LP-TP3-POWERSHELL – ECUE31 Administration Système

Lundi 8 mars 2021

Amine ABDOUL-AZID

Martial SENE

Kavirajan SARAVANANE

Partie 1 : maquette

script\_1.PS1 :

On commence par l’autorisation de scripts PowerShell

Set-ExecutionPolicy Unrestricted

#auto exécution de scripts powershell

#1 partie maquette

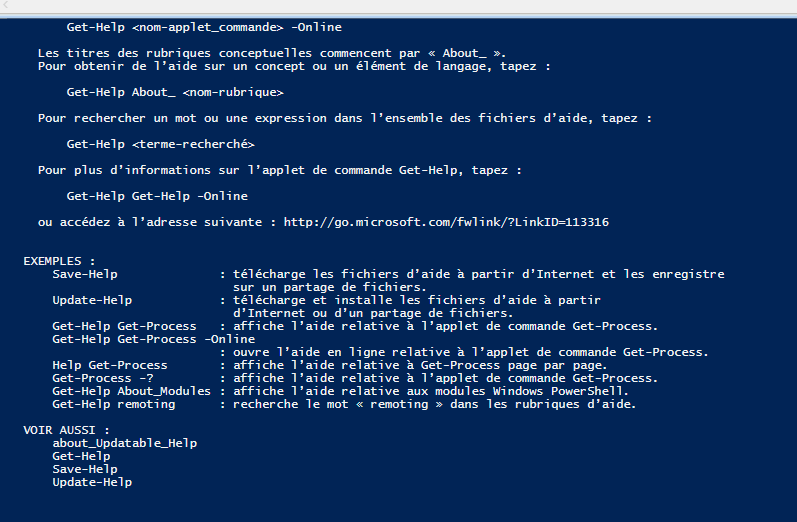
Partie 2 découverte :

script\_2.PS1 :

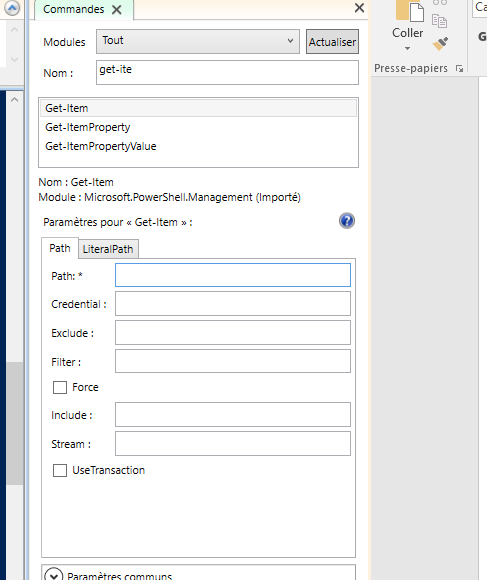
CTRL+ESPACE comme Intellisense/les autres IDE(Eclipse) donne un aperçu de la tabulation.

