Formation Powershell & Hyper-V - sys 30 – T.p. maquette

Michel Cabaré / www.cabare.net / michel@cabare.net

Powershell Local – Direct – Remote - Hyper-V
-sys 26 - sys 27- sys 30 – T.p V1-1 - Janvier 2022

ICPF
Certification **PRO**
Formateur Expert

CABARÉ FORMATION

Microsoft Partner

TABLE DES MATIÈRES

POWERSHELL 3

POWERSHELL -CMD: 3
Prompt powershell 3

COMMANDE DE BASE 4

CLS 4
GET-COMMAND 4
GET-SERVICE 4
GET-VM 4
HOSTNAME 4

COMPLETION - HELP 5

COMPLETION TABULATION 5
COMPLETION CTRL + ESPACE 5
GET-HELP - AIDE 6

PIPE - | 7

PIPE – PASSER LE RESULTAT D'UNE COMMANDE A UNE AUTRE 7

PIPE - | 7

FOR ET IF THEN 7

POWERSHELL ISE 8

PASSAGE AU SCRIPT – POWERSHELL ISE 8
UTILISATION BASIQUE DE POWERSHELL ISE 9

SESSION POWERSHELL DIRECT (VM) 10

POWERSHELL DIRECT – VM ET HYPER-V 10
MODES DE CONNEXION - SESSION INTERACTIVE - COMMANDE DIRECTE / SCRIPT 10
Session interactive 10
Commande directe / script 10
SESSION INTERACTIVE : ENTER-PSSESSION 11
Prompt powershell [nom vm] 11
Exit 12
COMMANDE DIRECTE : INVOKE-COMMAND -SCRIPTBLOCK 12
FOURNIR UNE AUTHENTIFICATION : CREDENTIAL 12
SAISIE DES LOGIN / MDP 13

SESSION DISTANTE WINRM (RESEAU) 15

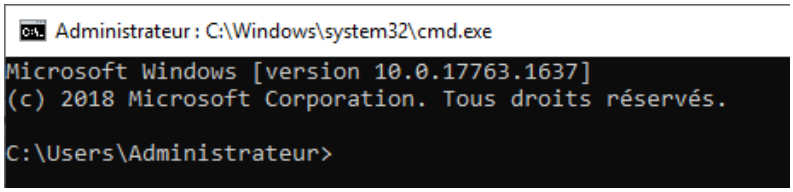
DEMARRER WINRM WINDOWS REMOTE MANAGEMENT 15
UTILITAIRE WINRM OU ENABLE-PSREMOTING 16
TEST WINDOWS REMOTE MANAGEMENT 16
NEW-PSSESSION OU SESSION MANUELLE DISTANTE 17
FOURNIR UNE AUTHENTIFICATION : CREDENTIAL 20
INVOKE-COMMAND POUR 1 OU 2 COMMANDES A DISTANCE 20
SCRIPT A EXECUTER A DISTANCE 20

Powershell -cmd:

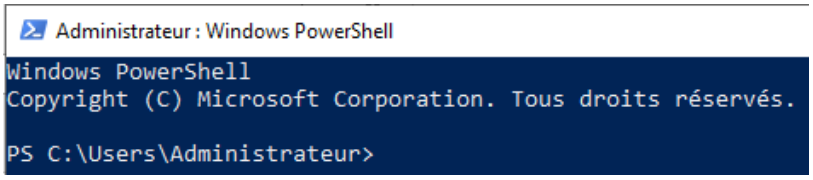
Powershell permet d'automatiser les actions sur les systèmes Windows, voir permettre des actions non possibles via l'interface graphique

On utilisera aussi **Powershell** lorsque l'on voudra être sûr de passer/effectuer la même commande / action sans se tromper, ou sur plusieurs machines

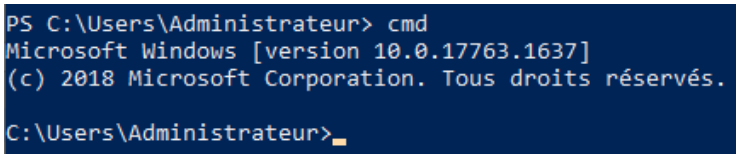
Ne pas confondre l'invite de commande **cmd.exe**



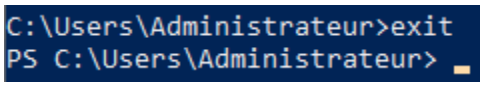
Avec **Powershell** (disponible en clic/droit menu démarrer)



N.B : On peut demander en **powershell** de «passer» en invite de commande avec la commande **cmd**,

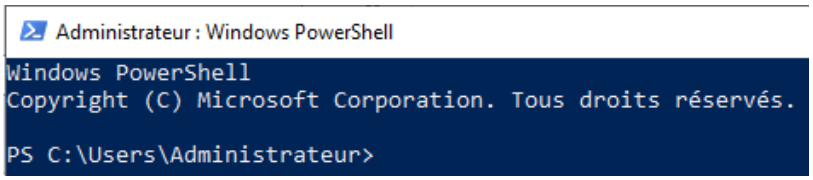


on en sortira par la commande **Exit** ou **Powershell**

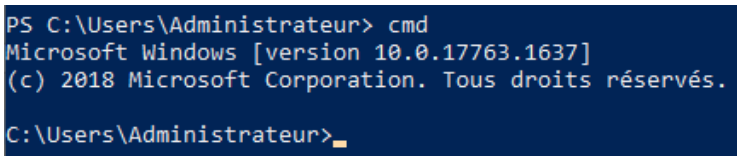


Prompt powershell

Il est intéressant de noter le prompt **Powerschell** sur la gauche pour savoir dans quel environnement on se trouve (et plus tard sur quelle machine on se trouve…)



PS indique que nous sommes en **Powershell**



L'absence de prompt indique que nous sommes en **invite de commande**

COMMANDE DE BASE

CLS

Get-command

Get-service

Get-service

Get-service -name xxxx

get-service | Where-Object {$_.status -eq 'stopped'}

Start-service / Stop-service

Get-VM

Liste toutes les VM sur un hyper-Viseur

Hostname

Renvoi le nom de la machine windows dans laquelle le powershell s'execute

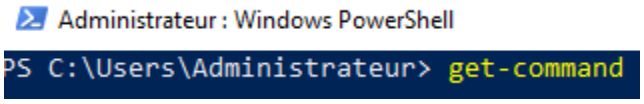
```
[serveur-2019] : PS C:\Users\Administrateur\Documents> hostname
UTILISA-3SSDP86
```

