.16.30.209
False
Pinging 172.16.30.210
True
PS C:\> _
```

Results from pingsweep.ps1 (Image Credit: Jeff Hicks)

Success! Eventually the variables in my script will most likely become function parameters, b
I have a rudimentary PowerShell script that pings a range of IP addresses and tells me if the
we are far from finished, so I hope you'll watch for the next article and come back for the ne

Tagged with **Advanced**, **Editor's Pick**, **PowerShell Ping Sweep Tool**, **Scripting**

1 Comment The Petri It Knowledgebase

Recommend Share

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

brokensyntax • 6 months ago

My simple modification to this.
Assuming you want to know each row of the sweep.
Test-Connection -Various parameters becomes $testResult = Test-Connect -various par
Followed directly by Write-Host "$a, $testResult" or ("$start, $testResult" dependent on
$a++ (or $start++)

This will make your IP output and your True/False declaration appear on the same line.

1 • Reply • Share ›

Subscribe Add Disqus to your siteAdd DisqusAdd Privacy