La commande help est l’équivalent du man sous Linux

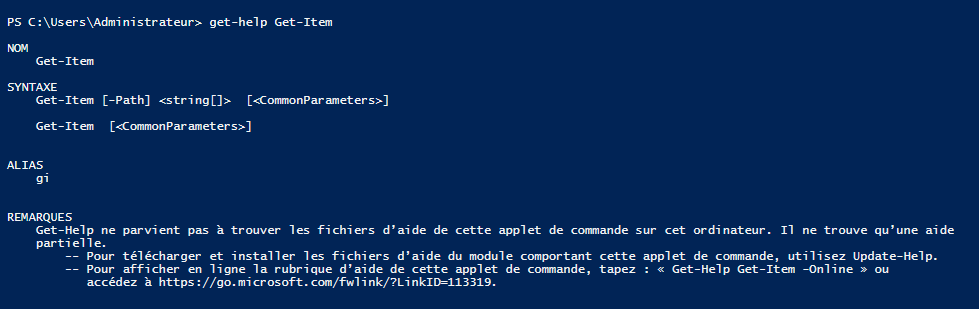
Get-Help <nom-applet\_commande>

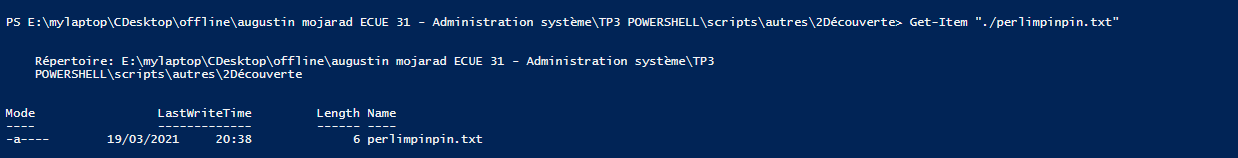


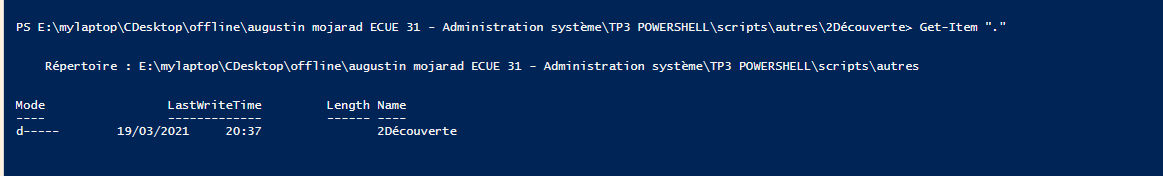
Par exemple Get-ITEM : ces paramètres sont aussi listés dans la partie commande



Avec get-help





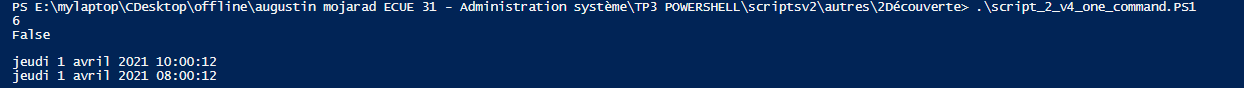


Réponses :

$command\_get\_item\_perlimpinpin= Get-Item "./perlimpinpin.txt"

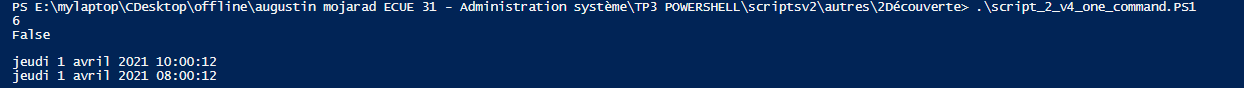
***# Taille du fichier en octets***

$command\_get\_item\_perlimpinpin.Length



***#Si le fichier est en lecture seule***

$command\_get\_item\_perlimpinpin.IsReadOnly

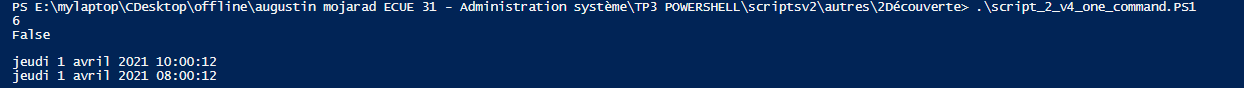


***#  L'heure du dernier accès au fichier***

$command\_get\_item\_perlimpinpin.LastAccessTime

*# ou*

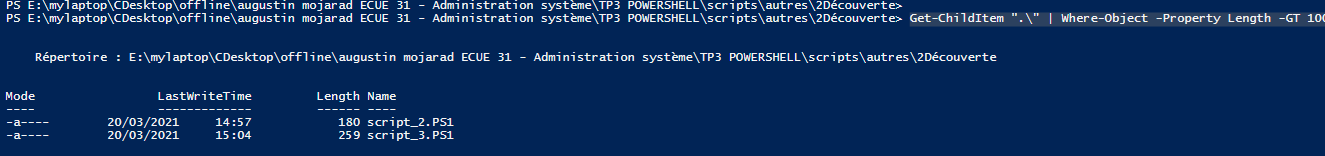
$command\_get\_item\_perlimpinpin.LastAccessTimeUtc



Partie 2 découverte :

script\_3.PS1 :

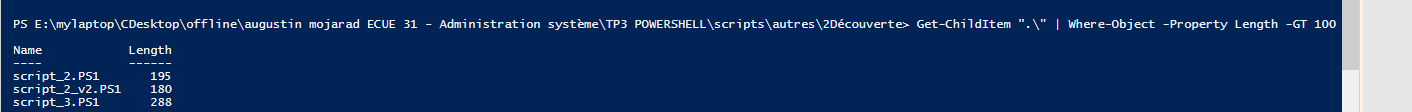
Get-ChildItem ".\" | Where-Object -Property Length -GT 100