COMPLETION - HELP

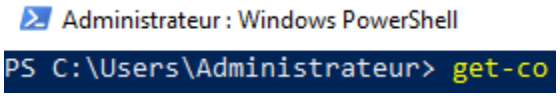
Complétion tabulation

Basée sur la touche **tabulation**

Au lieu taper en extenso **get-command** comme ci-dessous

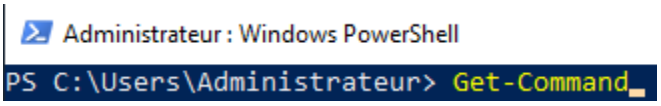


On peut taper juste



et Touche **Tabulation**

Pour obtenir

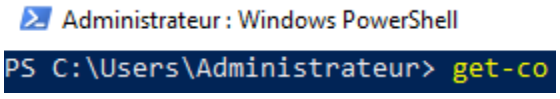


Depuis la version 5 et le module supplémentaire **psreadline** il est possible en plus de la complétion classique de demander les paramètres possible avec la combinaison de touche **CTRL + Espace**

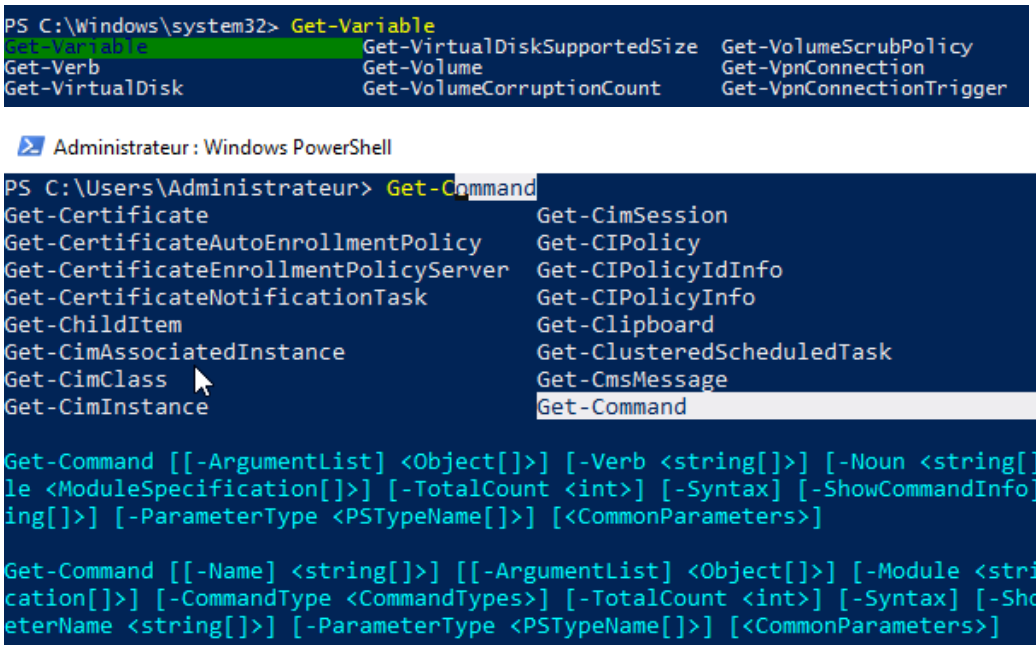
Complétion CTRL + espace

Basée sur la combinaison de touche **CTRL + espace**

On peut taper juste



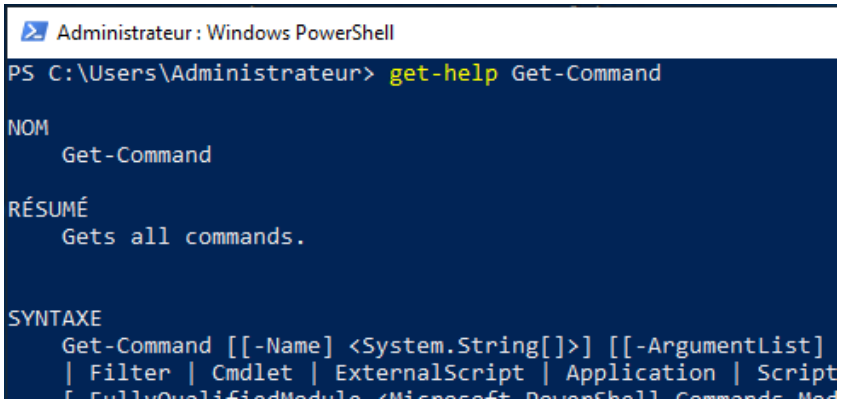
+ **CTRL +espace** donnera



Get-help - Aide

L'aide est toujours disponible via **get-help.**

Sur les commandes xxxx on pourra executer **get-help xxxx**.



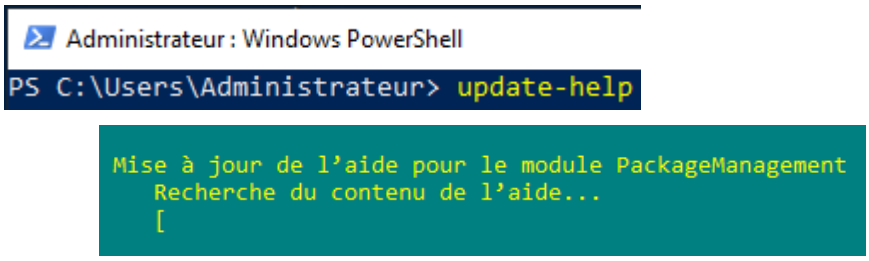
Par exemple

```
get-help Get-Command
```

On peut avoir que les exemples via le paramètre **-examples**

```
PS C:\Users\Administrateur> Get-Help Get-Command -examples
```

Si on a une connexion , Il peut être necessaire de mettre à jour l'aide avec **update-help**



PIPE - |

Pipe – passer le résultat d'une commande à une autre

Si on fait un get-command, le resultat est enorme… on peut présenter le résultant en le pipant sur une commande de présentation de donnée, par exemple essayer

