



POWERSHELL 4 ET 5 AUTOMATISER L'ADMINISTRATION WINDOWS

<http://gtek-it.fr>

Sommaire

▪ Les bases du langage	6
◆ Introduction à Powershell	7
◆ Le principe de l'objet	12
◆ Les applets de commandes "cmdlet"	24
◆ Le pipeline	30
◆ La gestion des sorties	33
◆ Prises en main de Powershell à travers la navigation dans les dossiers/fichiers	36
▪ Le types et opérateurs	54
◆ Les règles programmatiques	55
◆ Variables et constantes	60
◆ La gestion des dates	76
▪ Les structures de contrôles et fonctions	91
◆ L'opérateur "if"	92
◆ Les boucles "for", "foreach"	95
◆ Les fonctions	103
◆ La gestion des fichiers texte	111
◆ La gestion des fichiers CSV	126
◆ Export des données en tant que page HTML	131
◆ Scripts avec authentification	137
◆ Exécution de script à distance	143

Sommaire

▪ Utilisation des cmdlets et des modules	150
◆ Le principe du module	151
◆ Gestion des archives	154
◆ Les cmdlets de gestion web	157
◆ Quelques cmdlets pour gérer vos serveurs et vos stations de travail	163
◆ Les cmdlets de gestion du réseaux	176
◆ Les cmdlets du pare-feu	188
◆ Windows defender	198
◆ La gestion des journaux	202
◆ Gérer les points de restauration sur une station	205
◆ Gestion des packages –OneGet	207
▪ Utilisation des objets WMI	217
◆ Le modèle de données	218
◆ Listage des classe WMI	226
◆ Administrer Windows avec WMI	233

Sommaire

▪ Utilisation de .NET et COM via PowerShell	238
◆ Utilisation du .NET	237
◆ Utilisations de formulaires	253
◆ Les objets COM	267
◆ Traitement des fichiers XML	274
▪ Utilisation des modules spécifiques systèmes PowerShell	278
◆ Le module Active Directory	279
◆ Le module DHCP	303
◆ Le module DNS	310
◆ Le module Hyper V	318
▪ Cmdlets utiles et astuces PowerShell	329
◆ Quelques commandelettes	330
◆ PsEdit	347
◆ Fonction d'administration quotidienne	350
◆ Chaine de texte, cmdlets ConvertFrom-Strings	352

LES BASES DU LANGAGE

Les bases du langage

Introduction à Powershell

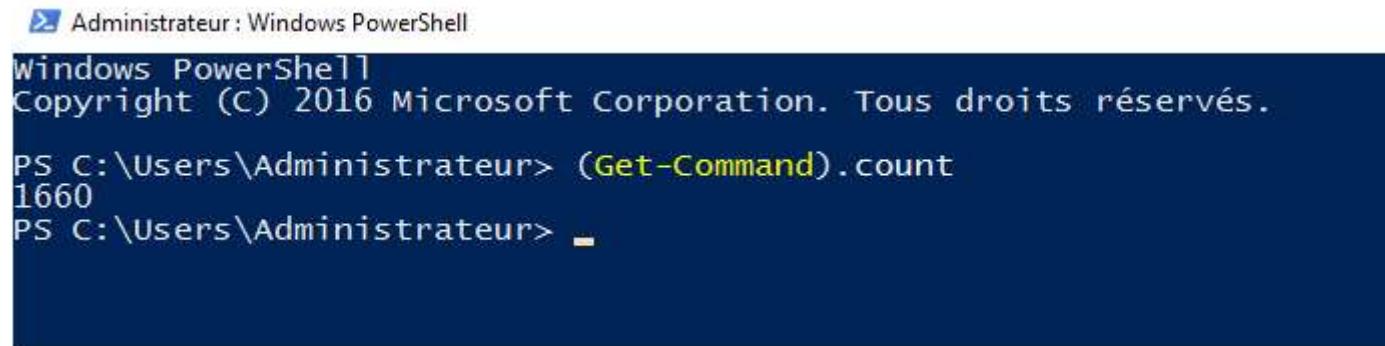
Introduction à Powershell

■ Que peut apporter Powershell ?

- ◆ L'invite CMD ne fournit que peu de commandes.
- ◆ Les administrateurs se sont orientés vers des langages de programmations tel que :
 - VBScript
 - Perl
 - KixStart
 -
- ◆ Langage de développement utilisés pour réaliser parfois des choses simples.

Introduction à Powershell

- Apparu avec vista, Powershell est devenu aujourd’hui incontournable.
 - ◆ Windows 2016 propose plus de 1600 cmdlets de base et plus de 60 modules



```
Administrator : Windows PowerShell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. Tous droits réservés.

PS C:\Users\Administrateur> (Get-Command).Count
1660
PS C:\Users\Administrateur>
```

- Windows Powershell
 - est orienté objet
 - donne accès aux objets .Net
 - offre une cohérence des commandes et des paramètres associés
 - offre une aide sur les commandes digne de ce nom

Introduction à Powershell

- PowerShell est :
 - ◆ Un interpréteur de commande
 - ◆ Et un langage de script
 - ◆ Sa puissance provient du Framework. Net
 - ◆ Connu des développeurs mais moins des administrateurs
 - ◆ Le Framework .NET est une énorme bibliothèque de classes à partir duquel nous allons utiliser des objets.
 - ◆ Avec Powershell, on manipule donc des objets sans vraiment sans rendre compte
 - ◆ Nous sommes à la version 5 de Powershell
 - ◆ Windows Powershell ISE 5 a été repensé

Introduction à Powershell

- Toutes les commandes cmdlets ont été écrites en s'appuyant sur les classes d'objet .Net
 - ◆ Ceci donne un accès à l'intégralité des classes du Framework .Net (ce qui offre des fonctionnalités étendues par rapport à WMI)
 - ◆ Powershell est également capable de faire appel à des objets
 - WSH,
 - COM,
 - WMI

Les bases du langage

Le principe de l'objet

Le principe de l'objet

- En Powershell tout est objet:
 - ◆ On manipule uniquement des objets c'est-à-dire des ensembles groupés de variables et de méthodes associées à des entités
 - ◆ Le principe de fonctionnement du mode objet peut être décomposé de la façon suivante:
 - Classe
 - Schéma de conception, c'est ce que l'on appelle le type de données
 - Instance
 - Objet construit à partir d'une classe
 - Collection
 - Ensemble d'objet
 - Variable
 - Mise en mémoire d'un objet ou d'une collection
 - Membres
 - Caractéristiques et méthodes applicable à un objet, c'est ce que l'on appelle communément les propriétés et méthodes

Le principe de l'objet

- Prenons l'exemple d'un fichier
 - ◆ Une **classe** de type “fichier”, permet de construire un ou plusieurs fichier similaires (**instance**).
 - ◆ Une propriété caractérise l'objet , pour notre objet fichier, on retrouve le nom, l'extension, la date de création, sa taille....
 - ◆ Une méthode va permettre d'agir sur l'objet, une méthode est souvent caractérisé par un verbe, par exemple, on pourrait créer un fichier, le supprimer, le déplacer, le copier, le modifier...
 - ◆ Les propriétés et méthodes sont appelés les **membres**
 - Les membres (propriétés ou méthodes) sont liés à leur parent (l'instance) par un point “Instance.Membre” et par un double “::” pour les objets du framework .NET “[Type.NET]::Membre”
 - Ex: [Math]::round(1,8998,2)

Le principe de l'objet

- Un objet est une variable complexe qui encapsule des données ou attributs ainsi que les fonctions membres pour y accéder.
 - ◆ Dans le monde orienté objet, les fonctions s'appellent aussi des méthodes.
 - ◆ Powershell peut de plus, permettre de modifier un objet à posteriori et y ajouter des méthodes ou des variables, ceci peut s'avérer très utile dans des cas précis.
 - ◆ Créons un objet :
 - `$objet = 1`
- *Nous créons une variable \$objet qui contient l'entier 1.*
- *Powershell va en fait créer un objet de type Int32 et y stocker la valeur assignée.*

Le principe de l'objet

- Tous les objets Powershell définissent une méthode membre appelée GetType(),
 - ◆ cette fonction ne prend pas de paramètres et retourne le type d'objet de l'objet en question.
 - `$objet.GetType()`
 - Cette fonction retourne ceci :

```
PS C:\Windows\system32> $objet.GetType()
| IsPublic IsSerial Name                                     BaseType
|----- - - - - - -----
| True     True      Int32                               System.ValueType
```

- Nous pouvons faire de même avec une variable d'un autre type, par exemple une chaîne de caractères :
- `$liste=< http://gtek-it.fr"`

```
PS C:\Users\LoicThomas> $liste.GetType()
IsPublic IsSerial Name                                     BaseType
----- - - - - - -----
True     True      String                                System.Object
```

Le principe de l'objet

- Nous pouvons utiliser les méthodes de la classe String directement sur l'objet.
 - ◆ Par exemple si nous voulons savoir si \$liste contient le caractère «g» on appelle la méthode ‘Contains’ :

```
PS C:\Users\LoicThomas> $liste.Contains("g")
True
PS C:\Users\LoicThomas>
```

- ◆ Puisque c'est le cas, Powershell retourne **True**.
- ◆ Les objets permettent de cacher la complexité de l'implémentation des fonctions.
 - Ceci s'appelle l'**abstraction**, c'est un concept important de l'**orienté objet**.
- ◆ Dans notre exemple nous voulions savoir si le caractère g se trouvait dans la chaîne \$liste, le comment du pourquoi du détail de l'implémentation de la fonction Contains ne nous intéresse pas.
 - On voulait le résultat c'est tout. C'est ça l'**abstraction**.

Le principe de l'objet

- Autre exemple. Imaginons maintenant la string suivante:
 - ◆ \$chaine = "Ceci est une chaîne de caractère"
 - ◆ Si on veut récupérer les mots, un par un, de cette chaîne, nous pouvons utiliser la méthode Split:
 - \$chaine.Split(" ")
 - Powershell retourne un array (tableau) de n éléments et les affiche un par un.

```
PS C:\Windows\system32> $chaine.Split(" ")
Ceci
est
une
chaîne
de
caractère
```

Le principe de l'objet

- ◆ \$a=\$chaine.Split(" ")
- ◆ \$a.GetType()

```
PS C:\Windows\system32> $a=$chaine.Split(" ")  
PS C:\Windows\system32> $a.GetType()  


| IsPublic | IsSerial | Name     | BaseType     |
|----------|----------|----------|--------------|
| True     | True     | string[] | System.Array |


```

- ◆ Du coup on peut demander à l'array combien d'éléments il possède :

- \$a.count

```
PS C:\Windows\system32> $a.count  
6
```

- Count est une variable membre de l'objet Array. Et nous avons bien 4 mots dans notre string.

Le principe de l'objet

- Nous pouvons créer nos propres objets.
 - ◆ Cela est les données de scripts d'une façon organisée et ainsi exécuter des fonctions sur l'ensemble des objets tels que l'export vers le HTML ou le CSV.
- Pour créer un objet nous utilisons la cmdlet New-Object :
 - ◆ Elle prend comme paramètres le type ainsi qu'une hash table définissant les méthodes et variables membres.

```
$contact = New-Object -TypeName PSCustomObject -Property @{  
    Prenom      = "Loic";  
    Nom         = "Thomas";  
    Numero      = "06xxx";  
    Adresse     = "125..."  
}
```

```
PS C:\Windows\system32> $contact  
Numero Nom      Adresse Prenom  
----- --      ----- -----  
06xxx Thomas 125... Loic
```

Le principe de l'objet

- Ajoutons maintenant une nouvelle variable membre à l'objet.

- ◆ Nous utilisons pour ceci la cmdlet Add-Member.

```
Add-Member -MemberType ScriptProperty -InputObject $contact -Name "Date" -Value {Get-Date}
```

- Le type de membre qu'on ajoute est un script.
 - Le nom est « Date » et la valeur est un script block qui appelle la cmdlet Get-Date qui retourne la date et l'heure actuelle.
- Nous pouvons accéder aux propriétés membres en utilisant leurs noms, exemple :
 - \$contact.Prenom
 - \$contact.date

```
PS C:\Windows\system32> $contact.Prenom  
Loic
```

```
PS C:\Windows\system32> $contact.date
```

```
lundi 31 octobre 2016 09:06:37
```

Le principe de l'objet

- Imaginons maintenant qu'on veut traiter une liste de contacts et pas un seul.
 - ◆ Pour cela nous allons créer une array vide appelée \$contacts :
 - `$contacts = @()`
 - ◆ Nous pouvons ajouter les objets \$contact créés à notre array avec l'opérateur `+ =` :
 - `$contacts += $contact`

```
PS C:\Windows\system32> $contacts

Numero : 06xxx
Nom    : Thomas
Adresse : 125...
Prenom  : Loic
Date   : 31/10/2016 09:10:21
```

Le principe de l'objet

- Avec cette liste de contacts nous pouvons faire plusieurs opérations, par exemple nous pouvons l'exporter vers un fichier de type CSV :
 - ◆ `$contacts | Select Prenom, Nom, Adresse | Export-Csv -Path "c:\temp\contacts.csv"`
 - ◆ Select-Object nous permet de sélectionner les propriétés membres ainsi que leur ordre.
 - Ainsi nous voulons, le Prenom en premier suivi du Nom et de l'Adresse. Nous ne voulons pas par exemple sélectionner la date, elle ne sera pas sauvegardée.
 - ◆ Si nous voulons récupérer notre liste de contacts à partir d'un fichier CSV, nous pouvons utiliser la cmdlet Import-Csv:
 - `$contacts = Import-Csv -Path "c:\temp\contacts.csv"`

```
PS C:\Windows\system32> $contacts
```

Prenom	Nom	Adresse
Loic	Thomas	125...

Les bases du langage

Les applets de commandes “cmdlet”

Les applets de commandes “cmdlet”

- En Powershell on agit sur les objets à l'aide d'applet de commande que l'on appelle **cmdlet**
- les applets de commandes sont toujours composé de :
 - ◆ **Verbe – Nom**
 - ◆ Exemple :
 - **get-process**
 - Ici on récupère (**get**) les processus en fonctionnement sur la machine (**process**)
 - Le résultat renvoie une “collection d'instances”

Collections d'instances

Propriétés

Valeurs

Handles	NPM(K)	PM(K)	WS(K)	VM(M)	CPU(s)	Id	ProcessName
159	13	1896	548	81	0,13	4944	ACMON
76	8	1220	4880	76	0,03	3596	ActivateDesktop
95	7	1664	5128	58		1192	AdminService
82	7	1056	4160	44		2032	armsvc
73	6	872	3936	17		1500	AsLdrSrv
91	8	1596	1256	73	0,28	5060	ASUS Console Starter
229	18	3380	768	109	0,94	6424	AsusTPCenter
45	5	792	216	45	0,00	280	AsusTPHelper
140	10	1788	588	84	3,28	5744	AsusTPLoader
138	10	1980	5740	59		2760	Ath_CoexAgent
107	12	2280	9004	125	0,41	5636	ATKOSD2
156	12	9972	12444	52	0,24	2072	audiodg

Les applets de commandes “cmdlet”

- Avec Powershell on retrouve les verbes :
 - ◆ Get (récupérer)
 - ◆ Set (Action sur l'objet)
 - ◆ Add (ajouter)
 - ◆ Remove (effacer)
 - ◆ ...
- Tous ces verbes sont associés avec des noms comme :
 - ◆ Path
 - ◆ Variable
 - ◆ Item
 - ◆ Object
 - ◆ ...
- En croisant les verbes et les noms, on peut se souvenir de beaucoup de commandes
- PowerShell ne respecte pas la casse
 - ◆ GET-Command = Get-command = get-command

Les applets de commandes “cmdlet”

- Dans un premier temps il est important de bien connaître les commandes suivantes
 - ◆ Get-Command
 - ◆ Get-Help
 - ◆ Get-Member
- **Get-command**
 - ◆ Permet de trouver les commandes
 - ◆ Trouver les commandes dont le verbe est **write**
 - **Get-command –verb write**

PS C:\Users\LoicThomas> get-command –verb write	CommandType	Name	ModuleName
	Alias	Write-FileSystemCache	Storage
	Function	Write-DtcTransactionsTraceSession	MsDtc
	Function	Write-PrinterNfcTag	PrintManagement
	Function	Write-VolumeCache	Storage
	Cmdlet	Write-Debug	Microsoft.PowerShell.Utility
	Cmdlet	Write-Error	Microsoft.PowerShell.Utility
	Cmdlet	Write-EventLog	Microsoft.PowerShell.Management
	Cmdlet	Write-Host	Microsoft.PowerShell.Utility
	Cmdlet	Write-Output	Microsoft.PowerShell.Utility
	Cmdlet	Write-Progress	Microsoft.PowerShell.Utility
	Cmdlet	Write-Verbose	Microsoft.PowerShell.Utility
	Cmdlet	Write-Warning	Microsoft.PowerShell.Utility

- Renvoi toutes les cmdlet mais aussi le alias et les fonction commençant par write

Les applets de commandes “cmdlet”

- Trouver les commandes dont le nom est *service*
 - ***Get-command –noun service***

PS C:\Users\LoicThomas> get-command –noun service		
CommandType	Name	ModuleName
Cmdlet	Get-Service	Microsoft.PowerShell.Management
Cmdlet	New-Service	Microsoft.PowerShell.Management
Cmdlet	Restart-Service	Microsoft.PowerShell.Management
Cmdlet	Resume-Service	Microsoft.PowerShell.Management
Cmdlet	Set-Service	Microsoft.PowerShell.Management
Cmdlet	Start-Service	Microsoft.PowerShell.Management
Cmdlet	Stop-Service	Microsoft.PowerShell.Management
Cmdlet	Suspend-Service	Microsoft.PowerShell.Management

- Renvoi les cmdlet permettant de travailler sur les services
- **Get-help**
 - Permet d'obtenir de l'aide sur une cmdlet particulière, on peut utiliser ensuite – *examples*, *–full*, *-detailed* ou encore *–online* pour une aide avec différents niveaux.
 - ◆ Depuis la version 3, il faut charger les mises à jour avec *update-help*
 - ◆ Cette commande force à récupérer les mises à jour des fichiers d'aide en français
 - ***update-help -Force -Module Microsoft.powershell.management -UICulture fr-FR***
 - ◆ Vous pouvez utiliser *update-help* pour mettre à jour les aides de vos cmdlettes.

Les applets de commandes “cmdlet”

▪ Get-Member :

- ◆ Elle permet d'explorer les objets et donc de visualiser les propriétés et méthodes applicable à l'objet.
- ◆ Exemple : **Get-Service | Get-Member**

```
PS C:\Users\LoicThomas> Get-service | get-member

    TypeName : System.ServiceProcess.ServiceController
    Name          MemberType      Definition
    --          --           --
    Name          AliasProperty  Name = ServiceName
    RequiredServices AliasProperty RequiredServices = ServicesDependedOn
    Disposed       Event         System.EventHandler Disposed(System.Object, System.EventArgs)
    Close          Method        void Close()
    Continue       Method        void Continue()
    CreateObjRef   Method        System.Runtime.Remoting.ObjRef CreateObjRef(type requestedType)
    Dispose        Method        void Dispose(), void IDisposable.Dispose()
    Equals         Method        bool Equals(System.Object obj)
    ExecuteCommand Method        void ExecuteCommand(int command)
    GetHashCode   Method        int GetHashCode()
    GetLifetimeService Method      System.Object GetLifetimeService()
    GetType        Method        type GetType()
    InitializeLifetimeService Method      System.Object InitializeLifetimeService()
    Pause          Method        void Pause()
    Refresh         Method        void Refresh()
    Start          Method        void Start(), void Start(string[] args)
    Stop           Method        void Stop()
    WaitForStatus  Method        void WaitForStatus(System.ServiceProcess.ServiceControllerStatus desiredStat...
    CanPauseAndContinue Property     bool CanPauseAndContinue {get;}
    CanShutdown    Property     bool CanShutdown {get;}
    CanStop        Property     bool CanStop {get;}
    Container      Property     System.ComponentModel.IContainer Container {get;}
    DependentServices Property     System.ServiceProcess.ServiceController[] DependentServices {get;}
    DisplayName    Property     string DisplayName {get; set;}
```

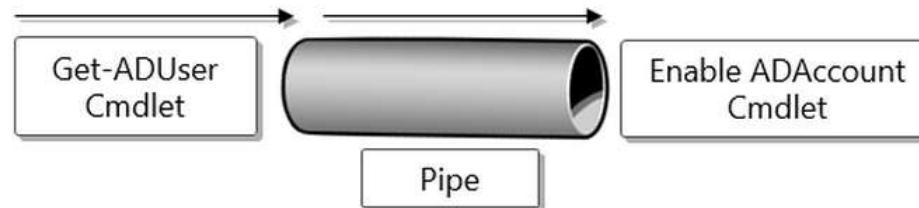
- ◆ On retrouve:

- Les méthodes applicables aux objets de type service (close, continue, start...)
- Les propriétés (caractéristiques) de objets de types services (CanShutdown, CanStop, DisplayName...)

Le pipeline

Le pipeline Windows Powershell

- Le pipeline est utilisé pour connecter la sortie d'une applet de commande à l'entrée d'une autre applet de commande
- La combinaison de la première applet de commande, du canal, et de la deuxième applet de commande crée un pipeline



- obtenir la liste de tous les utilisateurs dans le domaine, puis envoyez la liste dans un pipeline à l'applet de commande **Enable-ADAccount**, en exécutant la commande suivante :
 - `Get-ADUser –Filter * | Enable-ADAccount`
- Pour activer les comptes qui sont désactivés, vous pouvez utiliser l'applet de commande **Where-Object** pour renvoyer uniquement les comptes qui sont désactivés. Pour ce faire, exécutez la commande suivante
 - `Get-ADUser –Filter * | Where-Object {$_Enabled –eq $false} | Enable-ADAccount`

Le pipeline variable

- Depuis la V4, une nouveauté très intéressante est la possibilité de mettre des variables dans le pipeline.
- Exemple :

```
Get-adcomputer -filter * -pv comp | foreach-object{gwmi  
win32_logicalDisk -computername $_.name} |  
ForEach-Object { write-host "Analyse de l'ordinateur $comp.name"  
write-host " " nom du disque: $($_.deviceid)"}
```

The screenshot shows a PowerShell window with the title 'variables dans le pipeline.ps1'. The code in the editor is:

```
1 Get-content C:\temp\pc.txt -PipelineVariable comp | %{gwmi win32_logicaldisk -ComputerName $_} | %{  
2 write-host "Analyse de l'ordinateur: $comp" -f Green  
3 write-host "Nom de disque: $($_.deviceid)"}  
4  
5  
6
```

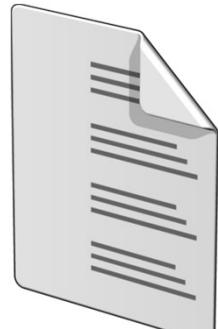
The output in the console is:

```
write-host "Analyse de l'ordinateur: $comp" -f Green  
write-host "Nom de disque: $($_.deviceid)"}  
  
Analyse de l'ordinateur: W81  
Nom de disque: C:  
Analyse de l'ordinateur: W81  
Nom de disque: D:  
Analyse de l'ordinateur: W81  
Nom de disque: E:
```

La gestion des sorties

Options de mise en forme de la sortie Windows PowerShell

- Applets de commande pour mettre la sortie en forme:



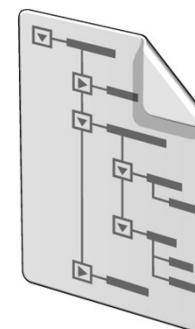
**Format-Wide
(FW)**



**Format-Table
(FT)**

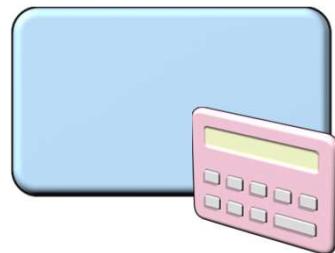


**Format-List
(FL)**

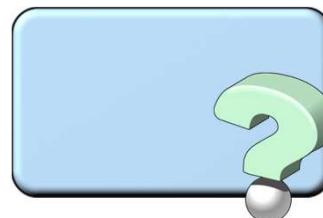


**Format-Custom
(FC)**

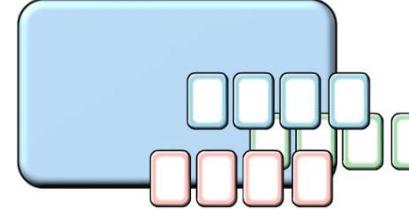
- Applets de commande pour manipuler la sortie



**Measure-Object
(measure)**



**Sort-Object
(sort)**



**Select-Object
(select)**



**Where-Object
(where)**

Les applets de commandes “cmdlet”

- Pour rappel :

- ◆ **Get-command** retourne les cmdlettes
- ◆ **Get-help** Cmdlettes obtenir de l'aide sur une cmdlette particulière, utiliser ensuite **-examples** ou **-full** pour une aide avec seulement les exemples ou détaillées
- ◆ **Cmdlet | get-member** pour obtenir les propriétés et méthodes sur une cmdlet
- ◆ Tout est objet en PowerShell, pensez systématiquement propriétés et méthodes sur ces objets
- ◆ Le **|** permet de récupérer les objets courants passé dans le pipe
- ◆ La variable **\$_** est la variable la plus importante, elle récupère l'objet courant passé dans le pipe
- ◆ Les filtres utilise **where-object** ou **where** ou **?**
- ◆ Les collections sont récupérés avec la boucle **foreach-object** ou **foreach** ou **%**

Les bases du langage

Prises en main de Powershell à travers la navigation dans les dossiers et les fichiers

Navigation dans les dossiers et les fichiers

- Les anciennes commandes de l'invite cmd fonctionnent toujours avec PowerShell.
 - ◆ Par ex la commande **dir** fonctionne mais en réalité renvoi à un alias nommé **Get-childItem**
 - Pour vérifier taper **get-alias dir**

```
PS C:\Users\Administrateur> Get-Alias dir
```

CommandType	Name	Version	Source
Alias	dir -> Get-ChildItem		

```
PS C:\Users\Administrateur> Get-Alias cd
```

CommandType	Name	Version	Source
Alias	cd -> Set-Location		

Navigation dans les dossiers et les fichiers

- Le tableau suivant donne l'alias des principales commandes cmd

CMD	CMDLET	Description
DIR	Get-childitem	Lister le contenu d'un dossier
CD	Set-location	Changer de dossier courant
MD	New-item	Créer un fichier /dossier
RD	Remove-item	Supprimer un fichier/dossier
Move	Move-item	Déplacer un fichier/dossier
Ren	Rename-item	Renommer un fichier/dossier
Copy	Copy-item	Copier un fichier/dossier

Navigation dans les dossiers et les fichiers

■ Get-childItem (alias ls ou dir)

- ◆ Permet d'obtenir la liste des fichiers et dossiers
- ◆ Ex : **Get-ChildItem -Path 'E:\01_Support Formation\00 - Intervention Orsys'**

■ La colonne « Mode » la nature des objets à l'intérieur du système de fichiers, les valeurs possibles sont :

- d pour dossier
- a pour archive
- r pour lecture seule
- h pour objet caché
- s pour objet système

Répertoire : E:\01_Support Formation\00 - Intervention Orsys				
Mode	LastWriteTime	Length	Name	
d----	19/05/2016 09:40		00_Réseaux	
d----	18/05/2015 08:55		01_Windows_7	
d----	19/04/2015 09:44		02_Windows_2008	
d----	22/10/2016 09:41		03_Windows_2012	
d----	30/10/2016 12:20		04_Windows_Powershell	
d----	02/02/2015 18:13		05_SCOM	
d----	06/10/2016 14:02		06_HyperV	
d----	28/10/2016 12:05		07_Linux	
d----	23/11/2015 08:49		08_Windows_8	
d----	26/01/2016 22:26		09_Lync_2013	
d----	02/02/2015 18:13		10_IIS8	
d----	01/06/2016 17:53		11_Seminaire	
d----	18/04/2015 12:58		12_SQL_ADMIN	
d----	22/05/2015 16:16		13_Exchange_2013	
d----	23/03/2016 16:12		14_Skype_For_Business_2015	
d----	05/09/2016 14:09		15_Windows_2016	
d----	23/12/2015 20:47		20_Eval_Intra	
d----	18/04/2016 17:44		30_Windows_2016	
d----	02/02/2015 18:13		application	
d----	02/02/2015 18:13		cours	
d----	02/02/2015 18:13		doc_administratif	
d----	02/02/2015 18:13		HyperV	
d----	02/02/2015 18:13		tutoriel_powershell	
d----	02/02/2015 18:13		WIA_Alx_en_provence	
a----	25/06/2013 16:09	41984	10165_Programme_W7+AIN+ISU_spé 5 jours.doc	
a----	02/07/2013 18:54	40960	10352_Programme_W7I_spé 4 jours.doc	
a----	09/02/2014 10:39	1856512	ADMTmigguide migration compte et utilisateur domaine.doc	
	11/07/2012 10:28	16521_BILL_migguide_migration_Windows_2008.doc		

Navigation dans les dossiers et les fichiers

- Pour afficher la liste des fichiers cachés, taper:
 - ***Get-childitem -force***
 - Le fait de ne pas préciser de dossier, retourne la racine de c:

-a-h-	02/09/2007	13:12	0	ntuser.dat.LOG2
-a-hs	02/09/2007	13:12	65536	NTUSER.DAT<0f69446d-6a70-11db-8eb3
-a-hs	31/10/2007	07:35	524288	NTUSER.DAT<0f69446d-6a70-11db-8eb3
				ms
-a-hs	02/09/2007	13:12	524288	NTUSER.DAT<0f69446d-6a70-11db-8eb3
				ms
-a-hs	01/08/2008	07:47	65536	ntuser.dat<4ed726bf-b9e2-11dc-9901
-a-hs	01/08/2008	07:47	524288	ntuser.dat<4ed726bf-b9e2-11dc-9901
				ms
-a-hs	04/01/2008	09:07	524288	ntuser.dat<4ed726bf-b9e2-11dc-9901
				ms
---hs	02/09/2007	13:51	20	ntuser.ini
-arhs	23/04/2008	12:07	664	ntuser.pol

- ◆ Ex sur un fichier, nommé powershell_intro.ppt.
- ◆ On stocke dans une variable \$b le résultat ensuite on affiche **\$b.length** pour la longueur
- ◆ **\$b.creationTime** pour date de création ...

```
PS G:\vista\cours\powershell> $b = get-childitem powershell_intro* ; $b.lastaccessed
PS G:\vista\cours\powershell> $b = get-childitem powershell_intro* ; $b.creationtime
samedi 11 avril 2009 20:53:50

PS G:\vista\cours\powershell> $b = get-childitem powershell_intro* ; $b.length
91136
PS G:\vista\cours\powershell> $b = get-childitem powershell_intro* ; $b.lastaccesstime
samedi 11 avril 2009 20:53:50
```

Navigation dans les dossiers et les fichiers

- Afficher récursivement tous les fichiers du dossier e:\lotus avec l'extension *.doc et *.ppt
 - ◆ ***Get-childitem e:\lotus -recurse -include *.doc, *.ppt***

```
PS C:\> Get-ChildItem e:\lotus\* -recurse -include *.doc, *.ppt

Répertoire : Microsoft.PowerShell.Core\FileSystem::E:\lotus\lotus\Designer\dev base

Mode          LastWriteTime    Length Name
----          -----        --  --
-a--- 08/03/2005 17:49      748544 1 Lotus Notes Domino V6 designer.ppt
-a--- 23/07/2008 10:55      1196544 10 La sécurité Domino Notes.ppt
-a--- 25/04/2007 12:13      969728 13 Accéder aux données externes.ppt
-a--- 09/06/2006 16:40      150016 14 Nouveauté V6.ppt
-a--- 10/06/2006 11:32      143360 14 Nouveauté V7.ppt
-a--- 08/03/2005 19:54      2409472 2 Lotus Notes Domino V6 Concept.ppt
-a--- 23/04/2007 12:22      1541632 4 Le masque.ppt
-a--- 26/06/2008 17:03      1841664 5 Programmer avec les formules.ppt
-a--- 01/03/2007 09:55      1397248 6 les vues.ppt
-a--- 24/04/2007 15:24      951808 7 Fonctions avancées du masque.ppt
-a--- 09/06/2006 10:18      914944 8 Présenter l'information.ppt
-a--- 21/09/2006 09:51      1201152 9 Gérer la navigation.ppt
-a--- 28/02/2007 10:39      196608 Domino à travers le web.ppt
-a--- 28/10/2005 11:26      22528 Modèle de conception.doc
-a--- 21/07/2005 21:09      1203712 Présentation1.ppt

Répertoire : Microsoft.PowerShell.Core\FileSystem::E:\lotus\lotus\Designer\javascript

Mode          LastWriteTime    Length Name
----          -----        --  --
-a--- 20/02/2007 16:54      1288704 javaScript.doc
-a--- 30/03/2007 16:24      2724352 JavaScript.ppt

Répertoire : Microsoft.PowerShell.Core\FileSystem::E:\lotus\lotus\Designer\lotusScript

Mode          LastWriteTime    Length Name
----          -----        --  --
-a--- 27/05/2008 18:57      484352 coursHTMLChapitre2.ppt
-a--- 22/05/2008 17:42      6145024 Excel et LotusScript.ppt
-a--- 25/05/2008 18:49      1298432 LotusScript 2.ppt
-a--- 19/03/2007 11:49      431616 LotusScript.ppt
```

Navigation dans les dossiers et les fichiers

- Afficher tous les fichiers du dossier lotus sauf les *.jpg, *.doc
 - ◆ **Get-childitem e:\lotus –recurse –exclude *.jpg, *.doc**

```
PS C:\> Get-ChildItem e:\lotus\* -recurse -exclude *.doc, *.jpg
```

- Obtenir les fichiers dont la taille est supérieur à 100ko
- Get-childitem e:\lotus | where-object {\$_.length -gt 1000KB}**

```
PS C:\> Get-ChildItem e:\lotus | where-object {$_.length -gt 1MB}

Répertoire : Microsoft.PowerShell.Core\FileSystem::E:\lotus

Mode                LastWriteTime     Length Name
----                -----        ---- 
-a---       25/10/2007    17:00      2135077 db2.pdf
-a---       01/02/2008    14:18      8126464 gael.nsf
-a---       17/04/2008    12:19      2621440 ged.nsf
-a---       22/04/2008    11:12      6029312 helpdesksynersys1 <2>.nsf
-a---       27/02/2008    18:41      5767168 helpdesksynersys1.nsf
-a---       23/04/2008    16:38      3271298 helpdesksynersys12.zip
-a---       19/09/2007    11:13      5224448 Manuel Utilisateur Lotus 6 client lourd.doc
-a---       19/12/2007    16:40      3228228 notes8.zip
-a---       26/03/2008    11:41      2883584 reportGM <2>.nsf
-a---       25/05/2008    09:26      1898496 reportGM.nsf
-a---       25/05/2008    09:10      11272192 reportin.nsf
-a---       25/05/2008    09:09      1198080 reports.nsf
-a---       04/09/2007    19:15      731652096 vista_6000.16386.061101-2205-LRMAIK_FR.img
```

1000KB peut être
remplacé par 1MB.

Attention KB et MB sont
collés avec le chiffre
Gt pour greater than

Navigation dans les dossiers et les fichiers

- Afficher tous les fichiers du dossier lotus dont la date de dernier enregistrement est postérieur au 01/06/2008
 - ◆ (attention format de date américain (Mois/jours/année))
 - ◆ **Get-childitem | where-object {\$_.LastWriteTime -gt '06/01/2008'}**



Mode	LastWriteTime	Length	Name
d----	07/10/2008 16:13		lotus
-a---	06/10/2008 14:51	32	test.bat

- ◆ Le pipe « | » permet de passer un ou plusieurs objets à la commande.
- ◆ Dans nos exemple nous passons la commandelette « **Where-Object** » (filtre).
- ◆ Where-object analyse les propriétés **Length** et **LastWriteTime** et les compare au test (1000KB ou '06/01/2008) et retourne l'objet si le test est vrai.
- ◆ Le **\$_.** retourne l'objet courant
 - **-gt** signifie (greater than, plus grand que ou strictement supérieur), **-ge** (supérieur ou égal)
 - **-eq** (=)
 - **-lt** (strictement inférieur) , **-le** (inférieur ou égal)

Navigation dans les dossiers et les fichiers

- La commande suivante permet de lister le répertoire courant puis de faire un tri décroissant et enfin affiche les 5 premiers éléments

```
get-childitem | sort-object -descending | select-object -first 5
```

- ◆ *Sort-object* permet de faire un tri
- ◆ *Select-object* permet de sélectionner des objets après le pipe.
- ◆ Ces 2 cmdlets pourraient être utiliser avec autre chose que *get-childitem* comme *get-process* par exemple

Navigation dans les dossiers et les fichiers

- Pour afficher les propriété et méthode de ***Get-childitem***, n'oubliez pas la commandelette ***get-member***
 - ◆ ***Get-childitem / get-member***

```
OpenWrite           Method      System.IO.FileStream OpenWrite()
Refresh            Method      System.Void Refresh()
Replace            Method      System.IO.FileInfo Replace(String destinationFileName, S
SetAccessControl   Method      System.Void SetAccessControl(FileSecurity fileSecurity)
set_Attributes     Method      System.Void set_Attributes(FileAttributes value)
set_CreationTime   Method      System.Void set_CreationTime(DateTime value)
set_CreationTimeUtc Method      System.Void set_CreationTimeUtc(DateTime value)
set_IsReadOnly     Method      System.Void set_IsReadOnly(Boolean value)
set_LastAccessTime Method      System.Void set_LastAccessTime(DateTime value)
set_LastAccessTimeUtc Method     System.Void set_LastAccessTimeUtc(DateTime value)
set_LastWriteTime  Method      System.Void set_LastWriteTime(DateTime value)
set_LastWriteTimeUtc Method     System.Void set_LastWriteTimeUtc(DateTime value)
ToString           Method      System.String ToString()
PSChildName        NoteProperty System.String PSChildName=autoexec.bat
PSDrive            NoteProperty System.Management.Automation.PSDriveInfo PSDrive=C
PSIsContainer     NoteProperty System.Boolean PSIsContainer=False
PSParentPath       NoteProperty System.String PSParentPath=Microsoft.PowerShell.Core\Fi
PSPath             NoteProperty System.String PSPath=Microsoft.PowerShell.Core\FileSyste
PSProvider         NoteProperty System.Management.Automation.ProviderInfo PSProvider=Mi
Attributes          Property    System.IO.FileAttributes Attributes {get;set;}
CreationTime       Property    System.DateTime CreationTime {get;set;}
CreationTimeUtc   Property    System.DateTime CreationTimeUtc {get;set;}
Directory          Property    System.IO.DirectoryInfo Directory {get;}
DirectoryName     Property    System.StringDirectoryName {get;}
Exists             Property    System.Boolean Exists {get;}
Extension          Property    System.String Extension {get;}
FullName           Property    System.String FullName {get;}
IsReadOnly         Property    System.Boolean IsReadOnly {get;set;}
LastAccessTime    Property    System.DateTime LastAccessTime {get;set;}
LastAccessTimeUtc Property    System.DateTime LastAccessTimeUtc {get;set;}
LastWriteTime     Property    System.DateTime LastWriteTime {get;set;}
LastWriteTimeUtc  Property    System.DateTime LastWriteTimeUtc {get;set;}
Length             Property    System.Int64 Length {get;}
Name               Property    System.String Name {get;}
Mode               ScriptProperty System.Object Mode {get=$catr = "";}...
```

Navigation dans les dossiers et les fichiers

■ Set-Location (alias sl ou cd)

- ◆ Cette commandelette permet de se déplacer dans une arborescence de dossier
- ◆ Ex: ***Set-location e:***

■ Get-Location (alias gl ou PWD)

- ◆ Retourne l'emplacement actuel à l'intérieur d'une arborescence.

```
PS E:\Lotus> pwd
Path
-----
E:\Lotus

PS E:\Lotus> get-location
Path
-----
E:\Lotus

PS E:\Lotus>
```

Navigation dans les dossiers et les fichiers

■ New-item (alias ni ou md)

- ◆ Permet de créer des dossier, des fichiers
- ◆ Il est possible de créer un fichier et remplir ce fichier.
- ◆ Les paramètres que nous utilisons sont :
 - **Path** -> chemin d'accès de l'élément à créer (ex: c:\temp. Le « . » indique ce dossier
 - **Itemtype** ou **type** -> type d'élément à créer : **file** pour fichier, **directory** pour dossier
 - **Name** -> nom du nouvel élément à créer
 - **Value** contenu de l'élément (ex: « hello the world » dans le cas d'un fichier texte

Navigation dans les dossiers et les fichiers

- Création d'un dossier sous le répertoire courant
 - ◆ **New-item –type directory "dossier pwshell"**

```
PS E:\Lotus> new-item -type directory "dossier pwshell"

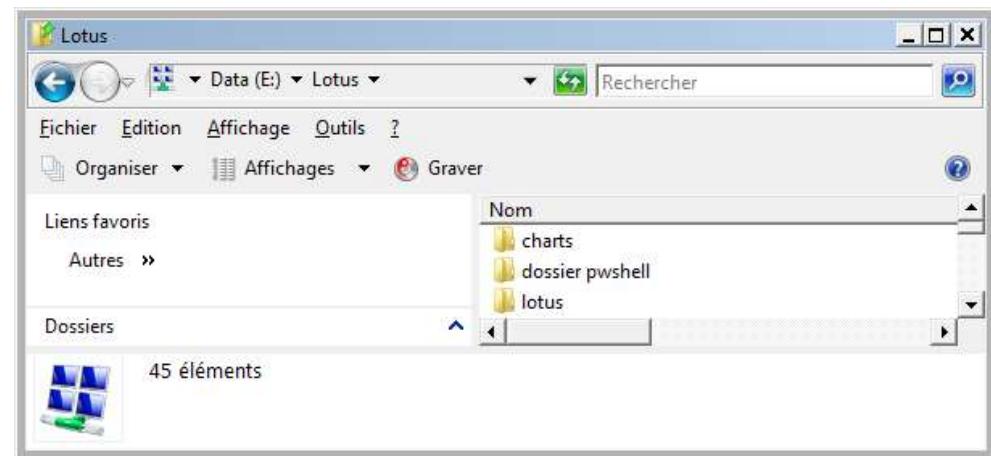
Répertoire : Microsoft.PowerShell.Core\FileSystem::E:\Lotus

Mode          LastWriteTime    Length Name
d---        18/11/2008     00:34      dossier pwshell

PS E:\Lotus> new-item -type directory "dossier pwshell"

Répertoire : Microsoft.PowerShell.Core\FileSystem::E:\Lotus

Mode          LastWriteTime    Length Name
d---        18/11/2008     00:34      dossier pwshell
```



Création d'un dossier sous le dossier e:\lotus\lotus

- **New-item –path e:\lotus\lotus –type directory –name « dossier pwshell 2 »**

```
PS E:\Lotus> new-item e:\lotus\lotus -type directory -name "dossier pwshell 2"

Répertoire : Microsoft.PowerShell.Core\FileSystem::E:\lotus\lotus

Mode          LastWriteTime    Length Name
d---        18/11/2008     00:38      dossier pwshell 2
```

Navigation dans les dossiers et les fichiers

- Création d'un fichier sous e:\scripts nommé test.txt dans lequel on ajoute une phrase « Powershell est un outil puissant ».
 - ◆ *New-item -type file -name test.txt -value « Powershell est un outil puissant »*

```
PS E:\Lotus> new-item -path e:\scripts -type file -name test.txt -value "Powershell est un outil puissant"

Répertoire : Microsoft.PowerShell.Core\FileSystem::E:\scripts

Mode          LastWriteTime    Length Name
----          -----        -----
-a---  18/11/2008   00:45         32 test.txt

PS E:\Lotus>
```

Navigation dans les dossiers et les fichiers

- **Remove-item (alias: ri, rm, rd, erase, del)**
 - ◆ Permet de supprimer ou des dossiers.
 - ◆ Ex : supprimer tout les fichiers avec l'extension *.log
 - *remove-item c:\temp*.log*

```
PS E:\Lotus> remove-item c:\temp\*.log  
PS E:\Lotus> _
```

Ex : supprimer tout les fichiers avec l'extension *.txt contenu dans tout les dossiers et sous dossier de c:\temp

- *Get-childitem c:\temp –recurse –include *.txt | remove-item*

```
PS E:\Lotus> Get-childitem c:\temp –recurse –include *.txt | remove-item  
PS E:\Lotus> _
```

- Nous utilisons ici Get-childitem pour lister les fichier *.txt (ceux à supprimer), ensuite pour chaque fichier txt trouvé, nous utilisons remove-item.
- On aurait pu ajouter l'option **–force** pour effacer les fichiers cachés

Navigation dans les dossiers et les fichiers

■ move-item (alias mi ou move, mv)

- ◆ Déplacer un fichier ou un dossier d'un emplacement vers un autre emplacement.
- ◆ Dans le cas d'un dossier, le contenu du dossier est également déplacé.
- ◆ Move-item permet également de renommer l'objet
 - Ex: déplacement des fichiers *.txt vers un autre dossier
 - **Move-item –path c:\temp\temp1*.txt –destination c:\temp\temp2**

```
PS C:\Users\gtek> move-item -path c:\temp\temp2\*.txt -destination c:\temp\temp1
PS C:\Users\gtek>
```

- ◆ Ex: Déplacer un dossier vers un autre dossier, indiquer simplement les noms de dossier
 - **Move-item –path c:\temp\temp1 –destination c:\temp\temp2**

Navigation dans les dossiers et les fichiers

■ Rename-item (alias ren ou rni)

- ◆ Ex: renommer le fichier journal.txt en log.txt
 - *Rename-item –path c:\temp\journal.txt –newname log.txt*
 - Le path et newname ne sont pas obligatoire
 - Rename-item c:\temp\journal.txt log.txt
- ◆ Ex: renommer un dossier
 - Rename-item -path c:\temp\temp1 temp2

Navigation dans les dossiers et les fichiers

■ Copy-item (alias copy, cpi, cp)

- ◆ Permet de copier des fichiers et dossiers
- ◆ Ex : copier un fichier d'un dossier vers un autre dossier
 - *Copy-item –path c:\temp\journal.txt –destination c:\temp\temp1*
- ◆ Ex: copier une arborescence de dossier (dossiers et sous-dossier)
 - *Copy-item –path c:\temp –destination c:\test -recurse*
 - Si le dossier n'existe pas, il est automatiquement créé

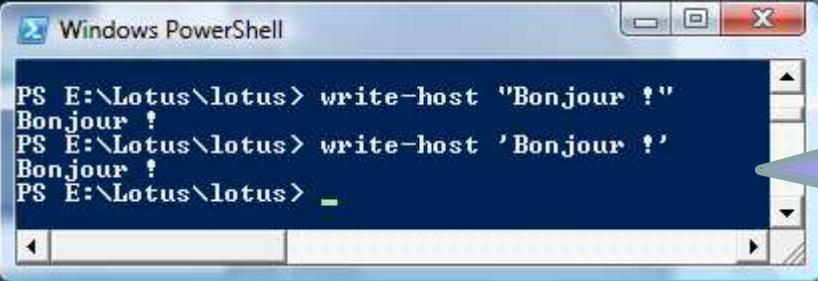
LES TYPES ET OPÉRATEURS

Les types et opérateurs

Les règles programmatiques

Les règles programmatiques

- Dans Powershell, 2 méthodes peuvent être utilisées pour les chaînes de caractères:
 - ◆ Les guillemets " " doubles ("Bonjour")
 - ◆ Les guillemets simples ' ' ('Bonjour')

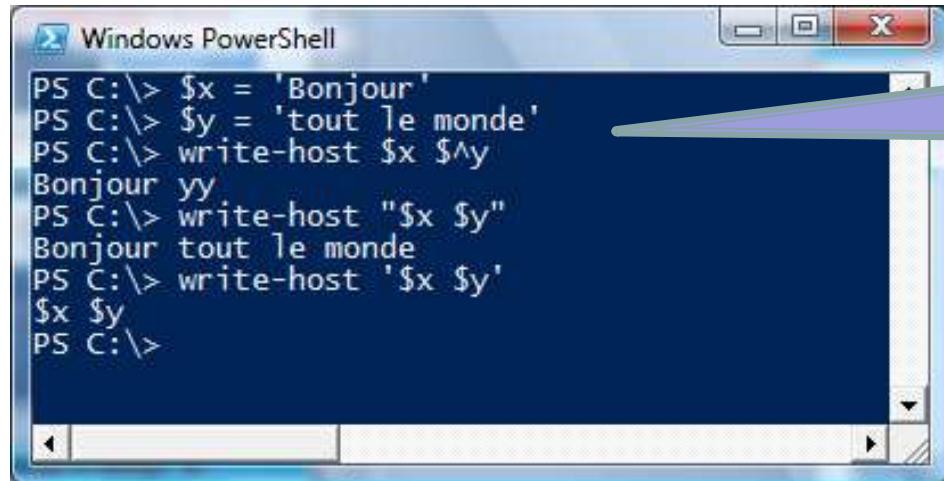


```
Windows PowerShell
PS E:\Lotus\lotus> write-host "Bonjour !"
Bonjour !
PS E:\Lotus\lotus> write-host 'Bonjour !'
Bonjour !
PS E:\Lotus\lotus>
```

Write-host permet d'écrire, a priori pas de différence entre " " et ' '.
Cependant il y'a une différence si nous travaillons avec des variables

- ◆ Les doubles guillemets remplace une variable par son contenu (nommé substitution de variable).
- ◆ Les guillemets simple ignorent les variables et conservent la chaine qu'ils contiennent

Les règles programmatiques

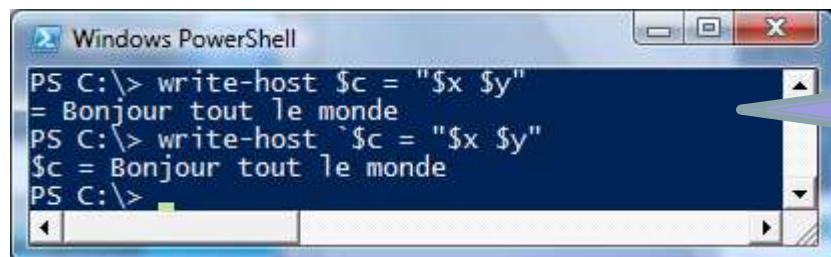


```
Windows PowerShell
PS C:\> $x = 'Bonjour'
PS C:\> $y = 'tout le monde'
PS C:\> write-host $x $y
Bonjour yy
PS C:\> write-host "$x $y"
Bonjour tout le monde
PS C:\> write-host '$x $y'
$x $y
PS C:\>
```

Vérification des différences entre " " et

' '

Si vous souhaitez un caractère spéciale dans une variable comme le \$ par ex, il devra être précédé du caractère spécial " ` " (backtick)



```
Windows PowerShell
PS C:\> write-host $c = "$x $y"
= Bonjour tout le monde
PS C:\> write-host `\$c = "$x $y"
$c = Bonjour tout le monde
PS C:\>
```

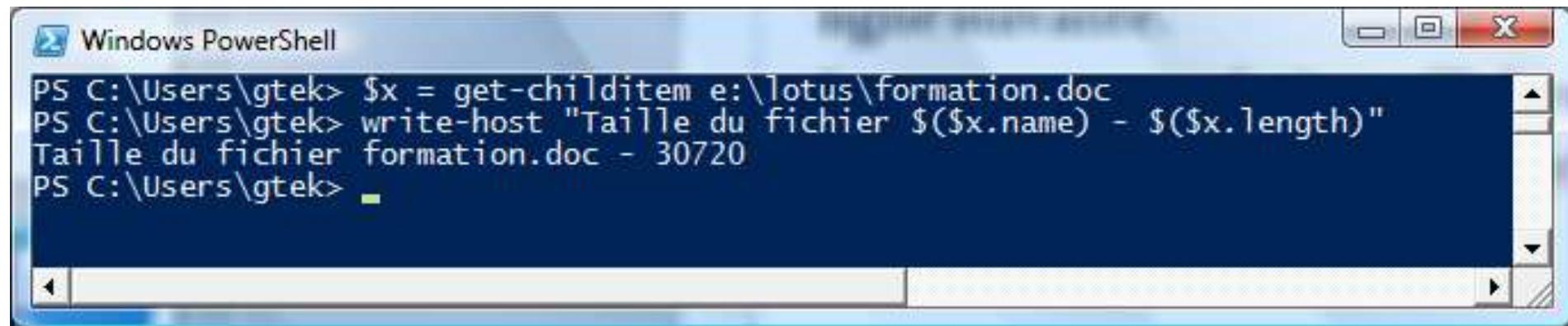
Avec et sans " ` " (équivalent de \ en C)

Les règles programmatiques

- Liste des caractère d'échappement et leurs descriptions
 - ◆ `n -> saut de ligne
 - ◆ `f -> saut de page
 - ◆ `r -> retour chariot
 - ◆ `a -> bip sonore
 - ◆ `b -> retour arrière
 - ◆ `t -> tabulation horizontale
 - ◆ `v -> tabulation verticale
 - ◆ `0 (zéro) -> Null ou espace
 - ◆ `"-> guillemet simple
 - ◆ `"" -> guillemet double
 - ◆ `` -> backtick simple

Les règles programmatiques

- Le backtick utilisé en fin d'une ligne de commande indique à PowerShell que la commande continue sur la ligne suivante.
 - ◆ Lorsque vous souhaitez afficher la propriété d'un objet, il faut utiliser la syntaxe : `$objet.propriété`.