Partie 2 découverte :

script\_4.PS1 :

Get-ChildItem ".\" | Where-Object -Property Length -GT 100 | Select-Object Name, Length



Partie 2 découverte :

script\_5.PS1 :

$ifIndex\_value=Get-NetIPAddress -InterfaceAlias "Ethernet" -AddressFamily "IPv4" | Select-Object -Property ifIndex | ForEach-Object -MemberName ifIndex

echo "L'index de la carte Ethernet est : $ifIndex\_value"

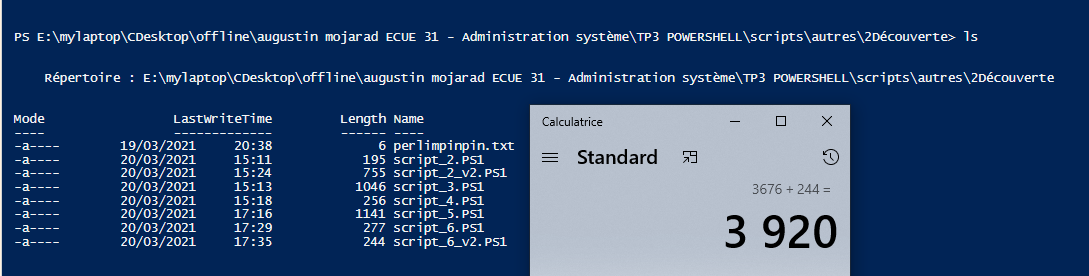


***Notes :*** *L’index vaut 11 sur le Windows Server.*

Partie 2 découverte :

script\_6.PS1 :

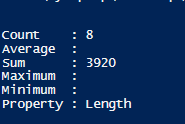
Résultat prévu : 3920 octets



Résultats obtenus :

Sous forme complète

PS E:\mylaptop\CDesktop\offline\augustin mojarad ECUE 31 - Administration système\TP3 POWERSHELL\scripts\autres\2Découverte> Get-ChildItem ".\" | measure-object -Property length -Sum



Filtrer encore mieux sur une ligne :

Get-ChildItem ".\" | measure-object -Property length -Sum | Select-Object -Property Sum | ForEach-Object -MemberName Sum

3916

Partie 3 Statistiques de RTT :

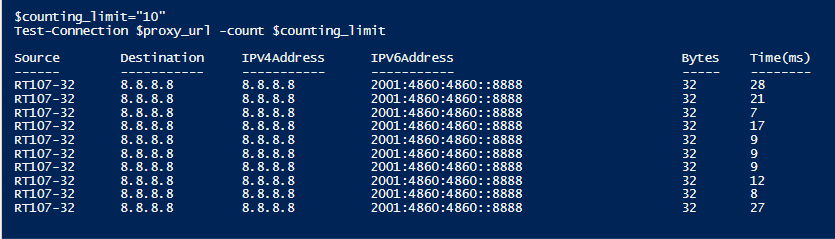
script\_7.PS1 :

$proxy\_url="proxya.u-pec.fr"

$proxy\_url="8.8.8.8"

$counting\_limit="10"

Test-Connection $proxy\_url -count $counting\_limit



Partie 3 Statistiques\_RTT :

script\_8.PS1 :

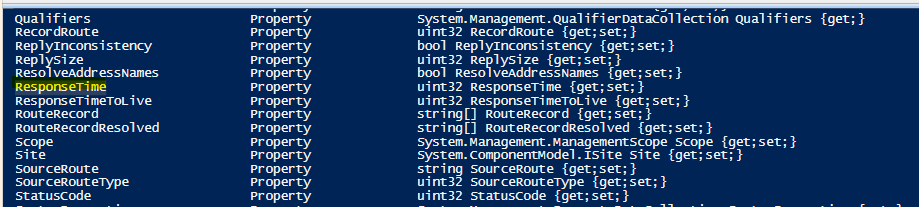
La colonne ResponseTime donne le temps d’un RTT/d’aller-retour de chaque test de connexion.