Get-Command | Out-GridView

Get-process

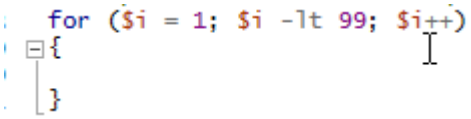
Puis Get-process |

PIPE - |

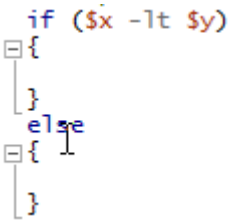
For et If then

Si on demande **CTRL+J** une liste de construction apprait, il n'y a plus qu'a choisir :

For



If else

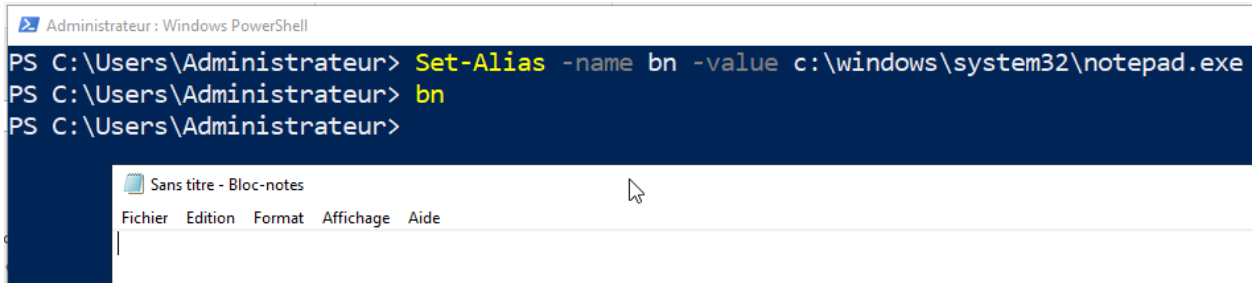


Passage au script – powershell ISE

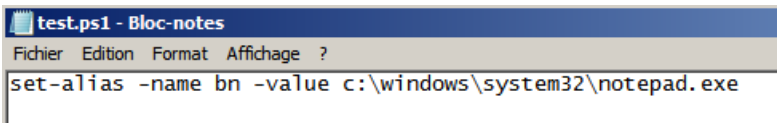
Si l'interpréteur de commandes **PowerShell** est pratique, lorsque l'on a passé 20 minutes à mettre au point sa ligne de commandes dans la console on ne souhaite pas tout réécrire la fois suivante.

Partons d'une commande qui permet de créer un alias ***bn*** lançant le bloc-note, telle que

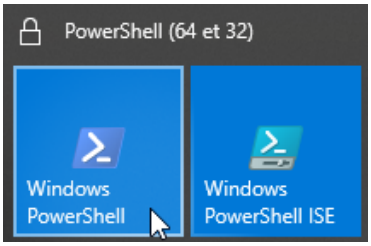
Set-Alias -name bn -value c:\windows\system32\notepad.exe



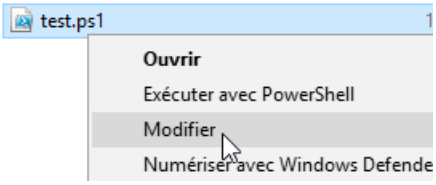
Pour faire un script, il suffit de placer sa ligne de commandes dans un éditeur de texte et de l'enregistrer avec l'extension **.ps1**



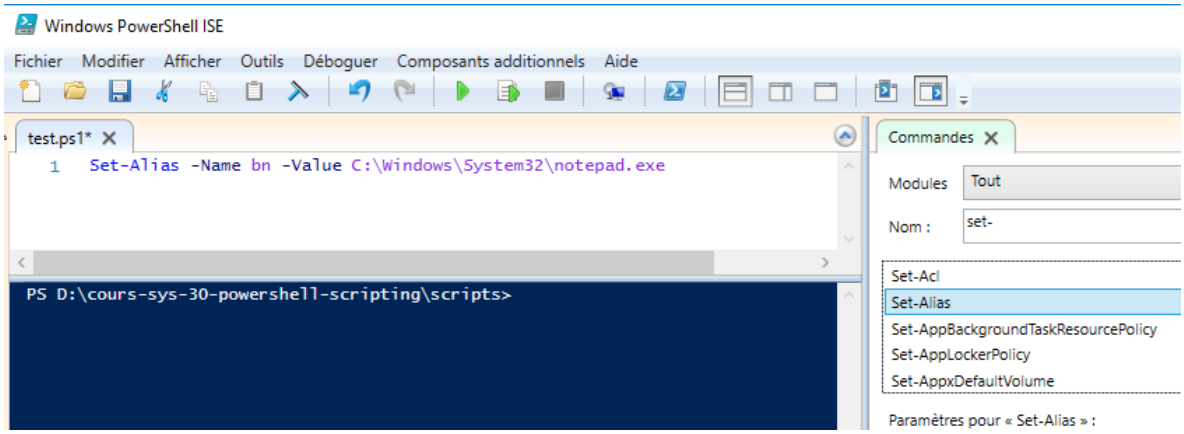
Depuis **powershell 2.0** une interface graphique dite **Powershell ISE** pour éditer les scripts est disponible,



Si on est sur un poste avec **Powershell ISE** installé, lorsque l'on demande de **modifier** un script **powershell**,