The screenshot shows a Windows PowerShell window titled "Windows PowerShell". The command entered is:

```
PS C:\Users\gtek> $x = get-childitem e:\lotus\formation.doc
PS C:\Users\gtek> write-host "Taille du fichier $($x.name) - $($x.length)"
Taille du fichier formation.doc - 30720
PS C:\Users\gtek>
```

The output of the command is displayed below the command line, showing the size of the file "formation.doc" as 30720 bytes.

Les types et opérateurs

Variables et constantes

Introduction aux variables

- Powershell ne dispose pas d'un langage typé, vous n'avez pas besoin de définir et de typer vos variable (integer, string ...)
 - ◆ La syntaxe sera toujours :
 - \$Mavariable = valeur quelconque
 - ◆ Powershell reconnaîtra automatiquement les types de variables (string, entier ...)
 - ◆ **Dir variable:** (permet d'afficher toutes les variables)
 - ◆ **Del variable:test** (effacer la variable test)

```
PS C:\Users\gtek> $x = 10
PS C:\Users\gtek> $y = 20
PS C:\Users\gtek> $x + $y
30
PS C:\Users\gtek> $x = "Gaël"
PS C:\Users\gtek> $y = "Gtek"
PS C:\Users\gtek> $x + $y
GaëlGtek
PS C:\Users\gtek>
```

Cet exemple montre que
Powershell a reconnu les
types de variables

Introduction aux variables

- En utilisant la méthode GetType, PW vous indique le type de votre variable
 - ◆ Type reconnu :
 - **Int32** (entier sur 32 bits soit 4 Milliards de -2 à +2 milliards)
 - **Double**
 - **String** (chaine de caractère)

```
PS C:\Users\gtek> $x = 10
PS C:\Users\gtek> $x.GetType()
IsPublic IsSerial Name
----- ---- 
True     True    Int32
                                     BaseType
                                     System.ValueType

PS C:\Users\gtek> $x=10.58
PS C:\Users\gtek> $x.GetType()
IsPublic IsSerial Name
----- ---- 
True     True    Double
                                     BaseType
                                     System.ValueType

PS C:\Users\gtek> $x = "GTEK"
PS C:\Users\gtek> $x.GetType()
IsPublic IsSerial Name
----- ---- 
True     True    String
                                     BaseType
                                     System.Object
```

Introduction aux variables

- Il est parfois utile pour des traitements importants ou pour ne pas avoir d'erreur de calcul et pour une facilité de lecture des scripts de vouloir typer une variable.
 - ◆ Utiliser pour ceci avant votre variable :
 - [int]\$var
 - [char]\$var
 - [double]\$var
 - ◆ Ex: vous souhaitez que l'utilisateur saisisse un chiffre sur la console, cela se fait avec read-host, si vous saisissez dans votre code \$nbre, l'utilisateur peut saisir dix ou 10, le code script acceptera les 2.
 - ◆ Si vous entrez dans votre code [int]\$nbre, l'utilisateur ne peut plus que saisir du chiffre

```
PS C:\Users\gtek> $nbre = read-host 'entrez un nombre'  
entrez un nombre: dix  
PS C:\Users\gtek>  
PS C:\Users\gtek> [int]$nbre = read-host 'entrez un nombre'  
entrez un nombre: dix  
Impossible de convertir la valeur « dix » en type « System.Int32 ». Erreur : « Le fo  
Au niveau de ligne : 1 Caractère : 11  
+ [int]$nbre <<<= read-host 'entrez un nombre'  
PS C:\Users\gtek> [int]$nbre = read-host 'entrez un nombre'  
entrez un nombre: 10  
PS C:\Users\gtek> -
```

Ex avec et sans le [int]
En ajoutant [int],
l'utilisateur ne peut
saisir que du chiffre

Introduction aux variables

- Vous avez la possibilité d'utiliser également le typage pour convertir des chiffres ou inversement du code Ascii:
 - ◆ Ex : [char]\$x = 65
 - \$x affichera la lettre A majuscule
 - ◆ Ex: [int][char]\$x = "A"
 - \$x affichera le n° Ascii 65

```
PS C:\> [char]$x = 65
PS C:\> $x
A
PS C:\>
PS C:\>
PS C:\>
PS C:\> [int][char]$x = "A"
PS C:\> $x
65
PS C:\>
```

Conversion avec -f ou [System.convert]::ToString

Le Framework n'est pas limité au système décimal et il est possible d'utiliser ou de convertir au format :

- Hexadécimal
 - Octal (base 8)
 - Binaire
-
- ◆ Conversion du chiffre 1000 en octal
 - ◆ [system.convert]::ToString(chiffre, base)

```
PS C:\> $hex  
3E8  
PS C:\> $decimal = 1000  
PS C:\> $octal = [system.convert]::ToString($decimal,8)  
PS C:\> $octal  
1750  
PS C:\>
```

Conversion avec -f ou [System.convert]::ToString

- Conversion en binaire

```
PS C:\> $decimal = 1000
PS C:\> $binaire = [system.convert]::ToString($decimal,2)
PS C:\> $binaire
PS C:\> $binaire
1111101000
PS C:\>
```

- Conversion en hexadécimal

```
PS C:\> $decimal = 1000
PS C:\> $hex = [system.convert]::ToString($decimal,16)
PS C:\> $hex
3e8
PS C:\>
```

Variables prédéfinies – liste non exhaustive

- Powershell dispose d'un nombre de variable prédéfini
 - ◆ \$Error -> variable type tableau listant les erreurs affichées lors de la session
 - ◆ \$false -> contient faux
 - ◆ \$true -> contient vraie
 - ◆ \$home -> chemin du dossier de base de l'utilisateur
 - ◆ \$host -> information sur l'hôte

```
PS C:\> $host

Name          : ConsoleHost
Version       : 1.0.0.0
InstanceId    : 823103bb-4876-4a9e-ae62-d8863c5658a8
UI            : System.Management.Automation.Internal.Host.Internal
CurrentCulture : fr-FR
CurrentUICulture : fr-FR
PrivateData   : Microsoft.PowerShell.ConsoleHost+ConsoleColorProxy
```

Opérateurs arithmétiques

- Les opérateurs standard possibles sont :
 - ◆ + -> addition ou concaténation
 - ◆ - -> soustraction
 - ◆ * -> Multiplication
 - ◆ / -> Division
 - ◆ % -> Modulo

```
PS C:\Users\gtek> $x=10
PS C:\Users\gtek> $y = 4
PS C:\Users\gtek> $x*$y
40
PS C:\Users\gtek> $x/$y
2,5
PS C:\Users\gtek> $x%$y
2
PS C:\Users\gtek> $x-$y
6
PS C:\Users\gtek> $a= "LI"
PS C:\Users\gtek> $b = "LOU"
PS C:\Users\gtek> $a + $b
LILOU
PS C:\Users\gtek>
```

Opérateurs de comparaison

- Les opérateurs de comparaison régulièrement utilisés sont:
 - ◆ -like comparaison d'une chaîne (accepte les caractères générique comme *, ?)
 - ◆ -notlike comparaison d'inégalité
 - ◆ -match comparaison d'une chaîne (équivalent de contenu)

```
>>
PS C:\Users\gtek> "bonjour tout le monde" -like "*mon*"
True
PS C:\Users\gtek> "bonjour tout le monde" -like "*mon"
False
PS C:\Users\gtek> $mavar = "Bonjour tout le monde"
PS C:\Users\gtek> $mavar -like "*mon??""
True
PS C:\Users\gtek>
```

```
PS C:\Users\Administrateur> $env
PS C:\Users\Administrateur> $env:username
administrateur
PS C:\Users\Administrateur> "Bonjour tout le monde" -match "tout"
True
PS C:\Users\Administrateur> "Bonjour tout le monde" -match "toute"
False
PS C:\Users\Administrateur>
```

Opérateurs de plage

- Ils permettent de couvrir une plage
 - ◆ Les opérateurs de plage se note : ".." (point point).
 - ◆ Ex sur Powershell: 1 .. 5

```
PS C:\Users\gtek> 1..5
1
2
3
4
5
PS C:\Users\gtek> $x = 1
PS C:\Users\gtek> $y = 7
PS C:\Users\gtek> $x .. $y
1
2
3
4
5
6
7
PS C:\Users\gtek>
```

Opérateurs de remplacement

- Permet de remplacer tout ou une partie d'une chaîne
 - ◆ Ex: remplace "société RECA" par "société GTEK"
 - \$soc = "Société RECA"
 - \$soc -replace "RECA", "GTEK" ou \$soc.replace("RECA", "GTEK")

```
PS C:\Users\gtek> $soc = "Société RECA"
PS C:\Users\gtek> $soc -replace "RECA", "GTEK"
Société GTEK
PS C:\Users\gtek> $soc.replace("GTEK", "RECA")
Société RECA
PS C:\Users\gtek> ■
```

Opérateurs de type

Ils permettent de vérifier le typage d'une variable

Ex: \$x = 12

\$x -is [int] retourne true si x est un entier ou false dans le cas contraire

```
PS C:\Users\gtek> $x = 12
PS C:\Users\gtek> $x.GetType()
IsPublic IsSerial Name
-----
True     True     Int32

PS C:\Users\gtek> $x -is [int]
True
PS C:\Users\gtek> $x -is [string]
False
PS C:\Users\gtek> $x -is [boolean]
False
PS C:\Users\gtek> $x -is [string]
False
PS C:\Users\gtek> $x -is [double]
False
PS C:\Users\gtek> $x -is [decimal]
False
PS C:\Users\gtek> $x -is [int32]
True
PS C:\Users\gtek> ■
```

Opérateurs logique

- Permettent de comparer et vérifier une expression

- ◆ -and → Et logique
- ◆ -or → Ou logique
- ◆ -not ou ! → non logique
- ◆ -xor →ou exclusif

```
PS C:\Users\gtek> !(2 -eq 6)
True
PS C:\Users\gtek> (1 -eq 1) -or (2 -eq 6)
True
PS C:\Users\gtek> (1 -eq 1) -and (2 -eq 2)
True
PS C:\Users\gtek> (1 -eq 1) -and (2 -eq 6)
False
```

Opérateurs d'affectation

- Permet d'affecter une valeur à une variable

- ◆ \$x = \$x + 2 -> \$x += 2
- ◆ \$x = \$x - 2 -> \$x -= 2
- ◆ \$x = \$x * 2 -> \$x *= 2
- ◆ \$x = \$x / 2 -> \$x /= 2
- ◆ \$x = \$x % 2 -> \$x %= 2
- ◆ \$x = \$x + 1 -> \$x ++ -> \$x += 1
- ◆ \$x = \$x - 1 -> \$x-- -> \$x -= 1

```
PS C:\Users\gtek> $x = 1
PS C:\Users\gtek> $x += 1
PS C:\Users\gtek> $x
2
PS C:\Users\gtek> $x++
PS C:\Users\gtek> $x
3
PS C:\Users\gtek>
```

Commandes permettant de travailler sur des chaînes de textes

Opérateur	Description	Exemple
*	Répète une chaîne	"-" * 20
+	Concatène des chaînes	"Bonjour " + "Pierre"
-replace	Remplace une chaîne (Insensible à la case)	"Bonjour Pierre" -replace "Pierre","Paul"
-creplace	Remplace une chaîne (Sensible à la case)	"Bonjour Pierre" -replace "pierre","paul"
-eq	Vérifie l'égalité (Insensible à la case)	"Pierre" -eq "pierre"
-ceq	Vérifie l'égalité (Sensible à la case)	"Pierre" -ceq "pierre"
-like	Vérifie qu'une chaîne contient une autre chaîne (Insensible à la case, Accept *)	"Pierre" -like "Pier*"
-clike	Vérifie qu'une chaîne contient une autre chaîne (Sensible à la case, Accept *)	"Pierre" -clike "Pier*"
-notlike	Vérifie qu'une chaîne ne contient pas une autre chaîne (Insensible à la case, Accept *)	"Pierre" -notlike "Pier*"
-cnotlike	Vérifie qu'une chaîne contient une autre chaîne (Insensible à la case, Accept *)	"Pierre" -cnotlike "Pier*"
-match	Vérifie qu'une chaîne contient une autre chaîne (Insensible à la case)	"Pierre" -match "rr"
-cmatch	Vérifie qu'une chaîne contient une autre chaîne (Sensible à la case)	"Pierre" -cmatch "rr"
-notmatch	Vérifie qu'une chaîne ne contient pas une autre chaîne (Insensible à la case)	"Pierre" -notmatch "rr"
-cnotmatch	Vérifie qu'une chaîne contient une autre chaîne (Insensible à la case)	"Pierre" -cnotmatch "rr"

Les types et opérateurs

La gestion des dates

La gestion des dates

- La date et l'heure sont gérées par la cmdlet **get-date**
 - ◆ Un get-date retourne le jour, l'heure

```
PS E:\vista\cours\Powershell> get-date  
mercredi 10 décembre 2008 13:39:08
```

- Get-date est de type **dateTime** offrant un nombre de méthode et de propriété très importante.
 - ◆ **Get-date | get-member**

Name	MemberType	Definition
Add	Method	System.DateTime Add(TimeSpan value)
AddDays	Method	System.DateTime AddDays(Double value)
AddHours	Method	System.DateTime AddHours(Double value)
AddMilliseconds	Method	System.DateTime AddMilliseconds(Double value)
AddMinutes	Method	System.DateTime AddMinutes(Double value)
AddMonths	Method	System.DateTime AddMonths(Int32 months)
AddSeconds	Method	System.DateTime AddSeconds(Double value)
AddTicks	Method	System.DateTime AddTicks(Int64 value)
AddYears	Method	System.DateTime AddYears(Int32 value)
CompareTo	Method	System.Int32 CompareTo(Object value), System.Int32
Equals	Method	System.Boolean Equals(Object value), System.Boolean
GetDateTimeFormats	Method	System.String[] GetDateTimeFormats(), System.String
GetHashCode	Method	System.Int32 GetHashCode()
GetType	Method	System.Type GetType()
GetTypeCode	Method	System.TypeCode GetTypeCode()
get_Date	Method	System.DateTime get_Date()
get_Day	Method	System.Int32 get_Day()
get_DayOfWeek	Method	System.DayOfWeek get_DayOfWeek()
get_DayOfYear	Method	System.Int32 get_DayOfYear()
get_Hour	Method	System.Int32 get_Hour()
get_Kind	Method	System.DateTimeKind get_Kind()
get_Millisecond	Method	System.Int32 get_Millisecond()
get_Minute	Method	System.Int32 get_Minute()
get_Month	Method	System.Int32 get_Month()
get_Second	Method	System.Int32 get_Second()
get_Ticks	Method	System.Int64 get_Ticks()
get_TimeOfDay	Method	System.TimeSpan get_TimeOfDay()
get_Year	Method	System.Int32 get_Year()
IsDaylightSavingTime	Method	System.Boolean IsDaylightSavingTime()
Subtract	Method	System.TimeSpan Subtract(DateTime value), System.TimeSpan
ToBinary	Method	System.Int64 ToBinary()
ToFileTime	Method	System.Int64 ToFileTime()
ToFileTimeUtc	Method	System.Int64 ToFileTimeUtc()
ToLocalTime	Method	System.DateTime ToLocalTime()
ToLongDateString	Method	System.String ToLongDateString()
ToLongTimeString	Method	System.String ToLongTimeString()
ToOADate	Method	System.Double ToOADate()
ToShortDateString	Method	System.String ToShortDateString()
ToShortTimeString	Method	System.String ToShortTimeString()
ToString	Method	System.String ToString(), System.String ToString(S)
ToUniversalTime	Method	System.DateTime ToUniversalTime()
DisplayHint	NoteProperty	Microsoft.PowerShell.Commands.DisplayHintType Disp
Date	Property	System.DateTime Date {get;}
Day	Property	System.Int32 Day {get;}
DayOfWeek	Property	System.DayOfWeek DayOfWeek {get;}
DayOfYear	Property	System.Int32 DayOfYear {get;}
Hour	Property	System.Int32 Hour {get;}

La gestion des dates

- Propriétés:

- (get-date).hour -> retourne l'heure
- (get-date).minute -> retourne les mns
- (get-date).second -> retourne les secondes
- (get-date).month -> retourne le mois en cours
- (get-date).day -> retourne le jour (1 à 31)
- (get-date).DayOfWeek -> retourne le jour de la semaine (dimanche à Samedi)
- (get-date).DayOfYear -> retourne le jour de l'année (1 à 365).

```
PS E:\vista\cours\Powershell> (get-date).hour  
13  
PS E:\vista\cours\Powershell> (get-date).year  
2008  
PS E:\vista\cours\Powershell> (get-date).month  
12  
PS E:\vista\cours\Powershell> (get-date).day  
10  
PS E:\vista\cours\Powershell> (get-date).dayofweek  
Wednesday  
PS E:\vista\cours\Powershell> (get-date).dayofyear  
345  
PS E:\vista\cours\Powershell> (get-date).millisecond  
84  
PS E:\vista\cours\Powershell> (get-date).datetime  
mercredi 10 décembre 2008 13:49:14  
PS E:\vista\cours\Powershell> (get-date).timeofday  
  
Days : 0  
Hours : 13  
Minutes : 49  
Seconds : 26  
Milliseconds : 641  
Ticks : 497666418000
```

Format personnalisé

- En plus des formats standard, vous pouvez personnaliser les formats de date:

Format	Desc	Format	Desc
d	Nbre compris entre 1 et 31	M	Mois entre 1 et 12
dd	Nbre compris entre 01 et 31	MM	Mois entre 01 et 12
ddd	Jour abrégé (lun, mar, mer ...)	MMM	Mois abrégé (janv...)
dddd	Jour complet (lundi, mardi)	MMMM	Mois complet (Janvier...)
h	Heure compris entre 0-12	Y	Année abrégé 0 à 99
hh	Heure compris entre 00-12	YY	Année abrégé (00 à 99)
H	Heure compris entre 0-23	YYYY	Année complète (2008)
HH	Heure compris entre 00-23	s	Seconde 0 à 59
m	Minute entre 0-59	ss	Seconde de 00 à 59
mm	Minute entre 00-59		

Format personnalisé

```
PS E:\vista\cours\Powershell> get-date -format "dddd dd MMM yyyy ... HH:mm:ss"
mercredi 10 déc. 2008 ... 14:05:06
PS E:\vista\cours\Powershell> get-date -format "dddd dd MMM yyyy ... HH:mm:ss"
mercredi 10 déc. 2008 ... 14:05:07
PS E:\vista\cours\Powershell>
```

- Cet exemple crée un fichier nommé rapport_dateDuJour.txt
 - ◆ Ex: rapport_10-12-08.txt

```
PS E:\vista\cours\Powershell> $date = get-date -format "dd-MM-yy"
PS E:\vista\cours\Powershell> $date
10-12-08
PS E:\vista\cours\Powershell> new-item -type file c:\temp\rapport_$date.txt
```

Répertoire : Microsoft.PowerShell.Core\FileSystem::C:\temp

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
-a---	10/12/2008	14:10	0 rapport_10-12-08.txt

Manipulation de date

- La méthode la plus simple pour créer une date est d'utiliser Get-date
 - ◆ Get-date sans rien retourne la date système.
 - ◆ Si vous omettez un paramètre, c'est celui du système qui est pris, ici les heures minutes et seconde ont été pris de l'heure système

```
PS E:\vista\cours\Powershell> $date = get-date -year 09 -month 01 -day 01
PS E:\vista\cours\Powershell> $date
jeudi 1 janvier 0009 14:13:34
```

- ◆ Pour modifier une date, utiliser les méthode Add
 - Ajout de 2 mois par rapport à la variable \$date

```
PS E:\vista\cours\Powershell> $date.addMonths(2)
dimanche 1 mars 0009 14:13:34
```

Manipulation de date

- Ex: connaitre le jour de la semaine d'une date de naissance.

```
Thursday  
PS E:\vista\cours\Powershell> $date = [DateTime]"07/20/1969"  
PS E:\vista\cours\Powershell> $date.DayOfWeek  
Sunday  
PS E:\vista\cours\Powershell> $date  
dimanche 20 juillet 1969 00:00:00
```

- ◆ \$date est la variable récupérant la date
- ◆ [DateTime]"07/20/1969" récupère au format date la date du 20 juillet 1969 (attention format américain mm/jj/aa)
- ◆ \$date.DayOfWeek récupère le jour de la semaine

Calcul de temps entre les dates

- Le calcul du temps se fait avec la cmdlet ***new-timespan***
 - newTimeSpan (date-debut) (date_fin)
 - Ex: calcul le temps écoulé entre votre jour de naissance et la date du jour

```
PS E:\vista\cours\Powershell> $timer = new-timespan (get-date -year 1969 -month 07 -day 20) (get-date)
```

```
PS E:\vista\cours\Powershell> $timer

Days          : 14388
Hours         : 0
Minutes       : 0
Seconds       : 0
Milliseconds  : 1
Ticks         : 12431232000010000
TotalDays     : 14388,0000000116
TotalHours    : 345312,000000278
TotalMinutes  : 20718720,0000167
TotalSeconds  : 1243123200,001
TotalMilliseconds : 1243123200001
```

TotalMinutes	Property	System.Double Total
TotalSeconds	Property	System.Double Total
PS E:\vista\cours\Powershell> \$timer.days		
14388		
PS E:\vista\cours\Powershell> \$timer.days / 365		
39,4191780821918		

LES STRUCTURES DE CONTRÔLES ET FONCTIONS

Les structures de contrôles et fonctions

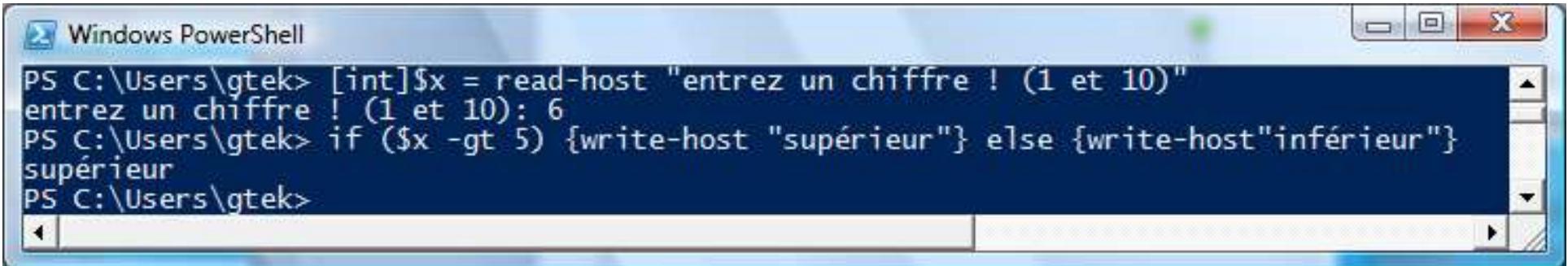
Les opérateurs if

L'opérateur if

- Permet de tester une condition et réaliser une action par rapport à cette condition.

- if (<test1>)
 - {<bloc_code1>}
 - [elseif (<test2>)
 - {<bloc_code2>}]
 - [else
 - <bloc_code3>}]

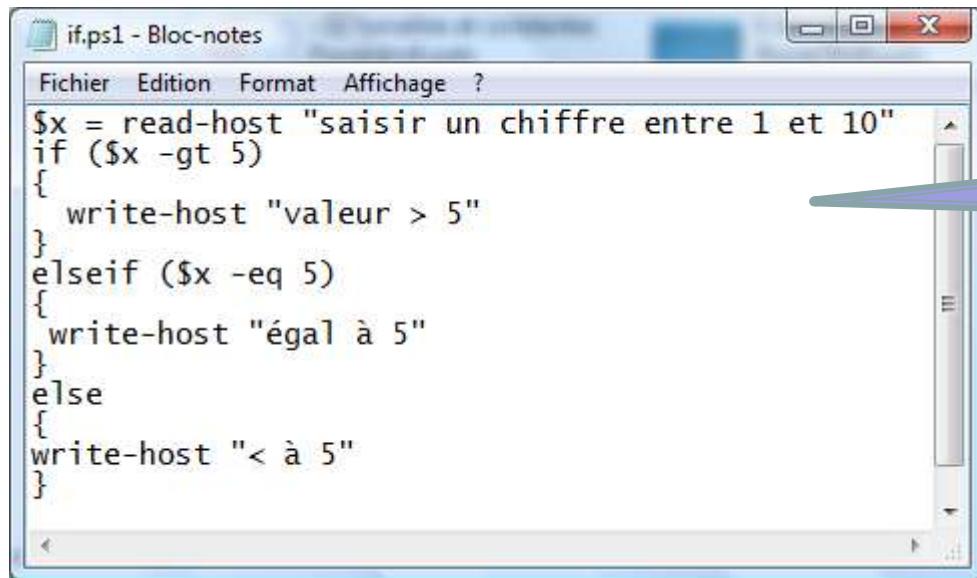
- Ex: entrez un chiffre entre 1 et 10, si le chiffre est supérieur à 5, entrez "supérieur" sinon "inférieur"



The screenshot shows a Windows PowerShell window titled "Windows PowerShell". The command entered is:

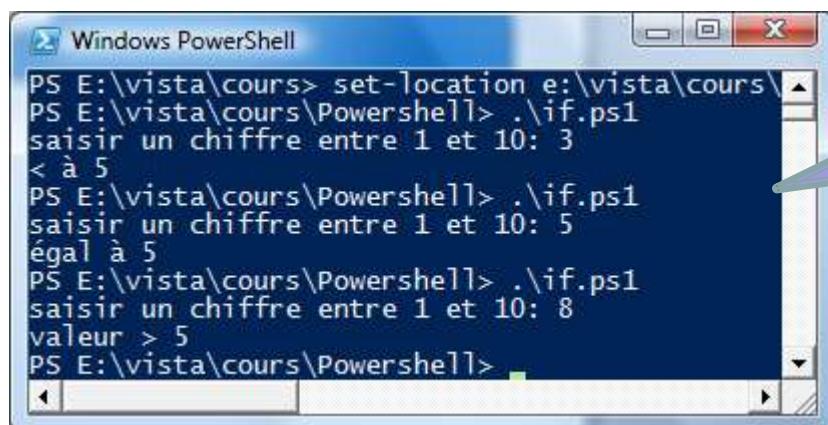
```
PS C:\Users\gtek> [int]$x = read-host "entrez un chiffre ! (1 et 10)"  
entrez un chiffre ! (1 et 10): 6  
PS C:\Users\gtek> if ($x -gt 5) {write-host "supérieur"} else {write-host "inférieur"}  
supérieur  
PS C:\Users\gtek>
```

L'opérateur if



```
$x = read-host "saisir un chiffre entre 1 et 10"
if ($x -gt 5)
{
    write-host "valeur > 5"
}
elseif ($x -eq 5)
{
    write-host "égal à 5"
}
else
{
    write-host "< à 5"
}
```

Autre exemple avec Elseif



```
PS E:\vista\cours> set-location e:\vista\cours\PowerShell> .\if.ps1
saisir un chiffre entre 1 et 10: 3
< à 5
PS E:\vista\cours\PowerShell> .\if.ps1
saisir un chiffre entre 1 et 10: 5
égal à 5
PS E:\vista\cours\PowerShell> .\if.ps1
saisir un chiffre entre 1 et 10: 8
valeur > 5
PS E:\vista\cours\PowerShell>
```

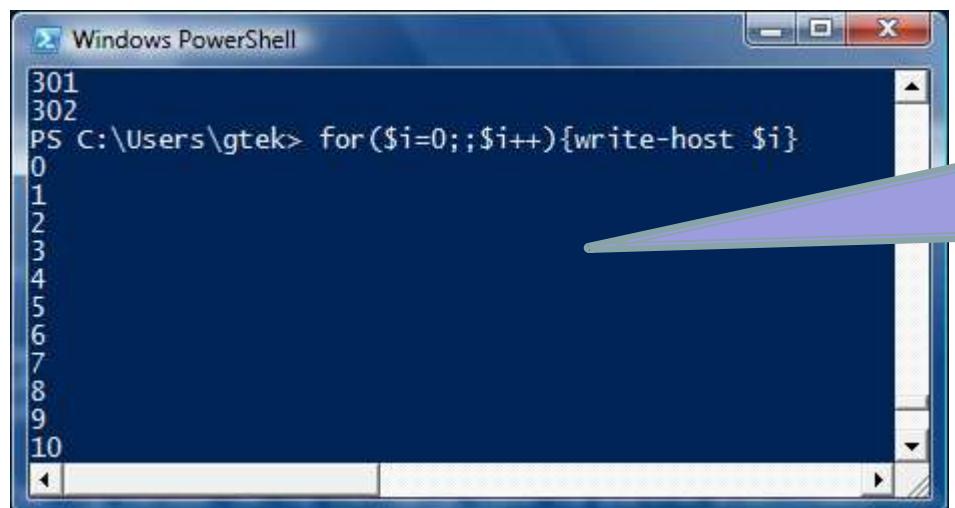
Pour exécuter le script, se placer
à l'endroit du script puis tapez
.\\if.ps1.

Les structures de contrôles et fonctions

Les boucles for, foreach

La boucle for

- L'instruction for crée une boucle exécutant les commandes d'un bloc de commandes tant qu'une condition spécifiée prend la valeur true (vrai).
 - ◆ for (<init>; <condition>; <récitation>)
 - {<bloc_commandes>}
 - ◆ **Init** initialise une variable = \$i=0
 - **Condition** \$i<20;
 - **Récitation** \$i++
 - For(i=0;i<20;i++)

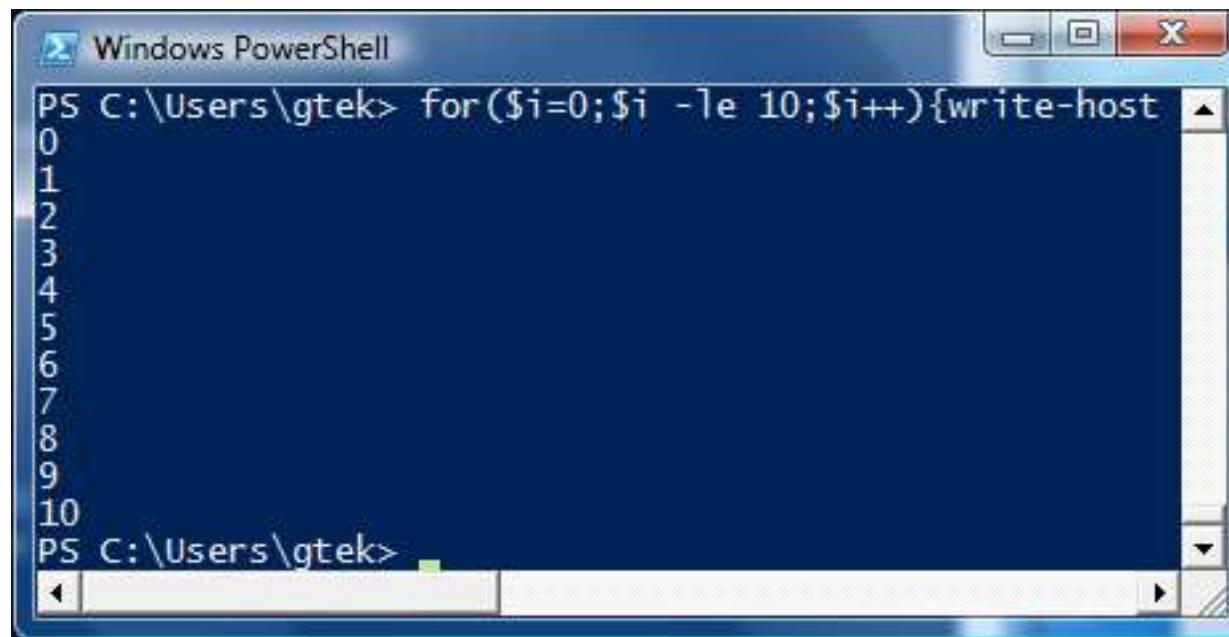


```
Windows PowerShell
301
302
PS C:\Users\gtek> for($i=0;;$i++){write-host $i}
0
1
2
3
4
5
6
7
8
9
10
```

Ce premier exemple crée une boucle infini en initialisant le compteur \$i à 0 et s'incrémentant de 1 (condition nulle)
CTRL + C pour stopper

La boucle for

- Cet exemple créé une boucle de 10 valeur –le (<=)
 - ◆ For($i=0;i \leq 10;i=i+1$)



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The command entered is:

```
PS C:\Users\gtek> for($i=0;$i -le 10;$i++){write-host $i}
```

The output shows the numbers 0 through 10, each on a new line, demonstrating the execution of the for loop.

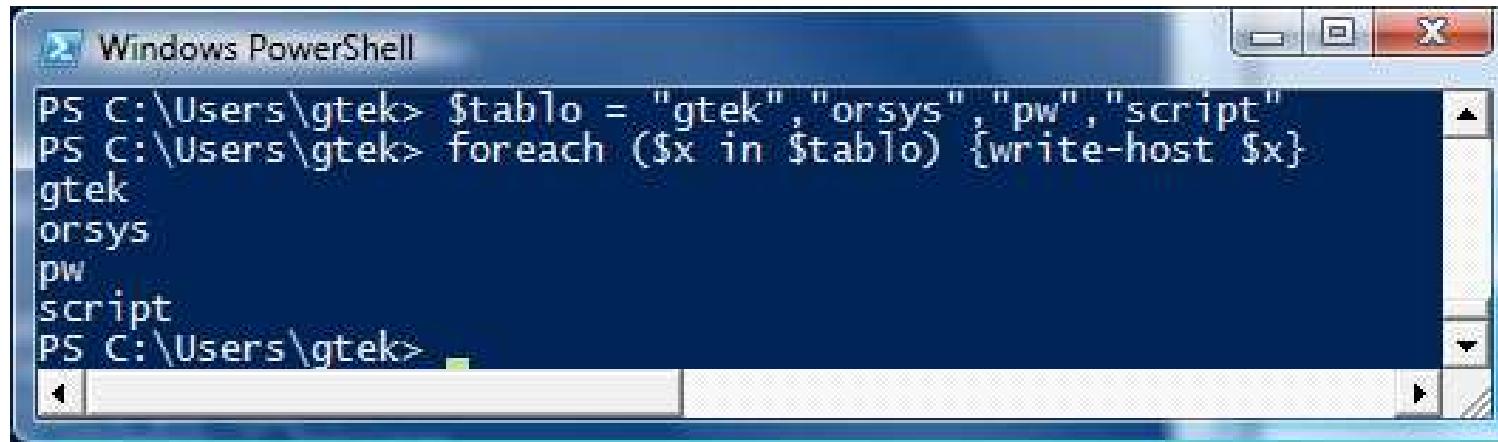
```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

Boucle foreach ou foreach-object

- Commande de langage permettant de parcourir tous les éléments d'une collection (stockée par ex dans un tableau).
- Le type de collection le plus simple et le plus classique à parcourir est le tableau.
- Dans une boucle foreach, il est fréquent d'exécuter une ou plusieurs commandes sur chaque élément d'un tableau.
 - ◆ Instruction :
 - foreach (\$<élément> in \$<collection>)
 - {<bloc_commandes>}

Boucle foreach ou foreach-object

- Cet exemple remplit des valeurs dans un tableau nommé \$tab.
 - ◆ Pour chaque élément du tableau on affiche la valeur stockée dans la tableau.
 - X est un simple pointeur permettant de scruter le tableau



The screenshot shows a Windows PowerShell window titled "Windows PowerShell". The command entered is:

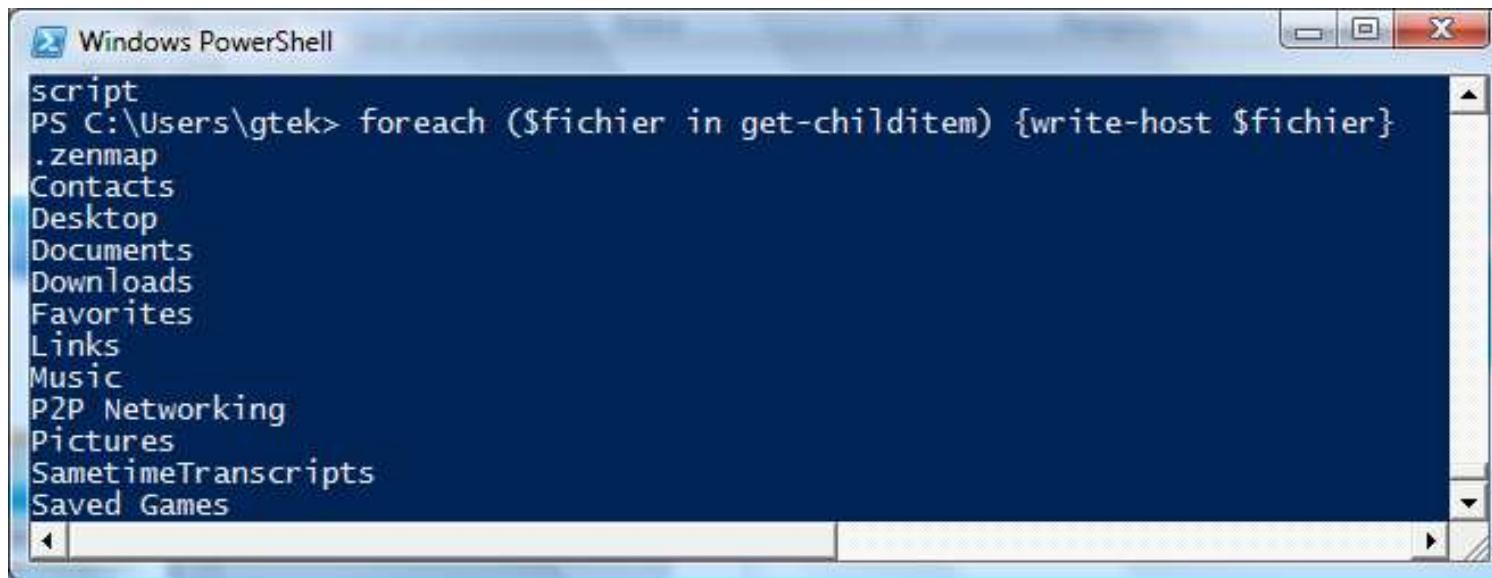
```
PS C:\Users\gtek> $tablo = "gtek", "orsys", "pw", "script"
PS C:\Users\gtek> foreach ($x in $tablo) {write-host $x}
```

The output displayed is:

```
gtek
orsys
pw
script
```

Boucle foreach ou foreach-object

- L'exemple suivant utilise ***get-childitem*** sur le dossier courant.
 - ◆ Pour chaque élément trouvé avec \$fichier, on affiche la valeur trouvé à l'écran.



The screenshot shows a Windows PowerShell window titled "Windows PowerShell". The command entered is:

```
script  
PS C:\Users\gtek> foreach ($fichier in get-childitem) {write-host $fichier}
```

The output displays the names of the files and folders in the current directory (C:\Users\gtek), including:

- .zenmap
- Contacts
- Desktop
- Documents
- Downloads
- Favorites
- Links
- Music
- P2P Networking
- Pictures
- SametimeTranscripts
- Saved Games

Instruction foreach

- Exemple de script boucle foreach

- ◆ L'instruction foreach parcourt un tableau (une collection), élément par élément, en attribuant une variable nommée spécifiquement pour l'élément actuel de la collection

```
# Obtenir une liste des membres du groupe Administrateurs du domaine
$DAdmins = Get-ADGroupMember "Domain Admins"
# Parcourir tous les membres et définir la description
foreach ($user in $DAdmins)
{
    Set-ADUser $user -Description "In the Domain Admins Group"
}
```

Boucle foreach ou foreach-object

- Foreach-Object permet également l'usage de 3 blocs
 - ◆ Un bloc begin exécuté une seule fois au début
 - ◆ Un bloc process exécuté à chaque entrée d'objet dans le pipeline
 - ◆ Un bloc end exécuté une seule fois à la fin
- L'exemple suivant retourne les adresses mac de type Ethernet de la machine locale

```
PS F:\W2K8\cours> gwmi win32_networkAdapter | where{$_.adapterType -match "eth"} | foreach-object{write-host "Adresse mac" -fore green}{$_.macaddress}{write-host "fin" -fore yellow}
Adresse mac
EC:55:F9:84:89:A1
EC:55:F8:81:98:A7
F6:55:F9:84:89:A1
fin
PS F:\W2K8\cours>
```

- ◆ Attention toutefois, le bloc begin et end ne sont pas transmis au pipeline suivant. Si par ex nous ajoutons | out-file c:\temp\mac.txt ne stockera pas le bloc début "Adresse mac" ni le bloc de fin "fin"

Les structures de contrôles et fonctions

Les fonctions

Les fonctions

- Une fonction est un ensemble d'instruction auquel est associé un nom
- On peut ensuite appeler cette fonction par son nom à plusieurs reprise permettant un gain de ligne de code et une clarté dans vos scripts
- Une fonction est constituée :
 - ◆ Un nom
 - ◆ Un type de portée (facultatif)
 - ◆ Un ou plusieurs argument (facultatif)
 - ◆ Un bloc d'instruction
- Une fonction à la structure suivante

Function [<portée> :] <nom de fonction> (<argument>)

{

param (<liste des paramètres>)

bloc d'instructions

}

Exemple de fonction

```
function creer-vm
{
    param (
        [string[]]$nomvm
    )

    foreach ($item in $nomvm)
    {
        #création d'un disque dur de différentiation
        new-vhd -ParentPath 'D:\VM\PARENT2K16-GUI.vhdx'
            -Path "D:\VM\2K16\$item.vhdx"
            -Differencing
            -SizeBytes 50GB

        #création d'une nouvelle VM
        new-vm -Name $item
            -generation 2
            -MemoryStartupBytes 2048MB
            -VHDPath "D:\VM\2K16\$item.vhdx"
            -SwitchName wifi
            -Path "D:\VM\2K16\$item"

        #démarrage des VM
        start-vm $item
    }
}

creer-vm -nomvm vm2016-1, vm2016-2
```

	Site1 - DS - CD core	Enregistre		
	vm2016-1	Exécution	11 %	1304 Mo
	vm2016-2	Exécution	11 %	1346 Mo

Les fonctions- les arguments

- Les arguments dans les fonctions permettent le passage de variables, de valeurs.
- Ils se placent derrière la fonction
 - ◆ Function
- Lorsque des arguments sont placés dans une fonction, ils sont stockés dans un tableau nommé \$args
 - ◆ \$args[0] correspond au premier argument
 - ◆ \$args[1] correspond au Deuxième argument

#L'exemple suivant utilise le tableau \$args pour passer les variables dans la fonction

```
function get-fic{  
Get-ChildItem $args[0] | where Length -gt $args[1]  
}  
get-fic C:\temp 100KB
```



Les structures de contrôles et fonctions

La gestion des fichiers

Gestion de fichiers

- La gestion des fichiers avec PS est plus simple que VBScript.
 - ◆ Il n'y'a pas besoin d'instancier les objets avec FileSystem ni de spécifier le mode d'accès (lecture, écriture).
 - ◆ Powershell utilise un jeu de commandelette dédié à la gestion des fichiers.
 - ◆ Powershell traite les fichiers texte en unicode de façon native contrairement à CMD qui ne traite que l'ASCII.
 - ◆ On peut forcer les commandelettes à utiliser d'autres encodages comme ASCII, UTF8, UTF32 ...

Envoi de données dans un fichier

- 3 cmdlets pour écrire des données dans un fichier :
 - ◆ Set-content
 - ◆ Out-file
 - ◆ Tee-object
- Lorsque out-file est utilisé, elle tente comme toutes les commandes out-* de formater le flux avant de l'écrire dans le fichier.
- Set-content ne cherche pas à formater le flux mais applique la méthode ToString pour écrire des caractères.
- Tee-object ressemble à out-file mais applique une double direction , on écrit dans le fichier et sur la console powershell.

Out-File

- Out-file permet de créer des fichiers, d'associer dans ce fichier créé un contenu.
 - ◆ Out-file utilise les paramètres suivant :

Paramètre	Description
Filepath	Fichier de destination
Encoding	Type d'encodage (par défaut unicode)
Append	Ajoute du contenu à un fichier existant
Width	Nombre de caractère par ligne
InputObject	Objet à écrire dans le fichier
noClobber	Indique de ne pas remplacer le fichier existant

Out-file

- Les paramètres d'encodage possibles sont:

Nom	Description
Ascii	Encodage ascii (jeu de caractères de 0 à 127, 7bits)
UTF7	Encodage unicode UTF7 (Unicode Transformation Format)
UTF8	Encode unicode UTF8
Unicode	Encode unicode UTF16 LittleEndian
BigEndianUnicode	Encode unicode UTF16 BigEndian
UTF32	Encode unicode UTF32
default	Code ANSI
OEM	Utilise l'identificateur de code du fabricant OEM actuel pour le système d'exploitation

- Microsoft travaille en unicode UTF32

Set-Content

- ***Set-content*** écrit les données telle qu'elle les reçoit sans les reformater.
- Cette commande écrit donc des octets dans un fichier quelque soit le type de fichiers (texte ou binaire).
- Set-content écrase le contenu du fichier de destination (pas de switch –append comme out-file).
- Set-content fait partie de la famille des commandelettes *-content :
 - ◆ Add-content (ajouter des données dans un fichier)
 - Add-content c:\temp\rpoc.jpg "nouvelle données..."
 - ◆ Get-content (lire le contenu d'un fichier)
 - Get-content code.txt
 - ◆ Clear-content (effacer les données d'un fichier)

Set-Content

- Les paramètres utilisables avec set-content sont :

Paramètre	Description
Path	Fichier de destination recevant les données
Value	Données à écrire
Include	Modifie uniquement les éléments spécifiés
Exclude	Omet les éléments spécifiés
Filter	Spécifie un filtre dans le format ou le langage du fournisseur
PAssThru	Passe l'objet créé à travers le pipeline
Force	Force la commande à réussir
Credential	Utiliser des informations d'identification pour valider l'accès au fichier
Encoding	Type d'encodage.

Recherche de contenu avec Select-String

- Permet de recherche dans un ou plusieurs fichiers une chaîne de caractère.
 - ◆ Connu sous unix comme commande grep
 - ◆ Paramètre de select string

Paramètre	Description
Pattern	Chaine ou Expression régulière à rechercher
Path	Cible de la recherche : chaîne ou fichier
Include	Récupère uniquement les éléments spécifiés
Exclude	Omet les éléments spécifiés
CaseSensitive	Respecte la casse
List	Spécifie qu'une seule correspondance doit être retourné par fichier
Quiet	Retourne une valeur booléenne vrai si la chaîne est trouvé

Recherche de contenu avec Select-String

- Exemple : recherche dans le dossier Temp tout les fichiers *.txt contenant le mot clé "formation"

```
PS E:\vista\cours\Powershell> select-string -path c:\temp\*.txt -pattern "formation"
C:\temp\a.txt:5:formation PW
C:\temp\extension.txt:1:powershell formation
```

- ◆ Retourne 2 fichiers contenant formation a.txt et extension.txt :5 indique le n° de ligne où se trouve le mot recherché
- Cet exemple permet de rechercher également dans les sous-dossiers le mot clé formation. Ajouter le | fl pour avoir un format sous forme de liste.