$proxy\_url="proxya.u-pec.fr"

$proxy\_url="8.8.8.8"

$counting\_limit="10"

Test-Connection $proxy\_url -count $counting\_limit | Get-Member -View All



Ou juste Avec Get-Member :

Test-connection 8.8.8.8 |Get-Member

[Capturing the PingReplyDetails (RTT) value from Test-NetConnection : PowerShell (reddit.com)](https://www.reddit.com/r/PowerShell/comments/732b1m/capturing_the_pingreplydetails_rtt_value_from/).

[Test-NetConnection vs. Test-Connection - Testing a network connection with PowerShell | 4sysops](https://4sysops.com/archives/test-netconnection-vs-test-connection-testing-a-network-connection-with-powershell/)

Partie 3 Statistiques\_RTT :

**script\_9\_v5\_testconnection\_debug.PS1** :

$proxy\_url="proxya.u-pec.fr"

$proxy\_url="8.8.8.8"

$counting\_limit="10"

$command = (Test-Connection $proxy\_url -count $counting\_limit) | measure-object -Property ResponseTime -Average

$average\_RTT=$command.Average

Write-Host "For" $counting\_limit "pings, the average RTT is of" $average\_RTT

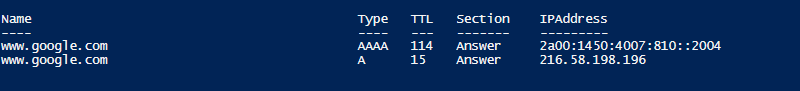


Partie 4 Résolution DNS :

script\_10.PS1 :

$url="www.google.com"

Resolve-DnsName -Name $url



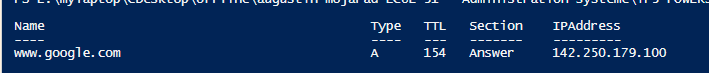
**La propriété qui donne le type d'enregistrement est le champ type**

partie 4 Résolution DNS :

script\_11.PS1 :

$url="www.google.com"

Resolve-DnsName -Name $url | Where-Object -Property "Type" -eq "A"



Partie 4 Résolution DNS :

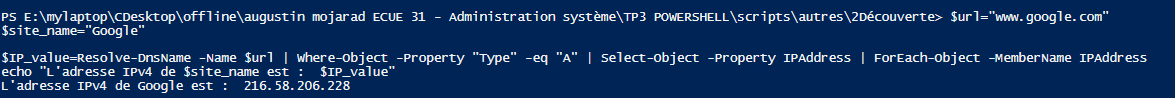
script\_12.PS1 :

$url="www.google.com"

$site\_name="Google"

$IP\_value=Resolve-DnsName -Name $url | Where-Object -Property "Type" -eq "A" | Select-Object -Property IPAddress | ForEach-Object -MemberName IPAddress

echo "L'adresse IPv4 de $site\_name est :  $IP\_value"



Partie 4 Résolution DNS : fichier lié hosts.txt

script\_13.PS1 : dns.ps1

$hostfile="./hosts.txt"

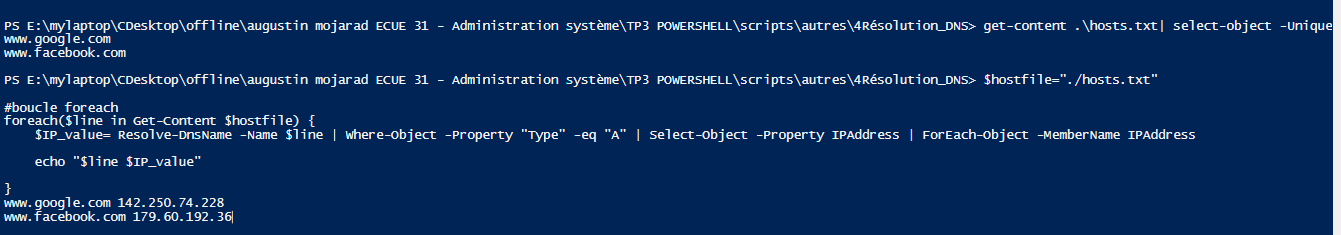
#boucle foreach

foreach($line in Get-Content $hostfile) {

    $IP\_value= Resolve-DnsName -Name $line | Where-Object -Property "Type" -eq "A" | Select-Object -Property IPAddress | ForEach-Object -MemberName IPAddress

    Write-Host "$line $IP\_value"