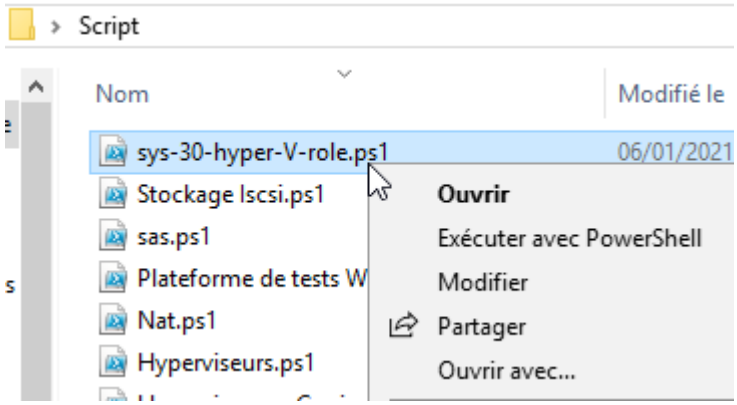


c'est **Powershell ISE** qui se lance



Utilisation basique de powershell ISE

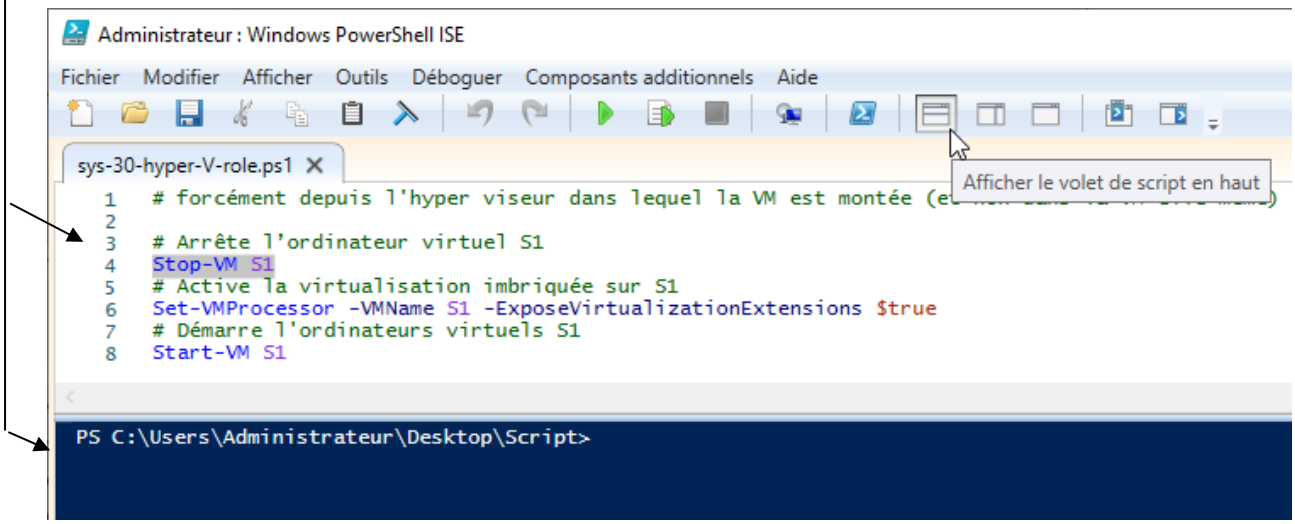
Si on a un script Poweshell, il est donc dans un fichier texte avec l'extension .ps1



Ouvrir : exécute le script dans sa totalité (à éviter)

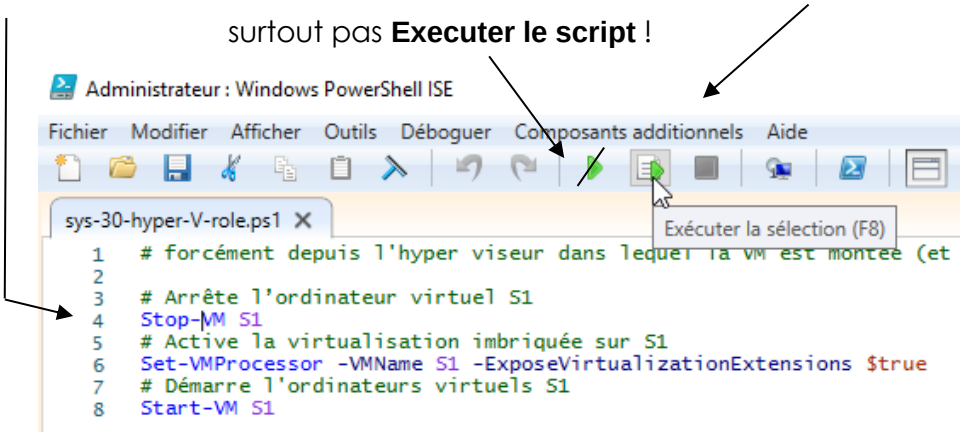
Modifier Ouvre le fichier dans **powershellISE**

On affiche le script en haut, et la fenêtre **powershell** en bas



On se place sur une ligne à exécuter, et on demande **Executer la sélection (F8)**,

surtout pas **Executer le script** !



SESSION POWERSHELL DIRECT (VM)

Powershell Direct – Vm et hyper-V

La nouvelle version d'Hyper-V apparue en même temps que Windows Server 2016 est accompagnée d'une fonctionnalité appelée "**PowerShell Direct**", celle-ci permet d'exécuter une commande sur une machine virtuelle en faisant abstraction de la couche réseau et des paramètres de gestion à distance.

Cela repose sur la technologie VMBUS (qui permettait de copier des fichiers dans une VM « directement depuis 2012)

Fini la configuration des pares-feux, les tests de démarrage du service **winRm** et autres paramétrages. Il n'est tout simplement pas nécessaire d'avoir une connexion réseau !

Les versions concernant l'OS de l'hôte physique et l'OS de l'hôte virtuel :

- OS Hyper-V : Windows 10 ou Windows Server 2016
- OS VM : Windows 10 ou Windows Server 2016

Les conditions techniques nécessaires :

- Le compte utilisateur avec lequel vous êtes connecté sur **l'Hyper-V** doit avoir les autorisations du groupe "**Administrateurs d'Hyper-V**"
- La **VM** machine virtuelle cible doit être démarrée et exécutée sur l'hôte **Hyper-V** sur lequel vous exécutez les commandes **PowerShell Direct**
- Des identifiants de connexion (login/password) de la machine virtuelle cible seront nécessaire pour s'authentifier

```
Enter-PSSession : Une erreur s'est produite et Windows PowerShell n'est pas en mesure de la gérer. Une session à distance
a peut-être pris fin.
```

Modes de connexion - session interactive - commande directe / script

Pour PowerShell Direct, il y a deux modes de connexion :

Session interactive

Il est possible d'ouvrir une session PowerShell à distance sur une VM par l'intermédiaire d'une session interactive via "Enter-PSSession" (ou New-PSSession pour créer une session et l'utiliser ensuite). Ce mode de fonctionnement implique qu'ensuite on se retrouve avec une console connectée sur la VM.

Ce n'est pas forcément utile si l'on souhaite seulement exécuter une commande.

Cette technique est aussi celle utilisée pour se connecter à une machine distance via Powershell – WinRM , mais dans le cadre de PowerShell Direct et de l'accès aux VMs, on utilisera des paramètres spécifiques (différents de ceux utilisés pour WinRm).