```
PS E:\vista\cours\Powershell> get-childitem c:\temp -include *.txt -recurse | select-string -pattern "formation"
C:\temp\temp1\TA - Copie.txt:2:windows 2008 - formation
C:\temp\temp1\TA.txt:2:formation dbut
C:\temp\a.txt:5:formation PW
C:\temp\extension.txt:1:powershell formation
```

- Ci-dessous expression régulière pour la recherche d'adresse

```
get-childitem *.ps1 | Select-String -pattern ("[0-9]{3}.[0-9]{3}.[0-9].[3]")
```

Recherche de contenu avec Select-String

- Affichage des processus en cours contenant les mot clé avg*
 - En tapant get-process « avg* », nous aurions obtenu les process aussi, affiché différemment.

```
PS E:\vista\cours\Powershell> get-process | select-string -pattern "avg*"  
System.Diagnostics.Process (avgrsx)  
System.Diagnostics.Process (avgscanx)  
System.Diagnostics.Process (avgtray)  
System.Diagnostics.Process (avgui)  
System.Diagnostics.Process (avgwdsvc)
```

- Cet exemple recherche tous les processus commençant par mst* puis arrête ces processus avec la commandelette stop-process

```
get-process mst* | stop-process
```

- Cet exemple force l'arrêt d'un processus n'appartenant pas à l'utilisateur

```
get-process lsass | stop-process -force
```

Les structures de contrôles et fonctions

La gestion des fichiers CSV

Gestion des fichiers csv

- Les fichiers CSV (Comma Separated Values) sont des fichiers texte séparés par des virgules.
- La première ligne représente souvent l'en-tête (nom des colonnes), les lignes suivantes représentent les données.
- Exemple :
 - ◆ Nom, prénom, fonction
 - ◆ Marot, Gaël, Consultant SI
 - ◆ Jegu, Pierrick, Consultant SCM
- Pour gérer les fichiers CSV, PW utiliser les cmdlets
 - ◆ [Export-csv](#)
 - ◆ [Import-csv](#)

Gestion des fichiers CSV

- Les paramètres pour **Export-csv** sont :

Paramètre	Description
Path	Chemin du fichier de destination
InputObject	Accepte un objet comme entrée
Force	Remplace le fichier si celui existe déjà
Encoding	Type d'encodage (voir Out-file). Par défaut Ascii
NoTypeInformation	Ne prend pas en compte l'en-tête
Nolobber	Ne pas écraser le fichier si il existe

- Les paramètres pour **import-csv** sont :

Paramètre	Description
Path	Chemin du fichier source

Gestion des fichiers CSV

- Cet exemple liste le journal des événements applications, et enregistre les 100 premiers événement dans un fichier Event.csv

```
#recherche des 1000 derniers evt du journal applicaton, puis un tri avec sort-objet sur la propriété instance id, suppression
#des doublons avec -unique, on sélectionne les propriétés timeGenerated, entryType...
#on exporte ensuite en csv avec un delimiter ";" l'argument noTypeInformation permet de supprimer la
#première ligne du fichier csv, on encode ensuite en utf8
Get-eventlog application -newest 1000 | Sort-Object -property instanceid -Unique |
select-object TimeGenerated, EntryType, Source, EventID, message |
export-csv c:\temp\event.csv -Delimiter ";" -NoTypeInformation -Encoding utf8
#lecture du fichier csv
import-csv C:\temp\event.csv -Delimiter ";"
```

```
TimeGenerated : 09/02/2017 16:24:08
EntryType     : Information
Source        : NvStreamSvc
EventID       : 2003
Message       : La description de l'ID d'événement '1073874899' dans la source 'NvStreamSvc' est introuvable. L'ordinateur local ne dispose
                Registre ou des fichiers DLL nécessaires à l'affichage du message, ou vous n'êtes peut-être pas autorisé à y accéder. Les i
                l'événement :'NvStreamSvc', 'EnsureNssProcessRunning [0]'

TimeGenerated : 12/02/2017 16:24:06
EntryType     : Warning
Source        : AutoEnrollment
EventID       : 64
Message       : La description de l'ID d'événement '-2147483584' dans la source 'AutoEnrollment' est introuvable. L'ordinateur local ne dis
                du Registre ou des fichiers DLL nécessaires à l'affichage du message, ou vous n'êtes peut-être pas autorisé à y accéder. Le
                à l'événement :'Système local', '71 35 0a 59 6b 73 be 40 5a 35 10 c3 c6 a9 21 01 94 ea e9 73'
```

Les structures de contrôles et fonctions

Export des données en tant que page HTML

Export des données en tant que page HTML

- La cmdlet **ConvertTo-HTML** permet de générer des pages HTML.
 - ◆ Les paramètres sont :

Paramètres	Description
Property	Propriété de l'objet passé en paramètre à écrire dans la page HTML
InputObject	Accepte un objet comme entrée
Body	Spécifie le texte à inclure dans la balise <Body>
Head	Spécifie le texte à inclure dans la balise <head>
Title	Spécifie le texte à inclure dans la balise <title>

Convertto-html

- Cette cmdlette permet de convertir du texte en html,
- L'exemple suivant utilise une feuille de style pour afficher la page html

```
get-service -Pipelinevariable service | ConvertTo-HTML -Property name, displayname, status
-Cssuri c:\temp\fic.css
-Title "service de l'os $($env:computername)"
-Body "<H1>Service de l'os</H1>" | %{

if($service.status -eq "Running")
{
    $_ -replace "<tr>", "<tr class=demarre>"
}

elseif($service.status -eq "Stopped")
{
    $_ -replace "<tr>", "<tr class=arret>"
}

else
{
    $_
}} | tee-object c:\temp\serv.html
ii c:\temp\serv.html
```

Feuille de style

```
20
21
22
23
24
25
26
27 .demarre {
28     background-color: #DADBC;
29     font-size: 12px;
30     color:red;
31 }
32 .arret {
33     background-color: #AA9BAE;
34     font-size: 12px;
35     color:blue;
36 }
```

Service de l'os		
name	displayname	status
AdobeFlashPlayerUpdateSvc	Adobe Flash Player Update Service	Stopped
AJRouter	Service de routeur AllJoyn	Stopped
ALG	Service de la passerelle de la couche Application	Stopped
AppIDSvc	Identité de l'application	Stopped
Appinfo	Informations d'application	Running
AppMgmt	Gestion d'applications	Stopped
AppReadiness	Préparation des applications	Stopped
AppVClient	Microsoft App-V Client	Stopped
AppXSvc	Service de déploiement AppX (AppXSVC)	Stopped
ASLDRService	ASLDR Service	Running
ATKGFNEXSrv	ATKGFNEX Service	Running

ConvertFrom-string

- Cette nouvelle cmdlette V5 permet de transformer une chaîne de texte en objet, en ajoutant des propriétés.
- L'exemple suivant permet de lire un fichier texte et d'ajouter une propriété « Ordinateur » et « adresse »

```
get-content C:\temp\pc.txt |  
ConvertFrom-String -Delimiter ";" -PropertyNames "Ordinateur","Adresse"
```

Ordinateur	Adresse
PC1	194.2.0.10
PC2	194.2.0.11
PC3	194.2.0.12
PC4	194.2.0.13

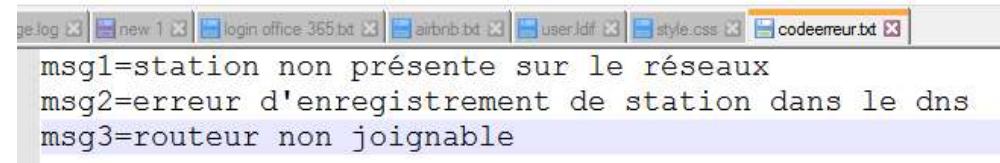
- L'exemple suivant permet de récupérer les données netstat et de créer des propriétés pour pouvoir utiliser les filtres

```
$a=netstat  
#on supprime les 5 premières lignes qui affichent des espaces et les entêtes  
#on utilise ensuite convertFrom-string, la propriété « t » ne sert à rien mais est créé avec  
convertfrom-string  
$a[4..$a.count] | ConvertFrom-String  
-PropertyNames "t","protocole","adlocale","addistante","état" |  
Select-Object protocole,adlocale,addistante,état | where état -Match established
```

ConvertFrom-stringData

- Convertit une chaîne contenant une ou plusieurs paires clé/valeur en une table de hachage.
- L'exemple suivant récupère dans une table de hash nommé alerte des codes erreurs stockés dans un fichier

```
$code=get-content C:\temp\codeerreur.txt -Encoding UTF8  
$alerte=$code | ConvertFrom-StringData  
write-warning "la première erreur est : $($alerte.msg1)"
```

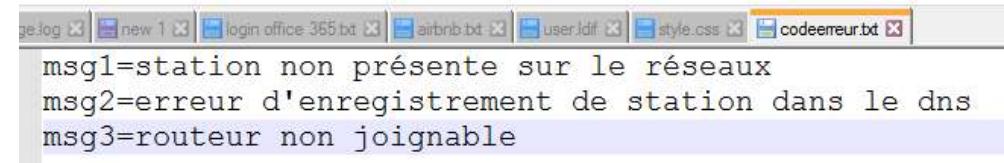


```
#ASTERISQUE-$code | ConvertFrom-StringData  
  
Write-Warning "la première erreur est : $($alerte.msg1)"  
AVERTISSEMENT : la première erreur est : station non présente sur le réseaux
```

ConvertFrom-stringData

- Convertit une chaîne contenant une ou plusieurs paires clé/valeur en une table de hachage.
- L'exemple suivant récupère dans une table de hash nommé alerte des codes erreurs stockés dans un fichier

```
$code=get-content C:\temp\codeerreur.txt -Encoding UTF8  
$alerte=$code | ConvertFrom-StringData  
write-warning "la première erreur est : $($alerte.msg1)"
```



```
#ASTERISQUE$code | ConvertFrom-StringData  
  
Write-Warning "la première erreur est : $($alerte.msg1)"  
AVERTISSEMENT : la première erreur est : station non présente sur le réseaux
```

Les structures de contrôles et fonctions

Scripts avec authentification

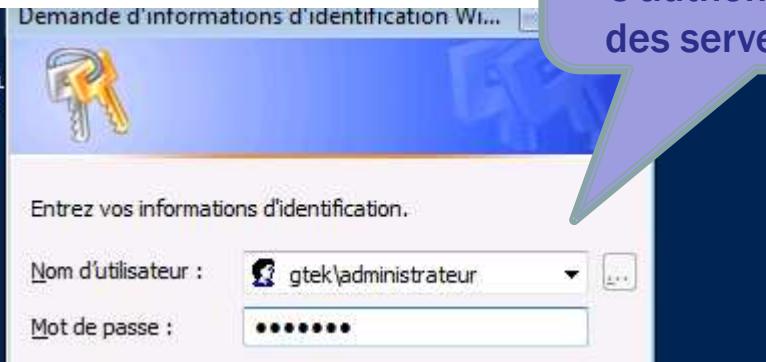
Scripts avec authentification

- Il est possible de retourner des informations ou d'agir sur un ordinateur distant
 - ◆ Vous devez au préalable vous authentifier
 - ◆ On utilise pour cela le "credential"
 - ◆ Le credential permet une authentification comme par ex:
 - interroger une liste de serveurs distants avec **get-wmiobject** sachant que notre propre compte n'ait pas accès à ces serveurs, on utilise alors la commande **get-credential** pour saisir un nouveau couple username/password, et à passer le résultat à **get-wmiobject** via son paramètre **-credential** :

Get-credential

- Tapez la commande suivante:
 - ◆ \$auth=get-credential (ouvrira la fenêtre d'authentification)

```
PS C:\Users\administ...> $auth=get-credential
Applet de commande Get-Credential à la position 1 du
Fournissez des valeurs pour les paramètres suivants
Credential
```



Saisir un login et mot de passe pour s'authentifier auprès des serveurs distants

- ◆ Nous pouvons maintenant utiliser get-wmiobject en utilisant notre authentification
 - \$auth=get-credential
 - **Get-wmiobject win32_logicalDisk –computername win-tcko58cnjlj -credential \$auth**

```
PS C:\Users\administ...> get-wmiobject win32_logicalDisk -computername win-tcko58cnjlj -credential $auth
```

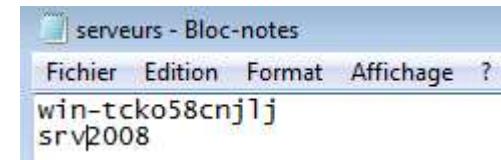
DeviceID	:	A:
DriveType	:	2
ProviderName	:	
FreeSpace	:	
Size	:	
VolumeName	:	
DeviceID	:	C:
DriveType	:	3
ProviderName	:	
FreeSpace	:	4726198272
Size	:	15055450112
VolumeName	:	

Get-credential

- Possibilité d'utiliser le paramètres –computername pour définir plusieurs serveurs

```
get-wmiobject win32_logicalDisk -computername win-tcko58cnj1j, srv2008 -credential $auth
```

- Cet exemple permet de s'authentifier sur des serveurs distants.
 - Le nom des serveurs est stocké dans un fichier texte nommé serveurs.txt appelé par la get-content
 - Pour chaque pc, nous retournons la liste des disques logiques.



```
titrel.ps1* X
$auth=get-credential
$listeSrv=get-content c:\part\serveurs.txt
foreach ($pc in $listeSrv)
{
    get-wmiObject win32_logicaldisk -computername $pc -credential $auth
}
```

Get-credential

- Possibilité de renseigner le nom de l'utilisateur et le domaine avec :
 - ◆ get-credential "domaine\nomutil".
 - ◆ La saisie d'un mot de passe avec get-credential n'est pas possible, nous devons passer par une autre commande



- Nous allons maintenant utiliser une authentification entièrement automatisée permettant une exécution du script sans besoin de renseigner un mot de passe (diapo suivante)

Convertto-securestring

- Cette commande doit être utiliser pour chiffrer le mot de passe avant de s'authentifier
 - ◆ **ConvertTo-SecureString** convertit une chaîne de texte standard en une chaîne sécurisée
 - ◆ La première étape consiste à chiffrer le mot de passe, la seconde permet l'authentification en passant le login et le mot de passe , nous faisons appel pour cela à un objet nommé system.management.PSCredential

```
$pass=convertto-secureString "az_1234" -asplainText -force
$auth= new-object -typename system.management.automation.PSCredential -argumentlist "gtek\administrateur", $pass
$listeSrv=get-content c:\part\serveurs.txt
foreach ($pc in $listeSrv)
{
    get-wmiObject win32_networkAdapterConfiguration -computername $pc -credential $auth
}
```

- ◆ On n'est pas obligé de mettre les paramètres –typename et -argumentlist

```
$pass=convertto-secureString "az_1234" -asplainText -force
$auth= new-object system.management.automation.PSCredential "gtek\administrateur", $pass
$listeSrv=get-content c:\part\serveurs.txt
foreach ($pc in $listeSrv)
{
    get-wmiObject win32_networkAdapterConfiguration -computername $pc -credential $auth
}
```

Les structures de contrôles et fonctions

Exécution de script à distance

Exécution de script à distance

- L'utilisation de PowerShell est une autre méthode permettant l'administration à distance,
 - ◆ soit grâce à l'argument computername,
 - ◆ soit via la commande invoke-command,
 - ◆ la commande enter-pssession ou l'argument cimsession.

- Cela nécessite la configuration de WinRm (Windows Remote Management) grâce à la commande suivante :

Enable-PSRemoting

- Cette commande :
 - Démarre le service WinRM et le définit en automatique,
 - Crée un écouteur WinRM pour accepter les demandes sur toute adresse IP,
 - Active une exception dans le parefeu...

Exécution de script à distance

- Il est possible d'ouvrir une session interactive à distance avec la commande Enter-PSSession :
 - ◆ Enter-PSSession -ComputerName Win8-01
 - ◆ [Win8-01]: PS C:\Users\Administrateur\Documents>
 - ◆ Vous devez spécifiez le nom de l'hôte tel qu'il apparait dans Active Directory ou dans la liste d'hôte de confiance (par défaut les IP ou les Alias DNS ne fonctionnent pas).

Exécution de script à distance

- Point à prendre en compte sur les connexions distantes :
 - ◆ Par défaut vous ne pouvez pas exécuter de commande qui initie une autre session distante à partir d'une session distante (second hop).
 - ◆ Vous ne pouvez pas lancer des applications graphiques.
 - ◆ Vous ne pouvez exécuter de commande qui possèdent leur propre shell tel que nslookup ou netsh.
 - ◆ La connexion est considérée comme "network logon" au même titre qu'un accès à un partage.
 - ◆ Rien de ce que vous effectuez n'est visible par un utilisateur en session interactive.

Exécution de script à distance

- Certaines commandes disposent de l'argument -computername permettant ainsi de l'exécuter à distance
 - ◆ Get-process -computername srv-1
 - ◆ Il est possible de transmettre une ou plusieurs commandes à plusieurs hôtes distants grâce à la commande Invoke-Command :
 - Invoke-Command -ScriptBlock { Get-Process -name s* } -computername localhost,win8-01
- ◆ Les commandes issues de la V3 disposent de l'argument cimsession qui permet d'exécuter la commande sur l'ordinateur distant :
 - Get-SmbShare -CimSession srv-1

Exécution de script à distance

- Il est possible d'ouvrir des sessions et de les réutiliser simplement
 - ◆ \$win8 = New-PSSession -ComputerName win8
 - ◆ \$domaincontrollers = New-PSSession -ComputerName win8,windc1
 - ◆ Enter-PSSession -Session \$win8
 - ◆ Invoke-Command -Session \$domaincontrollers -ScriptBlock { get-eventlog `
 - ◆ -LogName security -Newest 50 }
- Afficher la liste des sessions
 - ◆ Get-PSSession
- Utiliser une session existante
 - ◆ Enter-PSSession –Session (Get-PSSession –computername WINDC1)
 - ◆ Fermer toutes les sessions
 - ◆ Get-PSSession | Remove-PSSession
- Fermer une session
 - ◆ Get-PSSession -ComputerName win8 | Disconnect-PSSession

Exécution de script à distance

- Utilisation d'une commande avec argument computername :

Add-windowsfeature telnet-client -computer 2012-2

- Utilisation de invoke-command pour exécuter une commande à distance (équivalent de la commande précédente) :

Invoke-command -computer 2012-2 -script {Add-windowsfeature telnet-client }

- Utilisation d'un shell distant (équivalent de la commande précédente) :

enter-psession -computer 2012-2

[2012-2]:PS c:\>Add-windowsfeature telnet-client

- Utilisation d'une commande avec argument cimsession :

Get-Smbshare -cimsession 2012-2

UTILISATION DES CMDLETS ET DES MODULES

Utilisation des cmdlets et des modules

Le principe du module

Le principe du module

- Un module est un ensemble de fonctionnalités Windows PowerShell, c'est un package contenant :
 - ◆ des applets de commande
 - ◆ des fournisseurs
 - ◆ des scripts,
 - ◆ des fonctions
 - ◆ ...
- Un manifeste de module est un fichier qui contient .psd1 une table de hachage. Les clés et valeurs dans la table de hachage faire les choses suivantes:
 - ◆ Décrire le contenu et les attributs du module.
 - ◆ Définir les conditions préalables.
 - ◆ Déterminer comment les composants sont traités.
- Après avoir importé un module, vous pouvez l'utiliser dans votre session.
- Exemple :
 - ◆ import-module activeDirectory pour utiliser des cmdlettes dans l'ad
 - ◆ Depuis Windows 2012 (V3) il n'est plus forcément utile d'importer le module avant d'exécuter la commande

Le principe du module

- Pour lister les modules présent dans votre environnement tapez la commande
 - ◆ Get-module

```
PS C:\Users\Administrateur.LTCONSEIL> get-module
ModuleType Version Name
---- -- -- 
Manifest 3.1.0.0 Microsoft.PowerShell.Management
ExportedCommands
{Add-Computer, Add-Content, Checkpoint-Computer, Clear-Con...
```

- Pour lister les modules pouvant êtres traités dans votre environnement tapez la comdlte suivante
 - ◆ Get-Module -List

```
PS C:\Users\Administrateur.LTCONSEIL> get-module -list
Répertoire : C:\Windows\system32\WindowsPowerShell\v1.0\Modules

ModuleType Version Name
---- -- -- 
Manifest 1.0.0.0 ActiveDirectory
Manifest 1.0.0.0 ADDSDeployment
Manifest 2.0.0.0 AppLocker
Manifest 2.0.0.0 Appx
Manifest 1.0 BestPractices
Manifest 1.0.0.0 BitsTransfer
Manifest 1.0.0.0 BranchCache
Manifest 1.0.0.0 CimCmdlets
Manifest 1.0 DFSN
Manifest 1.0.0.0 DirectAccessClientComponents
ExportedCommands
{Add-ADCentralAccessPolicyMember, Add-ADComputerServiceAcc...
{Add-ADDSReadOnlyDomainControllerAccount, Install-ADDSFore...
{Get-AppLockerFileInformation, Get-AppLockerPolicy, New-Ap...
{Add-AppxPackage, Get-AppxPackage, Get-AppxPackageManifest...
{Get-BpaModel, Get-BpaResult, Invoke-BpaModel, Set-BpaResult}
{Add-BitsFile, Complete-BitsTransfer, Get-BitsTransfer, Re...
{Add-BCDataCacheExtension, Clear-BCCache, Disable-BC, Disa...
{Get-CimAssociatedInstance, Get-CimClass, Get-CimInstance, ...
{Get-DfsnRoot, Remove-DfsnRoot, Set-DfsnRoot, New-DfsnRoot...
{Disable-DAManualEntryPointSelection, Enable-DAManualEntry...
```

Utilisation des cmdlets et des modules

Gestion des archives

Les archives

- Un nouveau module permettant de gérer nativement les archives apparaît dans Powershell 5.0
 - ◆ Ce dernier permet de générer des archives (**Compress-Archive**) ou de les extraire (**Expand-Archive**).
 - ◆ Seul le format Zip est actuellement géré. Un paramètre nommé “**update**” permet de mettre à jour une archive existante en ajoutant seulement les nouveaux fichiers et les changements sur les fichiers déjà présents dans l'archive.
- Le paramètre “**path**” peut définir un ou plusieurs fichiers ainsi qu'un dossier entier en utilisant le caractère “*” (wildcard).
- Enfin le paramètre “**CompressionLevel**” permet d'influencer le taux de compression et la taille finale de l'archive.

Les archives

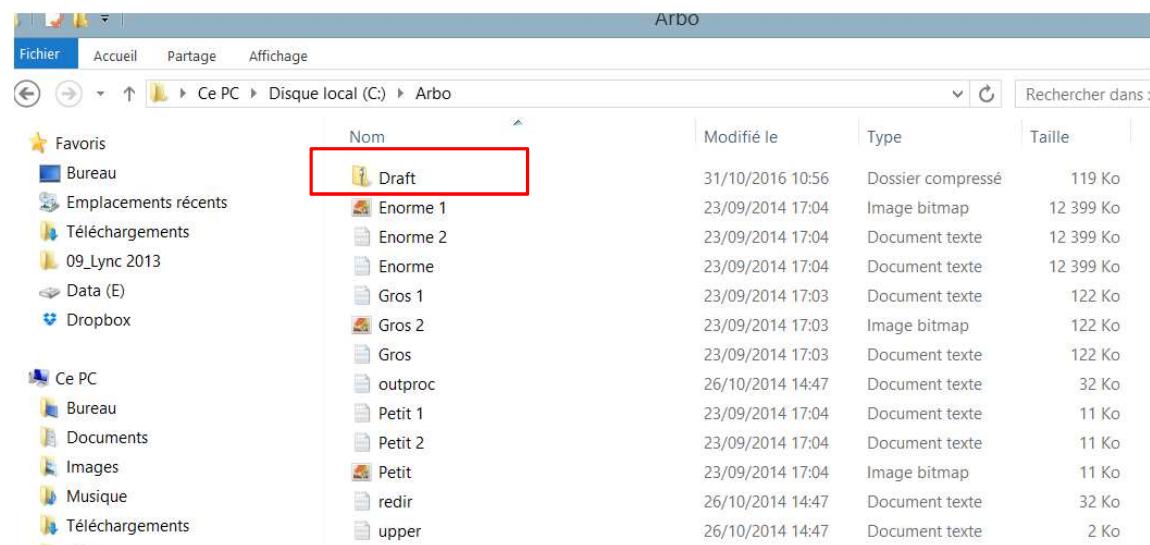
- Les cmdlets de gestion des archives :

```
PS C:\Users\LoicThomas> Get-Command -Module *archive*
```

CommandType	Name	Version	Source
Function	Compress-Archive	1.0.0.0	Microsoft.PowerShell.Archive
Function	Expand-Archive	1.0.0.0	Microsoft.PowerShell.Archive

- ◆ Compresser le contenu du dossier arbo

- `Compress-Archive -Path C:\arbo* -Update -DestinationPath C:\Arbo\Draft.Zip`



Utilisation des cmdlets et des modules

Les cmdlets de gestion web

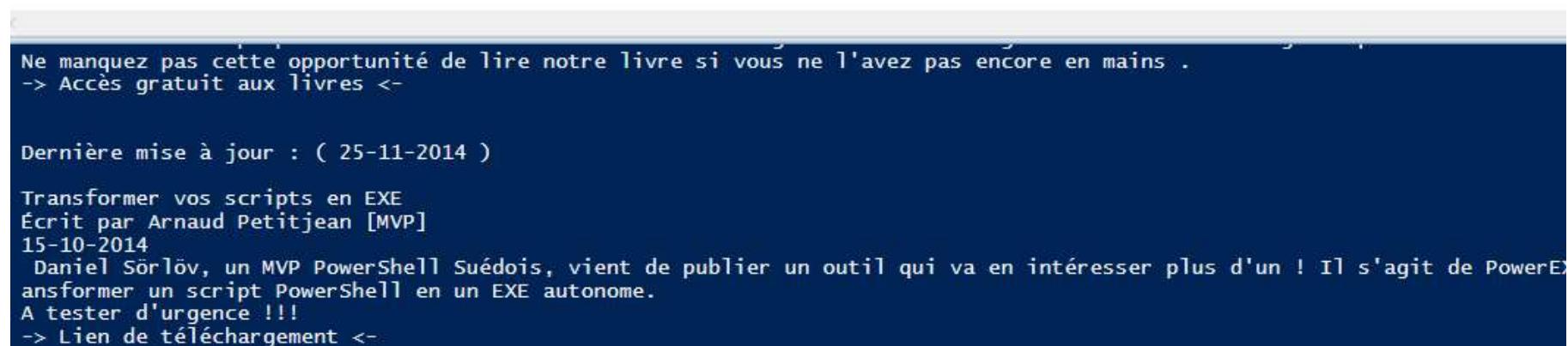
Les cmdlets de gestion web

▪ Invoke-WebRequest

- ◆ Cette cmdlets très puissante permet d'extraire des informations issues de page web.
- ◆ Le premier exemple très simple permet de récupérer en format text le contenu d'un site inter

```
$site = Invoke-WebRequest -URI "http://powershell-scripting.com/"  
$site.allElements.innertext
```

```
3 $site = Invoke-WebRequest -URI "http://powershell-scripting.com/"  
4 $site.allElements.innertext  
5
```



The screenshot shows the PowerShell command being run and its output. The output displays the HTML content of the specified website, including promotional text about a book, the last update date (25-11-2014), author information, and a note about transforming PowerShell scripts into EXE files.

```
Ne manquez pas cette opportunité de lire notre livre si vous ne l'avez pas encore en mains .  
-> Accès gratuit aux livres <-  
  
Dernière mise à jour : ( 25-11-2014 )  
  
Transformer vos scripts en EXE  
Écrit par Arnaud Petitjean [MVP]  
15-10-2014  
Daniel Sörlöv, un MVP PowerShell Suédois, vient de publier un outil qui va en intéresser plus d'un ! Il s'agit de PowerEX  
ansformer un script PowerShell en un EXE autonome.  
A tester d'urgence !!!  
-> Lien de téléchargement <-
```

Les cmdlets de gestion web

- Le second exemple permet d'utiliser des filtres et d'afficher les derniers numéro du loto

```
#"\d+" expression régulière le ^ indique le début d'une chaîne \d+$ indique que nous recherchons des chiffres positif
$Site = Invoke-WebRequest -URI "https://www.fdj.fr/jeux/jeux-de-tirage/loto/resultats"
#tagname retourne les tagues web comme H1, LINK, P, TABLE, TD, TR, BR, LI ...donc ici nous recherchons des tagues en <p>
$numLoto = $Site.AllElements | Where-Object {$_.tagName -eq "p"} | Where-Object { $_.InnerText -match "\d+" } | Select-Object -property InnerText -first 6
$numLoto
```

The screenshot shows a PowerShell window with the following content:

```
1 #"\d+" expression régulière le ^ indique le début d'une chaîne \d+$ indique que nous recherchons des chiffres positif
2 $Site = Invoke-WebRequest -URI "https://www.fdj.fr/jeux/jeux-de-tirage/loto/resultats"
3 #tagname retourne les tagues web comme H1, LINK, P, TABLE, TD, TR, BR, LI ...donc ici nous recherchons des tagues en <p>
4 $numLoto = $Site.AllElements | Where-Object {$_.tagName -eq "p"} | Where-Object { $_.InnerText -match "\d+" } |
5 Select-Object -property InnerText -first 6
6 $numLoto
```

innerText

```
1
9
14
20
36
2
```

Les cmdlets de gestion web

- ◆ Ajout de la date du dernier tirage du loto

```
#"^\\d+" expression régulière le ^ indique le début d'une chaîne \\d+$ indique que nous recherchons
des chiffres positif
$site = Invoke-WebRequest -URI "https://www.fdj.fr/jeux/jeux-de-tirage/loto/resultats"
#tagname retourne les tagues web comme H1, LINK, P, TABLE, TD, TR, BR, LI ...donc ici nous
recherchons des tagues en <p>
$numLoto = $site.AllElements | where-Object {$_ .tagName -eq "p"} | where-Object { $_ .InnerText -
match "^\\d+" } |
Select-Object -property InnerText -first 6
$dateLoto = $site.AllElements | where-Object {$_ .tagName -eq "H3"} | Select-Object -property
innertext -first 1
$dateLoto.innertext
write-host "date du dernier loto: " -nonewline
write-host $dateLoto.innertext -ForegroundColor Yellow
Write-host "les numéros gagnants: " -NoNewline
write-host $numLoto.InnerText -f Green
```

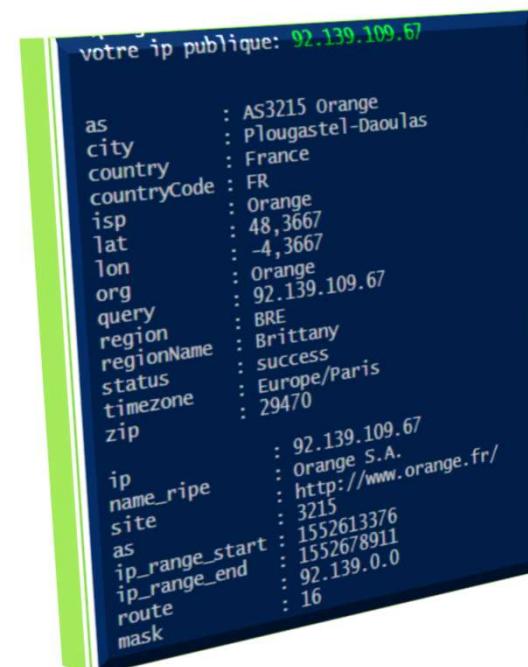
```
7   $dateLoto.innertext
8   write-host "date du dernier loto: " -nonewline
9   write-host $dateLoto.innertext -ForegroundColor Yellow
10  Write-host "les numéros gagnants: " -NoNewline
11  write-host $numLoto.InnerText -f Green|
```

```
date du dernier loto: Lundi 16 février 2015
les numéros gagnants: 1 9 14 20 36 2
```

Les cmdlets de gestion web

- L'exemple suivant recherche l'ip publique et affiche des informations détaillées sur votre FAI.

```
#connaitre son ip publique
$ipBox=Invoke-RestMethod -uri "https://api.ipify.org/?format=json"
write-host "votre ip publique: " -nonewline; write-host $ipBox.ip -ForegroundColor Green
#connaitre son FAI
$infoip=Invoke-RestMethod -Uri "http://ip-api.com/json/$(($ipBox.ip))"
$infoip
#information détaillé sur le FAI
$plage=Invoke-RestMethod -uri
"http://api.myspeed.today/provider.json?ip=$(($ipBox.ip))"
$plage
```

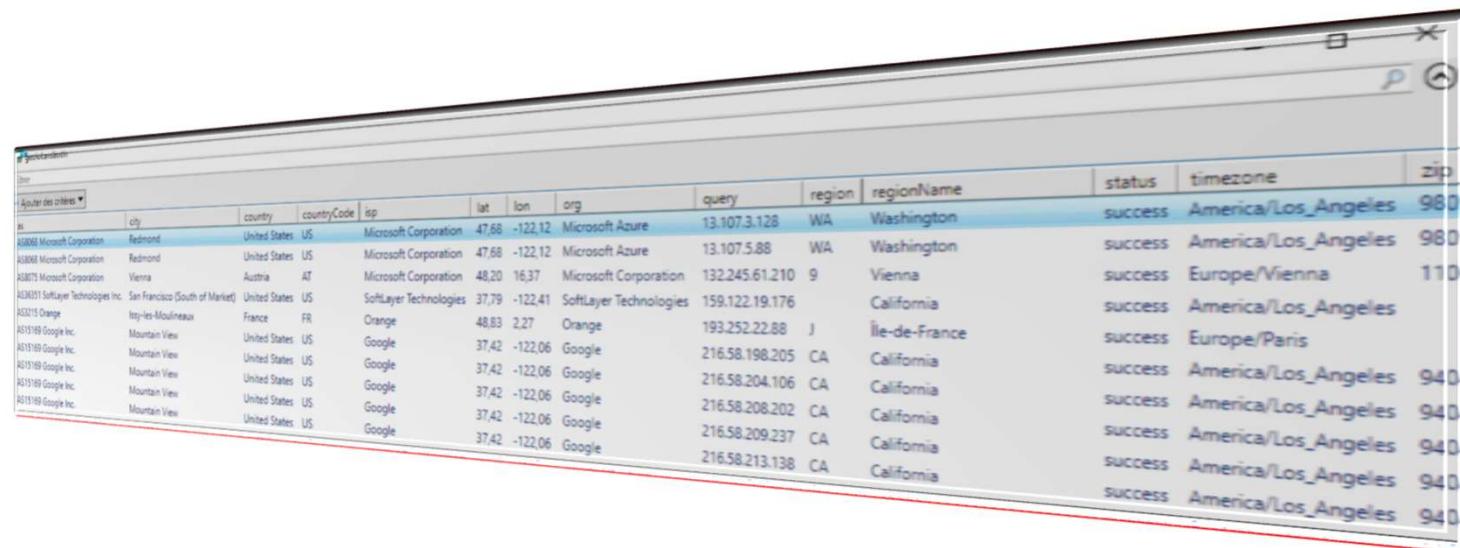


	:	
as	:	AS3215 Orange
city	:	Plougastel-Daoulas
country	:	France
countryCode	:	FR
isp	:	Orange
lat	:	48,3667
lon	:	-4,3667
org	:	Orange
query	:	92.139.109.67
region	:	BRE
regionName	:	Brittany
status	:	success
timezone	:	Europe/Paris
zip	:	29470
ip	:	92.139.109.67
name_ripe	:	Orange S.A.
site	:	http://www.orange.fr/
as	:	3215
ip_range_start	:	1552613376
ip_range_end	:	1552678911
route	:	92.139.0.0
mask	:	16

Les cmdlets de gestion web

- L'exemple suivant géolocalise les adresses ip internet sur votre pc
- Pour les essais, connectez vous à un site web
- On retire de nos tests les adresses locales en appliquant des regex (^10|^192... signifie ne doit pas commencer par 10 ou par 192...)
- On affiche le résultat dans une liste avec out-gridview

```
#information sur les adresses ip vers lequel votre poste pointe
(Get-NetTCPConnection -AppliedSetting internet | where {$_.LocalAddress -notmatch "^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}" -and $_.RemoteAddress -notmatch "^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}"} |
Sort-Object -Property remoteAddress -Unique |%{
Invoke-RestMethod -Uri "http://ip-api.com/json/$($_.remoteAddress)"
}
) | Out-GridView -Title "géolocalisation"
```



A screenshot of a PowerShell window displaying a grid of IP geolocation data. The data is presented in a table with the following columns:

id	city	country	countryCode	isp	lat	lon	org	query	region	regionName	status	timezone	zip
658068 Microsoft Corporation	Redmond	United States	US	Microsoft Corporation	47.68	-122.12	Microsoft Azure	13.107.3.128	WA	Washington	success	America/Los_Angeles	980
658068 Microsoft Corporation	Redmond	United States	US	Microsoft Corporation	47.68	-122.12	Microsoft Azure	13.107.5.88	WA	Washington	success	America/Los_Angeles	980
658075 Microsoft Corporation	Vienna	Austria	AT	Microsoft Corporation	48.20	16.37	Microsoft Corporation	132.245.61.210	9	Vienna	success	Europe/Vienna	110
652451 SoftLayer Technologies Inc.	San Francisco (South of Market)	United States	US	SoftLayer Technologies	37.79	-122.41	SoftLayer Technologies	159.122.19.176		California	success	America/Los_Angeles	
652315 Orange	Issy-les-Moulineaux	France	FR	Orange	48.83	2.27	Orange	193.252.22.88	J	Île-de-France	success	Europe/Paris	
651519 Google Inc.	Mountain View	United States	US	Google	37.42	-122.06	Google	216.58.198.205	CA	California	success	America/Los_Angeles	940
651519 Google Inc.	Mountain View	United States	US	Google	37.42	-122.06	Google	216.58.204.106	CA	California	success	America/Los_Angeles	940
651519 Google Inc.	Mountain View	United States	US	Google	37.42	-122.06	Google	216.58.208.202	CA	California	success	America/Los_Angeles	940
651519 Google Inc.	Mountain View	United States	US	Google	37.42	-122.06	Google	216.58.209.237	CA	California	success	America/Los_Angeles	940
								216.58.213.138	CA	California			

Utilisation des cmdlets et des modules

Les cmdlets pour gérer vos serveurs et vos stations de travail

Gestion des rôles et fonctionnalités

- Lister les rôles et fonctionnalités d'un ordinateur distant
 - ◆ *Get-windowsfeature -computer 2012-2*
- Ou pour pouvoir tout lire plus facilement
 - ◆ *Get-windowsfeature -computer 2012-2 | out-file features.txt*
- Ou pour avoir un tableau "intéragatif"
 - ◆ *Get-windowsfeature -computer 2012-2 | out-gridview*
- Installer un rôle ou fonctionnalité
 - ◆ *Add-windowsfeature nom_fonctionnalité -computer 2012-2*
- Supprimer un rôle ou fonctionnalité
 - ◆ *remove-windowsfeature nom_fonctionnalité -computer 2012-2*
 - ◆ L'argument `-IncludeAllSubFeature` permet d'inclure les sous-éléments.

Gestion des imprimantes

- Gestion des pilotes d'imprimantes :

- ◆ Connaître les pilotes d'imprimantes installés

`Get-PrinterDriver`

Name	PrinterEnvironment	MajorVersion	Manufacturer
Send to Microsoft OneNote 15 Driver	Windows x64	4	Microsoft
Microsoft XPS Document Writer v4	Windows x64	4	Microsoft
Epson ESC/P-R V4 Class Driver	Windows x64	4	EPSON
Canon Inkjet 0386 Class Driver	Windows x64	4	Canon
Remote Desktop Easy Print	Windows x64	3	Microsoft
PDFCreator	Windows x64	3	
Microsoft Shared Fax Driver	Windows x64	3	Microsoft
Microsoft enhanced Point and Pri...	Windows x64	3	Microsoft
HP Universal Printing PCL 6	Windows x64	3	HP
Canon MG2200 series Printer	Windows x64	3	Canon
Microsoft enhanced Point and Pri...	Windows NT x86	3	Microsoft

`Get-PrinterDriver -Name PDFCreator`

`Get-PrinterDriver -Name PDFCreator | fl`

Gestion des imprimantes

- ◆ Ajouter un pilote d'imprimante

- Stagger le drivers dans le magasin

```
pnputil -i -a C:\temp\driversHPPCL6\hpcu175u.inf
```

- ◆ Ajouter Drivers

```
Add-PrinterDriver -Name "HP Universal Printing PCL 6"
```

```
Get-PrinterDriver
```

- Supprimer un pilote d'imprimante

```
Remove-PrinterDriver -Name "HP Universal Printing PCL 6" -RemoveFromDriverStore
```

- ◆ RemoveFromDriverStore enlève le drivers du magasin

Gestion des imprimantes

- Gestion des ports d'imprimante
 - ◆ Obtenir la liste des ports

```
Get-PrinterPort | Select-Object name, Description
```

Gestion des imprimantes

- Ajouter port TCP/IP

```
Add-PrinterPort -Name "IP_192.168.0.50" -PrinterHostAddress  
192.168.0.51
```

```
Get-PrinterPort -Name "IP_192.168.0.51" | Format-List
```

```
PS C:\Windows\system32> Get-PrinterPort -Name "IP_192.168.0.51" | Format-List

Name          : IP_192.168.0.51
ComputerName   :
Description    : Port TCP/IP standard
Protocol      : RAW
PrinterHostAddress: 192.168.0.51
PrinterHostIP   :
PortNumber     : 9100
SNMPIndex      : 0
SNMPCommunity  :
SNMPEnabled    : False
LprQueueName   :
LprByteCounting: False
```

- Supprimer port TCP/IP

```
Remove-PrinterPort -Name "IP_192.168.0.50"
```

Gestion des imprimantes

- Ajouter une imprimante

```
Add-Printer -Name "HP LaserJet CP3525" -DriverName "HP  
Universal Printing PCL 6" -PortName "IP_192.168.0.50"
```

```
Get-Printer "HP LaserJet CP3525" | fl * | select-object  
Name,DriverName,PortName
```

- Modifier les paramètres d'une imprimante

```
Set-Printer -Name "HP LaserJet CP3525" -Comment "Imprimante HP  
ajoutée avec Windows PowerShell" -Location "Bureau 62"
```

- Supprimer une imprimante

```
Remove-Printer -Name "HP LaserJet CP3525"
```

- Définir une imprimante par défaut

```
(Get-WmiObject -Class Win32_Printer -Filter "Name='Microsoft  
XPS Document Writer'").SetDefaultPrinter()
```

Gestion des certificats

- Accéder aux certificats utilisateur
 - ◆ Set-Location Cert:
 - ◆ Get-ChildItem

```
PS C:\windows\system32> Set-Location Cert:  
PS Cert:> Get-ChildItem  
  
Location   : CurrentUser  
StoreNames : {TrustedPublisher, ClientAuthIssuer, Root, MSIEHistoryJournal...}  
  