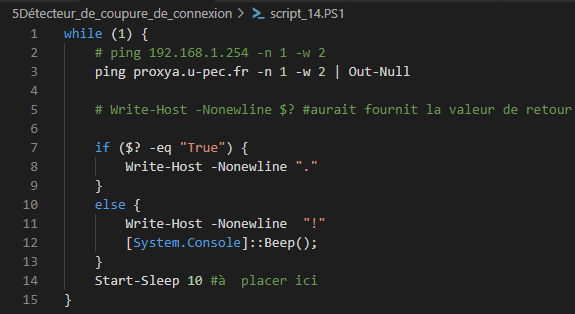
}



Partie 5. Détecteur de coupure de connexion :

script\_14.PS1 :

Partie sauté



#-n 1 maximum 1 requete au lieu de -c

#-w 2 sec en timeout

# > /dev/null | Out-Null

# > /dev/null ;  > NUL on cmd

# True au lieu de $? -eq 0



Partie 5. Détecteur de coupure de connexion :

script\_15.PS1 :

$route=$(Get-NetRoute | Where-Object -Property "RouteMetric" -eq "0" | Select-Object -Property NextHop | ForEach-Object -MemberName NextHop)

$ip=$args[0]

if  (!$route) {

    echo "Ce PC n'a pas de passerelle par défaut !"

    exit 1

}

echo "L'adresse passerelle est :" $route

if (!$args[0]) {

    while (1){

        ping $route -n 1 -w 2 | Out-Null

        if ( $? -eq 0) {

            Write-Host -Nonewline "!"

        }

        else{

            Write-Host -Nonewline "."

        }

        Start-Sleep 10

    }

}

echo "L'adresse IP en argument est :" $ip

while (1){

    ping $ip -n 1 -w 2 | Out-Null

    if ($? -eq 0) {

            Write-Host -Nonewline "!"

        }

    else{

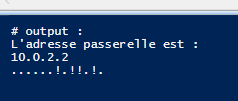
         Write-Host -Nonewline "."

    }

        Start-Sleep 10

    }

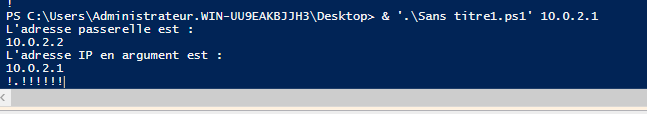
* Cas 1 : Avec le ping vers la passerelle par défaut :



* Cas 2 : Sans passerelle par défaut :



* Cas 3 : Avec adresse IP en argument



Partie 6. Temps de réponse de sites Web :

script\_16.PS1 :

$hostfile="./hosts\_url.txt"

#boucle foreach

foreach($line in Get-Content $hostfile) {

    $URL\_content\_value= Invoke-WebRequest -Uri $line -UseBasicParsing

    echo "$line output : "

    echo "$URL\_content\_value"

}

Avec qu’un seul URL car beaucoup d’URL, le résultat est trop long et illisible :

PS E:\mylaptop\CDesktop\offline\augustin mojarad ECUE 31 - Administration système\TP3 POWERSHELL\scripts\autres\4Résolution\_DNS> Invoke-WebRequest -Uri "http://www.google.com" -UseBasicParsing

StatusCode : 200

StatusDescription : OK

Content : <!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="fr"><head><meta content="text/html; charset=UTF-8"

http-equiv="Content-Type"><meta content="/logos/doodles/2021/spring-2021...

RawContent : HTTP/1.1 200 OK

X-XSS-Protection: 0

X-Frame-Options: SAMEORIGIN

Vary: Accept-Encoding

Transfer-Encoding: chunked

Accept-Ranges: none

Cache-Control: private, max-age=0

Content-Type: text/html; c...

Forms :

Headers : {[X-XSS-Protection, 0], [X-Frame-Options, SAMEORIGIN], [Vary, Accept-Encoding], [Transfer-Encoding, chunked]...}