Commande directe / script

Si vous souhaitez seulement effectuer une action sans rentrer dans une session interactive, il faudra passer directement par le commandlet "Invoke-Command", avec là aussi des paramètres spécifiques. Ceci peut-être utile, par exemple, pour récupérer l'adresse IP de la machine virtuelle, récupérer la liste des fonctionnalités et rôles installés,

ou exécuter une action précise... Le tout sans vouloir aller jusqu'à ouvrir une console interactive sur la VM.

Session interactive : Enter-Pssession

Pour rentrer en mode interactif sur une VM, on utilise la commandlet **Enter-PSSession**

avec le paramètre "**VMName**" correspondant au nom de la VM

ou le paramètres "**VMId**" correspondant à l'identifiant unique de la VM

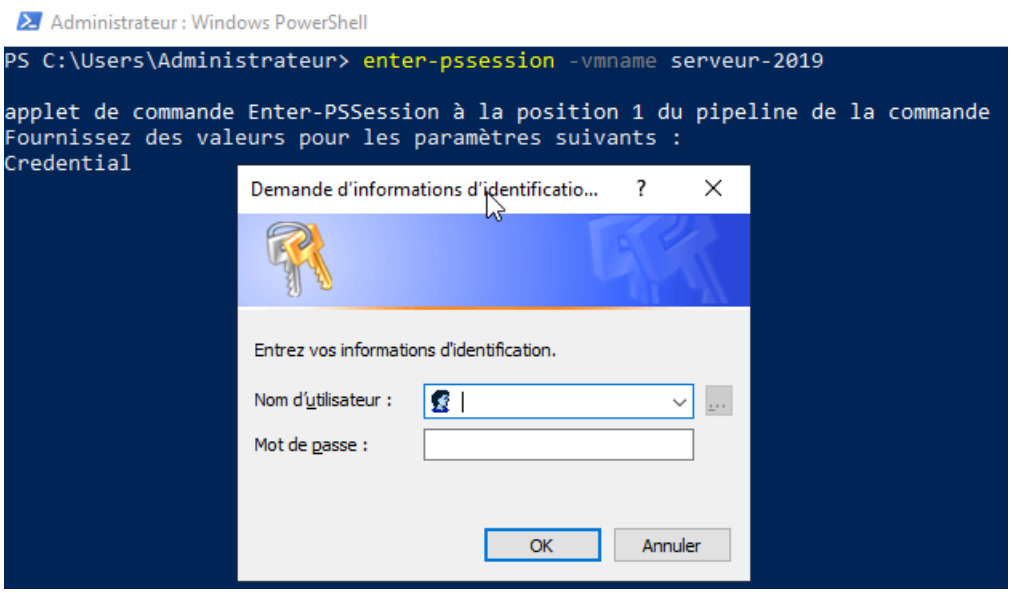
Ce qui donnerait, pour le nom de la VM :

Enter-PSSession -VMName <Nom-VM>

Et pour l'ID de la VM :

Enter-PSSession -VMId <ID-VM>

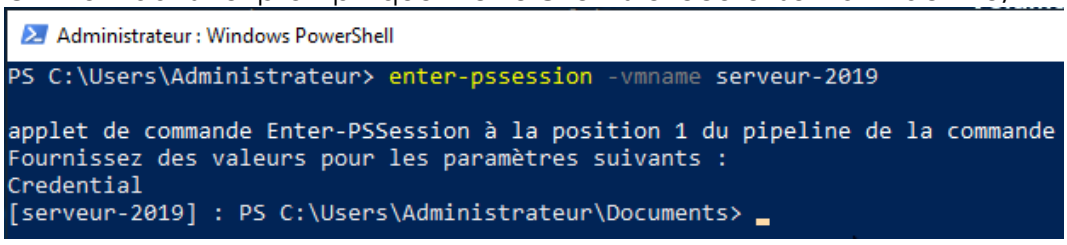
Il suffira ensuite de rentrer les identifiants pour se connecter à la VM, et le tour est joué !



Lorsque la session interactive sera ouverte sur la VM, vous pouvez interagir en PowerShell avec la machine virtuelle, comme si la console était ouverte en locale sur la VM.

Prompt powershell [nom vm]

On voit dans le prompt que **Powershell** s'exécute sur la machine/vm [serveur-2019]



N.B : Lorsque la **session** à la **VM** est établie, le **prompt** de la console **PowerShell** commence par le nom de la **VM**. (et non pas le nom de la machine windows/ OS)

On pourrait par exemple, lister les fonctionnalités et rôles installés :

Exit

Permet de sortie de la session

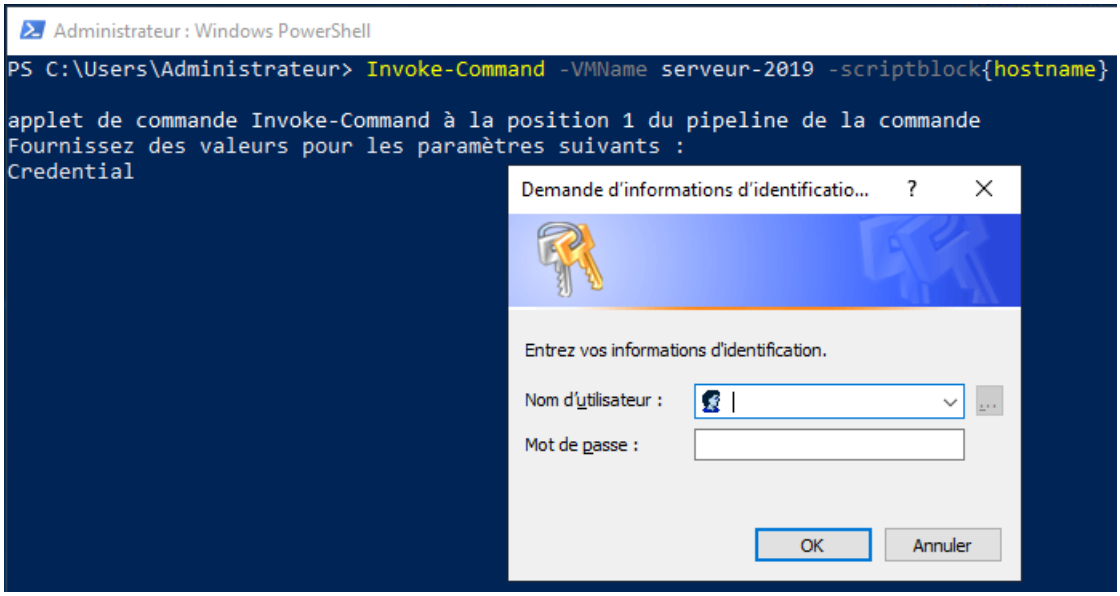
Commande directe : Invoke-Command -scriptblock