Location   : LocalMachine  
StoreNames : {TrustedPublisher, ClientAuthIssuer, avast! Mail Scanner Trusted, Remote Desktop...}
```

- créer certificats auto signé:

```
New-SelfSignedCertificate -DnsName WIN81-loic.ltconseil.fr -  
CertStoreLocation Cert:\LocalMachine\My
```

```
PS Cert:> New-SelfSignedCertificate -DnsName WIN81-loic.ltconseil.fr -CertStoreLocation Cert:\LocalMachine\My  
  
Répertoire : Microsoft.PowerShell.Security\Certificate::LocalMachine\My  
  
Thumbprint          Subject  
-----  
9058529151EBDA070EA977A77410BCD6215F79FE  CN=WIN81-loic.ltconseil.fr
```

Gestion des certificats

- Exporter un certificat :

```
Export-Certificate -Cert  
Cert:\LocalMachine\My\92990111298BC73FBADAF2212BE0424AE20E93DB  
-FilePath C:\Users\LoicThomas\Desktop\certloic.cer
```

- Exporter un ensemble de certificats dans un seul fichier

```
Get-ChildItem -Path Cert:\LocalMachine\ My | Export-Certificate  
-FilePath C:\Temp\myCertsMachine.sst -Type SST
```

- Exporter un certificat avec une clé privée

```
$motdepasse = ConvertTo-SecureString "Password1" -AsPlainText -  
Force  
Get-ChildItem  
Cert:\LocalMachine\My\92990111298BC73FBADAF2212BE0424AE20E93DB  
| Export-PfxCertificate -FilePath  
C:\Users\LoicThomas\Desktop\certloic.cer -Password  
$motdepasse
```

Gestion des certificats

- Importer des certificats

```
Import-Certificate -FilePath C:\Temp\myCertificate.cer -  
CertStoreLocation Cert:\LocalMachine\My
```

- Importer un ensemble de certificats

```
$certificats = Get-ChildItem -Path C:\Temp\Certificats  
$certificats | Import-Certificate -CertStoreLocation  
Cert:\LocalMachine\My
```

- Importer un certificat au format PFX

```
$motdepasse = ConvertTo-SecureString "Password1" -AsPlainText -Force  
Import-PfxCertificate -FilePath C:\Temp\myPfxCert.pfx  
Cert:\LocalMachine\My -Password $motdepasse
```

Gestion des tâches planifiés

- Ces tâches peuvent être déclenchés pour démarrer automatiquement, ou être effectués selon une planification récurrente.
- Lorsqu'un travail planifié est créé, il est enregistré sur le disque, puis inscrit dans le Planificateur de tâches.
- La seule différence entre les tâches planifiés et les job en arrière-plan est que les travaux planifiés enregistrent leurs résultats sur le disque.
 - ◆ Register-ScheduledJob –Name LastLogonJob –FilePath c:\script\LastLogon.ps1

```
PS C:\Users\Administrateur.LTCONSEIL> Register-ScheduledJob -Name LastLogonJob -FilePath c:\script\LastLogon.ps1
```

Id	Name	JobTriggers	Command	Enabled
1	LastLogonJob	0	c:\script\LastLogon.ps1	True

Gestion des tâches planifiés

- Pour que le travail planifié puisse s'exécuter, une planification ou un déclencheur doivent être définis.
 - ◆ Les déclencheurs sont créés à l'aide de l'applet de commande:
 - **New-JobTrigger**.
- Avec cette applet de commande, vous pouvez utiliser l'applet de commande **Add-JobTrigger** pour ajouter le déclencheur à un travail planifié déjà inscrit, ou utilisez-la pour attribuer un déclencheur lorsqu'un nouveau travail planifié est inscrit.
- L'exemple suivant montre comment créer un déclencheur qui s'exécute tous les lundis et vendredis à 9 h 00 du matin, puis inscrit le nouveau travail planifié avec le déclencheur

```
$Trigger = New-JobTrigger –Weekly –DaysOfWeek Monday,Friday –At 9:00AM  
Register-ScheduledJob –Name ScheduledLastLogonJob –FilePath `  
c:\script\LastLogon.ps1 -Trigger $Trigger
```

```
PS C:\Windows\system32> $Trigger = New-JobTrigger –Weekly –DaysOfWeek Monday,Friday –At 9:00AM  
Register-ScheduledJob –Name ScheduledLastLogonJob –FilePath `  
c:\script\LastLogon.ps1 -Trigger $Trigger
```

Id	Name	JobTriggers	Command	Enabled
1	ScheduledLas...	1	c:\script\LastLogon.ps1	True

Gestion des tâches planifiés

- Vous pouvez également utiliser l'applet de commande **Add-JobTrigger** pour modifier un travail planifié existant, comme l'illustre l'exemple suivant:

```
Add-JobTrigger -Name LastLogonJob -Trigger `  
  (New-JobTrigger -Daily -At 9:00AM)
```

- Les travaux planifiés peuvent être utilisés pour exécuter automatiquement la tâche de création des rapports, de vérification des paramètres de configuration, de réalisation de la maintenance des utilisateurs et des groupes, et à de nombreuses autres fins

Utilisation des cmdlets et des modules

Les cmdlets de gestion du réseaux

Les cmdlets de gestion du réseaux

- Get-netroute permet d'obtenir la table de routage de la station travail ou d'une station distante.

Get-NetRoute

ifIndex	DestinationPrefix	NextHop	RouteMetric	PolicyStore
13	255.255.255.255/32	0.0.0.0	256	ActiveStore
17	255.255.255.255/32	0.0.0.0	256	ActiveStore
11	255.255.255.255/32	0.0.0.0	256	ActiveStore
4	255.255.255.255/32	0.0.0.0	256	ActiveStore
1	255.255.255.255/32	0.0.0.0	256	ActiveStore
13	224.0.0.0/4	0.0.0.0	256	ActiveStore
17	224.0.0.0/4	0.0.0.0	256	ActiveStore
11	224.0.0.0/4	0.0.0.0	256	ActiveStore
4	224.0.0.0/4	0.0.0.0	256	ActiveStore

- Get-NetRoute -DestinationPrefix 0.0.0.0/0

```
1 Get-NetRoute -DestinationPrefix 0.0.0.0/0
```

PS C:\Users\gaeloum> Get-NetRoute -DestinationPrefix 0.0.0.0/0				
ifIndex	DestinationPrefix	NextHop	RouteMetric	PolicyStore
4	0.0.0.0/0	192.168.82.254	256	ActiveStore
11	0.0.0.0/0	192.168.43.1	0	ActiveStore

```
New-NetRoute -DestinationPrefix 192.168.2.0/24  
-NextHop 192.168.2.254  
-RouteMetric 256  
-InterfaceIndex 4  
-CimSession MW12100
```

Création d'une nouvelle entrée dans la table de routage du serveur MW12100

Les cmdlets de gestion du réseaux

- Pour supprimer une entrée dans la table de routage

```
remove-netroute -DestinationPrefix 192.168.2.0/24 -  
Confirm:$false
```

- Find-netroute permet de trouver une route pour une destination

- Ex, trouver une route pour l'adresse internet 194.2.0.20

```
find-netroute -RemoteIPAddress 194.2.0.20
```

11 Find-NetRoute -RemoteIPAddress 192.168.82.1	
Caption	
Description	
ElementName	
InstanceID	:8:8:8:9:55>55;C<8;@B8B<8<?>55;
AdminDistance	
DestinationAddress	
IsStatic	
RouteMetric	256
TypeOfRoute	3
AddressFamily	IPv4
CompartementId	1
DestinationPrefix	0.0.0.0/0
InterfaceAlias	vEthernet (éthernet)
InterfaceIndex	4
NextHop	192.168.82.254
PreferredLifetime	10675199.02:48:05.4775807
Protocol	NetMgmt
Publish	No
Store	ActiveStore
ValidLifetime	10675199.02:48:05.4775807
PSComputerName	
ifIndex	4

Les cmdlets de gestion du réseaux

- Plusieurs cmdlettes permettent de tester les connexions distantes soit par les protocoles icmp ou tcp
- Test-connection permet d'effectuer un test de ping sur une ou plusieurs stations

```
Test-Connection 194.2.0.20 -Count 1  
-BufferSize 200
```

- Ce premier test envoi un seul ping (count) à l'adresse ip 194.2.0.20 avec une taille de 200 octets

Source	Destination	IPV4Address	IPV6Address	Bytes	Time(ms)
PCGT	194.2.0.20	194.2.0.20		200	6

- Le second test effectue à partir de plusieurs sources un test ip

```
#pinguer à partir des différentes adresses source l'adresse computername
```

```
$auth=Get-Credential administrateur
```

```
Test-Connection -Source 192.168.82.1, 192.168.82.4, 192.168.82.100  
-ComputerName 192.168.82.1  
-Credential $auth  
-Count 1
```

Source	Destination	IPV4Address	IPV6Address	Bytes	Time(ms)
MW821	192.168.82.1	192.168.82.1		32	0
Test-Connection : Le serveur RPC n'est pas disponible. (Exception de HRESULT : 0x800706BA)					
Au caractère Ligne:3 : 1					
+ Test-Connection -Source 192.168.82.1, 192.168.82.4, 192.168.82.100					
+					
+ CategoryInfo : InvalidOperationException : (:) [Test-Connection], COMException					
+ FullyQualifiedErrorId : TestConnectionException,Microsoft.PowerShell.Commands.TestConnectionCommand					
MW82100	192.168.82.1	192.168.82.1		32	1

Les cmdlets de gestion du réseaux

- L'exemple suivant testent plusieurs sources et plusieurs destinations

```
$auth=Get-Credential administrateur
```

```
Test-Connection -Source 192.168.82.1, 192.168.82.3, 192.168.82.100  
-ComputerName 194.2.0.20, 192.168.82.1  
-Credential $auth  
-Count 1
```

Source	Destination	IPV4Address	IPV6Address	Bytes	Time(ms)
Mw821	194.2.0.20	194.2.0.20		32	4
Mw821	192.168.82.1	192.168.82.1		32	0
Mw823	194.2.0.20	194.2.0.20		32	4
Mw823	192.168.82.1	192.168.82.1		32	0
Mw82100	194.2.0.20	194.2.0.20		32	5
Mw82100	192.168.82.1	192.168.82.1		32	1

Les cmdlets de gestion du réseaux

- La cmdlette test-netconnection permet de tester les connexions sur des ports

```
Test-netconnection www.google.com -informationlevel detailed
```

```
Test-NetConnection www.google.fr -TraceRoute
```

```
Test-NetConnection www.google.fr -Port 80
```

#ou

```
Test-NetConnection www.google.fr -CommonTCPPort HTTP
```

```
Test-NetConnection smtp.orange.fr -port 25 -InformationLevel Detailed
```

24
25

```
Test-NetConnection www.google.fr -TraceRoute
```

```
ComputerName      : www.google.fr
RemoteAddress    : 172.217.16.67
InterfaceAlias   : vEthernet (éthernet)
SourceAddress    : 192.168.82.199
PingSucceeded    : True
PingReplyDetails (RTT) : 4 ms
TraceRoute       : 192.168.82.254
                  10.0.0.1
                  109.190.215.253
                  91.121.128.104
                  0.0.0.0
                  209.85.251.28
                  66.249.94.77
                  172.217.16.67
```

```
PS F:\Documents\OneDrive\Documents\script powershell\Réseaux> Test-NetConnection www.google.fr -Port 80 -InformationLevel Quiet
```

True

```
PS F:\Documents\OneDrive\Documents\script powershell\Réseaux> Test-NetConnection www.google.fr -Port 8025 -InformationLevel Quiet
```

AVERTISSEMENT : TCP connect to www.google.fr:8025 failed
False

Les cmdlets de gestion du réseaux

- Get-netTcpconnexion permet d'obtenir toutes les connexions tcp sur votre poste

Get-NetTCPConnection

LocalAddress	LocalPort	RemoteAddress	RemotePort	State	AppliedSetting	OwningProcess
0.0.0.0	54086	0.0.0.0	0	Bound		7324
0.0.0.0	54085	0.0.0.0	0	Bound		7324
0.0.0.0	54084	0.0.0.0	0	Bound		6880
0.0.0.0	54081	0.0.0.0	0	Bound		6880
0.0.0.0	54080	0.0.0.0	0	Bound		6880
0.0.0.0	54077	0.0.0.0	0	Bound		14488
0.0.0.0	53957	0.0.0.0	0	Bound		4912
0.0.0.0	52821	0.0.0.0	0	Bound		3600
0.0.0.0	51329	0.0.0.0	0	Bound		6880
127.0.0.1	65001	0.0.0.0	0	Listen		4832
127.0.0.1	65001	127.0.0.1	62100	Established	Internet	4832
127.0.0.1	62100	127.0.0.1	65001	Established	Internet	4832
192.168.82.199	59373	191.232.139.112	443	Established	Internet	2988
192.168.82.199	59372	191.232.139.112	443	Established	Internet	1612
192.168.82.199	58966	216.58.208.234	443	CloseWait	Internet	6880
192.168.82.199	58962	64.233.166.125	5222	Established	Internet	6880

Obtenir toutes connections établies sur Internet

Get-NetTCPConnection -State Established -AppliedSetting Internet

1 Get-NetTCPConnection -State Established -AppliedSetting Internet fl *						
PS C:\Documents\OneDrive\Documents\Scritps\powershell\Réseaux> Get-NetTCPConnection -State Established -AppliedSetting Internet						
LocalAddress	LocalPort	RemoteAddress	RemotePort	State	AppliedSetting	OwningProcess
127.0.0.1	65001	127.0.0.1	62100	Established	Internet	4832
127.0.0.1	62100	127.0.0.1	65001	Established	Internet	4832
192.168.82.199	59373	191.232.139.112	443	Established	Internet	2988
192.168.82.199	59372	191.232.139.112	443	Established	Internet	1612
192.168.82.199	58962	64.233.166.125	5222	Established	Internet	6880
192.168.82.199	58932	191.232.139.113	443	Established	Internet	8512
192.168.82.199	54133	191.232.139.180	443	Established	Internet	12968
192.168.82.199	54132	172.217.20.34	443	Established	Internet	7324
192.168.82.199	54131	172.217.20.34	443	Established	Internet	7324
192.168.82.199	54130	216.58.211.98	443	Established	Internet	7324
192.168.82.199	54129	216.58.211.98	443	Established	Internet	7324

Les cmdlets de gestion du réseaux

- Le script suivant permet de géolocaliser les adresses internet présent sur votre poste.
- Pour cela, avant de l'exécuter, connectez vous à internet et exécuter le script suivant

```
#la première partie récupère les connexions avec internet, sans prendre
127.0.0.1
```

```
#on supprime ensuite les doublons avec sort-object
```

```
#on utilise la cmdlette invoke-restMethod pour se connecter en json et
rechercher l'adresse ip (contenu dans remoteaddress)
```

```
#on affiche ensuite le résultat dans un out-gridview
```

```
Get-NetTCPConnection -State Established -AppliedSetting Internet | where
remoteAddress -NotLike "127.0.0.1" |
```

```
Sort-Object -Property RemoteAddress -Unique | %{
```

```
Invoke-RestMethod -uri "http://ip-api.com/json/$_remoteaddress"} | out-
GridView -Title "géolocalisation"
```

géolocalisation														
Filtrer														
Ajouter des critères														
as	city	country	countryCode	isp	lat	lon	org	query	region	regionName	status	timezone	zip	
AS8075 Microsoft Corporation	Amsterdam	Netherlands	NL	Microsoft Corporation	52.35	4.92	Microsoft Azure	104.40.210.32	NH	North Holland	success	Europe/Amsterdam	1091	
AS14618 Amazon.com, Inc.	Ashburn	United States	US	Amazon.com	39.04	-77.49	Amazon.com	107.22.185.206	VA	Virginia	success	America/New_York	20149	
AS16477 ACNIELSEN	Schaumb...	United States	US	A.C. Nielsen Company	42.05	-88.05	Acnelsen	138.108.96.100	IL	Illinois	success	America/Chicago	60173	
AS15169 Google Inc.	Mountain...	United States	US	Google	37.42	-122...	Google	172.217.18.230	CA	California	success	America/Los_Ange...	94043	
AS8075 Microsoft Corporation	Dublin	Ireland	IE	Microsoft bingbot	53.33	-6.25	Microsoft bin...	191.232.139.69	L	Leinster	success	Europe/Dublin		
AS8075 Microsoft Corporation	Dublin	Ireland	IE	Microsoft bingbot	53.33	-6.25	Microsoft bin...	191.232.139.63	L	Leinster	success	Europe/Dublin		
AS8075 Microsoft Corporation	Dublin	Ireland	IE	Microsoft bingbot	53.33	-6.25	Microsoft bin...	191.232.139.68	L	Leinster	success	Europe/Dublin		
								192.168.82.100				fail		
	Issy-les...	France	FR	Orange	48.83	2.27	Orange	193.252.22.88	J	Île-de-France	success	Europe/Paris		
AS15169 Google Inc.	Mountain...	United States	US	Google	37.42	-122...	Google	216.58.208.194	CA	California	success	America/Los_Ange...	94043	
AS15169 Google Inc.	Mountain...	United States	US	Google	37.42	-122...	Google	216.58.211.98	CA	California	success	America/Los_Ange...	94043	
AS26558 Freewheel Media Inc.	London	United Kin...	GB	Level 3 Communicat...	51.51	-0.13	Freewheel Me...	217.156.250...	ENG	England	success	Europe/London		
AS16509 Amazon.com, Inc.	Dublin	Ireland	IE	Amazon Technologies	53.33	-6.25	Amazon.com	54.239.33.151	L	Leinster	success	Europe/Dublin		
AS15169 Google Inc.	Mountain...	United States	US	Google	37.42	-122...	Google	74.125.206.125	CA	California	success	America/Los_Ange...	94043	
AS20940 AKAMAI-ASN1	London	United Kin...	GB	Akamai Technologies	51.51	-0.13	Akamai Techn...	88.221.14.144	ENG	England	success	Europe/London		
AS20940 AKAMAI-ASN1	London	United Kin...	GB	Akamai Technologies	51.51	-0.13	Akamai Techn...	88.221.14.155	ENG	England	success	Europe/London		

Les cmdlets de gestion du réseaux

- La cmdlette Get-NetConnectionProfile affiche les connexions et les différents profiles de vos cartes réseaux (profils public, privé, domaine)

```
Name          : Réseau 23
InterfaceAlias : vEthernet (wifi)
InterfaceIndex : 11
NetworkCategory : Private
IPv4Connectivity : Internet
IPv6Connectivity : NoTraffic
```

- La cmdlette set-connectionProfile permet de mettre à jour le profile de connexion, l'exemple suivant permet de changer le profil de private à oublic

```
Get-NetConnectionProfile | Set-NetConnectionProfile -NetworkCategory Public
```

```
PS C:\WINDOWS\system32> Get-NetConnectionProfile | Set-NetConnectionProfile -NetworkCategory Public
PS C:\WINDOWS\system32> Get-NetConnectionProfile

Name          : Réseau 23
InterfaceAlias : vEthernet (wifi)
InterfaceIndex : 11
NetworkCategory : Public
IPv4Connectivity : Internet
IPv6Connectivity : NoTraffic
```

Les cmdlets de gestion du réseaux

- Plusieurs cmdlettes permettent de reconfigurer l'adressage ip des serveurs et stations de travail.
- Get-netipaddress permet de voir la configuration ip du poste
- Pour ajouter une nouvelle adresse ip, utiliser new-netipaddress
- L'exemple suivant ajoute 2 adresses ip sur la carte réseau ethernet

```

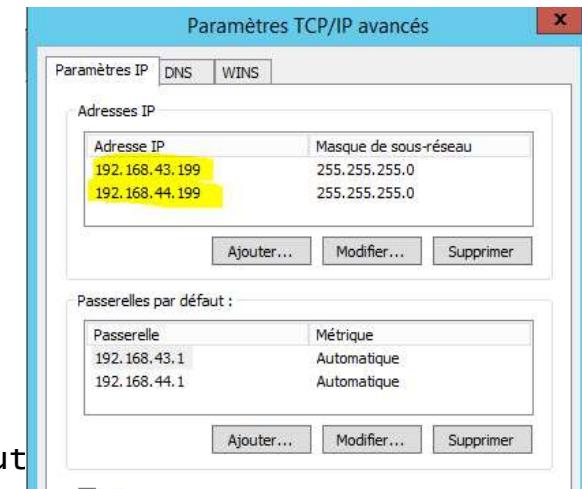
IPAddress          : 192.168.43.199
InterfaceIndex     : 12
InterfaceAlias     : Ethernet
AddressFamily      : IPv4
Type               : Unicast
PrefixLength       : 24
PrefixOrigin       : Manual
SuffixOrigin       : Manual
AddressState       : Preferred
ValidLifetime      : Infinite ([TimeSpan]::MaxValue)
PreferredLifetime  : Infinite ([TimeSpan]::MaxValue)
SkipAsSource       : False
PolicyStore        : ActiveStore

```

```
New-NetIPAddress -IPAddress 192.168.43.199
                  -DefaultGateway 192.168.43.1
                  -AddressFamily IPv4
                  -PrefixLength 24
                  -InterfaceAlias ethernet
```

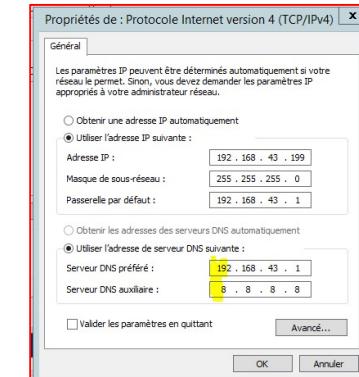
```
New-NetIPAddress -IPAddress 192.168.44.199
                  -DefaultGateway 192.168.44.1
                  -AddressFamily IPv4
                  -PrefixLength 24
                  -InterfaceAlias ethernet
```

- Si vous souhaitez mettre à jour une adresse ip, il faudra utiliser netipaddress



Les cmdlets de gestion du réseaux

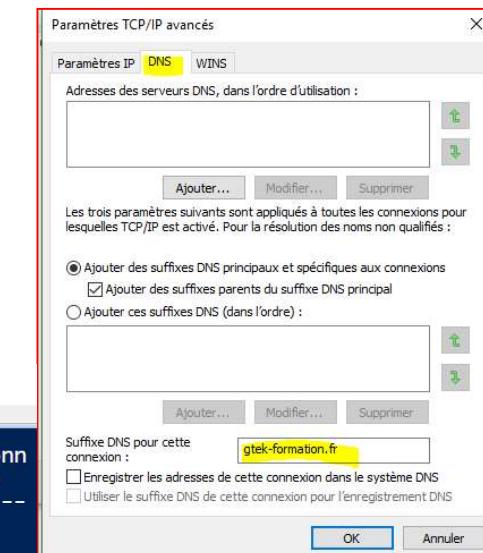
- La configuration et la mise à jour des adresses ip DNS sur la carte réseau se fait avec la cmdlette **Set-DnsClientServerAddress**
- Set-DnsClientServerAddress -InterfaceAlias ethernet -ServerAddresses ("192.168.43.1", "8.8.8.8")**
- Get-DnsClientServerAddress** affiche les DNS sur vos différents interfaces
- La cmdlette **set-dnsclient** permet de reconfigurer les paramètres tel que les suffixes dns



```
set-dnsclient -InterfaceIndex 11
              -ConnectionSpecificSuffix "gtek-formation.fr"
```

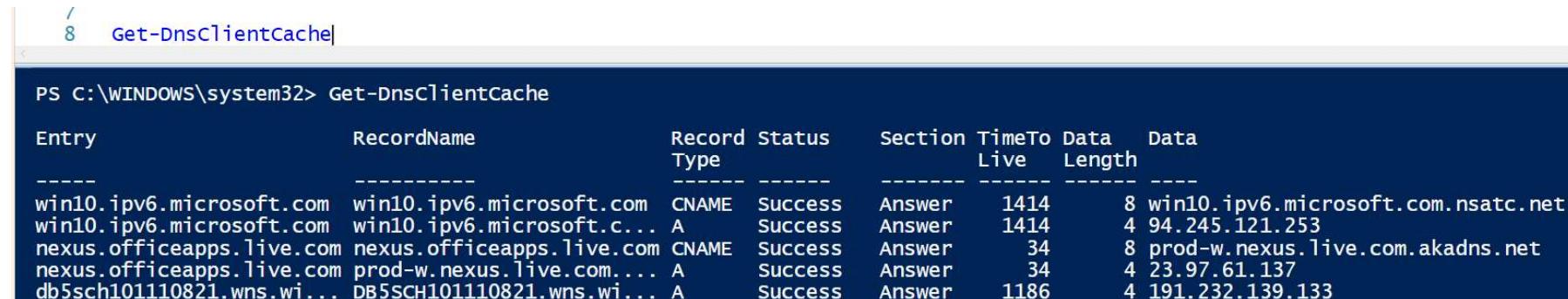
La cmdlette **get-dnsclient** permet de voir la configuration DNS sur votre poste

InterfaceAlias	InterfaceIndex	ConnectionSpecificSuffix	ConnectionSpecificSuffixSearchList	RegisterThisConnectionsAddress
Connexion au réseau local*	3	17	{}	True
vEthernet (wifi)	11	gtek-formation.fr	{}	False
Connexion réseau Bluetooth	13		{}	True
vEthernet (éthernet)	4		{}	True
Loopback Pseudo-Interface 1	1		{}	True
Teredo Tunneling Pseudo-I...	19		{}	True
isatap.{15D5A626-F7DF-4CA...	20		{}	True
isatap.gtek-formation.fr	12		{}	True



Information DNS

- La cmdlette `get-dnsclientcache` vous donne les résolutions DNS dans le cache DNS du poste



```
PS C:\WINDOWS\system32> Get-DnsClientCache

Entry          RecordName      RecordType Status Section TimeToLive DataLength Data
-----          -----        -----   -----  -----    -----      -----   -----
win10.ipv6.microsoft.com win10.ipv6.microsoft.com CNAME Success Answer 1414     8 win10.ipv6.microsoft.com.nsatc.net
win10.ipv6.microsoft.com win10.ipv6.microsoft.com A       Success Answer 1414     4 94.245.121.253
nexus.officeapps.live.com nexus.officeapps.live.com CNAME Success Answer 34      8 prod-w.nexus.live.com.akadns.net
nexus.officeapps.live.com prod-w.nexus.live.com A       Success Answer 34      4 23.97.61.137
db5sch101110821.wns.wi... db5sch101110821.wns.wi... A       Success Answer 1186     4 191.232.139.133
```

- `Clear-DnsClientCache` permet de purger le cache DNS, toutes ces cmdlettes peuvent pour rappel être exécutées sur un ordinateur distant
- La cmdlettes suivantes videraient les caches DNS de toute les stations de l'uo
`get-adcomputer -filter * -searchbase "ou=srv,dc=gtek-formation,dc=fr" | %{Clear-DnsClientCache -CimSession $_.name}`

Utilisation des cmdlets et des modules

Les cmdlets du pare-feu

Les commandelettes du pare-feu - Get-NetFirewallProfile

- Get-NetFirewallProfile -all
 - ◆ Affiche l'activation du pare-feu et des profiles
- Get-NetFirewallProfile | ft name, enabled
 - ◆ Affiche l'activation du pare-feu et des profiles en n'affichant que le nom et l'état

```
PS C:\Users\administateur.GTEK.000> Get-NetFirewallProfile | ft name, Enabled -a
```

name	Enabled
Domain	True
Private	True
Public	True

- ◆ Le test suivant effectue un contrôle sur tous les serveurs du domaine de type 2012.

```
Analyse du parc serveur avec pare-feu activé ou pas.ps1* X
1 #vérifier si les serveurs ont le parefeu d'activé.
2 #on récupère tout les serveurs du domaine de type 2012
3 $listeSrv=(get-adcomputer -filter {operatingSystem -like "*2012*"}).name
4 #pour chaque serveur trouver, on effectue un ping, retourne false ou true
5 foreach($srv in $listeSrv)
6 {
7
8     if(Test-Connection -ComputerName $srv -Quiet -Count 1)
9     {
10         write-host "Analyse de la station $srv" -ForegroundColor Green
11         Get-NetFirewallProfile -CimSession $srv | ft name, enabled -a
12     }
13     else
14     {
15         write-host "Station $srv non joignable pour le moment" -ForegroundColor DarkRed -BackgroundColor white
16     }
17 }
```

```
Analyse de la station SRV2K12R2CD1
name      Enabled
-----  -----
Domain    True
Private   True
Public    True

Station SRVEXCH1 non joignable pour le moment
Station SRV2K12EXCH13 non joignable pour le moment
Analyse de la station SRV2K12R2FIC
```

Les commandelettes du pare-feu - Set-NetFirewallProfile

- La commandelette **Set-NetFirewallProfile** permet d'activer ou désactiver votre pare-feu

Analyse du parc serveur avec pare-feu activé ou pas.ps1* Sans titre2.ps1*

```

1 Set-NetFirewallProfile -all -Enabled False
2 Get-NetFirewallProfile | ft name , enabled -AutoSize
3
4 Set-NetFirewallProfile -all -Enabled true
5 Get-NetFirewallProfile | ft name , enabled -AutoSize

```

Get-NetFirewallProfile | ft name , enabled -AutoSize

name	Enabled
Domain	False
Private	False
Public	False

name	Enabled
Domain	True
Private	True
Public	True

PS C:\Windows\system32>

Activation, désactivation du pare-feu sur 2 serveurs

Set-NetFirewallProfile -all -Enabled False -CimSession srv2K12-R2, srv2K12R2CD1

Get-NetFirewallProfile -CimSession srv2K12-R2, srv2K12R2CD1 | ft name , enabled -AutoSize

```

1 Set-NetFirewallProfile -all -Enabled False -CimSession srv2K12-R2, srv2K12R2CD1
2 Get-NetFirewallProfile -CimSession srv2K12-R2, srv2K12R2CD1 | ft name , enabled -AutoSize
3
4 Set-NetFirewallProfile -all -Enabled true -CimSession srv2K12-R2, srv2K12R2CD1
5 Get-NetFirewallProfile -CimSession srv2K12-R2, srv2K12R2CD1 | ft name , enabled -AutoSize

```

C:\Windows\system32> D:\W2K12\cours\scripts ps\les commandelettes powershell\Parfeu\Activer ou

name	Enabled
Domain	False
Private	False
Public	False
Domain	False
Private	False
Public	False

name	Enabled
Domain	True
Domain	True
Private	True
Public	True
Private	True
Public	True

Les commandelettes du pare-feu - Set-NetFirewallProfile

- Activer le pare-feu sur le profile de domaine
 - Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled True
- L'exemple suivant créé une règle de pare-feu autorisant le ping

```
New-NetFirewallRule -Name "Autorisation du ping"
    -Direction Inbound
    -Protocol icmpV4
    -Profile Domain, private
    -Action Allow
    -IcmpType 8
    -InterfaceType Any
    -DisplayName "Autorisation du pir"
```

Direction inbound pour règle en entrée
Protocol icmpV4 (TCP, UDP...)
-action allow (pour règle activée)
-interface (pour les interfaces réseaux)

```
Name          : Autorisation du ping
DisplayName   : Autorisation du ping
Description   :
DisplayGroup :
Group        :
Enabled       : True
Profile       : Domain, Private
Platform      : {}
Direction     : Inbound
Action        : Allow
EdgeTraversalPolicy : Block
LooseSourceMapping : False
LocalOnlyMapping : False
Owner         :
PrimaryStatus :
Status        :
EnforcementStatus :
PolicyStoreSource :
PolicyStoreSourceTy
```



Vérifier la règle définie dans votre pare-feu.

Les commandelettes du pare-feu - **Get-NetFirewallRule**

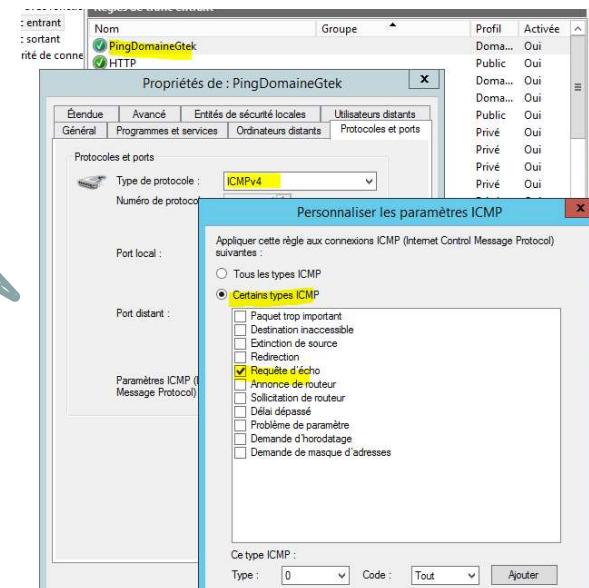
- Cette cmdlet permet d'afficher tout les rôles définis dans votre pare-feu (rôle pouvant être considéré comme une règle de pare-feu)
- Pour faire ce test, créer une règle entrant autorisant le ping sur le protocole ICMP type 8 code 0

Règle du pare-feu définie manuellement

```
1 Get-NetFirewallRule -DisplayName "PingDomaineGtek"
2

PS C:\Windows\system32> Get-NetFirewallRule -DisplayName "PingDomaineGtek"

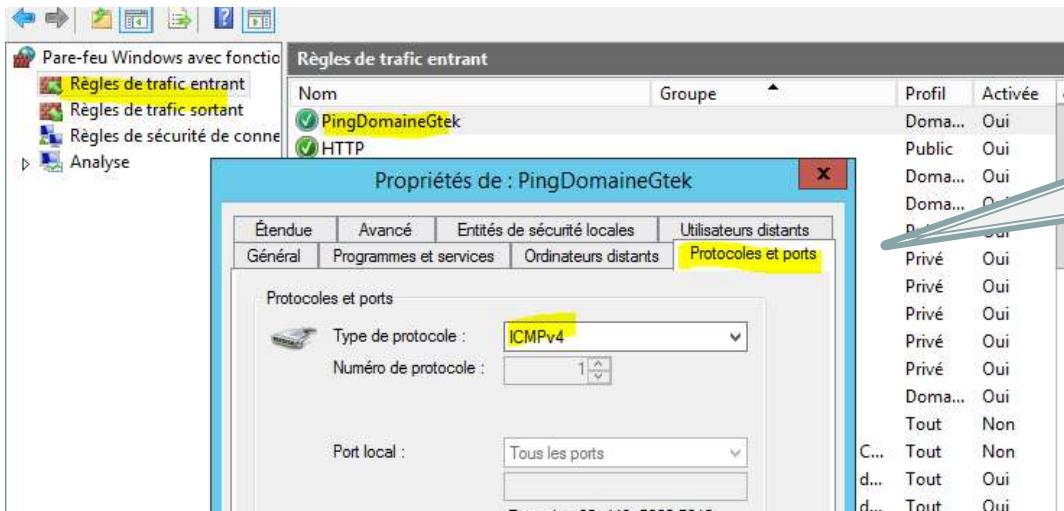
Name : {CD9D8A2B-AE84-48BD-B131-136591647ED8}
DisplayName : PingDomaineGtek
Description :
DisplayGroup :
Group :
Enabled : True
Profile : Domain, Private
Platform :
Direction : Inbound
Action : Allow
EdgeTraversalPolicy :
LooseSourceMapping :
LocalOnlyMapping :
Owner :
PrimaryStatus : OK
Status : La règle a été analysée à partir de la banque. (65536)
EnforcementStatus : NotApplicable
PolicyStoreSource : PersistentStore
PolicyStoreSourceType : Local
```



Cette première cmdlette permet d'afficher la règle nommée "pingDomaineGtek"

Les commandelettes du pare-feu - **Get-NetFirewallRule**

- Cette cmdlet permet d'afficher le rôle PingDomaineGTEK et la config de l'onglet Port et protocole (rôle pouvant être considéré comme une règle de pare-feu)
- Get-NetFirewallRule -DisplayName "PingDomaineGtek" | Get-NetFirewallPortFilter**



Nous retrouvons ici la règle de pare-feu pour l'autorisation du Ping

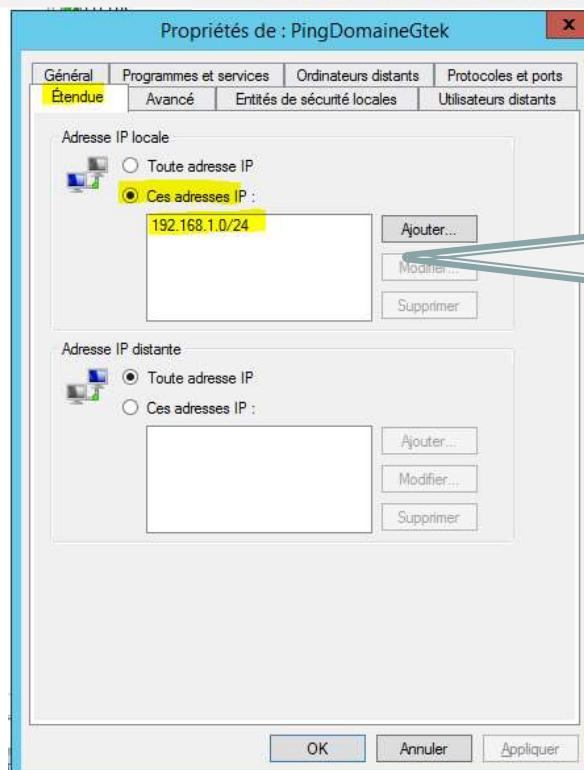
```
Sans titre1.ps1* X
1 #on va chercher une politique de sécurité nommée "pingDomaineGtek"
2 #politique de sécurité permettant de pinguer en utilisant le protocole ICMP de type 8 code 0
3 #on affiche ensuite les ports filtrer et utiliser
4 Get-NetFirewallRule -DisplayName "PingDomaineGtek" | Get-NetFirewallPortFilter
```

Get-NetFirewallPortFilter permet de voir l'onglet port de cette règle

```
PS C:\Windows\system32> Get-NetFirewallRule -DisplayName "PingDomaineGtek" | Get-NetFirewallPortFilter

Protocol      : ICMPv4
LocalPort     : RPC
RemotePort    : Any
IcmpType      : 8
DynamicTarget : Any
```

Les commandelettes du pare-feu - Get-NetFirewallRule



Modifiez votre règle pour l'adresse ip local en mettant votre id réseaux (192.168.1.0/24)

- Get-NetFirewallRule -DisplayName "PingDomaineGtek" | Get-NetFirewallAddressFilter

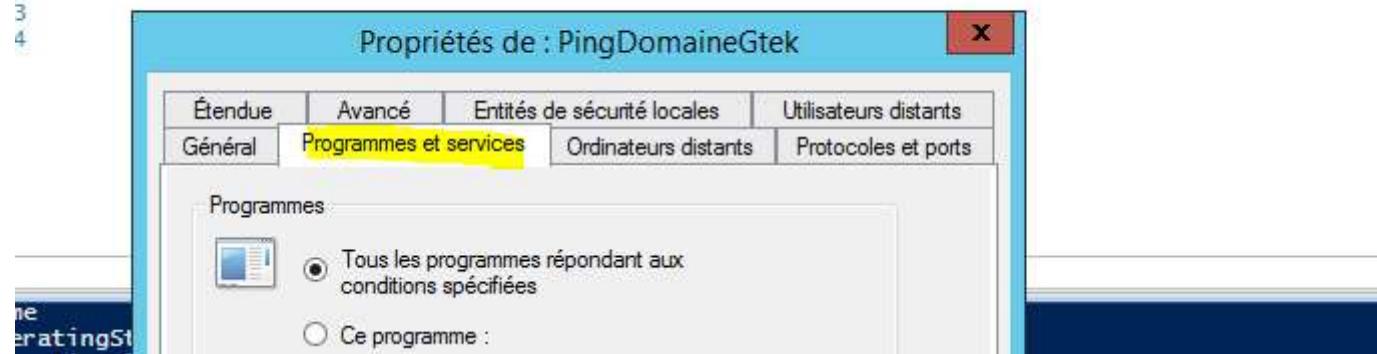
```
9 Get-NetFirewallRule -DisplayName "PingDomaineGtek" | Get-NetFirewallAddressFilter
10
11 |
<|
PS C:\Windows\system32>
Get-NetFirewallRule -DisplayName "PingDomaineGtek" | Get-NetFirewallAddressFilter

LocalAddress : 192.168.1.0/255.255.255.0
RemoteAddress : Any
```

Les commandelettes du pare-feu - Get-NetFirewallRule

- Pour voir les programmes et service
 - ◆ Get-NetFirewallRule -DisplayName "PingDomaineGtek" | Get-NetFirewallApplicationFilter

```
1 Get-NetFirewallRule -DisplayName "PingDomaineGtek" | Get-NetFirewallApplicationFilter
```

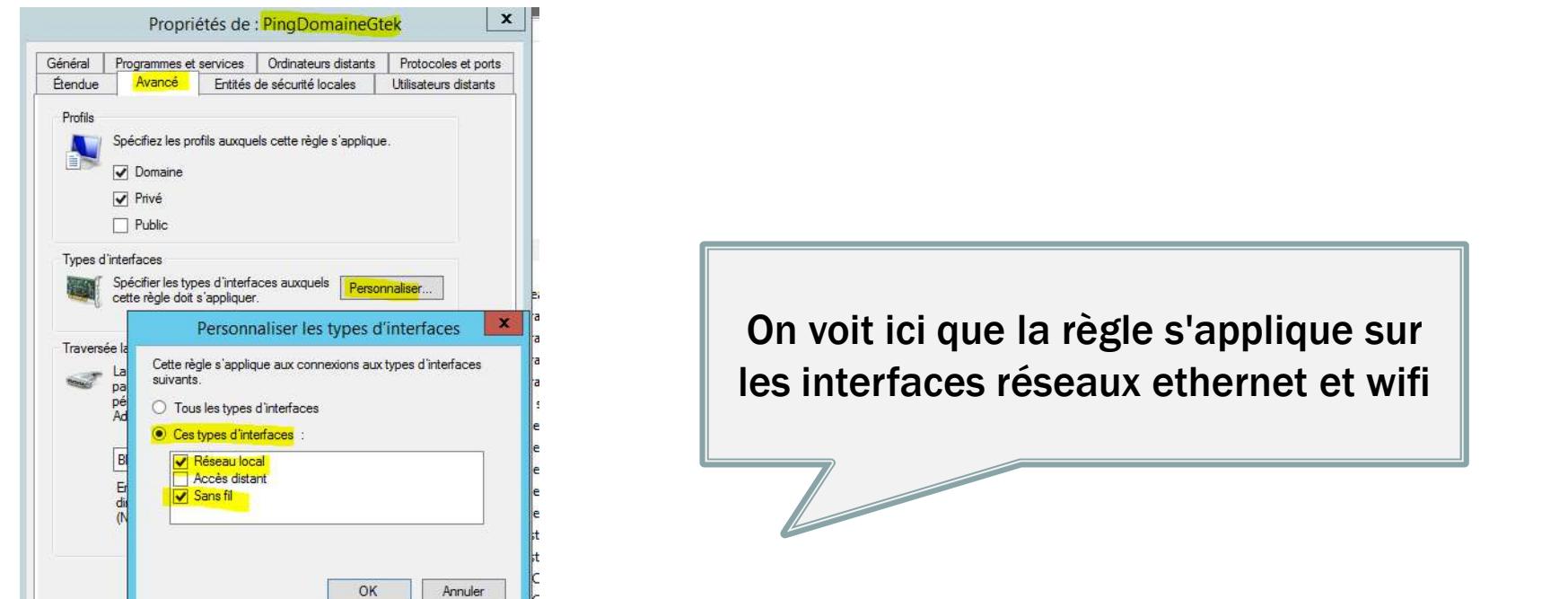


```
PS C:\Windows\system32> Get-NetFirewallRule -DisplayName "PingDomaineGtek" | Get-NetFirewallApplicationFilter
```

```
Program : Any
Package :
```

Les commandelettes du pare-feu - **Get-NetFirewallRule**

- La commande suivante permet de vérifier sur quel interface réseaux s'applique la règle
 - ◆ `Get-NetFirewallRule -DisplayName "PingDomaineGtek" | Get-NetFirewallInterfaceTypeFilter`

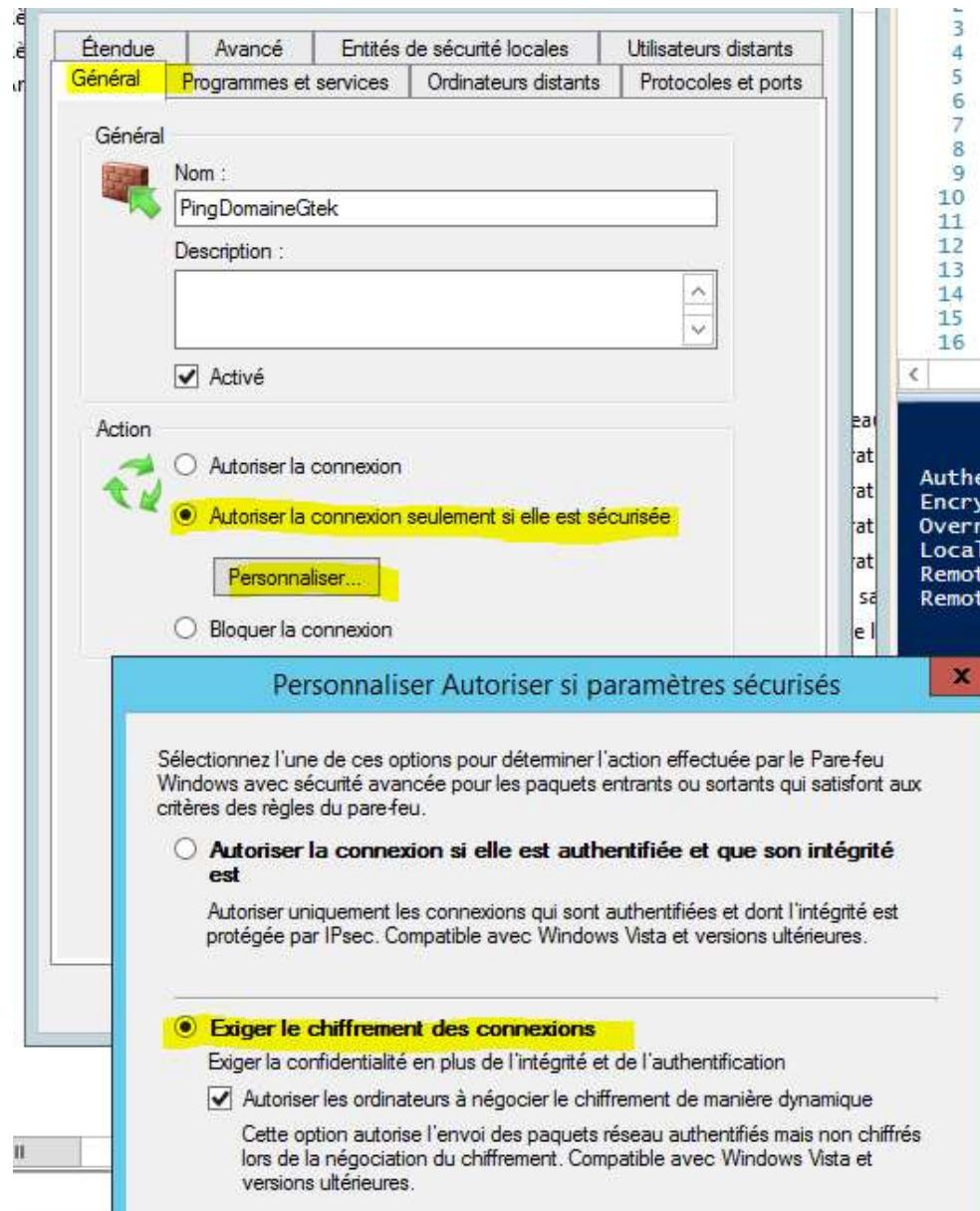


The screenshot shows the Windows Firewall Properties dialog box for a rule named "PingDomaineGtek". The "Avancé" tab is selected. In the "Types d'interfaces" section, a "Personnaliser..." button is clicked, opening a sub-dialog titled "Personnaliser les types d'interfaces". This sub-dialog shows that the rule applies to "Réseau local" and "Sans fil" interfaces. A callout bubble points from this sub-dialog to a text box containing the French translation: "On voit ici que la règle s'applique sur les interfaces réseaux ethernet et wifi".