Images : {}

InputFields : {}

Links : {@{outerHTML=<a class="gbzt gbz0l gbp1" id=gb\_1 href="https://www.google.fr/webhp?tab=ww"><span class=gbtb2></span><span

class=gbts>Recherche</span></a>; tagName=A; class=gbzt gbz0l gbp1; id=gb\_1; href=https://www.google.fr/webhp?tab=ww}, @{outerHTML=<a class=gbzt id=gb\_2

href="http://www.google.fr/imghp?hl=fr&tab=wi"><span class=gbtb2></span><span class=gbts>Images</span></a>; tagName=A; class=gbzt; id=gb\_2;

href=http://www.google.fr/imghp?hl=fr&tab=wi}, @{outerHTML=<a class=gbzt id=gb\_8 href="http://maps.google.fr/maps?hl=fr&tab=wl"><span

class=gbtb2></span><span class=gbts>Maps</span></a>; tagName=A; class=gbzt; id=gb\_8; href=http://maps.google.fr/maps?hl=fr&tab=wl}, @{outerHTML=<a

class=gbzt id=gb\_78 href="https://play.google.com/?hl=fr&tab=w8"><span class=gbtb2></span><span class=gbts>Play</span></a>; tagName=A; class=gbzt;

id=gb\_78; href=https://play.google.com/?hl=fr&tab=w8}...}

ParsedHtml :

RawContentLength : 49178

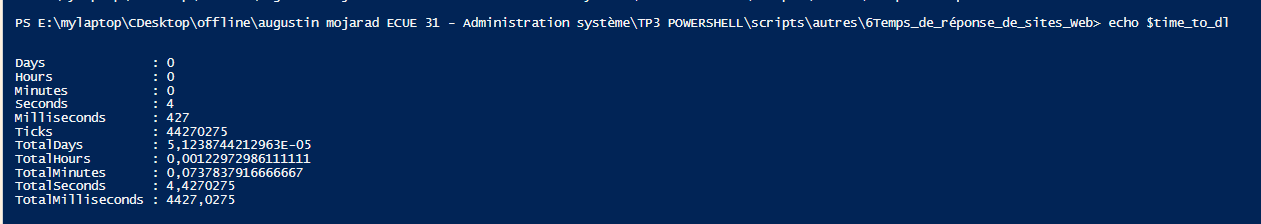
Partie 6 Temps de réponse de sites Web :

script\_17.PS1 :

* Sans formatage en une ligne de commande

PS E:\mylaptop\CDesktop\offline\augustin mojarad ECUE 31 - Administration système\TP3 POWERSHELL\scripts\autres\6Temps\_de\_réponse\_de\_sites\_Web> $host\_url="http://quelleheureestilenjoy.com/00.mp4"

Measure-Command {Invoke-WebRequest -Uri $host\_url -UseBasicParsing }

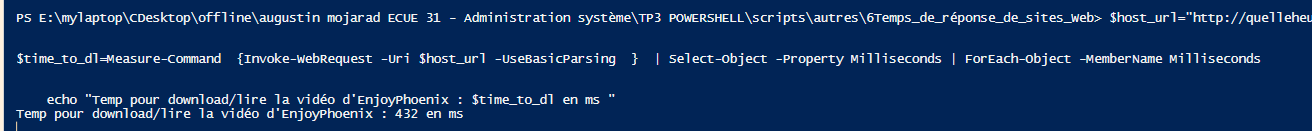


* Avec un meilleur formatage :

$host\_url="http://quelleheureestilenjoy.com/00.mp4"

$time\_to\_dl=Measure-Command  {Invoke-WebRequest -Uri $host\_url -UseBasicParsing  }  | Select-Object -Property Milliseconds | ForEach-Object -MemberName Milliseconds

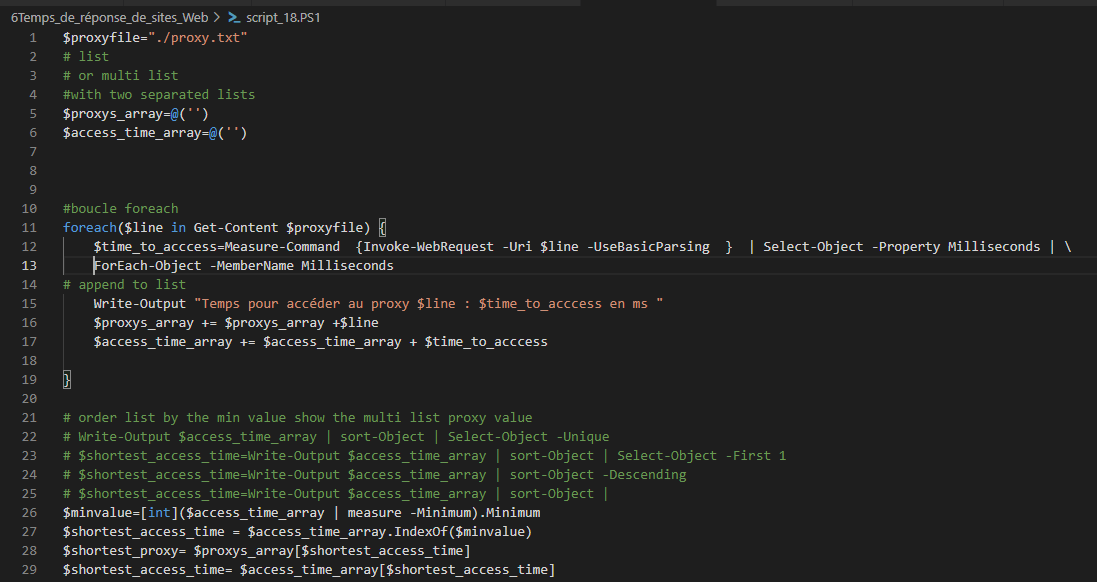
echo "Temp pour download/lire la vidéo d'EnjoyPhoenix : $time\_to\_dl en ms "