Si on veut juste passer une commande sur un poste, on utiliser la **Comandlet invoke-Command** avec la ou les commandes listées à l'interieur du paramètre **-ScriptBlock**

Ce qui donnerait, avec le nom de la VM :

Invoke-Command -VMName <Nom-VM>**-scriptblock{hostname}**

Invoke-Command -VMName <Nom-VM> **-scriptblock{hostname ; xxxxxxx}**



A la place d'une commande, on peut indiquer un fichier script, avec le paramètre suivant

Invoke-Command -VMName <Nom-VM**> -FilePath** C:\Scripts\Script.ps1

Fournir une authentification : Credential

Le paramètre -**Credential** va permettre de fournir une authentification a **Powershell direct**, de manière à ce qu'elle ne soit pas demandée de manière inter-active

Tout le monde dira qu'il ne faut pas le faire, mais c'est tellement pratique !

Ecriture abrégé

$password = ConvertTo-SecureString "pw" -AsPlainText -Force

$cred= New-Object System.Management.Automation.PSCredential ("administrateur", $password)

Ecriture plus, classique

$login = "cabare-intra.net\admin"

$password = "M0T2pass!" | Convertto-SecureString -AsPlainText –Force

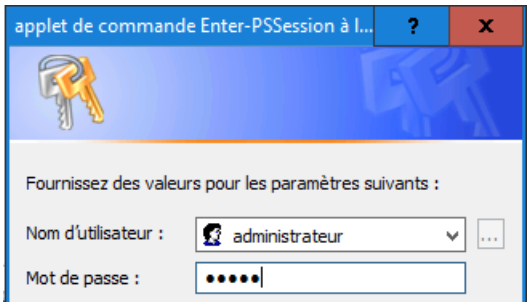
$id = New-Object System.Management.Automation.Pscredential -Argumentlist $login,$password

New-PSSession -ComputerName poste11 -Credential:$id

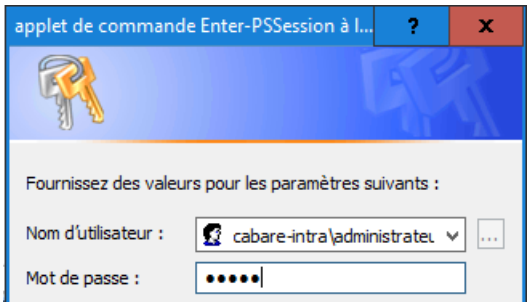
```
# POWERSHELL DIRECT avec Credential
$password = ConvertTo-SecureString "Local2019" -AsPlainText -Force
$identifiants= New-Object System.Management.Automation.PSCredential ("administrateur", $password )
'


# POWERSHELL DIRECT avec Credential autre ecriture moins condensée
$login = "administrateur"
$password = "Local2019" | Convertto-SecureString -AsPlainText -Force
$identifiants = New-Object System.Management.Automation.Pscredential -Argumentlist $login,$password
```

Saisie des Login / Mdp

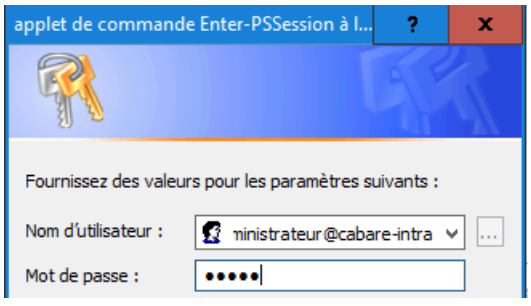
En **workgroup** : Nom d'utilisateur + Mot de passe



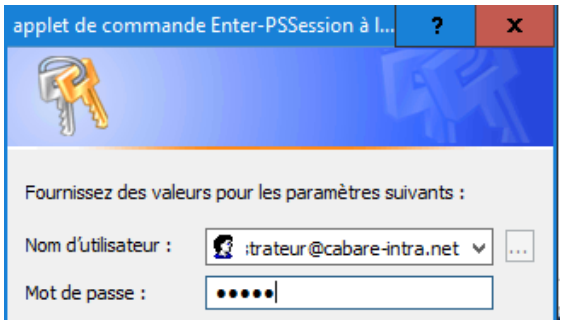
En **domaine :** **Domaine ** Nom d'utilisateur + Mot de passe



Nom d'utilisateur@**Domaine** + Mot de passe



Nom d'utilisateur@**Domaine.suf** + Mot de passe




```
Enter-PSSession : Les informations d'identification ne sont pas valides.
```

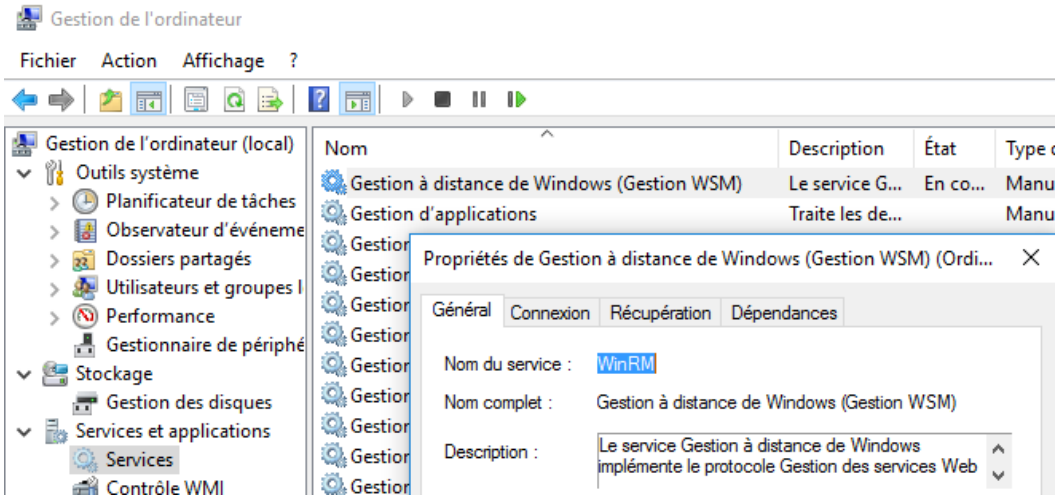
SESSION DISTANTE WINRM (RESEAU)

Démarrer WinRm Windows Remote Management

Une des grosses nouveautés à partir de PowerShell 2.0 tient en sa capacité à exécuter des commandes à distance. Il devient alors très pratique d'administrer ses machines sans avoir à s'y connecter.