```
12
13 Get-NetFirewallRule -DisplayName "PingDomaineGtek" | Get-NetFirewallInterfaceTypeFilter
14
```

```
PS C:\Windows\system32> Get-NetFirewallRule -DisplayName "PingDomaineGtek" | Get-NetFirewallInterfaceTypeFilter
InterfaceType : Wired, Wireless
```

Les commandelettes du pare-feu - Get-NetFirewallRule



```

14
15 Get-NetFirewallRule -DisplayName "PingDomaineGtek" | Get-NetFirewallSecurityFilter
16

Authentication : NotRequired
Encryption     : NotRequired
OverrideBlockRules : False
LocalUser      : Any
RemoteUser     : Any
RemoteMachine   : Any

PS C:\Windows\system32> Get-NetFirewallRule -DisplayName "PingDomaineGtek" | Get-NetFirewallSecurityFilter

Authentication : Required
Encryption     : Dynamic
OverrideBlockRules : False
LocalUser      : Any
RemoteUser     : Any
RemoteMachine   : Any

```

- **Get-NetFirewallRule -DisplayName "PingDomaineGtek" | Get-NetFirewallSecurityFilter**
- Permet de voir la sécurité de votre règle

Utilisation des cmdlets et des modules

Windows defender

Windows defender

- Le module defender sur Windows 10 permet de configurer, analyser...windows defender

Get-Command -Module defender

CommandType	Name	Version	Source
Function	Add-MpPreference	1.0	Defender
Function	Get-MpComputerStatus	1.0	Defender
Function	Get-MpPreference	1.0	Defender
Function	Get-MpThreat	1.0	Defender
Function	Get-MpThreatCatalog	1.0	Defender
Function	Get-MpThreatDetection	1.0	Defender
Function	Remove-MpPreference	1.0	Defender
Function	Remove-MpThreat	1.0	Defender
Function	Set-MpPreference	1.0	Defender
Function	Start-MpScan	1.0	Defender
Function	Start-MpWDOScan	1.0	Defender
Function	Update-MpSignature	1.0	Defender

- Get-MpComputerStatus** permet de voir l'état de windows defender, la dernière mise à jour, la dernière analyse rapide, détaillée...

```
AMEngineVersion      : 1.1.13103.0
AMProductVersion    : 4.10.14393.0
AMServiceEnabled    : True
AMServiceVersion    : 4.10.14393.0
AntispywareEnabled  : True
AntispywareSignatureAge : 0
AntispywareSignatureLastUpdated : 07/10/2016 00:47:31
AntispywareSignatureVersion : 1.229.1054.0
AntivirusEnabled    : True
AntivirusSignatureAge : 0
AntivirusSignatureLastUpdated : 07/10/2016 00:47:32
AntivirusSignatureVersion : 1.229.1054.0
BehaviorMonitorEnabled : True
ComputerID          : 76EA75E4-1845-44FE-BF05A3E62828
ComputerState        : 0
FullScanAge         : 4294967295
FullScanEndTime     :
FullScanStartTime   :
IoavProtectionEnabled : True
LastFullScanSource  : 0
LastQuickScanSource : 2
NISEnabled          : True
NISEngineVersion    : 2.1.12706.0
NISSignatureAge     : 2
NISSignatureLastUpdated : 05/10/2016 12:45:09
NISSignatureVersion : 116.33.0.0
OnAccessProtectionEnabled : True
QuickScanAge        : 4
QuickScanEndTime    : 03/10/2016 15:44:03
QuickScanStartTime  : 03/10/2016 15:35:27
RealTimeProtectionEnabled : True
RealTimeScanDirection : 0
PSComputerName       :
```

Windows defender

- Get-mpthreat permet de voir les derniers virus, spyware, trojan...détecté par windows defender

Get-MpThreat

```
CategoryID      : 8
DidThreatExecute : False
IsActive        : False
Resources       : {file:_G:\2014.2 KEYGEN\KEYGEN.exe}
RollupStatus    : 1
SchemaVersion   : 1.0.0.0
SeverityID     : 5
ThreatID        : 2147684005
ThreatName      : Trojan:Win32/Dynamer!ac
TypeID          : 0
PSComputerName  :
```

- Afficher la bases de virus pouvant etre détecter par windows defender

```
Get-MpThreatCatalog | fw threatname -Column 5
```

Trojan:Win32/startpage.TG	Trojan:Win32/PayTime	Trojan:
Backdoor:Win32/Exploiter.A	Backdoor:Win32/Exploiter.B	Backdo
Backdoor:Win32/BlackAngel.1_2	Backdoor:Win32/Netbus.1_80	Backdo
TrojanDropper:Win32/DOASearch	Trojan:Win32/ntfs32	Spywar
TrojanDownloader:Win32/Agent.IS	Trojan:Win32/SvcProc	Trojan
TrojanDownloader:Win32/msshed32	Trojan:Win32/eetu	Backdo
TrojanDownloader:Win32/DRUsearch	TrojanDownloader:Win32/Vxgame	Backdo
SoftwareBundler:Win32/RosoftAudio	TrojanDownloader:Win32/Small.HS	Brows
TrojanDownloader:Win32/Dlsw.B	BrowserModifier:Win32/QuickMetaS...	Spywar
BrowserModifier:Win32/Istbar.C	BrowserModifier:Win32/MySearchPage	Brows
Spyware:Win32/Coolwebsearch.A	Trojan:Win32/Rdbo	Backdo
TrojanDownloader:Win32/Small.DP	Backdoor:Win32/Wincrash.1_2	Dialer

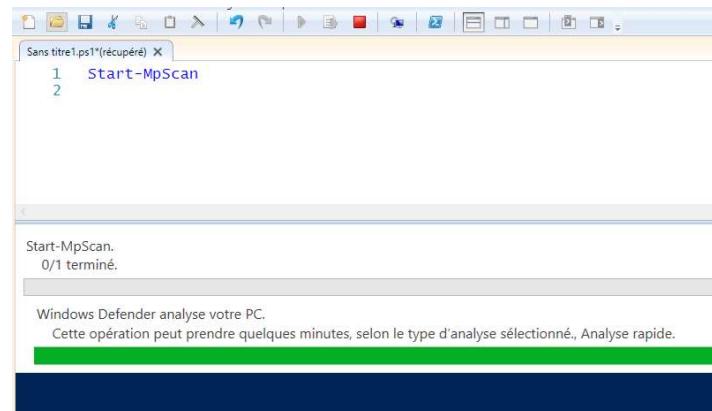
- Get-MpThreatCatalog | measure

```
PS C:\WINDOWS\system32> Get-MpThreatCatalog | measure

Count      : 158734
Average    :
```

Windows defender

- Start-mpscan permet de lancer windows defender
 - ◆ `Start-MpScan -ScanType QuickScan` (possibilité d'utiliser des paramètres et des scans distants)



- ◆ `Start-MpWDOScan` permet de lancer une analyse hors ligne (au redémarrage)

Utilisation des cmdlets et des modules

La gestion des journaux

Gestion des journaux

- La commandelette Get-eventLog permet de lire les différents journaux windows

```
PS C:\Users\W7> Get-EventLog -list

Max<(K) Retain OverflowAction      Entries Log
----- ----- -----
20 480    0 OverwriteAsNeeded      9 840 Application
20 480    0 OverwriteAsNeeded      0 HardwareEvents
512     7 OverwriteOlder          0 Internet Explorer
20 480    0 OverwriteAsNeeded      0 Key Management Service
8 192    0 OverwriteAsNeeded      0 Media Center
16 384    0 OverwriteAsNeeded      0 ODiag
16 384    0 OverwriteAsNeeded      227 OSession
                                         Security
```

Get-eventLog -list liste les différents journaux

```
PS C:\Users\W7> Get-EventLog system

Index Time           EntryType   Source          InstanceID Message
----- ----- -----
25029 mai 25 18:46 Information  Service Control M... 1073748860 Le service Planificateur d...
25028 mai 25 18:39 Information  Service Control M... 1073748864 Le type de démarrage du se...
25027 mai 25 18:15 Information  Service Control M... 1073748860 Le service Adobe Flash Pl...
25026 mai 25 18:15 Information  Service Control M... 1073748860 Le service Adobe Flash Pl...
25025 mai 25 17:57 Information  Service Control M... 1073748860 Le service Client DNS est ...
25024 mai 25 17:56 Information  Service Control M... 1073748860 Le service Expérience d'a...
25023 mai 25 17:51 Information  Service Control M... 1073748860 Le service Planificateur d...
```

Get-eventLog system liste les documents système

Gestion des journaux

- L'exemple suivant liste le journal système contenant le mot clé DHCP dans le message

```
PS C:\Users\W7> Get-EventLog system | ?{$_.Message -match "DHCP"}
```

Index	Time	EntryType	Source	InstanceID	Message
24934	mai 25 16:56	Information	Service Control M...	1073748860	Le service Client DHCP est entré dans l'état :
24933	mai 25 16:56	Information	Microsoft-Windows...	51046	Le service client DHCPv6 est démarré
24932	mai 25 16:56	Information	Microsoft-Windows...	50036	Le service client DHCPv4 est démarré
24859	mai 25 00:53	Information	Service Control M...	1073748860	Le service Client DHCP est entré dans l'état :
24857	mai 25 00:53	Information	Microsoft-Windows...	50037	Le service client DHCPv4 est arrêté. La valeur
24856	mai 25 00:53	Information	Microsoft-Windows...	51047	Le service client DHCPv6 est arrêté. La valeur
24797	mai 24 20:40	Information	Service Control M...	1073748860	Le service Client DHCP est entré dans l'état :

- L'exemple retourne les 5 derniers événements de chaque journal

```
wmi description des méthodes.ps1 WMI description des propriétés.ps1 wmi gestion des événements.ps1
1 $ErrorActionPreference="SilentlyContinue"
2 #on stocke dans la variable tous les journaux d'événement
3 $Logs = Get-EventLog -List
4 #pour chaque journal, nous allons afficher le nom du journal et les 5
5 foreach ($journ in $Logs)
6 {
7 Write-Host "`n journal : " $($journ.log) "les 5 dernières entrées ..."
8
9 Get-EventLog -LogName $($journ.log) -newest 5
10 }
11

journal : Application les 5 dernières entrées ...

Index Time EntryType Source InstanceID Message
----- ---- -----
11574 juin 03 11:02 Information HHCTRL 1904 La de
11573 juin 03 11:02 Information HHCTRL 1904 La de
11572 juin 03 11:02 Information HHCTRL 1904 La de
11571 juin 03 11:02 Information HHCTRL 1904 La de
11570 juin 03 11:02 Information HHCTRL 1904 La de

1 journal : HardwareEvents les 5 dernières entrées ...
8 journal : Internet Explorer les 5 dernières entrées ...
```

Création d'une source dans un journal

- New-eventLog permet de créer une nouvelle source ou un nouveau nom de log
- Nous pourrons ensuite utiliser write-eventlog pour écrire dans le journal
- L'exemple suivant permet de créer un journal, d'enregistrer un événement et enfin de le lire

```
New-EventLog -LogName Application -Source "Mes scripts powershell"
```

```
write-EventLog -LogName Application -Source "Mes scripts powershell"  
    -EntryType Information -EventId 123  
    -Message "script Effectué avec succès "
```

```
Get-EventLog -LogName Application -Source "Mes scripts powershell"
```

Effacer les données d'un journal

```
Clear-EventLog -LogName system -Confirm:$false
```

- Permet d'effacer le journal system

Utilisation des cmdlets et des modules

Gérer les points de restauration sur une station

Gérer les points de restauration sur une station

- Récupérer les points de restauration existants
 - ◆ [Get-ComputerRestorePoint](#)

```
PS C:\Windows\system32> Get-ComputerRestorePoint
```

CreationTime	Description	SequenceNumber	EventType	RestorePointType
17/10/2016 18:53:17	ASU_MSI_TRAN	100	BEGIN_SYSTEM_C...	APPLICATION_INSTALL
22/10/2016 09:47:43	ASU_MSI_TRAN	101	BEGIN_SYSTEM_C...	APPLICATION_INSTALL
29/10/2016 10:11:44	Windows Update	102	BEGIN_SYSTEM_C...	18

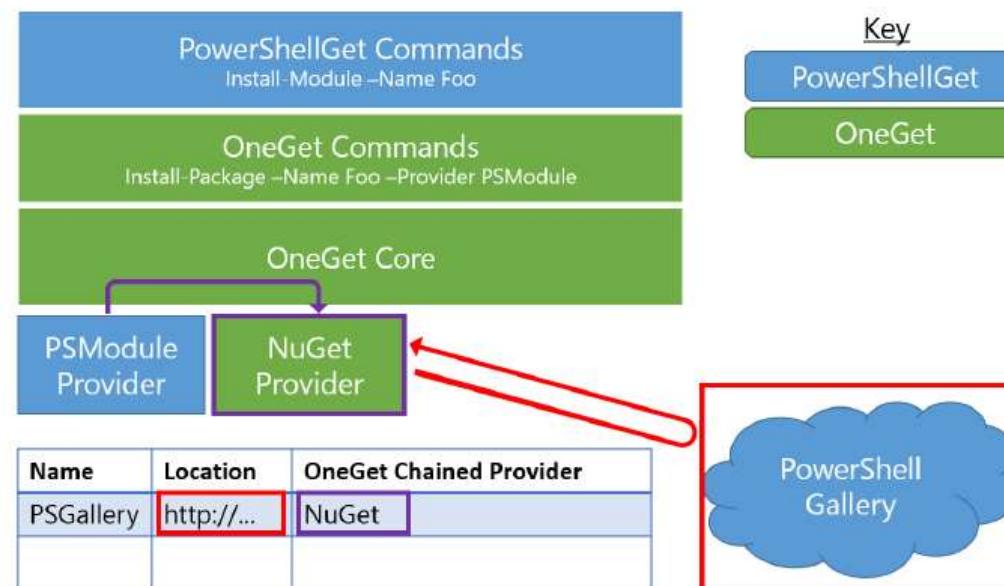
- Activer la protection système
 - ◆ [Enable-ComputerRestore -drive C:, D:](#)
- désactiver les points de restauration
 - ◆ [Disable-ComputerRestore -Drive c:](#)
- Restaurer station avec point de restauration
 - ◆ [Restore-Computer -RestorePoint 11](#)
- statut de la restauration
 - ◆ [Get-ComputerRestorePoint -LastStatus](#)

Utilisation des cmdlets et des modules

Gestion des packages -OneGet

- Microsoft permet aujourd'hui comme dans le monde Open Source d'installer des logiciels à partir de dépôts.
 - ◆ Ils permettent d'installer des paquets en évitant les manipulations peu reluisante comme le téléchargement, et les nombreux suivants, suivants, suivants...
 - ◆ Cette fonctionnalité s'apparente aux gestionnaires de packages classiques que l'on trouve dans les distributions Linux, notamment APT (Advanced Packaging Tool) sous Ubuntu, Yum sous RedHat ou encore Aptitude sous Debian.
 - ◆ Cette fonctionnalité permet, par exemple, de télécharger et d'installer un paquet sur un ensemble de machines, en une seule commande.

- Ce gestionnaire de paquet est appelé OneGet,
 - ◆ "OneGet est une nouvelle façon de découvrir et installer des paquets logiciels à travers le Web avec OneGet, vous pouvez.:
 - Gérer une liste de référentiels de logiciels dans lequel les paquets peuvent être recherchés, acquis et installés
 - Rechercher et filtrer vos dépôts à trouver les paquets dont vous avez besoin
 - Installer et désinstaller des packages d'un ou plusieurs référentiels avec une commande PowerShell.



Gestion des packages

- Pour obtenir la liste des commandes disponibles :
 - ◆ Get-Command -Module Packagemanagement

```
PS C:\Users\LoicThomas> Get-Command -Module Packagemanagement
```

CommandType	Name	Version	Source
Cmdlet	Find-Package	1.0.0.1	PackageManagement
Cmdlet	Find-PackageProvider	1.0.0.1	PackageManagement
Cmdlet	Get-Package	1.0.0.1	PackageManagement
Cmdlet	Get-PackageProvider	1.0.0.1	PackageManagement
Cmdlet	Get-PackageSource	1.0.0.1	PackageManagement
Cmdlet	Import-PackageProvider	1.0.0.1	PackageManagement
Cmdlet	Install-Package	1.0.0.1	PackageManagement
Cmdlet	Install-PackageProvider	1.0.0.1	PackageManagement
Cmdlet	Register-PackageSource	1.0.0.1	PackageManagement
Cmdlet	Save-Package	1.0.0.1	PackageManagement
Cmdlet	Set-PackageSource	1.0.0.1	PackageManagement
Cmdlet	Uninstall-Package	1.0.0.1	PackageManagement
Cmdlet	Unregister-PackageSource	1.0.0.1	PackageManagement

Gestion des packages

- Une des premières étapes intéressante est de lister les sources de paquets disponibles dans votre environnement:
 - ◆ Get-PackageProvider
 - Permet de lister les dépôts de votre environnement

```
PS C:\Users\LoicThomas> Get-PackageProvider
Name              Version           DynamicOptions
----              -----           -----
msi               3.0.0.0          AdditionalArguments
msu               3.0.0.0
NuGet             2.8.5.205        Destination, ExcludeVersion, Scope, Headers, FilterOnTag, Contains, AllowPr...
PowerShellGet     1.0.0.1          PackageManagementProviderType, Scope, InstallUpdate, PackageManagementPr...
Programs          3.0.0.0          IncludeWindowsInstaller, IncludeSystemComponent
```

- ◆ Find-PackageProvider
 - Lister les providers de paquets disponibles

```
PS C:\Users\LoicThomas> Find-PackageProvider
Name              Version           Source            Summary
----              -----           ----            -----
GistProvider      0.6              PSGallery        Gist-as-a-Package - PackageManagement...
ContainerImage    0.6.4.0         PSGallery        This is a PackageManagement provider ...
DockerMsftProvider 1.0.0.1         PSGallery        PowerShell module with commands for d...
NanoServerPackage 1.0.1.0         PSGallery        A PackageManagement provider to Disc...
GitHubProvider    0.5              PSGallery        GitHub-as-a-Package - PackageManageme...
TSDProvider       0.2              PSGallery        PowerShell PackageManager provider to...
PowerShellGet     1.1.1.0         PSGallery        PowerShell module with commands for d...
MyAlbum           0.1.2            PSGallery        MyAlbum provider discovers the photos...
nuget             2.8.5.207        https://oneget.org/nuget-2.... NuGet provider for the OneGet meta-pa...
ps1                1.0.0.210        https://oneget.org/ps1-1.0.... ps1 provider for the OneGet meta-pack...
chocolatey       2.8.5.130        https://oneget.org/Chocolat... ChocolateyPrototype provider for the ...
OfficeProvider    1.0.0.1          PSGallery        OfficeProvider allows users to instal...
WSAProvider        1.0.0.4          PSGallery        Provider to Discover, Install and inv...
ChocolateyGet     1.0.0.1          PSGallery        An PowerShell OneGet provider that di...
GitLabProvider    1.3.3            PSGallery        GitLab PackageManagement provider
Zinstall          2.12.0           PSGallery        Zero Install is a decentralized cross...
```

Gestion des packages

- Lister les packages disponibles:
 - ◆ Pour trouver ou lister les packages disponibles vous pouvez utiliser la cmdlet **find-package**
 - ◆ Lister tous les packages disponibles

```
PS C:\Users\LoicThomas> find-package
```

Name	Version	Source	Summary
AzureRM.profile	2.3.0	PSGallery	Microsoft Azure PowerShell - Profile ...
Azure.Storage	2.3.0	PSGallery	Microsoft Azure PowerShell - Storage ...
AzureRM.ApiManagement	3.1.0	PSGallery	Microsoft Azure PowerShell - Api Mana...
AzureRM.Automation	2.3.0	PSGallery	Microsoft Azure PowerShell - Automati...
AzureRM	3.1.0	PSGallery	Azure Resource Manager Module
AzureRM.KeyVault	2.3.0	PSGallery	Microsoft Azure PowerShell - KeyVault...
AzureRM.Compute	2.3.0	PSGallery	Microsoft Azure PowerShell - Compute ...
AzureRM.Storage	2.3.0	PSGallery	Microsoft Azure PowerShell - Storage ...
AzureRM.Batch	2.3.0	PSGallery	Microsoft Azure PowerShell - Batch se...
AzureRM.Backup	2.3.0	PSGallery	Microsoft Azure PowerShell - Azure Ba...
AzureRM.Resources	3.3.0	PSGallery	Microsoft Azure PowerShell - Azure Re...
Posh-SSH	1.7.6	PSGallery	PowerShell module for automating task...

- ◆ Lister les packages disponibles pour gérer office365

```
PS C:\Users\LoicThomas> Find-Package -Name *365*
```

Name	Version	Source	Summary
0365ServiceCommunications	1.3	PSGallery	Gathers data regarding Office 365 Ser...
Connect0365	0.8.2.1	PSGallery	Supporting functions for connect-0365...
0365_Logon	1.0	PSGallery	0365 logon cmdlets to assist IT admin...
Office365	0.0.2	PSGallery	For Microsoft Partners and Internal A...
0365Tools	1.0.3	PSGallery	This module was set up to simplify us...

Gestion des packages

- Installer un paquets
 - ◆ Ici on désire installer le paquet des service de communications d'Office365
 - *Install-Package -Name O365ServiceCommunications -Source PsGallery*

```
PS C:\Windows\system32> Install-Package -Name O365ServiceCommunications -Source PsGallery
Le ou les packages proviennent d'une source de package qui n'est pas marquée comme étant approuvée.
Voulez-vous vraiment installer le logiciel à partir de 'PSGallery' ?
[O] Oui [T] Oui pour tout [N] Non [U] Non pour tout [S] Suspender [?] Aide
(la valeur par défaut est « N ») :o
Name          Version      Source      Summary
----          -----      ----      -----
O365ServiceCommunications    1.3        PSGallery  Gathers data regarding Offi
```

Gestion des packages

- **Installer un provider de paquet :**

- ◆ Nous allons ici installer le provider Chocolatey

- C'est un gestionnaire de paquets pour Windows
 - Un gestionnaire de paquets pour Windows

- Chocolatey s'interface avec de nombreuses solutions de gestion de configuration :

- » Ansible : win_chocolatey
 - » Chef : chocolatey_package
 - » PowerShell PackageManagement / OneGet :
 - » provider natif depuis PowerShell 5/Windows 10
 - » Puppet : module chocolatey
 - » – Saltstack : module chocolatey
 - » – SCCM

- Une liste complète : <https://github.com/chocolatey/choco/wiki/FeaturesInfrastructureAutomation>

Gestion des packages

- Installer le provider :
 - ◆ Install-PackageProvider -Name chocolatey

```
PS C:\Windows\system32> Install-PackageProvider -Name chocolatey
Le ou les packages proviennent d'une source de package qui n'est pas marquée comme étant approuvée.
Voulez-vous vraiment installer le logiciel à partir de
'https://oneget.org/ChocolateyPrototype-2.8.5.130.package.swidtag' ?
[O] Oui [T] Oui pour tout [N] Non [U] Non pour tout [S] Suspendre [?] Aide
(la valeur par défaut est « N ») : o
Name          Version      Source      Summary
----          -----      -----      -----
chocolatey    2.8.5.130   https://oneget.org/Chocolat... ChocolateyPrototype provide
```

- Lister les paquets disponibles depuis le dépôt:
 - ◆ Find-Package –ProviderName chocolatey

```
PS C:\Windows\system32> Find-Package -ProviderName chocolatey
Le fournisseur 'chocolatey v2.8.5.130' n'est pas installé.
chocolatey peut être téléchargé manuellement à partir de
https://oneget.org/ChocolateyPrototype-2.8.5.130.exe et installé.
Voulez-vous que PackageManagement télécharge et installe automatiquement 'chocolatey' maintenant ?
[O] Oui [N] Non [S] Suspendre [?] Aide (la valeur par défaut est « 0 ») : o
Le ou les packages proviennent d'une source de package qui n'est pas marquée comme étant approuvée.
Voulez-vous vraiment installer le logiciel à partir de
'https://oneget.org/ChocolateyPrototype-2.8.5.130.package.swidtag' ?
[O] Oui [T] Oui pour tout [N] Non [U] Non pour tout [S] Suspendre [?] Aide
(la valeur par défaut est « N ») : o
Name          Version      Source      Summary
----          -----      -----      -----
chocolatey    0.10.3       chocolatey
flashplayerplugin 23.0.0.205  chocolatey
GoogleChrome   54.0.2840.87  chocolatey
winrar        5.40         chocolatey
Firefox        49.0.2.20161024  chocolatey
jre8          8.0.101.20160917  chocolatey
flashplayeractivex 23.0.0.162  chocolatey
adobereader   2015.007.20033  chocolatey
7zip.install   16.02.0.20160811  chocolatey
javaruntyme   8.0.101       chocolatey
notepadplusplus.install 7.1.0.20161029  chocolatey
nodejs.install 7.0.0         chocolatey
autohotkey_portable 1.1.24.02  chocolatey
git.install    2.10.1       chocolatey
git            2.10.1       chocolatey
notepadplusplus 7.1.0.20161029  chocolatey
```

- ◆ Nbre de paquets disponible

```
PS C:\Windows\system32> (Find-Package -ProviderName chocolatey).Count
3360
PS C:\Windows\system32>
```

Gestion des packages

- Installer 7 zip depuis le provider Chocolatey
 - ◆ Install-Package 7zip.install -ProviderName chocolatey

```
Administrator : Windows PowerShell
Windows PowerShell
Copyright (C) 2015 Microsoft Corporation. Tous droits réservés.

Downloading 'http://www.7-zip.org/a/7z1602-x64.exe'
To C:\Users\LoicThomas\AppData\Local\Temp\chocolatey\7zip.install\7zip.install\7zip.installinstall.exe
[oooo]

[O] Oui [T] Oui pour tout [N] Non [U] Non pour tout [S] Suspendre [?] Aide
(la valeur par défaut est « N ») :o
```

```
PS C:\Windows\system32> Install-Package 7zip.install -ProviderName chocolatey
Le ou les packages proviennent d'une source de package qui n'est pas marquée comme étant approuvée.
Voulez-vous vraiment installer le logiciel à partir de 'chocolatey' ?
[O] Oui [T] Oui pour tout [N] Non [U] Non pour tout [S] Suspendre [?] Aide
(la valeur par défaut est « N ») :o

Name          Version      Source
----          -----      -----
7zip.install  16.02.0.20160811 chocolatey
                                         Summary
                                         -----
                                         7-Zip is a file archiver wi
```

Application installé



UTILISATION DES OBJETS WMI

Utilisation des objets WMI

Le modèle de données

Introduction aux objets WMI

- WMI Windows Management Instrumentation permet d'accéder et de partager des informations liées à l'infrastructure matérielle et logicielle de votre réseau
- WMI permet simplement d'accéder à des informations telles que la taille de la mémoire, la taille des disques ou le type et le nombre de cartes réseaux installées...
- WMI permet une supervision simplifiée d'événement système
- Les API WMI sont basées sur la technologie COM, sous dotnet on y accède via les classes de l'espace de nom **System.Management**
- Pour récupérer l'instance d'une classe donnée (par ex les cartes réseaux, on doit préciser sa clé. WMI étant un référentiel, *c.-à-d. une base de données, chaque instance possède un identifiant qui peut être une combinaison de clés* (précisées lors de la conception de la classe WMI ici `win32_networkadapter`)

Introduction aux objets WMI

- WMI est constitué de 4 composants :
 - ◆ Les fournisseurs WMI (appelé Provider) assurent :
 - La disponibilité des ressources
 - L'adressage dans WMI
 - ◆ Le CIMOM (Common Interface Model Object Manager) est un directeur de trafic:
 - Il traite les interactions entre les fournisseurs WMI et les consommateurs (dans notre cas, les scripts WMI sont les consommateurs)
 - ◆ Le référentiel CIM
 - ◆ La bibliothèque de scripting WMI

Introduction aux objets WMI

- Windows Management Instrumentation (WMI) offre beaucoup de possibilités pour des scripts qui gèrent le matériel et les logiciels
- WMI est une énorme base de données répondant aux besoins d'uniformisation de la gestion des accès aux ressources de l'O.S
- WMI est présent depuis NT4 SP4 en tant que composant additionnel et est aujourd'hui complètement intégré à tous les systèmes d'exploitation
- WMI a été développé en répondant au modèle CIM (Common Information Model) introduit par le DMTF (Distributed Management Task Force)
- Le but de cette organisation est d'établir des standards quelque soient l'os (windows, ibm, ...)

Introduction aux objets WMI

- WMI permet aujourd'hui d'accéder assez simplement à l'ensemble des ressources de manière unifiée via des scripts.
- WMI est en fait une énorme bibliothèque référençant :
 - ◆ Toutes les informations concernant le matériel.
 - Pour un disque dur, on retrouve par ex le modèle, le numéro de série, l'espace disque disponible, le type de partition.
 - Pour une carte mère, nous retrouvons le type bios, le numéro de série, les différents type de ports ...
 - ◆ Toutes les informations de gestion du système
 - On retrouve l'ensemble des composants d'un système W2K comme les journaux, le système de fichier, la gestion des services (DNS, DHCP,IIS..), ...

Introduction aux objets WMI

■ Le modèle CIM:

- ◆ CIM est un modèle orienté objet décrivant l'environnement système et réseau d'une entreprise (matériel, logiciel et services).
- ◆ CIM peut être considéré comme le schéma de WMI
- ◆ CIM ne stocke pas les informations mais offre un chemin d'accès uniformisé à l'information recherchée
- ◆ Comme tout schéma (par ex l'ad), CIM est hiérarchisé en classes permettant de regrouper les ressources WMI par famille
- ◆ Ces classes sont regroupées en espaces de nom (NameSpace) servant à regrouper les classes sous un label commun.
- ◆ Le plus utilisé est **root/cimV2** qui regroupe les classes les plus utilisées
- ◆ Nous retrouvons par ex **root/virtualServer** regroupant les classes liés à la gestion des ressources de Virtual server

Introduction aux objets WMI

- 2 points importants à retenir :
 - ◆ WMI ne fait que retourner des informations
 - Il est toutefois possible d'accéder aux fonctionnalités du système pour modifier par ex la configuration d'un serveur DNS
 - ◆ WMI permet de faire de l'audit en tant réel grâce aux événements WMI
- Toutes ces informations sont classées sur la base du modèle CIM permettant l'échange d'information quelque soit la plateforme utilisé

Introduction aux objets WMI

- WMI est très vaste, vous pouvez donc pratiquement retrouver toutes les informations nécessaires à une bonne gestion de parc comme par ex:
 - ◆ Les @mac, IP de vos machines ([classe win32_networkadapterConfiguration](#))
 - ◆ Les applications installées aux postes clients ([classe win32_product](#))
 - ◆ L'espace dispo sur les disques ([win32_logicalDisk](#))
 - ◆ La version du bios, ([win32_bios](#))
 - ◆ de votre OS, ([win32_operatingSystem](#))
 - ◆ La marque et le modèle du PC ([win32_computersystem](#))
 - ◆ Les partages ([win32_share](#))
 - ◆ L'état des batteries de portable ([win32_battery](#))
- Chaque classe WMI possède souvent plusieurs propriétés et méthodes
- 2 cmdlettes utilisées avec WMI:
 - ◆ [Get-wmiobject](#) alias [gwmi](#)
 - ◆ [Get-ciminstance](#) alias [gcim](#)

Utilisation des objets WMI

Listage des classes WMI

Listage des classes WMI

- Sous PowerShell la cmdlet WMI est **get-wmiobject** (alias **gwmi**)
- Par défaut gwmi interroge l'espace de nom **root\cimv2** de l'ordinateur local
 - Ex: **gwmi win32_networkadAPTER**

```
PS C:\Users\Administrateur> $b=gwmi win32_motherboarddevice
PS C:\Users\Administrateur> $b.GetType()
IsPublic IsSerial Name                                     BaseType
-----  -----  ----
True      True    ManagementObject
System.Management.ManagementBaseObject
```

Gettype() retourne la classe dot net utilisée

```
PS C:\Users\Administrateur> $b.__namespace
root\cimv2
PS C:\Users\Administrateur> $b.__class
Win32_MotherboardDevice
PS C:\Users\Administrateur>
```

__namespace et **__class** retourne l'espace de nom sur lequel vous travaillez et la classe

Listage des classes WMI

- L'exemple suivant retourne toutes les propriétés de la station courant

```
gwmi -namespace "root\cimv2" -list | %{$_.name} | %{gwmi $_} | fl *
```

```
PS C:\Users\W7> gwmi -list
Win32_PerfRawData_rcppip_0D106
Win32_PerfFormattedData_TermServ...
Win32_PerfRawData_TermService_Te...
Win32_PerfFormattedData_UGathere...
Win32_PerfRawData_UGatherer_Sear...
Win32_PerfFormattedData_UGTHRSVC...
Win32_PerfRawData_UGTHRSVC_Searc...
Win32_PerfFormattedData_usbbhub_USB
Win32_PerfRawData_usbbhub_USB
Win32_PerfFormattedData_WindowsM...
Win32_PerfRawData_WindowsMediaPl...
Win32_PerfFormattedData_WindowsW...
Win32_PerfRawData_WindowsWorkflo...
Win32_PerfFormattedData_WindowsW...
Win32_PerfRawData_WindowsWorkflo...
Win32_PerfFormattedData_WSearchI...
Win32_PerfRawData_WSearchIdxPi_S...
```

Gwmi -list affiche les classes disponibles dans le "namespace" (ici root\cimv2)

Cet ex retourne les classes win32_network*

Name	Methods	Properties
Win32_NetworkAdapter	{SetPowerState, R...	{AdapterType, Adap...
Win32_NetworkConnection	{}>	{AccessMask, Capti...
Win32_NetworkProtocol	{}>	{Caption, Connecti...
Win32_NetworkClient	{}>	{Caption, Descript...
Win32_NetworkLoginProfile	{}>	{AccountExpires, A...
Win32_NetworkAdapterConfiguration	{EnableDHCP, Rene...	{ArpAlwaysSourceRo...
Win32_NetworkAdapterSetting	{}>	{Element, Setting}

Listage des classes WMI

```
NetEnabled : False
NetworkAddresses :
PermanentAddress :
PhysicalAdapter : ROOT\*\ISATAP\0001
PNPDeviceID :
PowerManagementCapabilities :
PowerManagementSupported : False
ProductName : Carte Microsoft ISATAP
ServiceName :
Speed : 10000
SystemCreationClassName : Win32_ComputerSystem
SystemName : PC?
TimeOfLastReset : 201206022154505.384658+
Scope :
Path : \\PC7\root\cimv2:Win32_
Options :
ClassPath :
Properties :
SystemProperties : System.Management.ObjectContainer
Qualifiers : {__GENUS, __CLASS, __SERVER, __NAMESPACE, __TYPEID, __DICTIONARY, __PROVIDERNAME, __PROVIDERVERSION, __PROVIDERSTATE, __PROVIDERNAME, __PROVIDERVERSION, __PROVIDERSTATE, __PROVIDERNAME, __PROVIDERVERSION, __PROVIDERSTATE}
Site :
Container : {dynamic, Locale, provider}
```

Gwmi NOM_Classe | fl * pour obtenir toutes les propriétés d'une classe

"... | gm" (get-member pour obtenir les propriétés et méthodes associées à la classe)

```
PS C:\Users\W7> gwmi win32_networkadapter | gm
```

Name	MemberType	Definition
Disable	Method	System.Management.ManagementBaseObject
Enable	Method	System.Management.ManagementBaseObject
Reset	Method	System.Management.ManagementBaseObject
SetPowerState	Method	System.Management.ManagementBaseObject
AdapterType	Property	System.String AdapterType {get;set;}
AdapterTypeId	Property	System.UInt16 AdapterTypeId {get;set;}
AutoSense	Property	System.Boolean AutoSense {get;set;}
Availability	Property	System.UInt16 Availability {get;set;}
Caption	Property	System.String Caption {get;set;}
Configurable	Property	System.Boolean Configurable {get;set;}
CurrentSpeed	Property	System.UInt32 CurrentSpeed {get;set;}
CurrentStatus	Property	System.UInt32 CurrentStatus {get;set;}
DesiredSpeed	Property	System.UInt32 DesiredSpeed {get;set;}
LinkSpeed	Property	System.UInt32 LinkSpeed {get;set;}
MaxSpeed	Property	System.UInt32 MaxSpeed {get;set;}
NetworkAdapterCategory	Property	System.UInt32 NetworkAdapterCategory {get;set;}
NetworkAdapterType	Property	System.UInt32 NetworkAdapterType {get;set;}
PhysicalAdapter	Property	System.Boolean PhysicalAdapter {get;set;}
Speed	Property	System.UInt32 Speed {get;set;}
Status	Property	System.UInt32 Status {get;set;}
SupportedSpeeds	Property	System.UInt32[] SupportedSpeeds {get;set;}
VirtualAdapter	Property	System.Boolean VirtualAdapter {get;set;}

Ex pour afficher les méthodes utilisables pour une classe déterminée

```
PS C:\Windows\system32> gwmi win32_networkadapter | gm -membertype method  
  
TypeName: System.Management.ManagementObject#root\cimv2\Win32_NetworkAdapter  
  
Name          MemberType  Definition  
---          Method      System.Management.ManagementBaseObject Disable()  
Enable        Method      System.Management.ManagementBaseObject Enable()  
Reset         Method      System.Management.ManagementBaseObject Reset()  
SetPowerState Method      System.Management.ManagementBaseObject SetPowerState(<
```

Listage des classes WMI

```
PS C:\Users\W7> gwmi win32_networkadapter | fl [a-z]*
```

Availability	: 3
Name	: WAN Miniport (SSTP)
Status	:
StatusInfo	:
DeviceID	: 0
AdapterType	:
AdapterTypeId	:
AutoSense	:
Caption	: [00000000] WAN Miniport (SSTP)
ConfigManagerErrorCode	: 0
ConfigManagerUserConfig	: False
CreationClassName	: Win32_NetworkAdapter
Description	: WAN Miniport (SSTP)

Cet exemple permet de ne pas afficher les premières propriétés comme `__name`, `__class`

```
PS C:\Users\W7> gwmi win32_process | ?($_.name -like "powershell*")
```

GENUS	: 2
CLASS	: Win32_Process
SUPERCLASS	: CIM_Process
DYNASTY	: CIM_ManagedSystemElement
RELPATH	: Win32_Process.Handle="5524"
PROPERTY_COUNT	: 45
DERIVATION	: {CIM_Process, CIM_LogicalElement, CIM_ManagedS...
SERVER	: PC7
NAMESPACE	: root\cimv2
PATH	: \\PC7\\root\\cimv2:Win32_Process.Handle="5524"
Caption	: powershell.exe
CommandLine	: "C:\\WINDOWS\\system32\\WindowsPowerShell\\v1.0\\po...
CreationClassName	: Win32_Process
CreationDate	: 20120602161835.418942+120
CSCreationClassName	: Win32_ComputerSystem
CSName	: PC7
Description	: powershell.exe
ExecutablePath	: C:\\WINDOWS\\system32\\WindowsPowerShell\\v1.0\\pow...
ExecutionState	: 5501

L'exemple suivant permet de filtrer avec `where-object` (alias `?`) les process portant le nom "powershell"

```
PS C:\Users\W7> gwmi win32_process -filter 'name = "powershell.exe"'
```

Vous pouvez aussi utiliser l'option `-filter` à la place du `where`

Listage des classes WMI

- Pour rechercher les données affichées par une classe wmi, prenez le nom de la classe | fl *:

```
gwmi win32_operatingSystem | fl *
gcim win32_computerSystem | fl *
```

- Pour rechercher toutes les classes wmi :

```
#liste toute les classes wmi sous root/cimv2
gwmi -List
#liste les classes wmi commençant par win32_net
gwmi -list win32_net*
#liste toute les classes wmi sous root/default
gwmi -list -Namespace root/default
#liste toute les classes wmi sous root/security
gwmi -list -Namespace root/security
```

Listage des classes WMI

- **Get-cimClass** permet de faire de la recherche sur des propriétés et méthodes.
 - ◆ Rechercher les classes utilisant la méthode reboot (**-methodname** recherche les méthodes et **-propertyname** les propriétés)

```
1 Get-CimClass -MethodName reboot
```

CimClassName	CimClassMethods	CimClassProperties
CIM_OperatingSystem	{Reboot, Shutdown}	{Caption, Description, InstallDate, Name...}
Win32_OperatingSystem	{Reboot, Shutdown...}	{Caption, Description, InstallDate, Name...}

```
3 Get-CimClass -PropertyName Macaddress
```

CimClassName	CimClassMethods	CimClassProperties
Win32_NetworkAdapter		{SetPower*
Win32_NetworkAdapterConfiguration		{EnableDHCP}

Utilisation des objets WMI

Administre Windows avec WMI

Administre Windows avec WMI

- L'exemple suivant retourne toutes les propriétés des disques locaux

```
get-wmiobject win32_diskpartition | format-list *
```

- L'exemple suivant retourne l'espace disponibles sur chaque partition en Go.

```
PS C:\Users\w7> Get-wmiobject win32_logicaldisk : foreach-object{$_deviceId + $_freespace/1GB}
C:11.5079345703125
D:154.78063583374
E:76.0479125976563
F:21.6006469726563
G:108.558261871338
H:0
I:1.87361145019531
```

- Logiciels installé sur un poste

- L'exemple suivant recherche une appli nommée "ccc help japanese", cette application est ensuite désinstallée

```
$tes = get-wmiobject -query "select * from win32_product where name='ccc help japanese'"
$tes.uninstall()
```

- L'exemple suivant recherche les applications installées au poste contenant les caractères pc

```
Get-WmiObject -Class Win32_Product | Where-Object -FilterScript {$_.Name -like "*pc*"}  
-----
```

Administre Windows avec WMI

- Test de ping sur plusieurs adresses IP avec la classe win32_pingStatus

```
#permet de tester une plage d'adresses IP
$id=read-host "choisir votre id réseau (ex:192.168.18)"
[int]$dep=read-host "choisir l'adresse ip de départ(ex:100)"
[int]$fin=read-host "choisir l'adresse ip de fin (ex:200)"
for($x=$dep;$x -le $fin;$x++)
{
$pc="$id.$x"
if ((gwmi win32_pingStatus -filter("Address='" + $pc + "' and Timeout=300")).statusCode -eq "0")
{
    write-host "Station $pc joignable, suite en cours ..." -foregroundcolor darkgreen -back white
}
else
{
    write-host "Station $pc non joignable" -foregroundcolor "red" -back Yellow
}
}
```

```

choisir votre id réseau (ex:192.168.18) : 192.168.18
choisir l'adresse ip de départ(ex:100) : 200
choisir l'adresse ip de fin (ex:200) : 210
Station 192.168.18.200 non joignable
Station 192.168.18.201 non joignable
Station 192.168.18.202 non joignable
Station 192.168.18.203 non joignable
Station 192.168.18.204 non joignable
Station 192.168.18.205 non joignable
Station 192.168.18.206 non joignable
Station 192.168.18.207 non joignable
Station 192.168.18.208 non joignable
Station 192.168.18.209 joignable, suite en cours ...
Station 192.168.18.210 non joignable

```

- And Timeout définit le temps à partir duquel on considère que la machine ne répond pas, accélère les temps de réponse (TimeOut en ms, par défaut 4000 ms soit 4 s ce qui est souvent trop long pour l'analyse de centaine de machine)
 - `if((gwmi win32_pingStatus -filter("Address='" + $pc + "' and Timeout=300")).statusCode -eq 0)`

Administre Windows avec WMI

- Outils de monitoring
 - ◆ Le script suivant récupère à partir d'un fichier une liste de serveur et retourne des informations sur ces serveurs (adresse ip, mac, partage, os...)