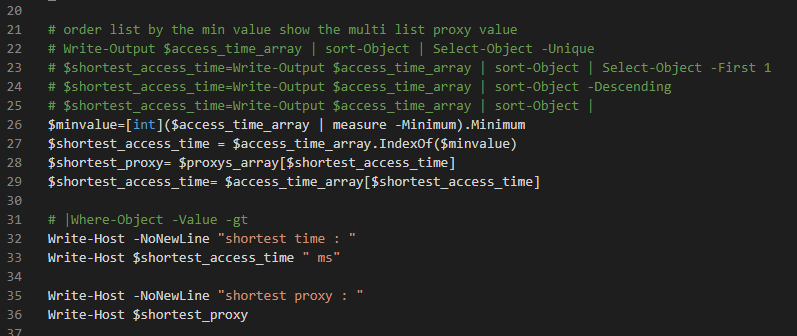


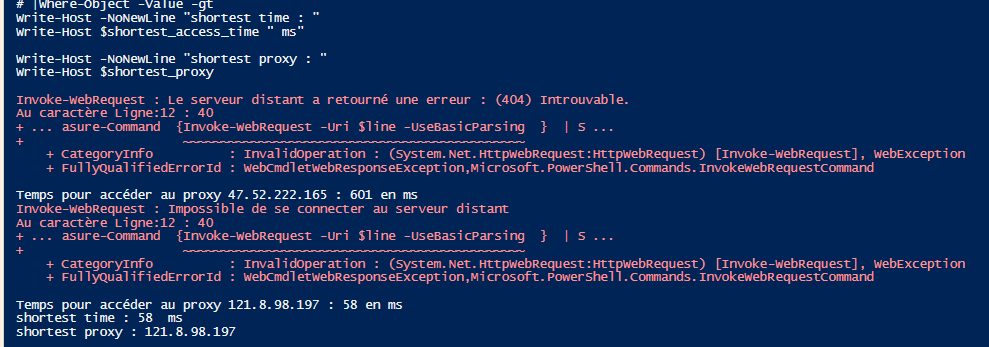
[3 ways to measure your Powershell script's speed | Pluralsight](https://www.pluralsight.com/blog/tutorials/measure-powershell-scripts-speed)

Partie 6 Temps de réponse de sites Web :

script\_18.PS1 :







***Notes :*** *Fastest\_proxy au lieu de shortest proxy*

Partie 7. Cartes réseau :

script\_19.PS1 : Fichier lié *nicdetector.ps1* : (network interface card detector)

$command=Get-NetAdapter -Physical| Select-Object -Property Name, Status,ifindex | Where-Object -Property "Status" -eq "Up"

$command\_count\_value = $command | Measure-Object -Property ifIndex -Sum | ForEach-Object -MemberName Count

*# case 2*

if ($command\_count\_value -lt 1){

    Write-Host "Pas d'interface branchée ! Vous devez en câbler au moins une."

}

*# case 3*

elseif ($command\_count\_value  -gt 1){

    Write-Host "Il y a plusieurs interfaces branchées ! Une seule carte doit être câblée."