Pour pouvoir utiliser PowerShell à distance, 2 prérequis :

- Il faut avoir une connexion réseau opérationnelle
- il vous faut configurer le service « **Windows Remote Management** » en français « **gestion à distance de Windows** » dit **WinRM** sur l'ordinateur cible.



Ou directement via la console **services.msc**

Les pré-requis sont :

- Les deux machines sont dans le même domaine
- **Windows Remote Management** est correctement installé sur l'ordinateur et le service est lancé.

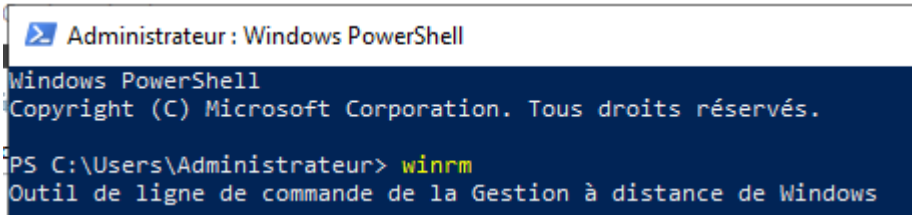
```
PS C:\Windows\system32> get-service -name 'winrm'

Status   Name               DisplayName
------   ----               -----------
Stopped  winrm              Gestion à distance de Windows
```

On peut alors exécuter **Start-Service WinRM**

```
PS C:\Windows\system32> start-service winrm
PS C:\Windows\system32> get-service -name 'winrm'

Status   Name               DisplayName
------   ----               -----------
Running  winrm              Gestion à distance de Windows
```

Utilitaire WinRm ou Enable-Psremoting

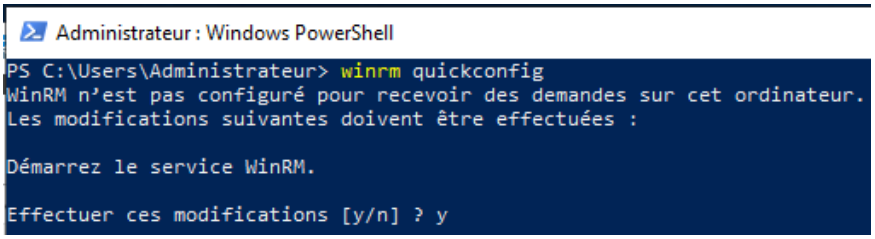
Windows Remote Management propose en standard plusieurs scripts d'administration, dont une configuration simplifiée. Tout d'abord, vérifiez que vous êtes connecté avec un compte administrateur local.



Lancez alors la commande

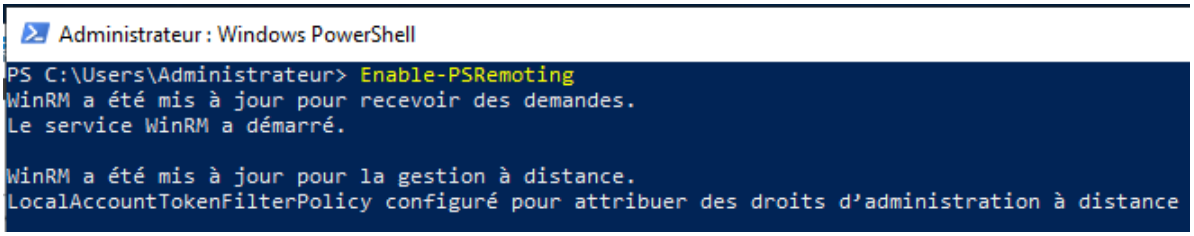
winrm quickconfig

Cela déroule un petit assistant qui propose (si on répond par **y**) de mettre le service en mode automatique, d'écouter les demandes entrantes, et de configurer le pare-feu



Cela peut aussi se faire en **Powershell** par la commande

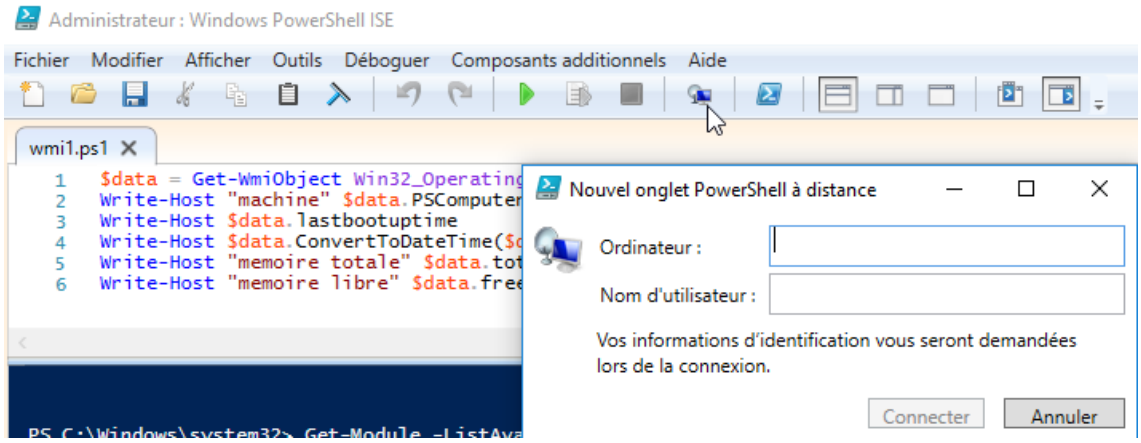
Enable-psremoting



Test Windows Remote Management

Pour vérifier, connectez-vous depuis l'autre machine et lancez « Windows PowerShell ISE ».