```
# $liste récupère la liste des serveurs
$liste=get-content C:\temp\serveurs.txt
$cpt=0
$taille=$liste.length
# la boucle foreach permet d'analyser chaque ordinateur dans le fichier
foreach($pc in $liste)
{
    $cpt++
    Write-Progress -Activity "Barre de progression" -status "%effectué" -PercentComplete $($cpt/$taille) * 100
    # si la station répond, on continue les tests
    if((gwmi win32_pingstatus -filter ("Address='`" + $pc + "' and timeout=300")).statuscode -eq 0)
    {
        $os=gwmi win32_operatingsystem -ComputerName $pc
        $disk=gwmi win32_logicaldisk -ComputerName $pc | ?{$_._drivetype -eq 3}
        $net=gwmi win32_networkadapterconfiguration -ComputerName $pc | ?{$_._ipenabled}
        $comp=gwmi win32_computersystem -ComputerName $pc
        $bat=gwmi win32_battery -ComputerName $pc
        $bios=gwmi win32_bios -ComputerName $pc
        $part=gwmi win32_share -ComputerName $pc
        $mon=gwmi win32_desktopmonitor -ComputerName $pc

        $part=gwmi win32_share -ComputerName $pc
        $mon=gwmi win32_desktopmonitor -ComputerName $pc

        write-host $($"-" * 50)
        write-host "Station $pc joignable, suite des tests en cours..." -fore darkGreen -BackgroundColor white
        write-host $($"-" * 50)
        write-host "Analyse de l'operating system" -fore green
        write-host "Os installé: $($os.caption)"
        write-host "Service pack : $($os.csdition)"
        write-host "Mémoires physiques dispos: $($os.FreePhysicalMemory)"
        write-host "Architecture: $($os.OSArchitecture)"
        write-host "Dernier démarrage le: $($os.lastbootuptime.substring(0,14))"
        write-host $($"-" * 50) -fore yellow
```

Administre Windows avec WMI

- Modifier une adresse IP
 - ◆ Activer DHCP
 - Get-wmiobject win32_networkadapterconfiguration -filter index=7 | foreach-object {\$_.enabledhcp () }
 - ◆ Adresse statique
 - Get-wmiobject win32_networkadapterconfiguration -filter index=7 | foreach-object {\$_.enablestatic ('192.168.18.198','255.255.255.0')}
 - ◆ Gateway
 - Gwmi win32_networkadapterconfiguration –filter index=7 | %{\$_.setGateways('192.168.18.254')}

```
PS C:\Users\Administrateur> gwmi win32_networkadapterconfiguration | get-member

 TypeName: System.Management.ManagementObject#root\cimv2\Win32_NetworkAdapterC
Name          MemberType  Definition
----          -----    -----
DisableIPSec Method     System.Management.ManagementBaseObject
EnableDHCP   Method     System.Management.ManagementBaseObject
EnableIPSec   Method     System.Management.ManagementBaseObject
EnableStatic   Method     System.Management.ManagementBaseObject
ReleaseDHCPLease Method     System.Management.ManagementBaseObject
RenewDHCPLease Method     System.Management.ManagementBaseObject
SetDNSDomain  Method     System.Management.ManagementBaseObject
SetDNSServerSearchOrder Method     System.Management.ManagementBaseObject
SetDynamicDNSRegistration Method     System.Management.ManagementBaseObject
SetGateways   Method     System.Management.ManagementBaseObject
SetIPConnectionMetric  Method     System.Management.ManagementBaseObject
SetIPXFrameTypeNetworkPairs Method     System.Management.ManagementBaseObject
SetTcpipNetbios  Method     System.Management.ManagementBaseObject
SetWINSServer   Method     System.Management.ManagementBaseObject
ArpAlwaysSourceRoute Property  System.Boolean ArpAlwaysSourceRoute {get;set;}
ArpUseEtherSNAP  Property  System.Boolean ArpUseEtherSNAP {get;set;}
Caption        Property  System.String Caption {get;set;}
```

Pour voir toutes les propriétés et méthodes, utiliser le get-member

UTILISATION DE .NET ET COM VIA POWERSHELL

Utilisation de .NET et COM via PowerShell

Utilisation du .NET

Les Assemblies

- Powershell utilise les classes .NET
 - ◆ Pour correctement fonctionner , certaines assemblies sont chargées au démarrage de PowerShell
 - ◆ une assemblie est constitué d'un exécutable et de plusieurs DLL
 - ◆ Pour voir les assemblies chargé, tapez la commande suivante

```
strateur> [System.AppDomain]::CurrentDomain.GetAssemblies()
```

Location
C:\Windows\Microsoft.NET\Framework64\v2.0.50727\mscorlib.dll
C:\Windows\assembly\GAC_MSIL\Microsoft.PowerShell.ConsoleHost\1.0.0.0_31bf3856ad3d0e61\System.dll
C:\Windows\assembly\GAC_MSIL\System.Management.Automation\1.0.0.0_31bf3856ad3d0e61\System.Management.Automation.dll
C:\Windows\assembly\GAC_MSIL\Microsoft.PowerShell.Commands.Diagnostics\1.0.0.0_31bf3856ad3d0e61\System.Diagnostics.PowerShell.dll
C:\Windows\assembly\GAC_MSIL\System.Core\3.5.0.0_b77a5c561934e089\System.Core.dll
C:\Windows\assembly\GAC_MSIL\System.Configuration.Install\2.0.0.0_b03f5f7f1c2b39f0\System.Configuration.Install.dll
C:\Windows\assembly\GAC_MSIL\Microsoft.WSMAN.Management\1.0.0.0_31bf3856ad3d0e61\System.Management.WSMAN.dll
C:\Windows\assembly\GAC_64\System.Transactions\2.0.0.0_b77a5c561934e089\System.Transactions.dll
C:\Windows\assembly\GAC_MSIL\Microsoft.PowerShell.Commands.Utility\1.0.0.0_31bf3856ad3d0e61\System.Management.Automation.Utility.dll
C:\Windows\assembly\GAC_MSIL\Microsoft.PowerShell.Commands.Management\1.0.0.0_31bf3856ad3d0e61\System.Management.Automation.Management.dll
C:\Windows\assembly\GAC_MSIL\Microsoft.PowerShell.Security\1.0.0.0_31bf3856ad3d0e61\System.Security.PowerShell.dll
C:\Windows\assembly\GAC_MSIL\Microsoft.PowerShell.ConsoleHost.resources\1.0.0.0_31bf3856ad3d0e61\System.Resources.PowerShell.dll
C:\Windows\assembly\GAC_MSIL\System.Xml\2.0.0.0_b77a5c561934e089\System.Xml.dll
C:\Windows\assembly\GAC_MSIL\System\2.0.0.0_b77a5c561934e089\System.dll

Les Assemblies

- Chaque assemblie possède plusieurs méthodes.
 - ◆ Les méthodes sont lisibles avec GetExportedTypes()

```
PS C:\Users\Administrateur> $assemb[1].GetExportedTypes()

IsPublic IsSerial Name                                     BaseType
-----  ----- 
True      False   ConsoleShell                         System.Object
True      False   StartTranscriptCommand               System.Management.Automation
True      False   StopTranscriptCommand                System.Management.Automation
True      False   PSHostPSSnapIn                      System.Management.Automation
True      False   EngineInstaller                     System.Management.Automation
True      False   UnmanagedPSEntry                    System.Object

PS C:\Users\Administrateur> $assemb[0].GetExportedTypes()

IsPublic IsSerial Name                                     BaseType
-----  ----- 
True      True    Object                                System.Object
True      False   ICloneable                           System.Object
True      False   IEnumerable                          System.Object
True      False   ICollection                          System.Object
True      False   IList                               System.Object
True      True    Array                               System.Object
True      False   IComparable`1                       System.Object
True      True    ValueType                           System.Object
True      False   IEnumerator                         System.Object
True      False   IDisposable                         System.Object
True      False   IComparable`1                       System.ValueType
True      True    ArraySegment`1                     System.ValueType
True      False   IComparable`1                       System.ValueType
True      False   IEquatable`1                        System.ValueType
True      False   IFormattable                        System.ValueType
True      False   IConvertible                        System.ValueType
True      True    Enum                                System.ValueType
```

Charger une assemblie

- **Charger une assemblie**
- Pour charger une assemblie, il n'existe pas de commandelettes mais nous utilisons une classe .NET :
 - ◆ [System.Reflection.Assembly]::LoadWithPartialName(nomAssembly)
 - ◆ Exemple:

```
#pour utiliser les types graphiques dans le framework, il faut charger l'assembly
#system.windows.forms qui contient toutes les classes pour la création d'interface
#graphique.
[System.Reflection.Assembly]::LoadWithPartialName('System.windows.forms')
-----
```

Manipuler les objets du .NET

- L'utilisation du .NET permet d'utiliser des centaines de classes
- Nous allons utiliser plusieurs exemple comme:
 - ◆ Utilisation de formulaire pour enregistrer des personnes dans l'ad
 - ◆ Envoi de message par smtp
 - ◆ ...
- Ce premier ex permet de transformer une chaine de type string en type entier

```
PS C:\Users\Administrateur> $r=read-host "saisir un chiffre"
saisir un chiffre: 12
PS C:\Users\Administrateur> $r.GetType()

IsPublic IsSerial Name                                     BaseType
-----  -----  -- String                                 System.Object

PS C:\Users\Administrateur> $r=[System.convert]::ToInt32($r)
PS C:\Users\Administrateur> $r.GetType()

IsPublic IsSerial Name                                     BaseType
-----  -----  -- Int32                                System.ValueType
```

Manipuler les objets du .NET

- Nous allons voir quelques exemples de classes
 - ◆ L'option **get-member –static** permet d'obtenir les classes statique associé avec la classe math (cos, pow, sqrt..)

```
PS C:\Users\w7> [system.math] | get-member -static

    TypeName: System.Math

Name      MemberType  Definition
---      ~~~~~~      ~~~~~~
Abs      Method      static System.SByte Abs(System.SByte va...
Acos     Method      static double Acos(double d)
Asin     Method      static double Asin(double d)
Atan     Method      static double Atan(double d)
Atan2    Method      static double Atan2(double y, double x)
BigMul   Method      static long BigMul(int a, int b)
Ceiling  Method      static decimal Ceiling(decimal d), stat...
Cos      Method      static double Cos(double d)
Cosh     Method      static double Cosh(double value)
DivRem   Method      static int DivRem(int a, int b, System.
Equals   Method      static bool Equals(System.Object objA, ...
Exp      Method      static double Exp(double d)
Floor    Method      static decimal Floor(decimal d), stat...
IEEERemainder Method  static double IEEERemainder(double x, d...
Log      Method      static double Log(double d), static dou...
Log10    Method      static double Log10(double d)
Max      Method      static System.SByte Max(System.SByte va...
Min      Method      static System.SByte Min(System.SByte va...
Pow      Method      static double Pow(double x, double y)
ReferenceEquals Method  static bool ReferenceEquals(System.OBJ...
```

- ◆ [system.math] permet d'appeler la classe math

```
PS C:\Users\w7> [system.math]::sqrt(27)
5,19615242270663
PS C:\Users\w7> [system.math]::sqrt(25)
5
PS C:\Users\w7> [system.math]::pow(2,5)
32
```

Manipuler les objets du .NET

- [system.environment] contient des informations générales sur l'environnement de travail

```
PS C:\Users\w7> [system.environment] | get-member -static

    TypeName: System.Environment

Name          MemberType  Definition
---          -----
Equals        Method     static bool Equals<Sys...
Exit          Method     static void Exit()
ExpandEnvironmentVariables Method   static string ExpandEn...
FailFast       Method     static void FailFast()
GetCommandLineArgs  Method   static string[] GetCom...
GetEnvironmentVariable Method   static string GetEnvir...
GetEnvironmentVariables Method   static System.Collections.I...
GetFolderPath  Method     static string[] GetFold...
GetLogicalDrives Method   static string[] GetLog...
ReferenceEquals Method   static bool ReferenceE...
SetEnvironmentVariable Method   static void SetEnvir...
CommandLine    Property   static System.String Command...
CurrentDirectory Property  static System.String CurrentD...
ExitCode       Property  static System.Int32 ExitCode
HasShutdownStarted Property static System.Boolean HasShutd...
MachineName    Property  static System.String MachineName
NewLine         Property  static System.String NewLine
OSVersion      Property  static System.OperatingSystem OS...
ProcessorCount Property static System.Int32 ProcessorC...
StackTrace     Property  static System.String StackTrace
SystemDirectory Property  static System.String SystemDir...
TickCount      Property static System.Int32 TickCount
UserDomainName Property  static System.String UserDomainName
UserInteractive Property static System.Boolean UserInteractive
UserName       Property  static System.String UserName
Version        Property  static System.Version Version
WorkingSet      Property static System.IntPtr WorkingSet
```

```
PS C:\Users\w7> [system.environment]::getlogicaldrives()
C:\
D:\
E:\
F:\
G:\
H:\
I:\
PS C:\Users\w7> [system.environment]::hasShutdownStarted
False
PS C:\Users\w7> [system.environment]::machinename
W7-PC
PS C:\Users\w7> [system.environment]::version
Major  Minor  Build  Revision
----  ----  -----  -----
2      0       50727  5446

PS C:\Users\w7> [system.environment]::OSversion
Platform ServicePack
Win32NT Service Pack 1
```

hasShutdownStarted indique si la machine est en train de s'arrêter.

Manipuler les objets du .NET

- Les classes du Framework sont extrêmement volumineuses
 - ◆ Le site de Microsoft permet de lister les classes utilisables, nous retrouvons par ex la classe **System.Convert**

The screenshot shows a web browser displaying the Microsoft MSDN library page for the `Convert` class. The URL is <http://msdn.microsoft.com/fr-fr/library/system.convert.aspx>. The page title is "Convert, classe". Key sections visible include:

- Hiérarchie d'héritage:** Shows inheritance from `System.Object` and `System.Convert`.
- Espace de noms :** `System`
- Assembly :** `mscorlib` (dans `mscorlib.dll`)
- Syntaxe:** Buttons for VB, C#, C++, F#, JScript.
- Méthodes:** A table showing two methods:

	Nom	Des
	<code>ChangeType(Object, Type)</code>	Retourne
	<code>ChangeType(Object, TypeCode)</code>	Retourne

Nous pouvons lister toutes les propriétés méthode de cette classe
Pour l'utiliser en powershell
[System.convert]:: Méthode
Ou propriété

Manipuler les objets du .NET

- Nous pouvons par exemple de cette classe utiliser les méthode To...
 - Exemple
 - ◆ \$r=[System.Convert]::ToInt32(\$r) convertit la valeur en entier long

	ToInt16(UInt64)	Convertit la valeur de l'entier non signé 64 bits spécifié en un entier signé 16 bits équivalent.
	ToInt16(Object, IFormatProvider)	Convertit la valeur de l'objet spécifié en un entier signé 16 bits à l'aide des informations de mise en forme spécifiées.
	ToInt16(String, IFormatProvider)	Convertit la représentation sous forme de chaîne spécifiée d'un nombre en un entier signé 16 bits équivalents à la culture spécifiées.
	ToInt16(String, Int32)	Convertit la représentation sous forme de chaîne d'un nombre dans une base spécifiée en un entier signé 16 bits équivalent.
	ToInt32(Boolean)	Convertit la valeur booléenne spécifiée en entier signé 32 bits équivalent.
	ToInt32(Byte)	Convertit la valeur de l'entier non signé 8 bits spécifié en entier signé 32 bits équivalent.
	ToInt32(Char)	Convertit la valeur du caractère Unicode spécifié en entier signé 32 bits équivalent.
	ToInt32(DateTime)	L'appel de cette méthode lève toujours <code>InvalidOperationException</code> .
	ToInt32(Decimal)	Convertit la valeur du nombre décimal spécifié en un entier signé 32 bits équivalent.
	ToInt32(Double)	Convertit la valeur du nombre à virgule flottante double précision spécifié en entier signé 32 bits équivalent.
	ToInt32(Int16)	Convertit la valeur de l'entier signé 16 bits spécifié en un entier signé 32 bits équivalent.
	ToInt32(Int32)	Retourne l'entier signé 32 bits spécifié. Aucune conversion n'est effectivement effectuée.
	ToInt32(Int64)	Convertit la valeur de l'entier signé 64 bits spécifié en un entier signé 32 bits équivalent.
	ToInt32(Object)	Convertit la valeur de l'objet spécifié en un entier signé 32 bits.

<http://msdn.microsoft.com/fr-fr/library/system.aspx>

Manipuler les objets du .NET

■ Envoi de mail

```
# envoi d'un mail en powershell

$expediteur = "expediteur@test.fr"
$destinataire = "gaelmarot@wanadoo.fr"
$serveur = "smtp.wanadoo.fr"
$fichier = "c:\temp\noms.txt"
$objet = "Envoi de mail via powershell " + [System.DateTime]::Now
$texte = "texte"

$message = new-object System.Net.Mail.MailMessage $expediteur, $destinataire, $objet, $texte

$attachment = new-object System.Net.Mail.Attachment $fichier
$message.Attachments.Add($attachment)
$SMTPclient = new-object System.Net.Mail.SmtpClient $serveur
#$SMTPclient.Credentials = [System.Net.CredentialCache]::DefaultNetworkCredentials

$SMTPclient.Send($message)
```

Si l'on souhaitait utiliser des credentials autres que ceux de l'utilisateur courant, il est possible de les spécifier ainsi :

```
$SMTPAuthUsername = "username"
$SMTPAuthPassword = "password"
$SMTPClient.Credentials = New-Object System.Net.NetworkCredential($SMTPAuthUsername, $SMTPAuthPassword)

$SMTPclient.Send($message)
```

Manipuler les objets du .NET

Réveil ordinateur

- Le WakeOnLan permet d'allumer un poste éteint via l'envoi sur le réseau d'une suite d'octet (nommé paquet magique)
- Suite de 102 octets -> 6 FF puis 16 * @mac

```
#utilisation de WakeOfLan (réveil d'un pc
#on saisit l'adresse mac de la machine à réveiller
#le wakeOfLan est l'envoi de 102 octets
#les 6 premiers sont FF suivi ensuite de 16 fois la répétition de l'adresse mac
$mac = [byte[]](0x14, 0x2A, 0x34, 0xE5, 0x56, 0xCF)
$paquet = [byte[]](0xFF,0xFF,0xFF,0xFF,0xFF,0xFF )
$paquet = $paquet + $mac*16
write-host $paquet
#on utilise la classe udpclient pour "réveiller" l'ordi
#nous faisons un broadcast (255.255.255.255) sur le port 38600 (port qqconque mais libre)
$UDPclient = new-Object System.Net.Sockets.UdpClient
$UDPclient.Connect(([System.Net.IPAddress]::Broadcast), 38600)
#.send envoi sur le réseau le paquet
$UDPclient.Send($paquet, $paquet.Length)
```

Frame 8 (144 bytes on wire, 144 bytes captured)

Ethernet II, Src: 1c:75:08:d9:04:05 (1c:75:08:d9:04:05), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

Internet Protocol Version 4, Src: 192.168.41.205 (192.168.41.205), Dst: 255.255.255.255 (255.255.255.255)

User Datagram Protocol, Src Port: 64645 (64645), Dst Port: 38600 (38600)

Source port: 64645 (64645)
Destination port: 38600 (38600)
Length: 110
Checksum: 0xeb4e [validation disabled]

Wake On LAN, MAC: Dell_b5:78:ef (a4:ba:db:b5:78:ef)
Sync stream: FFFFFFFFFFFF
MAC: Dell_b5:78:ef (a4:ba:db:b5:78:ef)
MAC: Dell_b5:78:ef (a4:ba:db:b5:78:ef)
MAC: Dell_b5:78:ef (a4:ba:db:b5:78:ef)
MAC: Dell_b5:78:ef (a4:ba:db:b5:78:ef)
MAC: Dell_b5:78:ef (a4:ba:db:b5:78:ef)

Paquet capturé avec Wireshark, on retrouve le paquet WOL...

Manipuler les objets du .NET

- Utilisation du DNS
- On peut interroger les serveurs comme le dns par ex
 - ◆ L'exemple suivant retourne le nom de la station via dns

```
PS C:\Users\Administrateur> [system.net.dns]::gethostname()
srv2K8r2m
PS C:\Users\Administrateur> [system.net.dns]::gethostname().toupper()
SRV2K8R2M
```

```
PS C:\Users\Administrateur> $dns=[system.net.dns]
PS C:\Users\Administrateur> $dns::gethostaddresses($dns::gethostname())

Address           : fe80::395b:264d:3697:825%15
AddressFamily     : InterNetworkV6
ScopeId           : 15
IsIPv6Multicast  : False
IsIPv6LinkLocal   : True
IsIPv6SiteLocal   : False
IPAddressToString : fe80::395b:264d:3697:825%15

Address           : fe80::c88a:2adf:948c:92bb%11
AddressFamily     : InterNetworkV6
ScopeId           : 11
IsIPv6Multicast  : False
IsIPv6LinkLocal   : True
IsIPv6SiteLocal   : False
IPAddressToString : fe80::c88a:2adf:948c:92bb%11

Address           : 3442059456
AddressFamily     : InterNetwork
ScopeId           : 
IsIPv6Multicast  : False
IsIPv6LinkLocal   : False
IsIPv6SiteLocal   : False
IPAddressToString : 192.168.41.205

Address           : 318875840
AddressFamily     : InterNetwork
ScopeId           : 
IsIPv6Multicast  : False
IsIPv6LinkLocal   : False
IsIPv6SiteLocal   : False
IPAddressToString : 192.168.1.19
```

L'exemple suivant retourne les adresses ip de l'ordinateur courant

Manipuler les objets du .NET

- L'exemple suivant résout une adresse IP en un nom d'hôte
 - ◆ (La zone inversée doit être créée sinon netbios)

```
C:\Users\Administrateur
PS C:\Users\Administrateur> $dns=[System.net.dns]
PS C:\Users\Administrateur> $dns::GethostByAddress("192.168.41.151")
HostName                               Aliases                              AddressList
-----                               ----                               -----
WIN-19RN8M9MUH4                          {}                                {192.168.41.151}
```

Manipuler les objets du .NET

- La classe ***system.security.principal.windowsIdentity***
 - ◆ Cette classe est intéressante retourne des informations sur l'utilisateur courant

```
PS F:\w2k8\cours\script powershell> [system.Security.principal.windowsIdentity]::getCurrent()

AuthenticationType : NTLM
ImpersonationLevel : None
IsAuthenticated    : True
IsGuest             : False
IsSystem            : False
IsAnonymous         : False
Name                : SRV2K8R2M\Administrateur
Owner               : S-1-5-32-544
User                : S-1-5-21-367607446-2157560905-199202089-500
Groups              : {S-1-5-21-367607446-2157560905-199202089-513, S-1-1-0, S-1-5-32-544, S-1-5-32-545...}
Token               : 2524

PS F:\w2k8\cours\script powershell> [system.Security.principal.windowsIdentity]::getCurrent().authenticationType
NTLM
PS F:\w2k8\cours\script powershell> [system.Security.principal.windowsIdentity]::getCurrent().user
      BinaryLength AccountDomainSid          Value
-----           -----           -----
          28   S-1-5-21-367607446-2157560905-199202089  S-1-5-21-367607446-2157560905-199202...
```

Utilisation de .NET et COM via PowerShell

Utilisation des formulaires

Utilisation des formulaires

- Pour utiliser les formulaires, on fait appel à l'assemblie **system.windows.forms**
- Cette classe permet la création d'interface graphique (GUI)
- Le framework .NET utilise cette classe pour la création d'objet graphique

```
#pour utiliser les types graphiques dans le framework, il faut charger l'assembly  
#system.windows.forms qui contient toutes les classes pour la création d'inter-  
#graphique.  
[System.Reflection.Assembly]::LoadWithPartialName('System.windows.forms')  
#création du formulaire principal  
$formulaire=New-Object windows.Forms.Form  
#Création d'un titre  
$formulaire.text="création d'utilisateur"  
#dimension du formulaire  
$formulaire.size=New-Object system.Drawing.Size @(500,250)  
#la méthode showdialog permet d'afficher le formulaire.  
$formulaire.ShowDialog()
```

Nous chargeons
l'assembleur pour les
formulaires

Déclaration d'un nouveau
contrôle type formulaire

Titre du formulaire

Taille du contrôle (tableau avec
@ pour la hauteur et margeur

Ce premier ex affiche simplement un formulaire.
Chaque objet dans le jargon est appelé un contrôle.

Un contrôle est un objet comme:

- . un bouton radio
- . Un champ de type texte
- . Une liste déroulante
-



Utilisation des formulaires

```
#pour utiliser les types graphiques dans le framework, il faut charger l'assembly
#system.windows.forms qui contient toutes les classes pour la création d'interface
#graphique.
[System.Reflection.Assembly]::LoadWithPartialName('System.windows.forms')
#####
#création du formulaire principal
#####
$formulaire=New-Object windows.Forms.Form
#Création d'un titre
$formulaire.Text="création d'utilisateur"
#dimension du formulaire
#nécessite pour cela d'appeler la propriété system.drawing.size
#@ (500,250) indique un tableau @(longueur, largeur)
$formulaire.size=New-Object system.Drawing.Size @(500,250)
#####
#Création du bouton Annuler
#####
$bouton_annuler=New-Object system.Windows.Forms.Button
$bouton_annuler.Text = "Annuler"
#positionnement du bouton dans le formulaire et dimensionnement du bouton
$bouton_annuler.Location = New-Object system.Drawing.Size(350,150)
$bouton_annuler.Size= New-Object system.Drawing.Size(100,25)
#maintenant, il faut ajouter ce bouton 'annuler' dans le formulaire
$formulaire.Controls.Add($bouton_annuler)
#gestion de l'événement clic du bouton
#nous souhaitons ici fermer le formulaire lorsque nous cliquons sur le bouton annuler
#le code se met entre accolade
$bouton_annuler.add_click{
    {
        $formulaire.Close()
    }
}
#####
#la méthode ShowDialog permet d'afficher le formulaire.
$formulaire.ShowDialog()
```

Cet exemple ajoute un bouton en bas à droite, nous codons dans un événement add_click (lorsque l'on clique sur le bouton, on déclenchera la fermeture du formulaire)



Utilisation des formulaires

- L'exemple suivant utilise un contrôle label plus un timer.
 - Nous souhaitons dans cet exemple mettre à jour toutes les 2 secondes la date
 - Nous commençons par le même code que tout à l'heure

```
#pour utiliser les types graphiques dans le framework, il faut charger l'assembly
#system.windows.forms qui contient toutes les classes pour la création d'interface
#graphique.
[System.Reflection.Assembly]::LoadWithPartialName('System.windows.forms')
#####
#création du formulaire principal
#####
$formulaire=New-Object windows.Forms.Form
#Création d'un titre
$formulaire.text="création d'utilisateur"
#dimension du formulaire
#nécessite pour cela d'appeler la propriété system.drawing.size
#@ (500,250) indique un tableau @ (longueur, largeur)
$formulaire.size=New-Object system.Drawing.Size @(500,250)
#####
#Création du bouton Annuler
#####
$bouton_annuler=New-Object system.Windows.Forms.Button
$bouton_annuler.Text = "Annuler"
#positionnement du bouton dans le formulaire et dimensionnement du bouton
$bouton_annuler.Location = New-Object system.Drawing.Size(350,150)
$bouton_annuler.Size= New-Object system.Drawing.Size(100,25)
#maintenant, il faut ajouter ce bouton 'annuler' dans le formulaire
$formulaire.Controls.Add($bouton_annuler)
#gestion de l'événement clic du bouton
#nous souhaitons ici fermer le formulaire lorsque nous cliquons sur le bouton annuler
#le code se met entre accolade
$bouton_annuler.add_click(
{
    $formulaire.Close()
})
```

Utilisation des formulaires

```
$bouton_annuler.add_click(
{
    $formulaire.Close()
}
)
#####
#Création d'un label (étiquette permettant d'afficher l'heure et une
#mise à jour toutes les 2 s
#####
$labelHeure=New-object system.Windows.forms.label
$labelHeure.location=New-Object system.Drawing.Size(50,50)
$labelHeure.autosize=$true
$formulaire.controls.add($labelHeure)
### création de l'objet Timer permettant de mettre à jour notre étiquette toute les 2 s
$temps=new-object system.Windows.forms.Timer
### Interval définit le temps de rafraîchissement
$temps.interval=2000
### nous entrons maintenant le code concernant les champs qui devront
## être mise à jour, ici nous souhaitons seulement mettre l'étiquette à jour toutes les 2s
$temps.add_tick(
{
    $labelHeure.Text="nous sommes le $(get-date)"
})
### nous démarrons le timer
$temps.start()

#la méthode ShowDialog permet d'afficher le formulaire.
$formulaire.ShowDialog()
```

Nous ajoutons le contrôle label, nous le positionnons dans le formulaire
Autosize permet d'ajuster la taille du label par rapport au contenu.

Nous ajoutons avec controls.add le label dans le formulaire

Nous déclarons un nouvel objet ...forms.Timer
Add_tick permet d'exécuter du code qui sera « rafraîchi par le timer »
Ici nous souhaitons mettre à jour le label.
\$temps.Start() déclenche le timer

Utilisation des formulaires

- L'exemple suivant nous permet d'enregistrer un utilisateur dans l'active directory
 - ◆ \$obj est notre variable nous permettant de se connecter à l'active directory

```
#pour utiliser les types graphiques dans le framework, il faut charger l'assembly
#system.windows.forms qui contient toutes les classes pour la création d'interface
#graphique.
#####
#connexion à l'ad
#####
$obj=[ADSI]"LDAP://192.168.41.1/ou=redon,dc=gtek,dc=local"
#####

[System.Reflection.Assembly]::LoadWithPartialName('System.windows.forms')
#####
#création du formulaire principal
#####
$formulaire=New-Object windows.Forms.Form
#Création d'un titre
$formulaire.text="création d'utilisateur"
#dimension du formulaire
#necessite pour cela d'appeler la propriété system.drawing.size
#@ (500,250) indique un tableau @(longueur, largeur)
$formulaire.size=New-Object system.Drawing.Size @(600,350)
#####
#Création d'un champ type texte pour le nom et prénom, mot de passe du user
#####
```

Nous déclarons notre
formulaire de type
Forms

Utilisation des formulaires

- Nous saisissons ici les différents champs nécessaires pour enregistrer notre user.
 - ◆ TextBox est un champ de type Texte

```
#####
#Création d'un champ type texte pour le nom et prénom, mot de passe du user
#####
$prenomLabel = new-object system.Windows.Forms.Label
$prenomLabel.text="Prénom"
$prenomLabel.Location = New-Object system.Drawing.Size(10,50)
$prenomLabel.Size= New-Object system.Drawing.Size(50,25)
$formulaire.Controls.Add($prenomLabel)
$prenom = New-Object system.Windows.Forms.TextBox
$prenom.Location = New-Object system.Drawing.Size(100,50)
$prenom.Size= New-Object system.Drawing.Size(150,25)
$formulaire.Controls.Add($prenom)

$nomLabel = new-object system.Windows.Forms.Label
$nomLabel.text="Nom"
$nomLabel.Location = New-Object system.Drawing.Size(10,80)
$nomLabel.Size= New-Object system.Drawing.Size(50,25)
$formulaire.Controls.Add($nomLabel)
$nom = New-Object system.Windows.Forms.TextBox
$nom.Location = New-Object system.Drawing.Size(100,80)
$nom.Size= New-Object system.Drawing.Size(150,25)
$formulaire.Controls.Add($nom)

$PassLabel = new-object system.Windows.Forms.Label
$PassLabel.text="Mot de passe"
$PassLabel.Location = New-Object system.Drawing.Size(10,110)
$PassLabel.Size= New-Object system.Drawing.Size(50,50)
$formulaire.Controls.Add($PassLabel)
$pass = New-Object system.Windows.Forms.TextBox
$pass.Location = New-Object system.Drawing.Size(100,110)
$pass.Size= New-Object system.Drawing.Size(150,25)
$formulaire.Controls.Add($pass)

$label2 = New-Object System.Windows.Forms.Label
$label2.Location = new-object System.Drawing.Size(200,220)
$label2.size = new-object System.Drawing.Size(200,60)
$label2.Text="Résumé:"
$formulaire.Controls.Add($label2)
```

Utilisation des formulaires

- Nous ajoutons ici les boutons valider et annuler

```
#####
#Création du bouton Annuler et valider
#####
$bouton_annuler=New-Object system.Windows.Forms.Button
$bouton_annuler.Text = "Annuler"
#positionnement du bouton dans le formulaire et dimensionnement du bouton
$bouton_annuler.Location = New-Object system.Drawing.Size(350,150)
$bouton_annuler.Size= New-Object system.Drawing.Size(100,25)
#maintenant, il faut ajouter ce bouton 'annuler' dans le formulaire
$formulaire.Controls.Add($bouton_annuler)
#gestion de l'évenement clic du bouton
#nous souhaitons ici fermer le formulaire lorsque nous cliquons sur le bouton annuler
#le code se met entre accolade
$bouton_annuler.add_click(
{
    $formulaire.Close()
}
)

$bouton_Valider=New-Object system.Windows.Forms.Button
$bouton_Valider.Text = "Valider"
#positionnement du bouton dans le formulaire et dimensionnement du bouton
$bouton_valider.Location = New-Object system.Drawing.Size(200,150)
$bouton_valider.Size= New-Object system.Drawing.Size(100,25)
#maintenant, il faut ajouter ce bouton 'annuler' dans le formulaire
$formulaire.Controls.Add($bouton_valider)
#gestion de l'évenement clic du bouton
#nous souhaitons ici fermer le formulaire lorsque nous cliquons sur le bouton annuler
#le code se met entre accolade
$bouton_valider.add_click(
{
    $formulaire.Close()
}
)
```

Utilisation des formulaires

- ◆ Ce code permet en premier d'afficher dans une étiquette le prénom et le nom de l'utilisateur
- ◆ Ensuite nous enregistrons dans l'AD les informations saisies par dans le formulaire

```
$bouton_valider.add_click()
{
    $label2.Text = "Résumé: " + $nom.get_text() + " " + $prenom.get_text()
    $label2.set_ForeColor("red")
    $user = $obj.Create("user","cn=" + $prenom.get_Text() + " " + $nom.get_text())
    $user.setinfo()
    $user.put("SamAccountName",$prenom.get_Text() + $nom.get_Text())
    $user.put("userPrincipalName",$prenom.get_Text() + $nom.get_Text() + "@gtek.local")
    $user.put("Description","utilisateur n° " + $nom.get_Text() + " testé pour monter en charge")
    $user.put("Displayname",$prenom.get_Text() + " " + $nom.get_Text())
    $user.put("GivenName",$prenom.get_Text())
    $user.put("sn",$nom.get_Text())
    $user.put("Mail",$prenom.get_Text() + $nom.get_Text() + "@gtek.com")
    $user.setinfo()
    $user.setpassword($pass.get_Text())
    $user.setinfo()
    $user.psbase.invokeSet("AccountDisabled",$false)
    $user.setinfo()
}
#la méthode showdialog permet d'afficher le formulaire.
$formulaire.ShowDialog()
```



Utilisation des formulaires

- L'exemple suivant permet toujours d'enregistrer des utilisateurs mais cette fois nous donnons en plus le choix de l'unité d'organisation

```
[System.Reflection.Assembly]::LoadWithPartialName('System.windows.forms')
#####
#création du formulaire principal
#####
$formulaire=New-Object windows.Forms.Form
#Création d'un titre
$formulaire.text="création d'utilisateur dans le domaine GTEK"
#dimension du formulaire
#nécessite pour cela d'appeler la propriété system.drawing.size
#@ (500,250) indique un tableau @(longueur, largeur)
$formulaire.size=New-Object system.Drawing.Size @(800,450)

#####
#Création d'un champ type texte pour le nom et prénom, mot de passe du users
# et la description, plus l'uo
#####
$prenomLabel = new-object system.Windows.Forms.Label
$prenomLabel.text="Prénom"
$prenomLabel.Location = New-Object system.Drawing.Size(10,50)
$prenomLabel.Size= New-Object system.Drawing.Size(50,25)
$formulaire.Controls.Add($prenomLabel)
$prenom = New-Object system.Windows.Forms.TextBox
$prenom.Location = New-Object system.Drawing.Size(100,50)
$prenom.Size= New-Object system.Drawing.Size(150,25)
$formulaire.Controls.Add($prenom)

$nomLabel = new-object system.Windows.Forms.Label
$nomLabel.text="Nom"
$nomLabel.Location = New-Object system.Drawing.Size(10,80)
$nomLabel.Size= New-Object system.Drawing.Size(50,25)
$formulaire.Controls.Add($nomLabel)
$nom = New-Object system.Windows.Forms.TextBox
$nom.Location = New-Object system.Drawing.Size(100,80)
$nom.Size= New-Object system.Drawing.Size(150,25)
$formulaire.Controls.Add($nom)
```

Utilisation des formulaires

```
$PassLabel = new-object system.Windows.Forms.Label
$PassLabel.text="Mot de passe"
$PassLabel.Location = New-Object system.Drawing.Size(10,110)
$PassLabel.Size= New-Object system.Drawing.Size(50,50)
$formulaire.Controls.Add($PassLabel)

$pass = New-Object system.Windows.Forms.TextBox
$pass.Location = New-Object system.Drawing.Size(100,110)
$pass.Size= New-Object system.Drawing.Size(150,25)
$formulaire.Controls.Add($pass)

$DescriptionLabel = new-object system.Windows.Forms.Label
$DescriptionLabel.text="Description"
$DescriptionLabel.Location = New-Object system.Drawing.Size(300,50)
$DescriptionLabel.Size= New-Object system.Drawing.Size(60,25)
$formulaire.Controls.Add($DescriptionLabel)

$Description = New-Object system.Windows.Forms.TextBox
$Description.Location = New-Object system.Drawing.Size(400,50)
$Description.Size= New-Object system.Drawing.Size(150,25)
$formulaire.Controls.Add($description)

#$formulaire.Controls.Add($uo)

$label2 = New-Object System.Windows.Forms.Label
$label2.Location = new-object System.Drawing.Size(200,220)
$label2.size = new-object System.Drawing.Size(200,60)
$label2.Text="Résumé:"
$formulaire.Controls.Add($label2)
```

Utilisation des formulaires

- ◆ Nous ajoutons ici un contrôle type liste déroulante (combobox)
- ◆ Nous recherchons toutes les uos dans notre active directory
- ◆ Ces uos seront ensuite afficher dans la liste déroulante

```
#nous allons lister toutes les uos de notre Active directory
$uniteorg = New-Object system.Windows.Forms.ComboBox
$strCategory = "organizationalunit"
$objDomain = New-Object System.DirectoryServices.DirectoryEntry("LDAP://192.168.41.1/dc=gtek,dc=local")
$objSearcher = New-Object System.DirectoryServices.DirectorySearcher($objDomain,"(objectCategory=$strCategory)",@('name'))
$objSearcher.FindAll()
#nous listons ensuite dans notre liste déroulante toutes les uos.
$uos = $objSearcher.findall() | %{$_.properties.adspath}
foreach ($uo in $uos)
{
    $uniteorg.Items.Add($uo)
}

$uniteorg.Location = New-Object system.Drawing.Size(280,110)
$uniteorg.Size= New-Object system.Drawing.Size(300,25)
$formulaire.Controls.Add($uniteOrg)
```

Utilisation des formulaires

```
#####
#Création du bouton Annuler et valider
#####
$bouton_annuler=New-Object system.Windows.Forms.Button
$bouton_annuler.Text = "Annuler"
#positionnement du bouton dans le formulaire et dimensionnement du bouton
$bouton_annuler.Location = New-Object system.Drawing.Size(350,150)
$bouton_annuler.Size= New-Object system.Drawing.Size(100,25)
#maintenant, il faut ajouter ce bouton 'annuler' dans le formulaire
$formulaire.Controls.Add($bouton_annuler)
#gestion de l'événement clic du bouton
#nous souhaitons ici fermer le formulaire lorsque nous cliquons sur le bouton annuler
#le code se met entre accolade

$bouton_annuler.add_click(
{
    $formulaire.Close()
}
)

$bouton_Valider=New-Object system.Windows.Forms.Button
$bouton_Valider.Text = "Valider"
#positionnement du bouton dans le formulaire et dimensionnement du bouton
$bouton_valider.Location = New-Object system.Drawing.Size(200,150)
$bouton_valider.Size= New-Object system.Drawing.Size(100,25)
#maintenant, il faut ajouter ce bouton 'annuler' dans le formulaire
$formulaire.Controls.Add($bouton_valider)
#gestion de l'événement clic du bouton
#nous souhaitons ici enregistre les données de notre utilisateur
$bouton_valider.add_click(
{
    ...
}
```

Utilisation des formulaires

- Nous enregistrons l'utilisateur dans la bonne uo
 - ◆ [ADSI]\$uniteorg.Get_text() permet de se connecter sur la bonne uo

```
$bouton_valider.add_click(
{
    $label2.Text = "Résumé: " + $nom.get_text() + " " + $prenom.get_text()
    $label2.set_ForeColor("red")
    $obj=[ADSI]$uniteorg.get_text()
    $user = $obj.Create("user","cn=" + $prenom.get_Text() + " " + $nom.get_Text())
    $user.setinfo()
    $user.put("SamAccountName",$prenom.get_Text() + $nom.get_Text())
    $user.put("userPrincipalName",$prenom.get_Text() + $nom.get_Text() + "@gtek.local")
    $user.put("Description",$description.get_text())
    $user.put("Displayname",$prenom.get_Text() + " " + $nom.get_Text())
    $user.put("GivenName",$prenom.get_Text())
    $user.put("sn",$nom.get_Text())
    $user.put("Mail",$prenom.get_Text() + $nom.get_Text() + "@gtek.com")
    $user.setinfo()
    $user.setpassword($pass.get_Text())
    $user.setinfo()
    $user.psbase.invokeSet("AccountDisabled",$false)
    $user.setinfo()
}
)
#la méthode showdialog permet d'afficher le formulaire.
$formulaire.ShowDialog()
```

Utilisation de .NET et COM via PowerShell

Les objets Com

Les objets Com

- La technologie COM (Component Object Model) permet la communication entre différentes applications
- Nous pouvons par exemple aller chercher une info dans un fichier Excel, puis recopier cette info dans un fichier Word, finir par une présentation Powerpoint, puis une présentation sur IE...
- Pour appeler un objet COM
 - ◆ `$MaVar=New-object -ComObject « application »`

Les objets Com

- Vous pouvez utiliser la voix des hauts-parleurs,
 - ◆ \$spVoice = new-object -com "SAPI.spvoice" \$spVoice.Speak("Hello the World!")
 - ◆ Possibilité de lire un fichier txt comme l'exemple ci dessous et d'entendre le contenu sur les hauts parleurs

```
$spvoice = new-object -com "SAPI.SpVoice"
foreach ($a in get-content c:\temp\test.txt)
{$spvoice.speak($a)}
```

- ◆ Cet exemple retourne sur les haut-parleurs le message "hello Mr Marot", nous sommes aujourd'hui le puis la date du jour
- ◆ Enfin nous listons tous les utilisateurs de la sam local qui nous sont retournés avec le H-P

```
$spvoice = new-object -com "SAPI.SpVoice"
$spvoice.speak("hello Mr Marot ?")
$spvoice.speak("nous sommes aujourd hui le")
$x = get-date
$spvoice.speak($x)
$spvoice.speak("les personnes présentes dans la sam locale sont")
$y = gwmi win32_useraccount | %{$spvoice.speak($_.name)}
write-host $y
sleep 12
```

Les objets Com

- Ce premier exemple permet de créer un fichier Excel et d'enregistrer une valeur dans la cellule A1.
 - ◆ Nous déclarons un nouvel Objet Com Excel.Application
 - ◆ Il est visible (en premier plan avec visible = \$true)
 - ◆ Nous ajoutons un nouveau classeur puis nous sélectionnons la première feuille de ce classeur
 - ◆ Nous prenons la cellule 1,1 (A1) dans lequel nous renseignons une valeur puis nous mettons cette cellule en gras et couleur rouge.
 - ◆ Ce classeur est enfin enregistré dans temp

```
$Excel = New-Object -comobject Excel.Application
$Excel.Visible = $True
$nouvClasseur = $Excel.Workbooks.Add()
$cellule = $nouvClasseur.Worksheets.Item(1)
$cellule.Cells.Item(1,1) = "une valeur dans A1."
#la cellule aura une police rouge et gras
$cellule.Cells.Font.Bold = $true
$cellule.Cells.Font.Color = -16776961
$nouvclasseur.SaveAs("C:\temp\Test42.xls")
$Excel.Quit()
```

Les objets Com

- Cet exemple ouvre un fichier existant et écrit sur la feuille n°2 en A1 « ceci est le titre »

```
#ouverture de l'application Excel
$excel = new-object -comobject Excel.application
#rendre excel visible
$excel.visible = $true
#récupération du fichier Excel Test
$file = "c:\temp\test42.xls"
$list = $excel.Workbooks.open($file)
#pour travailler sur Excel, on doit rendre le classeur actif
$workbook = $excel.ActiveWorkbook
#on travail sur la Feuille 2 du classeur
$worksheet = $workbook.Worksheets.item(2)
#sur la cellule 1,1 (range A1), on entre une phrase
$worksheet.cells.item(1,1) = "ceci est le titre"
```

Les objets Com

- L'exemple suivant récupère la taille des fichiers et les noms et les stocke dans un fichier Excel, si la taille dépasse les 10000 octets, on écrit en rouge.

```
#on stocke dans la variable $fichiers le nom et la taille des fichiers, on exclut les dossiers
$fichiers = get-childitem f:\w2k8\cours\part | where {$_.mode -notlike "*d*"}
#on instancie un nouvel objet com type Excel
$xls = New-Object -comobject Excel.Application
#l'application Excel est visible
$xls.Visible = $True
#ajout d'un classeur avec la méthode Add
$class = $xls.Workbooks.Add()
#on se positionne sur la feuille 1 du classeur
$feuil = $class.Worksheets.Item(1)
$ligne=1
$col=1
#on enlève les alertes et le rafraîchissement
#la boucle suivante permet d'afficher dans 2 colonnes les fichiers et la taille des fichiers
#si le fichier dépasse une certaine taille, on écrit en rouge et on mets alerte dans la col a coté
foreach ($fic in $fichiers)
{
    $feuil.Cells.Item($ligne,$col) = $fic.name
    if($fic.length -gt 10000)
    {
        $feuil.Cells.item($ligne, $col+1).font.color = -16776961
        $feuil.Cells.Item($ligne,$col +1 ) = $fic.length
        $feuil.Cells.item($ligne, $col +2)= "alerte"
    }
    else
    {
        $feuil.Cells.Item($ligne,$col +1 ) = $fic.length
    }
    $ligne++
}
#on se positionne sur la feuille 1 du classeur
$feuil = $class.Worksheets.Item(1)
$ligne=1
$col=1
$feuil.Columns.item(1).ColumnWidth = 50
```

Dimension de la colonne

	A	B	C	D
1	ActiveRoles	29955072	alerte	
2	affichage use	2151		
3	affichage use	1252		
4	affichage use	352		
5	ApplicationC	12812600	alerte	
6	choixClasse.	344		
7	computersLi	476		
	tion de	6392		
	amd64	15953408	alerte	
	os1	94		

Les objets Com

- L'exemple suivant permet de lancer Internet Explorer, de définir une taille puis de lancer google.Fr

```
#déclaration d'un objet Internet Explorer
$ie=new-object -comobject internetExplorer.application
#visible pour le voir au premier plan
$ie.visible=$true
#taille pour la hauteur et largeur
$ie.height=500
$ie.Width=600
#navigate permet de définir la page de navigation
$ie.navigate("http://www.google.fr")
#Toolbar permet de masquer ou pas les barre d'outils
$ie.toolbar=$false
```



Windows PowerShell pour des tâches d'administration

Traitement des fichiers XML

Traitement des fichiers XML

- Pour lire un fichier XML, utilisez [xml]
 - L'exemple suivant va lire ce fichier

```
<!-- file.xml -->
<salaries>
    <employe id="101">
        <nom>Marot Gael</nom>
        <age>36</age>
    </employe>
    <employe id="102">
        <nom>Rollais Fred</nom>
        <age>79</age>
    </employe>
    <employe id="301">
        <nom>jegu Daniel</nom>
        <age>102</age>
    </employe>
</salaries>
```

```
PS C:\Users\Administrateur> $xml=[xml](get-content c:\test\emp.xml)
PS C:\Users\Administrateur> $xml
```

```
#comment
-----
file.xml
```

```
salaries
-----
salaries
```

XML

- Pour lire le contenu

```
PS C:\Users\Administrateur> $xml.salaries
```

```
employe
-----
{employe, employe, employe}
```

```
PS C:\Users\Administrateur> $xml.salaries.employe
```

id	nom	age
--	---	---
101	Marot Gael	36
102	Rollais Fred	79
301	jegu Daniel	102

- Lit le contenu de la balise employe du tableau

```
PS C:\Users\Administrateur> $xml.salaries.employe[1]
```

id	nom	age
--	---	---
102	Rollais Fred	79

```
PS C:\Users\Administrateur> $xml.salaries.employe[1].nom
```

```
Rollais Fred
```

```
PS C:\Users\Administrateur> $xml.salaries.employe[1].age
```

```
79
```

```
PS C:\Users\Administrateur>
```

XML

- Nous pouvons rechercher dans notre fichier toutes les personnes de plus de 40 ans

```
PS C:\Users\Administrateur> $xml=[XML](get-content c:\test\emp.xml)
PS C:\Users\Administrateur> $xml.salaries.employe | where {$_.age -gt 40}

id                      nom                  age
--                      ---                  ---
101                     Marot Gael          43
102                     Rollais Fred        43
```

- L'exemple suivant récupère les données et les envois dans un fichier txt

```
$xml.salaries.employe | %{$_.'id' + ";" + $_.'age' + ";" + $_.'nom'} | out-file c:\test\emp.txt
```



UTILISATION DES MODULES SPÉCIFIQUES SYSTÈMES POWERSHELL

Gestion des modules PowerShell V3

Le module Active Directory

Le module Active Directory

- Powershell peut être utilisé pour dans un annuaire pour :
 - ◆ Créer des utilisateurs
 - ◆ Créer des groupes, des uos
 - ◆ Installer la corbeille dans l'ad (2008 R2)
 - ◆ Modifier des objets dans l'ad
- Le module ***activedirectory*** permet de gérer entièrement votre AD, plus de 70 cmdlets dédiés à l'Active Directory

Le module Active Directory

- Depuis 2008 R2 il est possible d'ajouter un module Active Directory pour travailler sur les objets de l'Active Directory
- Pour utiliser les nouvelles cmdlet de l'AD tapez "import-module activedirectory"

```
PS C:\Users\Administrateur> import-module activedirectory
```

- ◆ Get-command –module active directory permet de voir toutes les commandelettes utilisables dans l'ad

```
C:\Users\Administrateur.GTEK> import-module activedirectory
C:\Users\Administrateur.GTEK> get-command -module activedirectory
```

CommandType	Name	Definition
Cmdlet	Add-ADComputerServiceAccount	Add-ADCompute
Cmdlet	Add-ADDomainControllerPasswordReplicationPolicy	Add-ADDoma
Cmdlet	Add-ADFineGrainedPasswordPolicySubject	Add-ADFineGra
Cmdlet	Add-ADGroupMember	Add-ADGroupMe
Cmdlet	Add-ADPrincipalGroupMembership	Add-ADPrincip
Cmdlet	Clear-ADAuditExpiration	Clear-ADAudi
Cmdlet	Disable-ADAccount	Disable-ADAcc
Cmdlet	Disable-ADOptionalFeature	Disable-ADOpt
Cmdlet	Enable-ADAccount	Enable-ADAcco
Cmdlet	Enable-ADOptionalFeature	Enable-ADOpti
Cmdlet	Get-ADAuditAuthorizationGroup	Get-ADAuditA
Cmdlet	Get-ADAuditResultantPasswordReplicationPolicy	Get-ADAuditR

Le module Active Directory

▪ Get-AddomainController

- ◆ Affiche des informations sur les contrôleurs de domaine
- ◆ L'exemple ci-dessous affiche des informations sur les CD et demande une authentification –credential
- ◆ L' option –credential peut être utilisé avec toutes les commandes de l'ad

```
PS C:\Users\Administrateur> get-addomaincontroller -credential administrateur@gtek.local

ComputerObjectDN      : CN=WIN-TCK058CNJLJ,OU=Domain Controllers,DC=gtek,DC=local
DefaultPartition       : DC=gtek,DC=local
Domain                : gtek.local
Enabled               : True
Forest                : gtek.local
HostName              : WIN-TCK058CNJLJ.gtek.local
InvocationId          : fb113661-6a04-4a9c-9a4e-cc236810d75e
IPv4Address           : 10.44.0.198
IPv6Address           : ::1
IsGlobalCatalog        : True
IsReadOnly             : False
LdapPort              : 389
Name                  : WIN-TCK058CNJLJ
```

- ◆ Cette option permet de rechercher les CD qui sont aussi Catalogue global

```
teur> get-addomaincontroller -filter {isGlobalCatalog -eq $true}
```

Le module Active Directory

■ Get-Addomain

- ◆ Obtient des informations sur un domaine comme le nom, les maître FSMO...