}

*# case 1*

elseif ($command | Where-Object -Property "Status" -eq "Up" ){

    $name\_card=$command | Select-Object -Property Name | ForEach-Object -MemberName Name

    Write-Host "Interface branchée : " $name\_card

}

* Cas 1 :



* Cas 2 :



* Cas 3 :



**Notes :** *Les caractères accentués sont mal interprétés sur la VM Windows Server ce qui n’est pas le cas, des autres PC Windows 10 (Consumer)*

***Github :***

[amineAUPEC/TP3-POWERSHELL: TP3-POWERSHELL IUT CRETEIL-VITRY LICENCE PRO ASUR (github.com)](https://github.com/amineAUPEC/TP3-POWERSHELL)

<https://github.com/amineAUPEC/TP3-POWERSHELL>

# Annexes :

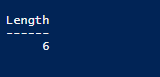
Partie 2 découverte :

script\_2.PS1 :

**Avec Select-object** :

# Taille du fichier en octets

Get-Item "./perlimpinpin.txt" | Select-Object -Property Length



Avec Where-Object

#Si le fichier est en lecture seule

Get-Item "./perlimpinpin.txt" | Where-Object IsReadOnly

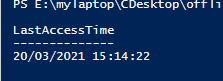
Aucune sortie :



**Avec Select-object** :

#  L'heure du dernier accès au fichier

Get-Item "./perlimpinpin.txt" | Select-Object -Property LastAccessTime



Partie 3 Statistiques\_RTT :

**Avec Test-Net-Connection** :

script\_7\_v2.PS1 :

$proxy\_url="proxya.u-pec.fr"

$proxy\_url="8.8.8.8"

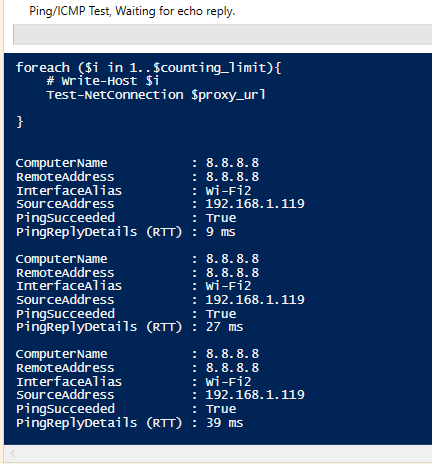
$counting\_limit="10"

foreach ($i in 1..$counting\_limit){

*# Write-Host $i*

    Test-NetConnection $proxy\_url

}



Partie 3 Statistiques\_RTT :

**Avec Test-Net-Connection** :

script\_8\_v2.PS1 :

La colonne PingReplyDetails puis RoundtripTime RTT donne le temps d’un aller-retour de chaque test de connexion.

[Test-NetConnection (NetTCPIP) | Microsoft Docs](https://docs.microsoft.com/en-us/powershell/module/nettcpip/test-netconnection?view=windowsserver2019-ps)

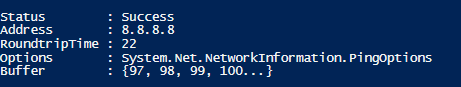
$proxy\_url="proxya.u-pec.fr"

$proxy\_url="8.8.8.8"

(Test-NetConnection $proxy\_url).pingreplydetails

*# (tnc $proxy\_url).pingreplydetails*

*# tnc $proxy\_url | Get-Member*



Avec Get-Member :

tnc 8.8.8.8 - |Get-Member

[Capturing the PingReplyDetails (RTT) value from Test-NetConnection : PowerShell (reddit.com)](https://www.reddit.com/r/PowerShell/comments/732b1m/capturing_the_pingreplydetails_rtt_value_from/).

[Test-NetConnection vs. Test-Connection - Testing a network connection with PowerShell | 4sysops](https://4sysops.com/archives/test-netconnection-vs-test-connection-testing-a-network-connection-with-powershell/)

Partie 3 Statistiques\_RTT :

**Avec Test-Net-Connection** :

script\_9\_v2.PS1 :

$proxy\_url="proxya.u-pec.fr"

$proxy\_url="8.8.8.8"

$counting\_limit="10"

$command = (Test-NetConnection $proxy\_url).PingReplyDetails.RoundTripTime

$counting\_RTT="0"

foreach ($i in 1..$counting\_limit){

*# Write-Host $i*

*# Write-Host "command\_value" $command*

*# Write-Host "Previous value " $counting\_RTT*

    $counting\_RTT= $counting\_RTT + $command -as [**int**]

*# Write-Host $counting\_RTT*

*# (Test-NetConnection $proxy\_url).PingReplyDetails.RoundTripTime | measure-object -Property length -Sum*

}

    Write-Host $counting\_RTT