Depuis la barre d'outils, lancez un nouvel onglet **PowerShell à distance** depuis le menu « Fichier » ou l'icône correspondante dans la barre d'outils



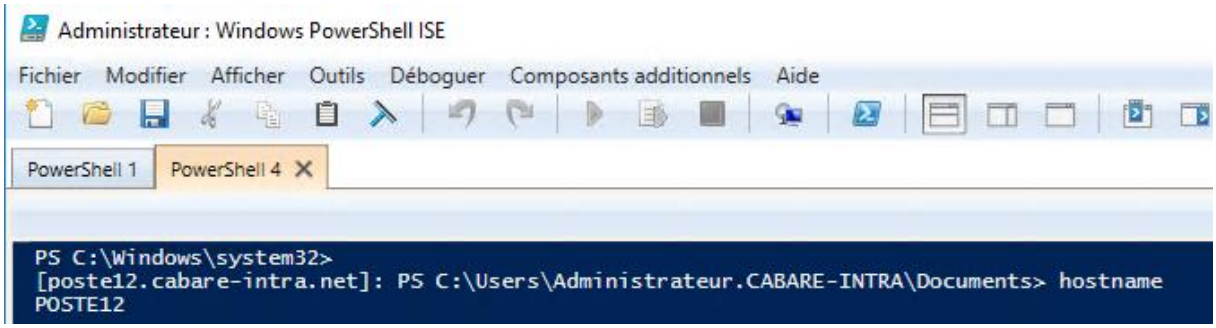
Rappel, il faut se connecter sur une machine du domaine, avec un compte utilisateur de domaine

N.B : il est possible d'autoriser des machines distantes et des comptes utilisateurs locaux en WRM mais cela demande des paramétrages supplémentaires.

Une fois la connexion effectuée, (cela peut prendre un peu de temps) le nom de la machine distante est présente dans la zone de commande, on peut afficher le nom de la machine courante grâce à la commande suivante.

Hostname Le résultat doit être le nom de la machine cible.

Vous pouvez comparer en exécutant la même commande dans un autre onglet PowerShell en local.



New-Pssession ou session manuelle distante

Lorsque l'on utilise Powershell ISE pour ouvrir une invite de commande à distance, PowerShell a lancé en fait la commande « **Enter-PSSession** » permettant de créer une session sur une machine dont le nom est passé en argument et a ajouté le paramètre « **Credential** » pour passer le compte à utiliser.

Il existe plusieurs commandes sur les sessions, mais on on va utiliser principalement **New-Pssession, Get-Pssession, Enter-Pssession, exit,** et **Remove-Pssession**

Il existe deux types de sessions qui permettent d'exécuter des commandes Windows PowerShell :

•La session temporaire : elle s'établit à l'aide des cmdlets Invoke-Command et Enter-PSSession. Une session temporaire dure le temps d'exécution d'une commande dans le

cas d'Invoke-Command, et dans le cas de Enter-PSSession, celle-ci dure jusqu'à ce que l'administrateur quitte la session interactive.

•La session permanente (créée avec New-PSSession) : elle dure le temps d'une session PowerShell, et ce jusqu'à ce qu'on la ferme avec remove-PSSession

- Création d'une nouvelle session

  New-PSSession –ComputerName MonPC –Credential:Domaine\Utilisateur

Comme dans

New-PSSession -ComputerName poste11 -Credential:xxx@cabare-intra.net

Il faut donner le mot de passe, puis un message indique la création de la session.

```
PS C:\Windows\system32> New-PSSession -ComputerName poste11 -Credential:administrateur@cabare-intra.net

 Id Name            ComputerName    State         ConfigurationName     Availability
 -- ----            ------------    -----         -----------------     ------------
  1 Session1        poste11         Opened        Microsoft.PowerShell     Available


PS C:\Windows\system32>
```

- Il faut ensuite se connecter à cette session, que l'on peut repérer via

  Get-Pssession

```
PS C:\Windows\system32> get-pssession

 Id Name            ComputerName    State         ConfigurationName
 -- ----            ------------    -----         -----------------
  1 Session1        poste11         Opened        Microsoft.PowerShell
```

  Et on s'y connecte via

  Enter-PSSession –Name Session1

```
PS C:\Windows\system32> Enter-PSSession -Name Session1
[poste11]: PS C:\Users\Administrateur.CABARE-INTRA\Documents>
```

On, passe alors en mode connecté.

On peut afficher le nom de la machine pour le vérifier. Avec

hostname

```
[poste11]: PS C:\Users\Administrateur.CABARE-INTRA\Documents> hostname
POSTE11
```

6.

Ici, on est sur la machine distante… on fait tout ce que l'on veut…

- Jusqu'à ce que l'on quitte la session (et dans ce cas on pourra s'y reconnecter)

  exit

  Jusqu'à ce que l'on ferme la session (et dans ce cas on ne pourra plus s'y reconnecter, il faudra en recréer une autre)

  Remove-PSSession –Name Sessionx

Dans l'exemple ci-dessous on quitte la session distante sur le poste11, puis on ferme la session ouverte a distance

```
[poste11]: PS C:\Users\Administrateur.CABARE-INTRA\Documents> exit
PS C:\Windows\system32> Remove-PSSession -Name Session1
```

Fournir une authentification : Credential

$login = "cabare-intra.net\admin"

$password = "M0T2pass!" | Convertto-SecureString -AsPlainText –Force

$id = New-Object System.Management.Automation.Pscredential -Argumentlist $login,$password

New-PSSession -ComputerName poste11 -Credential:$id

```
PS C:\Windows\system32> $login = "cabare-intra.net\administrateur"
PS C:\Windows\system32> $password = "zk281" | Convertto-SecureString -AsPlainText -Force
PS C:\Windows\system32> $credentials = New-Object System.Management.Automation.Pscredential -Argumentlist $login,$password
PS C:\Windows\system32> New-PSSession -ComputerName \\poste11 -Credential:$credentials
```

Invoke-Command pour 1 ou 2 commandes à distance

La commande **Invoke-Command** permet de lancer une commande à distance

On peut réutiliser la session comme dans l'exemple précédent.

$s = New-PSSession -ComputerName "poste11"

Invoke-Command -Session $s -ScriptBlock {$services = Get-Service}

Invoke-Command -Session $s -ScriptBlock {$services | Where-Object {$_.Status -eq "Stopped"}}

Remove-PSSession $s

Script à executer à distance

Plutôt qu'un bloc de script, on peut exécuter un script local sur une machine distante sans avoir à le copier.

Pour cela, utilisez le paramètre « FilePath » en lui passant le chemin du fichier ps1 situé sur la machine source.

Invoke-Command –computername MonPC –FilePath .\script.ps1