```
PS C:\Users\Administrateur> get-addomain

AllowedDNSSuffixes          : <>
ChildDomains                 : <>
ComputersContainer           : OU=pc,DC=gtek,DC=local
DeletedObjectsContainer      : CN=Deleted Objects,DC=gtek,DC=local
DistinguishedName           : DC=gtek,DC=local
DNSRoot                      : gtek.local
DomainControllersContainer   : OU=Domain Controllers,DC=gtek,DC=local
DomainMode                   : Windows2008R2Domain
DomainSID                    : S-1-5-21-3564125724-175561107-1429508558
ForeignSecurityPrincipalsContainer : CN=ForeignSecurityPrincipals,DC=gtek,DC=local
Forest                       : gtek.local
InfrastructureMaster          : WIN-TCK058CNJLJ.gtek.local
LastLogonReplicationInterval : <CN={31B2F340-016D-11D2-945F-00C04FB984F9},CN=Policies,CN=System,
LinkedGroupPolicyObjects     : CN=LostAndFound,DC=gtek,DC=local
LostAndFoundContainer         :
ManagedBy                    : gtek
Name                          : GTEK
NetBIOSName                  : domainDNS
ObjectClass                  : 8f9fa8d3-5d53-400d-94bf-bdff12818708
ParentDomain                 : WIN-TCK058CNJLJ.gtek.local
PDCEmulator                  : CN=NTDS Quotas,DC=gtek,DC=local
QuotasContainer               : <>
ReadOnlyReplicaDirectoryServers : <WIN-TCK058CNJLJ.gtek.local>
ReplicaDirectoryServers       : WIN-TCK058CNJLJ.gtek.local
RIDMaster                     : <DC=ForestDnsZones,DC=gtek,DC=local, DC=DomainDnsZones,DC=gtek,DC=duration,DC=gtek,DC=local>
```

```
PS C:\Users\Administrateur> get-addomain | FT forest, RIDmaster, pdcemulator -A
forest      RIDmaster          pdcemulator
-----      -----            -----
gtek.local  WIN-TCK058CNJLJ.gtek.local  WIN-TCK058CNJLJ.gtek.local
```

Le module Active Directory

■ Get-AdForest

- ◆ Obtient des informations d'une forêt
 - Get-Adforest obtient des informations de l'utilisateur courant
- ◆ Cet exemple permet d'avoir des informations sur la forêt nommée gtek.local

```
PS C:\Users\Administrateur> get-adforest gtek.local

ApplicationPartitions : {DC=ForestDnsZones,DC=gtek,DC=local, DC=DomainDnsZones,DC=gtek,DC=local}
CrossForestReferences : {}
DomainNamingMaster   : WIN-TCK058CNJLJ.gtek.local
Domains              : {gtek.local}
ForestMode            : Windows2008R2Forest
GlobalCatalogs        : {WIN-TCK058CNJLJ.gtek.local, MYRODC.dev.gtek.local}
Name                 : gtek.local
PartitionsContainer  : CN=Partitions,CN=Configuration,DC=gtek,DC=local
RootDomain            : gtek.local
SchemaMaster          : WIN-TCK058CNJLJ.gtek.local
Sites                : {Default-First-Site-Name}
SPNSuffixes          : {}
UPNSuffixes          : {}
```

Le module Active Directory

- **Get-Adcomputer**
- Permet de lister les ordinateurs de votre AD
 - ◆ **Get-adcomputer –filter *** (affiche les ordinateurs du domaine, **filter** filtre sur le nom)

```
PS C:\Users\Administrateur> get-adcomputer -filter *
```

DistinguishedName	:	CN=WIN-TCK058CNJLJ,OU=Domain Controllers,DC=gtek,DC=local
DNSHostName	:	WIN-TCK058CNJLJ.gtek.local
Enabled	:	True
Name	:	WIN-TCK058CNJLJ
ObjectClass	:	computer
ObjectGUID	:	c364ea0d-b439-4aa6-b1fb-ed5bda4869c7
SamAccountName	:	WIN-TCK058CNJLJ\$
SID	:	S-1-5-21-3564125724-175561107-1429508558-1000
UserPrincipalName	:	
DistinguishedName	:	CN=SRU2K8R2DEP,CN=Computers,DC=gtek,DC=local

- ◆ Le prochain exemple permet de rechercher les ordinateurs présent dans l'uo domain controllers et affiche l'adresse IP des stations
- ◆ **Get-Adcomputer –filter * -searchbase "ou=Domain controllers,dc=gtek,dc=local" – properties ipv4Address**

```
PS C:\Users\Administrateur> Get-ADComputer -Filter * -SearchBase "ou=Domain controllers,dc=gtek,dc=local" -Properties ipv4Address
```

DistinguishedName	:	CN=WIN-TCK058CNJLJ,OU=Domain Controllers,DC=gtek,DC=local
DNSHostName	:	WIN-TCK058CNJLJ.gtek.local
Enabled	:	True
IPv4Address	:	10.44.0.198
Name	:	WIN-TCK058CNJLJ
ObjectClass	:	computer
ObjectGUID	:	c364ea0d-b439-4aa6-b1fb-ed5bda4869c7
SamAccountName	:	WIN-TCK058CNJLJ\$
SID	:	S-1-5-21-3564125724-175561107-1429508558-1000
UserPrincipalName	:	

Le module Active Directory

- Cet exemple permet de récupérer le nom et l'adresse ip
 - ◆ Get-adcomputer –filter * -searchbase "ou=Domain controllers,dc=gtek,dc=local" –properties IPV4address | FT name, IPV4Adress

```
PS C:\Users\Administrateur> Get-ADComputer -Filter * -searchbase "ou=Domain controllers,dc=gtek,dc=local" -properties ipv4address | FT name,IpV4address
name                                     IpV4address
---                                     -----
WIN-TCK058CNJLJ                           10.44.0.198
```

Le module Active Directory

■ Get-ADAccountAuthorizationGroup

- ◆ Affiche les groupes appartenant à un utilisateur spécifié
- ◆ L'exemple suivant affiche tous les groupe de la personne nommé administrateur
- ◆ Get-ADAccountAuthorizationGroup administrateur

```
PS C:\Users\Administrateur> get-ADAccountAuthorizationGroup administrateur

distinguishedName : CN=Utilisateurs du domaine,CN=Users,DC=gtek,DC=local
GroupCategory      : Security
GroupScope         : Global
name               : Utilisateurs du domaine
objectClass        : group
objectGUID         : 413ba2b6-a13a-4bd0-87be-6cceef491ea9
SamAccountName    : Utilisateurs du domaine
SID                : S-1-5-21-3564125724-175561107-1429508558-513

distinguishedName :
GroupCategory      : Security
GroupScope         : DomainLocal
name               : Tout le monde
objectClass        :
objectGUID         : 00000000-0000-0000-0000-000000000000
SamAccountName    : Tout le monde
SID                : S-1-1-0

distinguishedName : CN=Administrateurs,CN=BuiltIn,DC=gtek,DC=local
GroupCategory      : Security
GroupScope         : DomainLocal
name               : Administrateurs
objectClass        : group
objectGUID         : cf9c4c4e-bc3b-4bc1-83bb-eb48427ad1c6
SamAccountName    : Administrateurs
SID                : S-1-5-32-544
```

Le module Active Directory

- Cet exemple affiche le nom du groupe et l'étendue du groupe

```
PS C:\Users\Administrateur> get-ADAccountAuthorizationGroup administrateur | FT name, GroupScope -A
name                                     GroupScope
---                                     -----
Utilisateurs du domaine                  Global
Tout le monde                           DomainLocal
Administrateurs                         DomainLocal
Utilisateurs                            DomainLocal
Accès compatible pré-Windows 2000       DomainLocal
Utilisateurs authentifiés                DomainLocal
Cette organisation                      DomainLocal
Propriétaires créateurs de la stratégie de groupe Global
Admins du domaine                       Global
Administrateurs de l'entreprise          Universal
Administrateurs du schéma               Universal
Groupe de réPLICATION dont le mot de passe RODC est refusé DomainLocal
Niveau obligatoire élevé                 DomainLocal
```

Le module Active Directory

■ Get-AdGroup

- ◆ Permet d'obtenir des informations sur les groupes de l'ad
- ◆ Get-Adgroup -filter * affiche tous les groupes de l'ad

```
PS C:\Users\Administrateur> get-adgroup -filter * | FT name  
name  
----  
Administrateurs  
Utilisateurs  
Invités  
Opérateurs d'impression  
Opérateurs de sauvegarde  
Duplicateurs  
Utilisateurs du Bureau à distance  
Opérateurs de configuration réseau  
Utilisateurs de l'Analyseur de performances  
Utilisateurs du journal de performances  
Utilisateurs du modèle COM distribué  
IIS_IUSRS  
Opérateurs de chiffrement  
Lecteurs des journaux d'événements  
Accès DCOM service de certificats  
Ordinateurs du domaine  
Contrôleurs de domaine  
Administrateurs du schéma  
Administrateurs de l'entreprise
```

- ◆ Get-ADGroup administrateurs affichent des informations sur le groupe administrateurs

```
PS C:\Users\Administrateur> get-ADgroup administrateurs  
  
DistinguishedName : CN=Administrateurs,CN=BuiltIn,DC=gtek,DC=local  
GroupCategory    : Security  
GroupScope       : DomainLocal  
Name             : Administrateurs  
ObjectClass      : group  
ObjectGUID       : cf9c4c4e-bc3b-4bc1-83bb-eb48427ad1c6  
SamAccountName   : Administrateurs  
SID              : S-1-5-32-544
```

Le module Active Directory

- **Get-ADGroup Administrateurs –*properties member*** permet d'afficher les membres du groupes administrateurs

```
PS C:\Users\Administrateur> get-ADgroup administrateurs -properties member

DistinguishedName : CN=Administrateurs,CN=BuiltIn,DC=gtek,DC=local
GroupCategory     : Security
GroupScope        : DomainLocal
member            : {CN=Support_gg,OU=DG,DC=gtek,DC=local, CN=Admins du domaine,CN=Users,DC=gtek,DC=lo
                    ateur,CN=Users,DC=gtek,DC=local}
Name              : Administrateurs
ObjectClass       : group
ObjectGUID        : cf9c4c4e-bc3b-4bc1-83bb-eb48427ad1c6
SamAccountName   : Administrateurs
SID               : S-1-5-32-544
```

- ◆ Get-adGroup –filter {groupScope –eq "Global"} | FT name
- ◆ Affiche le nom de tous les groupe globaux

```
PS C:\Users\Administrateur> get-ADgroup -filter {groupscope -eq "Global"} | FT name

name
-----
Administrateurs du domaine
Contrôleurs de domaine
Admins du domaine
Utilisateurs du domaine
Invités du domaine
Propriétaires créateurs de la stratégie de groupe
Contrôleurs de domaine en lecture seule
DnsUpdateProxy
gg_production
gg_compta
gg_logistique
faible
strong
Support_gg
testliog
GG_supportTech
GGAAAAA
```

- ◆ Cet exemple recherche tous les groupe globaux de l'uo production (tiens comptes aussi des sous uo

```
PS C:\Users\Administrateur> get-ADgroup -filter {groupscope -eq "Global"} -searchbase "ou=production,dc=gtek,dc=local" | FT name
```

Le module Active Directory

■ Get-AdgroupMember

- ◆ Affiche des informations sur les membres d'un groupe
- ◆ Get-adGroupMember –identity administrateurs
 - Retourne les membres du groupe administrateurs

```
PS C:\Users\Administrateur> get-adgroupmember "Administrateurs"

distinguishedName : CN=Support_gg,OU=DG,DC=gtek,DC=local
name              : Support_gg
objectClass        : group
objectGUID         : d81242f6-916a-492b-b96b-171335f20b48
SamAccountName    : Support_gg
SID               : S-1-5-21-3564125724-175561107-1429508558-1332

distinguishedName : CN=Admins du domaine,CN=Users,DC=gtek,DC=local
name              : Admins du domaine
objectClass        : group
objectGUID         : 74c5f9b7-2fd1-4150-b52a-9db02f7c4d29
SamAccountName    : Admins du domaine
SID               : S-1-5-21-3564125724-175561107-1429508558-512

distinguishedName : CN=Administrateur,CN=Users,DC=gtek,DC=local
name              : Administrateur
objectClass        : user
objectGUID         : 163e60b6-82ba-48cd-9833-1f930bd74bb5
SamAccountName    : Administrateur
SID               : S-1-5-21-3564125724-175561107-1429508558-500
```

Affiche les membres du groupe administrateurs

```
PS C:\Users\Administrateur> get-adgroupmember "Administrateurs" -recursive

distinguishedName : CN=Administrateur,CN=Users,DC=gtek,DC=local
name              : Administrateur
objectClass        : user
objectGUID         : 163e60b6-82ba-48cd-9833-1f930bd74bb5
SamAccountName    : Administrateur
SID               : S-1-5-21-3564125724-175561107-1429508558-500

distinguishedName : CN=sup_dep,OU=pc,DC=gtek,DC=local
name              : sup_dep
objectClass        : user
objectGUID         : c8017215-b85f-4f6e-92cf-d9a87d6906ad
SamAccountName    : supdep
SID               : S-1-5-21-3564125724-175561107-1429508558-2326

distinguishedName : CN=admin,CN=Users,DC=gtek,DC=local
name              : admin
objectClass        : user
objectGUID         : d8ecea34-e0fe-45f2-ad3b-546a648b5684
SamAccountName    : admin
SID               : S-1-5-21-3564125724-175561107-1429508558-1114
```

L'option –recursive permet de scruter tous les membres utilisateurs des groupes enfants, nous verrons donc tous les users utilisant ce groupe

Le module Active Directory

■ Get-ADObject

- ◆ Obtient des informations d'un objet de l'ad
- ◆ Get-adobject –filter * affiche tous les objets de votre AD

DistinguishedName	Name	ObjectClass	ObjectGUID
DC=gtek,DC=local	gtek	domainDNS	8f9fa8d3-5d53-400d.
CN=Users,DC=gtek,DC=...	Users	container	81609378-5876-417d.
CN=Computers,DC=gte...	Computers	container	6d7abda4-5ac2-4453.
OU=Domain Controle...	Domain Controllers	organizationalUnit	86b7b622-b531-49d4.
CN=System,DC=gtek,D...	System	container	adf7b91f-40e6-4a6d.
CN=LostAndFound,DC=...	LostAndFound	lostAndFound	349b47b3-a49c-47c6.
CN=Infrastructure,D...	Infrastructure	infrastructureUpdate	f0fac75-bfb0-4fb8.
CN=ForeignSecurityP...	ForeignSecurityPrin...	container	988b1274-b486-4b0a.
CN=Program Data,DC=...	Program Data	container	78408c1d-ea5b-4c17.
CN=Microsoft,CN=Pro...	Microsoft	container	a1af4e9a-7654-4238.
CN=NTDS Quotas,DC=g...	NTDS Quotas	msDS-QuotaContainer	8042a37d-c765-406b.
CN=Managed Service ...	Managed Service Acc...	container	0b4f4606-ad79-43b5.
CN=WinsockServices,...	WinsockServices	container	54d54461-8014-4645.
CN=RpcServices,CN=S...	RpcServices	rpcContainer	699c11f5-e641-4b19.
CN=FileLinks,CN=Sus...	FileLinks	fileLinkTracking	8022eac4-eh2h-428h.

- ◆ Get-adobject –filter {objectClass –eq "group"} pour afficher les objets de type group

DistinguishedName	Name	ObjectClass	ObjectGUID
CN=Administrateurs,...	Administrateurs	group	cf9c4c4e-bc
CN=Utilisateurs,CN=...	Utilisateurs	group	6bd02de9-37
CN=Invités,CN=Built...	Invités	group	bd898bb0-5f
CN=Opérateurs d' imp...	Opérateurs d' impres...	group	08a1f2e9-66
CN=Opérateurs de sa...	Opérateurs de sauve...	group	5e3a5be7-97
CN=Duplicateurs, CN=	Duplicateurs	group	819b10cf-36

Le module Active Directory

- Cet exemple recherche tous les groupes de l'uo et sous ou production
 - ◆ Get-AdObject –filter {objectClass –eq "group"} –searchbase "ou=production,dc=gtek,dc=local"

```
PS C:\Users\Administrateur> get-adobject -filter {objectClass -eq "group"} -searchbase "ou=production,dc=gtek,dc=local"

DistinguishedName      Name          ObjectClass      ObjectGUID
-----      ----          -----      -----
CN=gg_production,OU... gg_production    group      305edc99-f503-49dd...
CN=gdl_production,OU... gdl_production    group      d7d96418-dd3c-437d...
CN=GGNantes,OU=Nant... GGNantes        group      7df529cd-024d-4391...
```

Le module Active Directory

- Cet exemple que nous aurions pu utiliser avec les autres commandes get permet d'afficher les propriétés et méthodes que l'on peut utiliser
 - ◆ `Get-adObject -filter {ObjectClass -eq "user"} -properties * | get-member`

```
PS C:\Users\Administrateur> get-adobject -filter {ObjectClass -eq "User"} -properties * | get-member

TypeName: Microsoft.ActiveDirectory.Management.ADObject

Name          MemberType      Definition
---          ---           ---
Contains      Method         bool Contains(string propertyName)
Equals        Method         bool Equals(System.Object obj)
GetEnumerator Method        System.Collections.IDictionaryEn...
GetHashCode   Method        int GetHashCode()
GetType       Method        type GetType()
ToString      Method        string ToString()
Item          ParameterizedProperty Microsoft.ActiveDirectory.Manage...
accountExpires Property     System.Int64 accountExpires {get;...}
adminCount    Property     System.Int32 adminCount {get;set;}
badPasswordTime Property    System.Int64 badPasswordTime {ge...
badPwdCount  Property    System.Int32 badPwdCount {get;set;}
CanonicalName Property    System.String CanonicalName {get;...}
CN           Property    System.String CN {get;...}
codePage      Property    System.Int32 codePage {get;set;}
countryCode  Property    System.Int32 countryCode {get;set;}
Created      Property    System.DateTime Created {get;...}
createTimeStamp Property  System.DateTime createTimeStamp ...
Deleted      Property    System.Boolean Deleted {get;...}
Description  Property    System.String Description {get;...}
DisplayName  Property    System.String DisplayName {get;...}
DistinguishedName Property  System.String DistinguishedName ...
dSCorePropagationData Property Microsoft.ActiveDirectory.Manage...
instanceType  Property    System.Int32 instanceType {get;...}
isCriticalSystemObject Property System.Boolean isCriticalSystemO...
isDeleted    Property    System.Boolean isDeleted {get;...}
LastKnownParent Property  System.String LastKnownParent {g...
lastLogoff    Property    System.Int64 lastLogoff {get;set;}
lastLogon    Property    System.Int64 lastLogon {get;set;}
lastLogonTimestamp Property System.Int64 lastLogonTimestamp ...
lockoutTime  Property    System.Int64 lockoutTime {get;set;}
logonCount   Property    System.Int32 logonCount {get;set;}
logonHours   Property    System.Byte[] logonHours {get;set;}
memberOf     Property    Microsoft.ActiveDirectory.Manage...
Modified     Property    System.DateTime Modified {get;...}
modifyTimeStamp Property  System.DateTime modifyTimeStamp ...
Name          Property    System.String Name {get;...}
nTSecurityDescriptor Property System.DirectoryServices.ActiveD...
ObjectCategory Property  System.String ObjectCategory {get;...}
ObjectClass   Property    System.String ObjectClass {get;...}
ObjectGUID    Property    System.Nullable`1[[System.Guid, ...]
objectSid    Property    System.Security.Principal.Securi...
primaryGroupId Property  System.Int32 primaryGroupId {get;...}
ProtectedFromAccidentalDeletion Property System.Boolean ProtectedFromAcci...
pwdLastSet   Property    System.Int64 pwdLastSet {get;set;}
```

Le module Active Directory

- ◆ Get-adobject –filter {objectclass –eq "user"} –properties pwdlastset, ObjectSid, Lastlogon ... affichent des propriétés comme la dernière connexion, le SID....

```
PS C:\Users\Administrateur> get-adobject -filter {ObjectClass -eq "User"} -properties pwdlastset, ObjectSid, WhenChanged, lastlogon, isdeleted

DistinguishedName : CN=Administrateur,CN=Users,DC=gtek,DC=local
lastlogon          : 129392934242963343
Name               : Administrateur
ObjectClass        : user
ObjectGUID         : 163e60b6-82ba-48cd-9833-1f930bd74bb5
ObjectSid          : S-1-5-21-3564125724-175561107-1429508558-500
pwdlastset         : 129326487700923520
WhenChanged        : 10/01/2011 15:21:27

DistinguishedName : CN=Invité,CN=Users,DC=gtek,DC=local
lastlogon          : 0
Name               : Invité
ObjectClass        : user
ObjectGUID         : 1456780f-02c7-4452-a390-d67558a43379
ObjectSid          : S-1-5-21-3564125724-175561107-1429508558-501
pwdlastset         : 0
WhenChanged        : 02/09/2010 11:34:35

DistinguishedName : CN=WIN-TCK058CNJLJ,OU=Domain Controllers,DC=gtek,DC=local
lastlogon          : 129392967872934837
Name               : WIN-TCK058CNJLJ
ObjectClass        : computer
ObjectGUID         : c364ea0d-b439-4aa6-b1fb-ed5bda4869c7
ObjectSid          : S-1-5-21-3564125724-175561107-1429508558-1000
pwdlastset         : 129380018433248903
WhenChanged        : 28/12/2010 10:24:03

DistinguishedName : CN=krbtgt,CN=Users,DC=gtek,DC=local
lastlogon          : 0
Name               : krbtgt
ObjectClass        : user
ObjectGUID         : 838c8bcd-1b1b-4537-8469-ea22e4c3b3b7
ObjectSid          : S-1-5-21-3564125724-175561107-1429508558-502
pwdlastset         : 129278937649843750
WhenChanged        : 02/09/2010 11:51:14
```

Le module Active Directory

- ◆ Get-Adobject –filter {ObjectClass –eq "Group" –and Deleted –eq \$true} –includeDeletedObjects
- ◆ Cet exemple filtre les objets groupes effacés, ne pas oublier d'ajouter –IncludeDeletedObjects qui permet d'afficher les objets effacés de l'AD.

```
PS C:\Users\Administrateur> get-adobject -filter {ObjectClass -eq "Group" -and Deleted -eq $true} -includeDeletedObjects

Deleted          : True
DistinguishedName : CN=mes pc\0ADEL:efba3346-4d44-4a8f-acd6-fa878ad7f5bc,CN=Deleted Objects,DC=gtek,DC=local
Name             : mes pc
ObjectClass      : group
ObjectGUID       : efba3346-4d44-4a8f-acd6-fa878ad7f5bc
```

Le module Active Directory

■ Get-AddefaultDomainPasswordpolicy

- ◆ Permet de voir les stratégies de mot de passe
- ◆ *Get-AddefaultDomainPasswordPolicy*

```
PS C:\Users\Administrateur> Get-addefaultDomainPasswordPolicy

ComplexityEnabled      : False
DistinguishedName     : DC=gttek,DC=local
LockoutDuration        : 00:00:00
LockoutObservationWindow : 00:30:00
LockoutThreshold       : 5
MaxPasswordAge         : 00:00:00
MinPasswordAge         : 1.00:00:00
MinPasswordLength      : 3
objectClass             : <domainDNS>
objectGuid              : 8f9fa8d3-5d53-400d-94bf-bdff12818708
PasswordHistoryCount   : 24
ReversibleEncryptionEnabled : False
```

- ◆ *Get-addefaultDomainPasswordPolicy -current loggedOnUser*

- Affiche la stratégie de mot de passe pour l'utilisateur connecté

```
PS C:\Users\Administrateur> Get-addefaultDomainPasswordPolicy -current loggedOnUser

ComplexityEnabled      : False
DistinguishedName     : DC=gttek,DC=local
LockoutDuration        : 00:00:00
LockoutObservationWindow : 00:30:00
LockoutThreshold       : 5
MaxPasswordAge         : 00:00:00
MinPasswordAge         : 1.00:00:00
MinPasswordLength      : 3
objectClass             : <domainDNS>
objectGuid              : 8f9fa8d3-5d53-400d-94bf-bdff12818708
PasswordHistoryCount   : 24
```

- Même chose pour un domaine

```
PS C:\Users\Administrateur> Get-addefaultDomainPasswordPolicy -identity gtek.local
```

Le module Active Directory – créer utilisateurs

- exemple de script de création d'utilisateurs

Fichier CSV
d'importation
des utilisateurs



The screenshot shows a Notepad window with the following content:

Nom	prénom	tel	uo	mdp	service	mail
delhomme	Pierre	3456	compta	P@ssw0rd	compta	pddel@gtek.fr
Duchnac	christophe	5678	rh	Pa\$\$w0rd	rh	cduc@gtek.fr

```
Import-Module activedirectory
$utils = import-csv c:\temp\users.csv -delimiter ";"
$i=1
foreach($ut in $utils)
{
    Write-Host "util $($ut.nom)"
    Write-progress -activity "Création des users dans l'uo $($ut.uo)" -status "%effectué" -percentcomplet ($i/$utils.length*100)

    New-ADUser
        -Name "$($ut.nom) $($ut.prenom)"
        -DisplayName "$($ut.nom) $($ut.prenom)"
        -SamAccountName $($ut.prenom.substring(0,1).tolower() + $($ut.nom))
        -UserPrincipalName $($ut.prenom.substring(0,1).tolower() + $($ut.nom) + "@gtek-formation.local")
        -GivenName $($ut.prenom)
        -surname $($ut.nom)
        -officephone $($ut.tel)
        -EmailAddress $($ut.mail)
        -path "ou=$($ut.uo),dc=gtek-formation,dc=local"
        -AccountPassword (convertto-securestring $($ut.mdp) -AsPlainText -Force)
        -Enabled $true
        -OtherAttributes @{company="BMI";title=$($ut.service);department="DSI"}
        #-Server 192.168.0.199
        #-credential $auth

    write-host "utilisateur" $($ut.prenom) $($ut.nom) "est créé dans l'uo" $($ut.uo) -fore blue -back yellow
    $i++
}
```

Le module Active Directory– créer groupe

- L'exemple suivant permet de créer un dossier puis de créer les groupes associés à ce dossier. Cas fréquemment utilisé en entreprise.
 - ◆ Ex: création d'un dossier nommé **compta** implique la création de 4 groupes globaux + 4 groupes Locaux formaté de la manière suivante:
 - GG_compta_L
 - GG_compta_LE
 - -GG_compta_M
 - -GG_compta_CT
 - -DL_compta_L
 - -DL_compta_LE
 - -DL_compta_M
 - -DL_compta_CT
 - ◆ On applique ensuite la règle préconisée par Microsoft, AGDLP. Cad on place les groupes globaux dans des Groupes locaux.
 - ◆ On affecte enfin les permission NTFS avec les groupe du domaine local

Utiliser Powershell – créer groupe

```
1 Import-module ActiveDirectory
2 $dossier=read-host "saisir le nom d'un dossier? (ex: compta)"
3 $lettre=read-host "Saisir la lettre de lecteur (ex: C:\)"
4 new-item "$lettre\$dossier" -type directory
5 #création du partage, tout le monde en CT
6 net share $dossier=$lettre\$dossier "/grant:Tout le monde,FULL"
7 #le premier tableau servira pour créer les permissions NTFS
8 #le second pour nommer les group
9 $stabP=("RX","WRX","M","F")
10 $stabN=("_L","_LE","_M","_CT")
11 #la boucle for permet de créer 4 groupes globale et 4 groupe DL
12 #elle permet aussi de définir les permissions NTFS
13 for($i=0;$i-1e3;$i++)
14 {
15     New-ADGroup -Path "ou=gest groupes,dc=gtek-formation,dc=local"
16             -name ("GG_" + $dossier + $stabN[$i])
17             -GroupScope Global
18     New-adgroup -Path "ou=gest groupes,dc=gtek-formation,dc=local"
19             -name ("DL_" + $dossier + $stabN[$i])
20             -GroupScope DomainLocal
21     Add-ADGroupMember ("DL_" + $dossier + $stabN[$i]) ("GG_" + $dossier + $stabN[$i])
22     $droit="DL_"+$dossier+$stabN[$i]+":(OI)(CI)$(stabP[$i])"
23     icacls $lettre\$dossier /grant $droit
24 }
25
26 #suppression des groupes utilisateurs et utilisateurs authentifiés
27 #pour supprimer il faut désactiver l'héritage
28 icacls $lettre\$dossier /inheritance:d
29 icacls $lettre\$dossier /remove "AUTORITE NT\Utilisateurs authentifiés"
30 icacls $lettre\$dossier /remove "Utilisateurs"
31
```

Le module Active Directory – créer ordinateur

- New-AdComputer
 - ◆ Permet d'ajouter des ordinateurs dans le domaine
 - ◆ Cet exemple ajoute un ordinateur nommé PC-DE-G dans l'uo production du domaine gtek.local

```
PS C:\Users\Administrateur> new-adcomputer -name "PC-DE-G" -SamAccountName "PC-DE-G" -path "ou=production,dc=gtek,dc=local" -location "Redon" -enabled $true
```

- ◆ Get-help new-adcomputer –detailed pour voir l'aide complète

Le module Active Directory –supprimer utilisateurs

- Remove-ADUser

- ◆ Vous pouvez supprimer des utilisateurs avec la commande Remove-ADUser
- ◆ Get-AdUser –searchbase "ou=compta,dc=gtek,dc=local" –filter * | remove-ADUser –confirm:\$false
- ◆ Va chercher tous les utilisateurs dans l'uo compta (y compris sous ou) et les efface (-confirm;\$false) permet de ne pas avoir à confirmer la suppression
- ◆ Si l'on souhaite uniquement effacer les users de l'uo compta sans les sous ou, ajouter –searchscope (0 pour ou mères, 1 pour ou mères et ou enfant, 2 pour tous les niveaux)

```
PS C:\Users\Administrateur> get-aduser -searchbase "ou=compta,dc=gtek,dc=local" -filter * | remove-ADUser -confirm:$false
```

Gestion des modules PowerShell V3

Le module DHCP

Le service DHCP

- Depuis l'arrivée de Windows 2012, les modules pour gérer les serveurs ont été largement enrichi
 - ◆ Le module dhcpserver permet de configurer et obtenir des informations détaillées sur vos serveur DHCP
 - ◆ **Import-module dhcpserver** sur posershell ou powershell Ise.
 - ◆ Nous allons utiliser qq commandelettes "get-dhcp xxx"

```
PS C:\Users\Administrateur> get-command -module dhcpserver
 CommandType      Name
-----      -----
 Function        Add-DhcpServerInDC
 Function        Add-DhcpServerv4Class
 Function        Add-DhcpServerv4ExclusionRange
 Function        Add-DhcpServerv4Failover
 Function        Add-DhcpServerv4FailoverScope
 Function        Add-DhcpServerv4Filter
 Function        Add-DhcpServerv4Lease
 Module          dhcpserver
 Module          dhcpserver
 Module          dhcpserver
 Module          dhcpserver
 Module          dhcpserver
 Module          dhcpserver
 Module          dhcpserver
```

Get-command -module dhcpserver permet d'obtenir la liste des cmdlettes pour DHCP

Le service DHCP

- Les commandes suivantes obtiennent des informations de serveurs dhcp

```
import-module dhcpserver

write-host "indication de la base de données --> cmdlette Get-DhcpServerDatabase" -fore green
Get-DhcpServerDatabase -ComputerName srv2012RC

write-host "indication sur le cd et l'adresse ip du cd autorisant ce dhcp --> cmdlette Get-DhcpServerInDC" -fore green
Get-DhcpServerInDC

write-host "config et autorisation dans le domaine cmdlette Get-DhcpServerSetting" -fore green
Get-DhcpServerSetting

write-host "interface délivrant les adresses ip --> cmdlette Get-DhcpServerv4Binding" -fore green
Get-DhcpServerv4Binding

write-host "affichage des différentes classes dhcp --> cmdlette Get-DhcpServerv4Class" -fore green
Get-DhcpServerv4Class

write-host "config avec DNS mise à jour client(propriétés Dhcp ipv4 onglet dns) -->cmdlette Get-DhcpServerv4DnsSetting" -fore green
Get-DhcpServerv4DnsSetting -ComputerName srv2012RC

write-host "exclusion des adresses des plages DHCP cmdlette Get-DhcpServerv4ExclusionRange" -fore green
Get-DhcpServerv4ExclusionRange

write-host "adresses macs faisant partis des filtres autorisés ou refusés (deny ou allow) --> Get-DhcpServerv4Filter"
Get-DhcpServerv4Filter
```

```
config avec DNS mise à jour client(propriétés Dhcp ipv4 onglet dns) -->cmdlette Get-DhcpServerv4DnsSetting

DeleteDnsRROnLeaseExpiry : True
DynamicUpdates : OnClientRequest
NameProtection : False
UpdateDnsRRForOlderClients : False
PSComputerName :

exclusion des adresses des plages DHCP cmdlette Get-DhcpServerv4ExclusionRange

EndRange      : 10.0.0.221
ScopeId       : 10.0.0.0
StartRange    : 10.0.0.200
PSComputerName :

adresses macs faisant partis des filtres autorisés ou refusés (deny ou allow) --> Get-DhcpServerv4Filter

Description   : pcaut
List          : Allow
MacAddress   : 02-03-05-08-07-05
PSComputerName :

Description   : interdit pcgmm
List          : Deny
MacAddress   : 00-01-02-03-04-05
PSComputerName :
```

```
indication de la base de données --> cmdlette Get-DhcpServerDatabase

fileName      : C:\Windows\system32\dhcp\dhcp.mdb
backupPath    : C:\Windows\system32\dhcp\backup
backupInterval(m) : 60
cleanupInterval(m) : 60
loggingEnabled : True
restoreFromBackup : False

indication sur le cd et l'adresse ip du cd autorisant ce dhcp --> cmdlette Get-DhcpServerInDC

dnsName      : srv2012rc.gtek-formation.lan
ipAddress    : 192.168.18.129
pscomputerName :

config et autorisation dans le domaine cmdlette Get-DhcpServerSetting

activatePolicies : True
conflictDetectionAttempts : 3
dynamicBootp   : True
isAuthorized   : True
isDomainJoined : True
napEnabled     : False
npsUnreachableAction : Full
restoreStatus  : False
pscomputerName :
```

Le service DHCP

```

write-host "indication d'activation ou non sur les filtres macs --> cmdlette Get-DhcpServerv4FilterList" -fore green
Get-DhcpServerv4FilterList

write-host "indique la prochaine adresse ip de dispo et attribuable par dhcp -->cmdlette Get-DhcpServerv4FreeIPAddress" -fore green
Get-DhcpServerv4FreeIPAddress -ScopeId "192.168.18.0" -StartAddress "192.168.18.112"

write-host "option dhcp pour une étendue -->cmdlette Get-DhcpServerv4OptionValue" -fore green
Get-DhcpServerv4OptionValue -ScopeId "192.168.18.0"

write-host "indique les adresses réservé pour une étendue -->cmdlette Get-DhcpServerv4Reservation" -fore green
Get-DhcpServerv4Reservation -scopeId "192.168.18.0" -ComputerName srv2012RC

write-host "affiche les différentes étendues sur le dhcp courant -->cmdlette Get-DhcpServerv4Scope" -fore green
Get-DhcpServerv4Scope

write-host "affiche des stats sur les étendues (nb adresses utilisés, nbre d'adresse libres ...) -->cmdlette Get-DhcpServerv4ScopeStatistics" -fore green
Get-DhcpServerv4ScopeStatistics

write-host "affiche des stats globales (nb d'étendues, total adresses % utilisés ...) -->cmdlette Get-DhcpServerv4Statistics" -fore green
Get-DhcpServerv4Statistics

```

```

write-host "option dhcp pour une étendue -->cmdlette Get-DhcpServerv4OptionValue" -fore green
Get-DhcpServerv4OptionValue -ScopeId "192.168.18.0"
indication d'activation ou non sur les filtres macs --> cmdlette Get-DhcpServerv4FilterList

Allow          Deny
----          ---
True           True
indique la prochaine adresse ip de dispo et attribuable par dhcp -->cmdlette Get-DhcpServerv4FreeIPAddress
192.168.18.115
option dhcp pour une étendue -->cmdlette Get-DhcpServerv4OptionValue

Name          : Bail
OptionId      : 51
PolicyName    :
Type          : DWord
UserClass     : 
Value         : {1800}
VendorClass   :
PSComputerName :

indique les adresses réservé pour une étendue -->cmdlette Get-DhcpServerv4Reservation
IP Address    ScopeId    ClientId    Name        Type        Description
-----        -----      -----      ----       ----       -----
192.168.18.112 192.168.18.0 00-04-52-1f-2a-43  imp2      Both
192.168.18.114 192.168.18.0 00-98-2d-4c-5f-45  imp3      Both
192.168.18.113 192.168.18.0 01-02-05-06-08  imp4      Both

affiche les différentes étendues sur le dhcp courant -->cmdlette Get-DhcpServerv4Scope
Name          : Routeur
OptionId      : 3
PolicyName    :
Type          : IPv4Address
UserClass     : 
Value         : {192.168.18.254}
ActivatePolicies : True
Delay          : 0
Description    :
EndRange       : 10.0.1.250
LeaseDuration  : 8.00:00:00
MaxBootpClients : 4294967295
Name          : et2
NapEnable     : False
NapProfile    :
ScopeId       : 10.0.0.0
StartRange    : 10.0.0.100
State         : Active

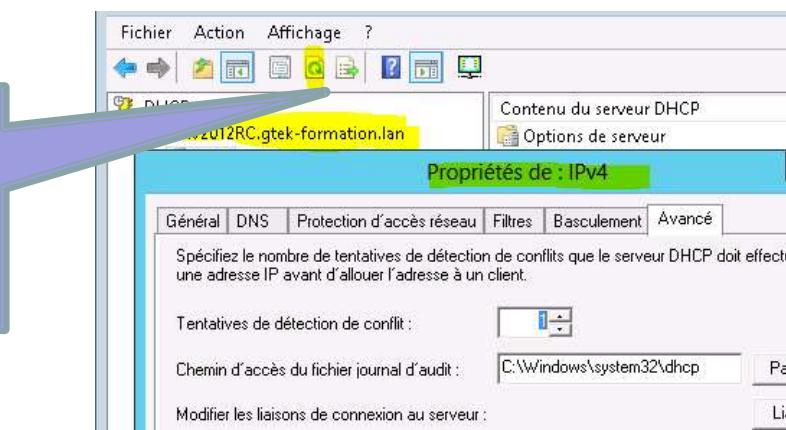
```

Le service DHCP

- La commande set-dhcp permet de paramétrer un serveur DHCP
 - ◆ La première commande paramètre le nbre de ping avant de délivrer un dhcp
 - ◆ Set-DhcpServerSetting -ConflictDetectionAttempts 1

```
Write-host "détection de conflits d'adresse ip, dhcp effectuera 1 ping avant de délivrer une adresse ip"  
Set-DhcpServerSetting -ConflictDetectionAttempts 1
```

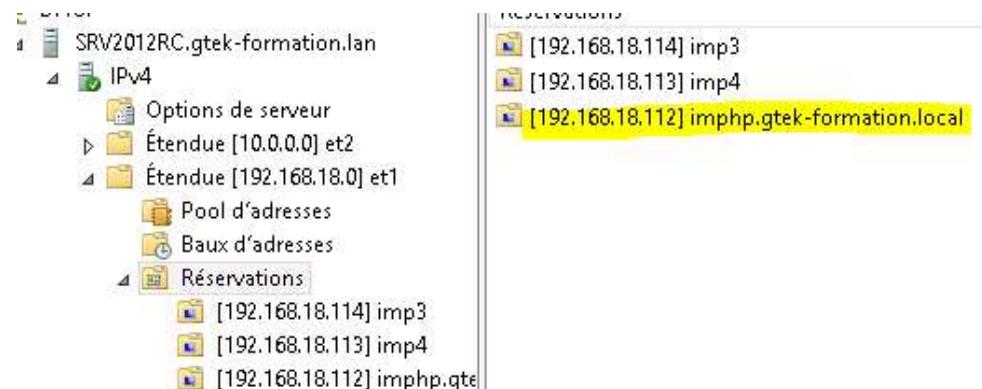
Réactualisez sur le nom du serveur puis ensuite IPV4/propriété/onglet avancé



Le service DHCP

- L'exemple suivant mets à jour une réservation d'adresse ip
 - ◆ Set-DhcpServerv4Reservation -IPAddress "192.168.18.112" -name "imphp.gtek-formation.local" -ComputerName "srv2012RC"

```
Write-host "Mise à jour du nom de l'adresse réservé 192.168.18.112 sur le serveur srv2012RC" -fore green  
Set-DhcpServerv4Reservation -IPAddress "192.168.18.112" -name "imphp.gtek-formation.local" -ComputerName "srv2012RC"
```



Le service DHCP

- L'exemple suivant va permettre de créer des réservation en masse

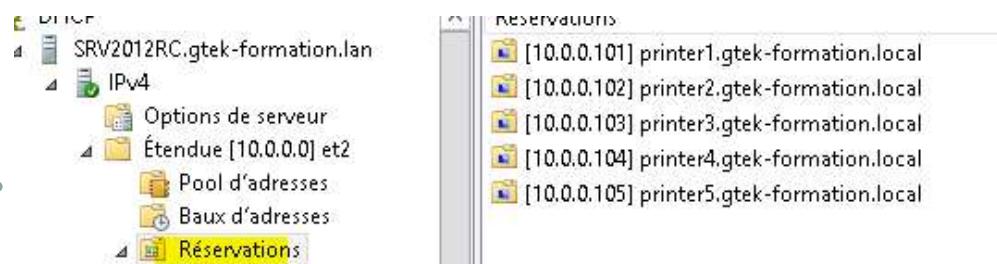
Créer ce fichier csv avec les en-tête de colonne

nom	mac	ip
printer1.gtek-formation.local	1D-77-08-D9-04-09	10.0.0.101
printer2.gtek-formation.local	1D-77-08-D9-04-10	10.0.0.102
printer3.gtek-formation.local	1D-77-08-D9-04-11	10.0.0.103
printer4.gtek-formation.local	1D-77-08-D9-04-12	10.0.0.104
printer5.gtek-formation.local	1D-77-08-D9-04-13	10.0.0.105

- Script:

```
$resa=import-csv C:\script\imp.csv -Delimiter ";"  
  
foreach($imp in $resa)  
{  
    Add-DhcpServerv4Reservation -ScopeId "10.0.0.0" -IPAddress $imp.ip -ClientId $imp.mac -name $imp.nom  
}
```

Après l'exécution du script, les réservation sont enregistré dans notre dhcp



Gestion des modules PowerShell V3

Le module DNS

Le service DNS

- Sur 2012, vous pouvez gérer vos serveurs DNS avec Powershell, plusieurs dizaine de cmdlet sont dédiées au DNS

<Users\Administrateur> get-command -module dnsserver	
Type	Name
	Export-DnsServerTrustAnchor
	Add-DnsServerConditionalForwarderZone
	Add-DnsServerDirectoryPartition
	Add-DnsServerForwarder
	Add-DnsServerPrimaryZone
	Add-DnsServerResourceRecord
	Add-DnsServerResourceRecordA
	Add-DnsServerResourceRecordAAAA
	Add-DnsServerResourceRecordCName
	Add-DnsServerResourceRecordDnsKey
	Add-DnsServerResourceRecordDS
	Add-DnsServerResourceRecordMX
	Add-DnsServerResourceRecordPtr
	Add-DnsServerRootHint
	Add-DnsServerSecondaryZone
	Add-DnsServerSigningKey

get-command -module dnsserver permet de voir toutes les cmdlettes utilisables sur un DNS

PS C:\>Users\Administrateur> Show-DnsServerCache				
HostName	RecordType	Timestamp	TimeToLive	RecordType
	NS	0	00:00:00	m.root-servers.net.
	NS	0	00:00:00	l.root-servers.net.
	NS	0	00:00:00	k.root-servers.net.
	NS	0	00:00:00	j.root-servers.net.
	NS	0	00:00:00	i.root-servers.net.
	NS	0	00:00:00	h.root-servers.net.
	NS	0	00:00:00	g.root-servers.net.
	NS	0	00:00:00	f.root-servers.net.
	NS	0	00:00:00	e.root-servers.net.
	NS	0	00:00:00	d.root-servers.net.
	NS	0	00:00:00	c.root-servers.net.
	NS	0	00:00:00	b.root-servers.net.
	NS	0	00:00:00	a.root-servers.net.
le17a.com	NS	0	14:18:16	ns2.dnsmadeeasy.com.
le17a.com	NS	0	14:18:16	ns3.dnsmadeeasy.com.
le17a.com	NS	0	14:18:16	ns0.dnsmadeeasy.com.
le17a.com	NS	0	14:18:16	ns4.dnsmadeeasy.com.
le17a.com	NS	0	14:18:16	ns1.dnsmadeeasy.com.
c10r.facebook.com	NS	0	15:05:35	a.ns.c10r.facebook.co

Show-dnsservercache permet de voir le cache DNS

Show-DnsServerCache - ComputerName srv2012-2dc pour analyser un DND distant

Le service DNS

- L'exemple suivant retourne les enregistrements présents dans la zone test.fr
 - Get-DnsServerResourceRecord -ZoneName test.fr

PS C:\Users\Administrateur> Get-DnsServerResourceRecord -ZoneName test.fr				
HostName	RecordType	Timestamp	TimeToLive	RecordData
srv2012-2dc	NS	0	01:00:00	srv2012-2dc.gtek.local.
srv2012dc	NS	0	01:00:00	srv2012dc.gtek.local.
hostmaster	SOA	0	01:00:00	[3][srv2012dc.gtek.local.][hostmaster.gtek.local.]
pctest	A	0	01:00:00	192.168.0.200

- L'exemple suivant retourne les enregistrements de type CNAME ou A présents dans la zone gtek.local
 - Get-DnsServerResourceRecord -ZoneName gtek.local | ?{\$_._recordType -eq "CNAME" -or \$_._recordType -eq "A"}

PS C:\Users\Administrateur> Get-DnsServerResourceRecord -ZoneName gtek.local ?{\$_._recordType -eq "CNAME" -or \$_._recordType -eq "A"}				
HostName	RecordType	Timestamp	TimeToLive	RecordData
srv2012-2dc	A	29/10/2012 13:00:00	00:10:00	192.168.0.200
srv2012dc	A	05/11/2012 09:00:00	00:10:00	192.168.0.199
DomainDnsZones	A	05/11/2012 09:00:00	00:10:00	192.168.0.199
DomainDnsZones	A	29/10/2012 13:00:00	00:10:00	192.168.0.200
ForestDnsZones	A	05/11/2012 09:00:00	00:10:00	192.168.0.199
ForestDnsZones	A	29/10/2012 13:00:00	00:10:00	192.168.0.200
mailhost	CNAME	0	01:00:00	srvombre2012.gtek.local.

- L'exemple suivant retourne les enregistrements de type A présents dans la zone gtek.local sur le serveur srv2012-2dc
 - Get-DnsServerResourceRecord -ZoneName gtek.local -ComputerName srv2012-2dc -RRType A

PS C:\Users\Administrateur> Get-DnsServerResourceRecord -ZoneName gtek.local -ComputerName srv2012-2dc -RRType A				
HostName	RecordType	Timestamp	TimeToLive	RecordData
srv2012-2dc	A	05/11/2012 09:00:00	00:10:00	192.168.0.199
srv2012dc	A	29/10/2012 13:00:00	00:10:00	192.168.0.200
DomainDnsZones	A	05/11/2012 09:00:00	00:10:00	192.168.0.199
DomainDnsZones	A	09/11/2012 09:00:00	00:10:00	192.168.0.200
ForestDnsZones	A	05/11/2012 09:00:00	00:10:00	192.168.0.199
ForestDnsZones	A	09/11/2012 08:00:00	00:10:00	192.168.0.200
PC-W81	A	06/11/2012 09:00:00	00:20:00	192.168.0.100
srv2012-2dc	A	0	01:00:00	192.168.0.200

Le service DNS

- Obtenir les serveurs racine paramétrer le DNS srv2012-2dc
 - ◆ Get-DnsServerRootHint -ComputerName srv2012-2dc | ft -a

```
PS C:\Users\Administrateur> Get-DnsServerRootHint -ComputerName srv2012-2dc | ft -a
NameServer          IPAddress
-----
m.root-servers.net. 202.12.27.33
l.root-servers.net. 199.7.83.42
k.root-servers.net. 193.0.14.129
j.root-servers.net. 192.58.128.30
i.root-servers.net. 192.36.148.17
h.root-servers.net. 128.63.2.53
g.root-servers.net. 192.112.36.4
f.root-servers.net. 192.5.5.241
e.root-servers.net. 192.203.230.10
d.root-servers.net. 2001:500:2d::d
c.root-servers.net. 192.33.4.12
b.root-servers.net. 192.228.79.201
a.root-servers.net. 2001:503:ba3e::2:30
```

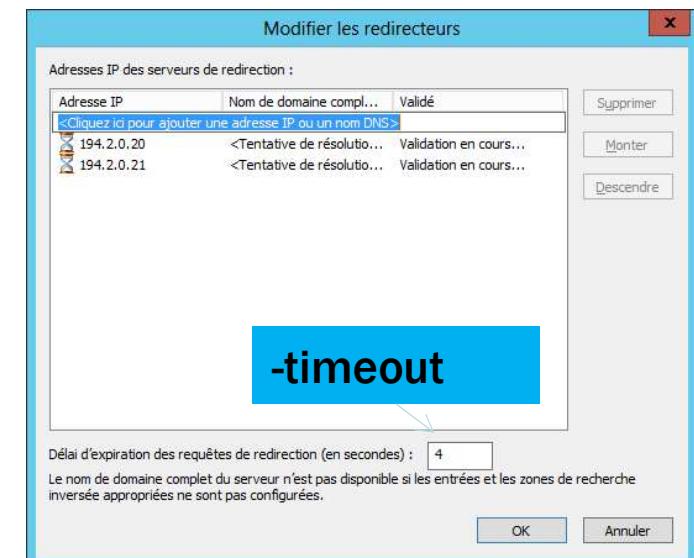
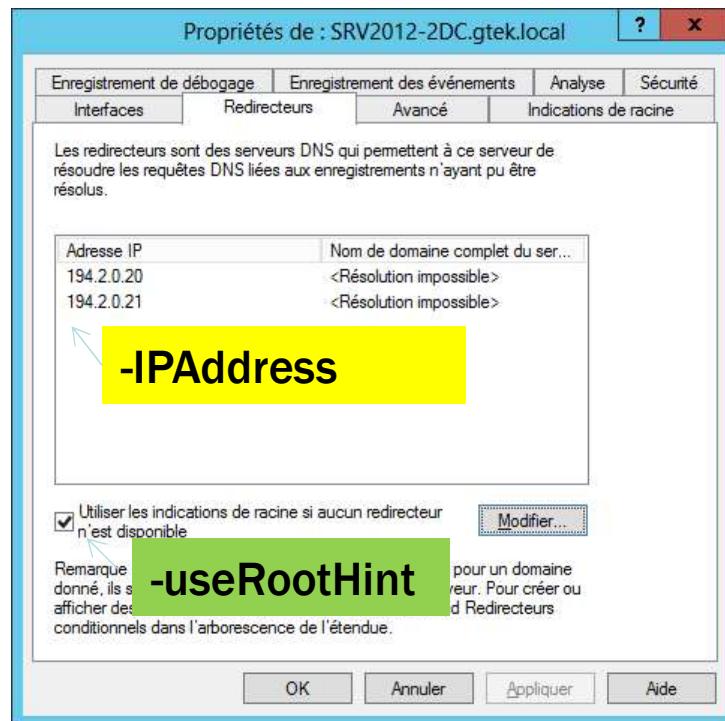
- ◆ L'exemple suivant retourne les redirecteurs sur le DNS courant

```
PS C:\Users\Administrateur> Get-DnsServerForwarder
UseRootHint      : True
Timeout(s)       : 3
EnableReordering : True
IPAddress        : {89.2.0.1, 89.2.0.2}
ReorderedIPAddress : {89.2.0.1, 89.2.0.2}
```

Le service DNS

- La commande suivante permet de paramétrer 2 redirecteurs sur le serveur srv2012-2dc
 - Set-DnsServerForwarder -IPAddress 194.2.0.20, 194.2.0.21 -ComputerName srv2012-2dc -UseRootHint \$true -Timeout 4

```
PS C:\Users\Administrateur> Set-DnsServerForwarder -IPAddress 194.2.0.20, 194.2.0.21 -ComputerName srv2012-2dc -UseRootHint $true -Timeout 4  
PS C:\Users\Administrateur>
```



Le service DNS

- L'exemple suivant permet de créer un enregistrement de type A nommé gaelpc dans la zone test.fr,
 - ◆ -creatptr permet de faire un enregistrement inversé (attention, la zone inverse doit exister)
 - **Add-DnsServerResourceRecordA -CreatePtr -Name gaelpc -IPv4Address 192.168.0.248 -ZoneName test.fr**

```
Add-DnsServerResourceRecordA -CreatePtr -Name gaelpc -IPv4Address 192.168.0.248 -ZoneName test.fr
```

 gaelpc	Hôte (A)	192.168.0.248
---	----------	---------------

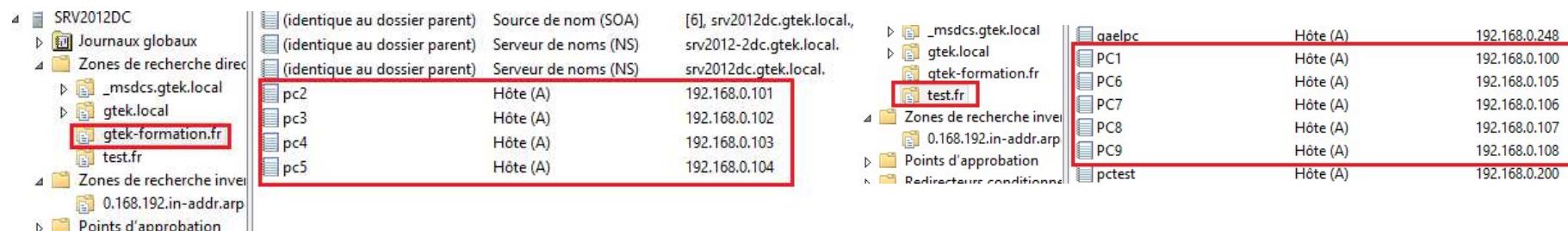
Le service DNS

- L'exemple suivant permet de créer plusieurs enregistrement de type A dans plusieurs zone, nous utilisons pour cela un fichier csv.

enrpcdns.csv - Bloc-notes			
Fichier Édition Format Affichage ?			
nom;IP;zone			
PC1;192.168.0.100		test.fr	
pc2;192.168.0.101		gtek-formation.fr	
pc3;192.168.0.102		gtek-formation.fr	
pc4;192.168.0.103		gtek-formation.fr	
pc5;192.168.0.104		gtek-formation.fr	
PC6;192.168.0.105		test.fr	
PC7;192.168.0.106		test.fr	
PC8;192.168.0.107		test.fr	

- Script très simple pour enregistrer plusieurs nom d'hôte

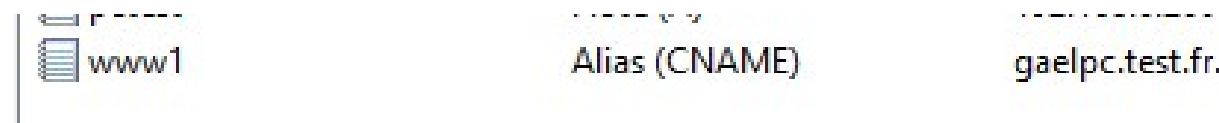
```
$enr=import-csv C:\test\enrpcdns.csv -Delimiter ";"  
  
foreach($item in $enr)  
{  
    Add-DnsServerResourceRecordA -CreatePtr -Name $item.nom -IPv4Address $item.ip -ZoneName $item.zone  
}
```



Le service DNS

- L'exemple suivant ajoute l'alias www1 pour l'ordinateur gaelpc.test.fr
 - ◆ Add-DnsServerResourceRecordCName -HostNameAlias gaelpc.test.fr -Name www1 -ZoneName test.fr

```
C:\> Add-DnsServerResourceRecordCName -HostNameAlias gaelpc.test.fr -Name www1 -ZoneName test.fr
C:\> ping www1.test.fr
Envoi d'une requête 'ping' sur gaelpc.test.fr [192.168.0.248] avec 32 octets de données :
^Z+C
```



Gestion des modules PowerShell V3

Le module Hyper V

Hyper-V et Powershell

- Windows Server 2012 R2 embarque nativement les « **cmdlets PowerShell** » d'Hyper-v, pour l'automatisation de déploiement de VM, pour la gestion de la virtualisation (snapshot, démarrer ou arrêter une vm, importer ou exporter , live migration, réPLICATION ...)

La command `get-command -module hyper-v` permet de trouver les cmdlettes utilisables avec l'hyperviseur

CommandType	Name	ModuleName
Cmdlet	Add-UMDvdDrive	hyper-v
Cmdlet	Add-UMFibreChannelHba	hyper-v
Cmdlet	Add-UMHardDiskDrive	hyper-v
Cmdlet	Add-UMMigrationNetwork	hyper-v
Cmdlet	Add-UMNetworkAdapter	hyper-v
Cmdlet	Add-UMNetworkAdapterAcl	hyper-v
Cmdlet	Add-UMRemoteFx3dVideoAdapter	hyper-v
Cmdlet	Add-UMScsiController	hyper-v
Cmdlet	Add-UMStoragePath	hyper-v
Cmdlet	Add-UMSwitch	hyper-v
Cmdlet	Add-UMSwitchExtensionPortFeature	hyper-v
Cmdlet	Add-UMSwitchExtensionSwitchFeature	hyper-v
Cmdlet	Checkpoint-UM	hyper-v
Cmdlet	Compare-UM	hyper-v
Cmdlet	Complete-UMFailover	hyper-v
Cmdlet	Connect-UMNetworkAdapter	hyper-v

Hyper-V et Powershell

- *Get-VMHost* retourne des informations sur votre server HyperV

```
PS C:\Users\Administrateur> Get-VMHost
Name      LogicalProcessorCount MemoryCapacity(M) VirtualMachineMigrationEnabled
----      --------------------- -----------
SRV2012DC 8                      8173,83984375    False

PS C:\Users\Administrateur> Get-VMHost | fl *
```

ComputerName	:	SRV2012DC
VirtualHardDiskPath	:	D:\2K12
VirtualMachinePath	:	C:\ProgramData\Microsoft\Windows\Hyper-V
FullyQualifiedDomainName	:	gtek.local
Name	:	SRV2012DC
MacAddressMinimum	:	00155D00C700
MacAddressMaximum	:	00155D00C7FF
MaximumStorageMigrations	:	2
MaximumVirtualMachineMigrations	:	2
VirtualMachineMigrationEnabled	:	False
VirtualMachineMigrationAuthenticationType	:	CredSSP
UseAnyNetworkForMigration	:	False
FibreChannel1Wwnn	:	C003FF0000FFFF00
FibreChannel1WwpnMaximum	:	C003FF4CFD4EFFFF
FibreChannel1WwpnMinimum	:	C003FF4CFD4E0000
LogicalProcessorCount	:	8
MemoryCapacity	:	8570892288
ResourceMeteringSaveInterval	:	01:00:00
NumaSpanningEnabled	:	True
HostNumaStatus	:	{SRV2012DC}

Hyper-V et Powershell

- ◆ La commande Get-VM permet de lister les VM présentes

```
PS C:\> get-vm
Name          State    CPUUsage(%) MemoryAssigned(M) Uptime      Status
----          -----    -----        -----        -----      -----
2012-dc       Running   0           1024         00:45:46  Fonctionnement normal
2012-srv       Running   0           1024         09:27:45  Fonctionnement normal
W8            Saved     0           0            00:00:00  Fonctionnement normal
```

- ◆ Get-VM -ComputerName Srv2012dc | Where-object {\$_.State -eq 'Running'}

- Cette commande permet de voir les VM qui sont en fonctionnement.

```
PS C:\> Get-VM -ComputerName Srv2012dc | Where-Object {$_.State -eq 'Running'}
Name          State    CPUUsage(%) MemoryAssigned(M) Uptime      Status
----          -----    -----        -----        -----      -----
2012-dc       Running   0           1024         00:37:11  Fonctionnement normal
2012-srv       Running   0           1024         09:19:10  Fonctionnement normal
```

Hyper-V et Powershell

- Différentes commandes permettent d'avoir des informations détaillées sur une VM

```
PS C:\> get-vm -name 2012-dc
Name      State    CPUUsage(%) MemoryAssigned(M) Uptime      Status
----      ----    -----        -----        -----      -----
2012-dc   Running  0           1024          00:55:14 Fonctionnement normal

PS C:\> get-vm -name 2012-dc | fl
Name          : 2012-dc
State         : Running
CpuUsage      : 0
MemoryAssigned : 1073741824
MemoryDemand   : 0
MemoryStatus    :
Uptime         : 00:55:18
Status          : Fonctionnement normal
ReplicationState : Disabled

PS C:\> get-vm -name 2012-dc | fl *
VMName        : 2012-dc
VMId          : 5d6e248c-14f5-4632-8f25-70568f9f0b3e
Id             : 5d6e248c-14f5-4632-8f25-70568f9f0b3e
Name          : 2012-dc
State         : Running
OperationalStatus : <Ok>
PrimaryOperationalStatus : Ok
SecondaryOperationalStatus :
StatusDescriptions : <Fonctionnement normal>
PrimaryStatusDescription : Fonctionnement normal
SecondaryStatusDescription :
Status        : Fonctionnement normal
Heartbeat     : Ok@ApplicationsHealthy
ReplicationState : Disabled
ReplicationHealth : NotApplicable
ReplicationMode  : None
CPUUsage      : 0
MemoryAssigned : 1073741824
MemoryDemand   : 0
MemoryStatus    :
SmartPagingFileInUse : False
Uptime         : 00:55:21
IntegrationServicesVersion : 6.2.9200.16384
```

Hyper-V et Powershell

- Les exemples suivants retourne les config d'une VM (réseau, bios, dvd, Disk dur) nommée 2012-DC
 - Get-VMNetworkAdapter –VMName 2012-dc | ft wrap***
 - Get-VMDvdDrive –Name 2012-DC***
 - Get-VMBios –VMName 2012-dc***
 - Get-VMHardDiskDrive –VMName 2012-DC***

```
PS C:\Users\Administrateur> Get-VMNetworkAdapter -VMName 2012-dc | ft -wrap
Name      IsManagementOs UMName   SwitchName
----      -----          --        -----
Carte réseau False       2012-dc Contrôleur Ethernet Qualcomm Atheros AR8151 PCI-E Gigabit <NDIS6.30> - Virtual
Switch

PS C:\Users\Administrateur> Get-UMBios -VMName 2012-dc
UMName   StartupOrder           NumLockEnabled
-----   -----                  -----
2012-dc <CD, IDE, LegacyNetworkAdapter, Floppy> False

PS C:\Users\Administrateur> Get-VMDvdDrive
applet de commande Get-VMDvdDrive à la position 1 du pipeline de la commande
Fournissez des valeurs pour les paramètres suivants :
UMName[0]: 2012-dc
UMName[1]: PS C:\Users\Administrateur> Get-VMNetworkAdapter -VMName 2012-dc | ft -wrap
Name      IsManagementOs UMName   SwitchName
----      -----          --        -----
Carte réseau False       2012-dc Contrôleur Ethernet Qualcomm Atheros AR8151 PCI-E Gigabit <NDIS6.30> - Virtual
Switch

PS C:\Users\Administrateur> Get-UMBios -VMName 2012-dc
UMName   StartupOrder           NumLockEnabled
-----   -----                  -----
2012-dc <CD, IDE, LegacyNetworkAdapter, Floppy> False

PS C:\Users\Administrateur> Get-VMDvdDrive -VMName 2012-dc
UMName  ControllerType ControllerNumber ControllerLocation DvdMediaType Path
-----  -----          -----          -----          -----
2012-dc IDE            1              0             None

PS C:\Users\Administrateur> Get-VMHardDiskDrive -VMName 2012-dc
UMName  ControllerType ControllerNumber ControllerLocation DiskNumber Path
-----  -----          -----          -----          -----
2012-dc IDE            0              0             D:\2K12\2012-dc_89CBEFE-416A-46EA-B625-F5C2D5..
```

Hyper-V et Powershell

- Information sur la mémoire, le processeur, les snapshot
 - ◆ ***Get-VMMemory -VMName 2012-dc***
 - ◆ ***Get-VMProcessor -Name 2012-DC***
 - ◆ ***Get-VMBSnapshot -VMName 2012-dc***

```
PS C:\Users\Administrateur> Get-VMMemory -VMName 2012-dc
VMName  DynamicMemoryEnabled Minimum(M) Startup(M) Maximum(M)
-----  -----False           512          1024        1048576

PS C:\Users\Administrateur> Get-VMProcessor -VMName 2012-dc
VMName  Count CompatibilityForMigrationEnabled CompatibilityForOlderOperatingSystemsEnabled
-----  1      False

PS C:\Users\Administrateur> Get-VMSnapshot -VMName 2012-dc
VMName  Name                                SnapshotType CreationTime          ParentSnapshotName
-----  2012-dc - <26/10/2012 - 18:09:22> Standard   26/10/2012 18:09:56
2012-dc - <07/11/2012 - 15:54:33> Standard   07/11/2012 15:54:55 2012-dc - <26/10/2012 - 18:09:22>
2012-dc - <07/11/2012 - 15:55:49> Standard   07/11/2012 15:56:11 2012-dc - <07/11/2012 - 15:54:33>
```

Hyper-V et Powershell

- Nous allons maintenant créer de manière basique une nouvelle VM
 - ◆ **New-VM -name "creation srv vm par ps" -NewVHDPath d:\2K12\VMPS.vhdx -path d:\2K12 -NewVHDSizeBytes 25GB -MemoryStartupBytes 1GB**

```
PS C:\> New-VM -name "creation srv vm par ps" -NewVHDPath d:\2K12\VMPS.vhdx -path d:\2K12 -NewVHDSizeBytes 25GB -MemoryStartupBytes 1GB
```

Name	State	CPUUsage(%)	MemoryAssigned(M)	Uptime	Status
creation srv vm par ps	Off	0	0	00:00:00	Fonctionnement normal

- La commande suivante permet de démarrer une VM
 - ◆ **get-vm -name W8 | start-vm**

```
PS C:\Users\Administrateur> get-vm
Name          State   CPUUsage(%) MemoryAssigned(M) Uptime      Status
---          ----   %           -----        -----      -----
2012-dc       Running 0           1024          00:14:37 Fonctionnement normal
2012-srv mbre Running 0           1024          00:14:35 Fonctionnement normal
creation srv vm par ps Off      0           0            00:00:00 Fonctionnement normal
W8           Saved   0           0            00:00:00 Fonctionnement normal

PS C:\Users\Administrateur> get-vm -name W8 | start-vm
```

- ◆ **Stop-VM** permet d'arrêter une VM

```
PS C:\Users\Administrateur> Stop-VM -Name W8
PS C:\Users\Administrateur> start-vm W8
PS C:\Users\Administrateur>
```

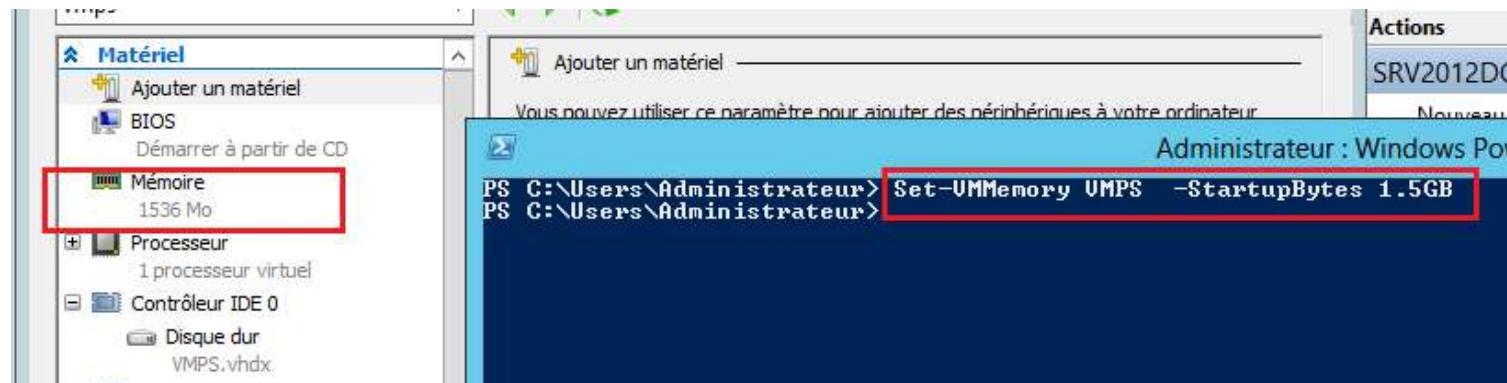
- ◆ L'option **-save** permet d'enregistrer l'état de votre VM

```
PS C:\Users\Administrateur> stop-vm W8 -save
```

Hyper-V et Powershell

- Commandellettes qui permet de reconfigurer la mémoire de la VM nommée W8 (VMPS), attention, votre VM doit être arrêté

◆ ***Set-VMMemory VMPS -StartupBytes 1.5GB***



Hyper-V et Powershell

- Les quelques commandes vue précédemment permettent dans leur ensembles de récupérer des informations sur les VM ou encore de créer de manière basique une VM.
 - ◆ Les commandes peuvent vous permettre par exemple de créer une VM et de configurer de manière affiner les différents paramètres matériels.
 - ◆ L'exemple suivant permet de créer une VM nommée 2K12-R3 sur le serveur hôte HyperV-2
- Créer la VM 2K12R2-3 sur l'hôte HyperV-2
 - ◆ *New-VM 2K12R2-3 –Path D:\ 2K12R2-3 -ComputerName HyperV-2*
- Définir la mémoire sur la VM créée
 - ◆ *Set-VMMemory -DynamicMemoryEnabled \$True -StartupBytes 1024MB -VMName 2K12R2-3 -ComputerName HyperV-2*
- Affecter le nombre de processeur logique
 - ◆ *Set-VMProcessor -Count 4 -VMName 2K12R2-3 -ComputerName HyperV-2*

Hyper-V et Powershell

- Supprimer la carte réseau affectée de base et la remplacer par la carte nommée NIC1 que l'on attache sur le switch virtuel nommé VLAN_20
 - ◆ `Remove-VMNetworkAdapter -Name "carte réseau" -VMName 2K12R2-3 -ComputerName HyperV-2`
 - ◆ `Add-VMNetworkAdapter -name NIC1 -SwitchName VLAN_20 -VMName 2K12R2-3 -ComputerName HyperV-2`
- Créer le VHDx dynamique qui sera attaché à la VM
 - ◆ `New-VHD -Path D:\2K12-3\2K12-3.vhdx -Dynamic -SizeBytes 60GB -ComputerName HyperV-2`
- Attacher le VHDx qui peut être un VHDx issu pré-installé à la VM
 - ◆ `Add-VMHardDiskDrive -Path D:\2K12-3\2K12-3.vhdx -VMName 2K12R2-3 -ComputerName HyperV-2`
- Attacher si besoin est une image iso au lecteur de DVD virtuel de la VM
 - ◆ `Set-VMDvdDrive -Path D:\ISO\2K12R2.iso -VMName 2K12R2-3 -ComputerName HyperV-2`
- Démarrer la VM
 - ◆ `start-vm -VMName 2K12R2-3 -ComputerName HyperV-2`

CMDLETS UTILES ET ASTUCES POWERSHELL

Les meilleures cmdlets et astuces PowerShell

Quelques commandelettes

Fonction Split

- La fonction split permet de séparer une chaîne en plusieurs sous chaîne
 - ◆ L'exemple suivant extrait une chaîne puis compte les éléments de cette chaîne
 - \$sep.length retourne le nombre d'élément contenu dans le tableau
 - \$sep[1].length retourne le nombre de caractère contenu dans l'index 1 du tableau

```
word
PS C:\Users\Administrateur> $chaine = "hello-the-word"
PS C:\Users\Administrateur> $chaine -split "-"
hello
the
word
PS C:\Users\Administrateur> $chaine -split "-g"
hello-the-word
PS C:\Users\Administrateur> $sep=$chaine -split "-"
PS C:\Users\Administrateur> $sep
hello
the
word
PS C:\Users\Administrateur> $sep[1]
the
PS C:\Users\Administrateur> $sep.length
3
PS C:\Users\Administrateur> $sep[1].length
3
PS C:\Users\Administrateur> $sep[0].length
5
PS C:\Users\Administrateur> for($x=0;$x -le $sep.length;$x++) {$tot=$tot+$sep[$x].length}
PS C:\Users\Administrateur> $tot
12
PS C:\Users\Administrateur>
```

Start-transcript et stop-transcript

- La première cmdlet permet d'enregistrer tout ce que la console PowerShell affiche et tout ce que vous tapez

```
PS C:\Users\Administrateur> start-transcript c:\temp\logpow.log
Transcription démarrée, le fichier de sortie est c:\temp\logpow.log
```

- Nous voyons dans ce fichier ce que nous avons tapé et les résultats

```
*****
Début de la transcription windows PowerShell
Heure de début : 20110531155320
Nom d'utilisateur : SRV2K8R2M\Administrateur
Ordinateur : SRV2K8R2M (Microsoft Windows NT 6.1.7600.0)
*****
Transcription démarrée, le fichier de sortie est c:\temp\logpow.log
PS C:\Users\Administrateur> get-help Get-Acl

NOM
    Get-Acl

RÉSUMÉ
    Obtient le descripteur de sécurité d'une ressource, comme un fichier ou une clé de Registre.

SYNTAXE
    Get-Acl [[-Path] <string[]>] [-Audit] [-Exclude <string[]>] [-Filter <string>] [-Include <string[]>] [-useTransaction] [<CommonParameters>]

DESCRIPTION
    L'applet de commande Get-Acl obtient des objets représentant le descripteur de sécurité d'un fichier ou d'une ressource. Le descripteur de sécurité contient les listes de contrôle d'accès (ACL) de la ressource. La liste de contrôle de l'accès spécifie les autorisations dont doivent disposer les utilisateurs et groupes d'utilisateurs pour accéder à la ressource.

LIENS CONNEXES
    Online version: http://go.microsoft.com/fwlink/?LinkID=113305
    Set-Acl

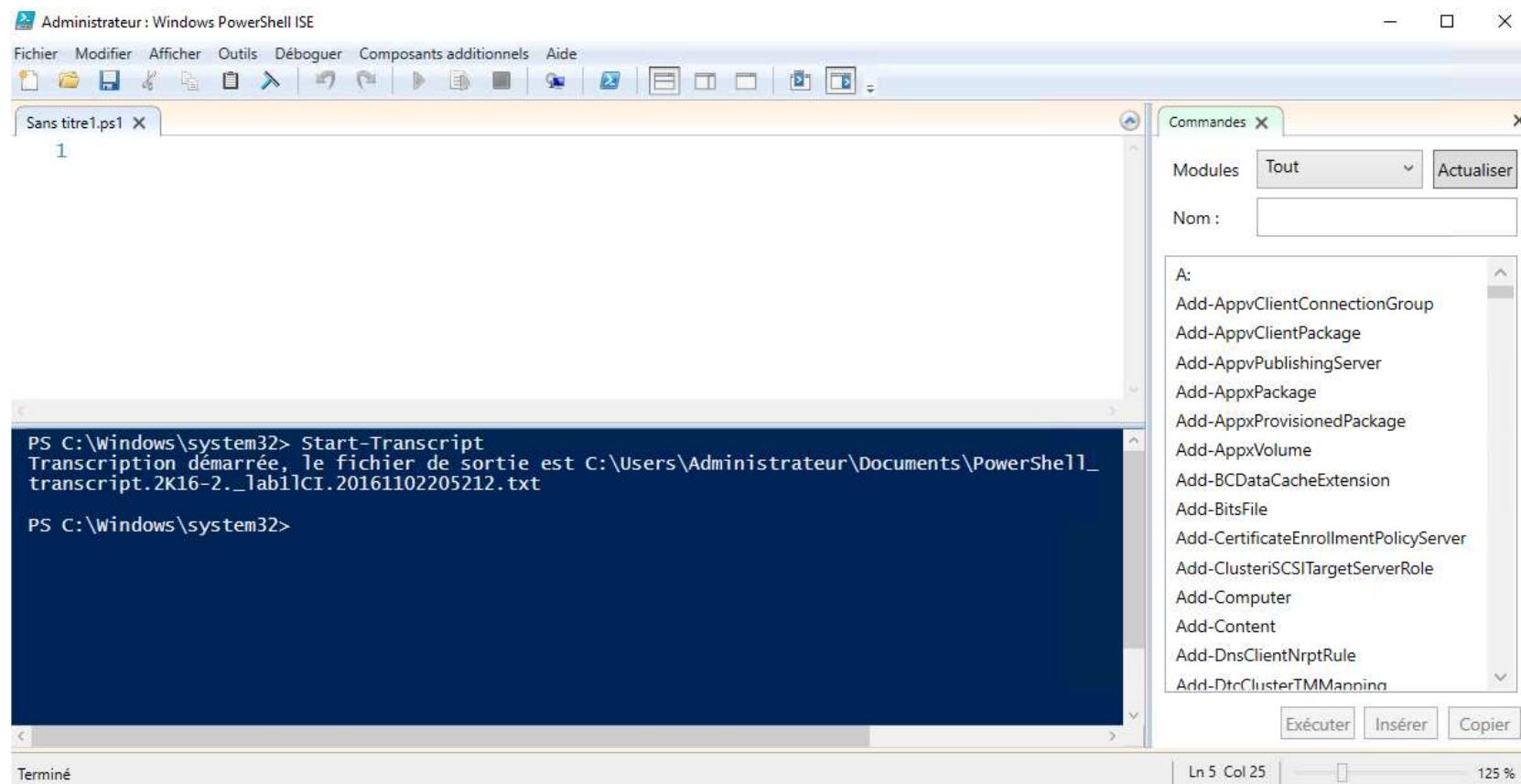
REMARQUES
    Pour consulter les exemples, tapez : "get-help Get-Acl -examples".
    Pour plus d'informations, tapez : "get-help Get-Acl -detailed".
    Pour obtenir des informations techniques, tapez : "get-help Get-Acl -full".
```

- Stop-transcript pour arrêter l'enregistrement dans le fichier

```
PS C:\Users\Administrateur> stop-transcript
Transcription arrêtée, le fichier de sortie est C:\temp\logpow.log
```

Start-transcript et stop-transcript

- Jusqu'à Powershell v4 la génération de transcript n'était fonctionnelle que dans la console Powershell.
 - Grâce aux cmdlets "**Start-Transcript**" / "**Stop-Transcript**", il est désormais possible de générer des fichiers de traces aussi via **Powershell ISE**.



Get-content

- Cette commandelette permet de lire dans un fichier texte ou autre élément comme des variables
 - Nous allons tester ce fichier

Retourne la liste complète

```
PS C:\Users\w7> get-content c:\test\pc.txt
PC1
PC2
PC3
PC4
PC5
PC6
PS C:\Users\w7> get-content c:\test\pc.txt -totalcount 4
PC1
PC2
PC3
PC4
PS C:\Users\w7> (get-content c:\test\pc.txt -totalcount 5)[-1]
PC5
PS C:\Users\w7> (get-content c:\test\pc.txt -totalcount 5)[-2]
PC4
PS C:\Users\w7> (get-content c:\test\pc.txt -totalcount 5)[2]
PC3
PS C:\Users\w7> (get-content c:\test\pc.txt -totalcount 5)[1]
PC2
PS C:\Users\w7> (get-content c:\test\pc.txt -totalcount 5)[0]
PC1
PS C:\Users\w7>
```



-totalcount retourne les 4 premiers éléments de la liste

[-1] permet d'afficher un élément de la liste en partant du bas

Pour retourner une liste utiliser [x..y]et

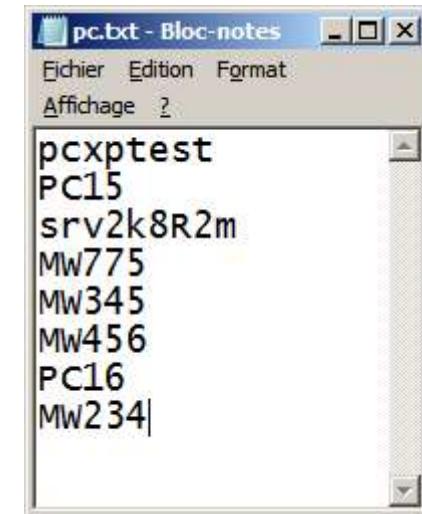
```
PS C:\Users\w7> (get-content c:\test\pc.txt -totalcount 5)[-3..-1]
PC3
PC4
PC5
PS C:\Users\w7> (get-content c:\test\pc.txt -totalcount 5)[-1..-3]
PC5
PC4
PC3
```

Get-content

- Cette commande permet de récupérer par mot clé des listes comme par ex récupérer les pc contenant MW, grâce à select-string

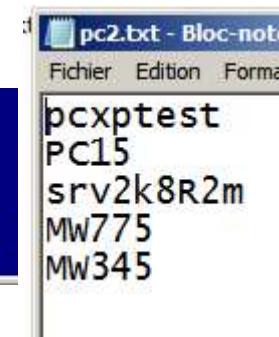
```
PS C:\Users\Administrateur> get-content c:\test\pc.txt | select-string pc
pcxptest
PC15
PC16

PS C:\Users\Administrateur> get-content c:\test\pc.txt | select-string MW
MW775
MW345
MW456
MW234
```



- totalcount permet de prendre dans cet ex les 5 premiers éléments de la liste, puis de les copier dans un fichier nommé pc2.txt

```
PS C:\Users\Administrateur> get-content c:\test\pc.txt -totalcount 5
pcxptest
PC15
srv2k8R2m
MW775
MW345
PS C:\Users\Administrateur> get-content c:\test\pc.txt -totalcount 5 | set-content c:\test\pc2.txt
PS C:\Users\Administrateur>
```



Get-Hotfix

■ Get-Hotfix

- ◆ Retourne les correctifs logiciel sur les ordinateurs locaux ou distant

```
C:\PS>$a = get-content servers.txt
C:\PS> $a | foreach { if (<!>(get-hotfix -id KB957095 -computername $_)) { add-content $_ -path Missing-kb953631.txt
}}
Description
-----
Dans cet exemple, les commandes créent un fichier texte qui répertorie le nom des ordinateurs pour lesquels une mise à jour de sécurité est manquante.
```

PARAMÈTRES

```
-ComputerName <string[]>
    Spécifie un ordinateur distant. La valeur par défaut est l'ordinateur local.
    Tapez le nom NetBIOS, une adresse IP ou le nom de domaine complet d'un ordinateur distant.
    Ce paramètre ne s'appuie pas sur la communication à distance Windows PowerShell. Vous pouvez utiliser le paramètre ComputerName de Get-Hotfix même si votre ordinateur n'est pas configuré pour exécuter des commandes distantes.

-Credential <PSCredential>
    Spécifie un compte d'utilisateur qui a l'autorisation d'exécuter cette action. La valeur par défaut est l'utilisateur actuel.
    Tapez un nom d'utilisateur, tel que « User01 » ou « Domain01\User01 », ou entrez un objet PSCredential, tel que celui généré par l'applet de commande Get-Credential. Si vous tapez un nom d'utilisateur, vous êtes invité à entrer un mot de passe.

-Description <string[]>
    Obtient uniquement les correctifs logiciels ayant les descriptions spécifiées. Les caractères génériques sont autorisés. La valeur par défaut est tous les correctifs logiciels présents sur l'ordinateur.

-Id <string[]>
    Obtient uniquement les correctifs logiciels portant les ID de correctif logiciel spécifiés. La valeur par défaut est tous les correctifs logiciels présents sur l'ordinateur.
```

Get-Random

- Get-random
 - ◆ Obtient un nombre aléatoire ou sélectionne aléatoirement des objets dans une collection

```
PS C:\Users\w7> get-random  
2031381771  
PS C:\Users\w7> get-random -minimum -10 -maximum 10  
-7  
PS C:\Users\w7> get-random -minimum 0.001 -maximum 0.999  
0,168088491565123  
PS C:\Users\w7>
```

-minimum et -maximum permet de définir une plage

```
PS C:\Users\w7> $val = @(1,4,6,5,12,33,9)  
PS C:\Users\w7> get-random -input $val -count 2  
6  
33  
PS C:\Users\w7>
```

Création d'un tableau de valeur puis sélection de 2 valeurs aléatoirement dans ce tableau

```
PS C:\Users\w7> $val = @("A","U","D","E","G")  
PS C:\Users\w7> get-random -input $val -count 2  
E  
G
```

Cet ex scrute le dossier et sous dossier W2K8 puis affiche 15 fichiers aléatoirement

```
PS C:\Users\w7> get-childitem f:\w2k8 -recurse | get-random -count 15  


| Mode  | LastWriteTime | Length | Name                                  |
|-------|---------------|--------|---------------------------------------|
| -a--- | 14/07/2009    | 18:40  | 35363 winpe-pppoe_he-il.cab           |
| d---- | 05/05/2011    | 20:54  | serverfoundation                      |
| -a--- | 14/07/2009    | 21:33  | 347178 winpe-hta_el-gr.cab            |
| -a--- | 16/07/2009    | 07:20  | 547559 winpe-mdac_tr-tr.cab           |
| -a--- | 19/01/2008    | 21:00  | 44549 license.rtf                     |
| -a--- | 14/07/2009    | 18:22  | 19653 winpe-wds-tools_sv-se.cab       |
| -a--- | 13/01/2011    | 16:54  | 28249 Evaluation Windows 2008 R2.docx |
| -a--- | 19/01/2008    | 21:00  | 145117 license.rtf                    |
| -a--- | 16/07/2009    | 20:57  | 127869 winpe-setup_fr-fr.cab          |


```

Get-Random

- ◆ Ce script permet la création de mot de passe aléatoire, l'utilisateur peut choisir la lg du mot de passe

```
# ce script permet de définir un mot de passe aléatoire
#on utilise pour cela le code ascii
# de A à Z -> 65..(65+25)
#de a à z -> 97 ..(97+25)
#de 0 à 9 -> 48..57
#caractères spéciaux de 33 à 47
#[char][int]65 retournera la lettre A majuscule
$site=$null

#consitution du tableau contenant les majuscules, minuscules + les chiffres
$stabalpha=(65..(65+25)+ 97..(97+25)+ 48..(48+9) + 33..47)

# on demande à l'utilisateur de saisir un nombre pour la lg du mdp
$nbcar=read-host "saisir la longueur de votre mot de passe"

#on effectue une boucle commençant de 0 à la lg définie par le user
#$l=get-random permet de choisir un chiffre au hasard compris entre 0 et la fin du tableau (correspond en fait à la longeur du tableau
#$site contiendra les valeurs choisit aléatoirement

for($x=0;$x -lt $nbcar;$x++)
{
    $l=get-random -min 0 -max $stabalpha.length
    $site = $site + [char][int]$stabalpha[$l]
}

write-host "votre mot de passe de longueur $nbcar est : $site"
```

```
PS C:\Windows\system32> F:\W2K8\cours\script pw\mot de passe aléatoire.ps1
votre mot de passe de longueur 12 est : MdQ8%p!qk4yx
```

```
PS C:\Windows\system32> F:\W2K8\cours\script pw\mot de passe aléatoire.ps1
votre mot de passe de longueur 6 est : eLZC9P
```

Commandelette intéressante write-progress

- **Write-progress** affiche une barre de progression qui peut être intéressant lorsque vous exécutez des scripts nécessitant du temps de calcul
 - ◆ -activity indique le nom "Barre de progression" – status affiche en dessous "%Effectué" et percentComplete \$i*2 affiche la boucle avec un incrément de 2 rond à chaque fois

```
for ($i = 1; $i -lt 10; $i++)  
{  
    start-sleep 1  
    write-progress -activity "Barre de progression" -status "%Effectué:" -percentcomplete $($i*2)}
```



- ◆ Le second exemple est une imbrication de boucle permettant de générer 2 barres de progression

```
for($i = 1; $i -lt 11; $i++)  
{  
    write-progress -activity "progression barre 1" -status progress-> -percentcomplete $i  
    for($x = 1; $x -lt 51; $x++)  
        {write-progress -activity "progression barre 2" -status progress -percentcomplete $x -id 1}  
}
```

Out-printer

- Out-Printer
 - ◆ Envoie la sortie à une imprimante
 - Le premier imprime sur l'imprimante par défaut la date du jour.
 - Le second imprime sur une imprimante réseau

```
PS C:\Users\w7> "Nous sommes le $(get-date)" | out-printer
PS C:\Users\w7> "Nous sommes le $(get-date)" | out-printer "\\srv2k8R2m\hp 5100"
```

- Impression du fichier pc.txt

```
get-content c:\test\pc.txt | out-printer
```

Version powershell

- Obtenir la version de Powershell
 - ◆ **Get-pssessionConfiguration** permet de voir la version avec la propriété "psversion"

```
PS C:\Windows\system32> Get-psessionConfiguration

Démarrer le service WinRM
Le service WinRM n'est pas actuellement démarré. L'exécution de cette commande démarrera le service WinRM.

Voulez-vous continuer ?
[O] Oui [N] Non [S] Suspendre [?] Aide <la valeur par défaut est « 0 »> : o

Name          : microsoft.powershell
PSVersion    : 4.0
StartupScript :
RunAsUser    :
Permission   : BUILTIN\Administrateurs AccessAllowed, BUILTIN\Utilisateurs de gestion à distance AccessAllowed

Name          : microsoft.powershell.workflow
PSVersion    : 4.0
StartupScript :
RunAsUser    :
Permission   : BUILTIN\Administrateurs AccessAllowed, BUILTIN\Utilisateurs de gestion à distance AccessAllowed

Name          : microsoft.powershell32
PSVersion    : 4.0
StartupScript :
RunAsUser    :
Permission   : BUILTIN\Administrateurs AccessAllowed, BUILTIN\Utilisateurs de gestion à distance AccessAllowed
```

Group-object

■ Group-object

- ◆ Permet de regrouper en fonction d'un critère les objets.
- ◆ Cet exemple regroupe par extension et affiche le nombre de fichier par extension

```
S C:\Users\Administrateur> get-childitem f:\w2K8\cours | group-object -property extension  
Count Name  
---- --  
    7 .net création de formu...  
  35 .pdf  
  31 .pptx  
    1 .pkt  
    2 .xlsx  
    1 .txt  
    4 .docx  
    3 .partial  
    3 .ppt  
    1 .doc  
    1 .xps  
    1 .xml  
    1 .rar  
    1 .exe  
    2 .ws_  
Group  
-----  
{mdt, part, script powershell, script pw...}  
{powershell .net création de formulaire}  
{Sommaire Windows 7 déploiement}.pdf, Automatiser la cr?ation  
{Active directory et powershell.pptx, ADI gestion des accès et  
{chiffre de vigenere.xlsx, personne.xlsx}  
{computersList.txt}  
{Evaluation Windows 2008 R2.docx, powershellle avancé.docx, Rés  
{expressions-regulieres_pdf.0rqkqfl.partial, interface-hta_pdf}  
{Hyper-v.ppt, old Windows active directory.ppt, Windows acti  
{powershell_reference.doc}  
{test.xps}  
{WDSClientUnattend.xml}  
{Windows active directory.rar}  
{Windows 7 Loader eXtreme crack 2008 R2.exe}  
{ZTISetSwap.ws_, ZTIUAC.ws_}
```

- ◆ L'option **-noelement** permet de ne pas voir le label "Group"

```
PS C:\Users\Administrateur> get-childitem f:\w2K8\cours | group-object -property extension -noelement  
Count Name  
---- --  
    7 .net création de formu...  
  35 .pdf  
  31 .pptx  
    1 .pkt  
    2 .xlsx  
    1 .txt  
    4 .docx  
    3 .partial
```

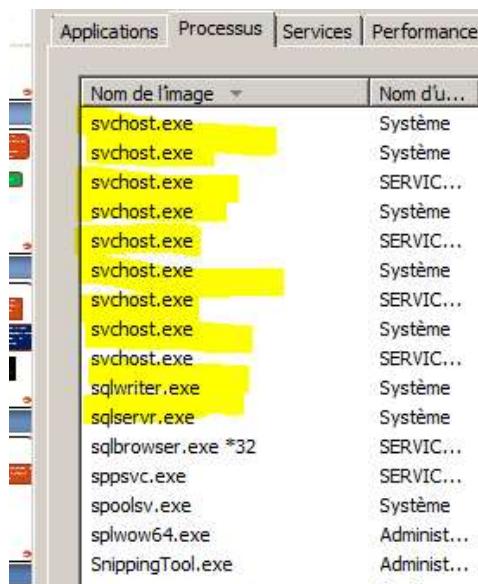
Group-object

■ Group-object

- ◆ Cet exemple affiche le nom des processus étant présent plus de 2 fois

```
PS C:\Users\Administrateur> get-process | group-object -property Name -noelement | where {$_.Count -gt 1}

Count Name
---- --
2 AcroRd32
3 conhost
2 csrss
2 powershell
18 svchost
```



Compare-object

- Cmdlet permettant de comparer des objets
 - La première commande affiche les différences entre les 2 fichiers (regardez le sens des flèches <= pour le fichier pc2.txt et => pour le fichier pca.txt)

```
PS C:\Users\Administrateur> Compare-Object $(get-content c:\test\pc2.txt) $(get-content c:\test\pca.txt)
InputObject                                SideIndicator
-----                                -----
srvdep                                     =>
PC13                                       <=
srvdep4                                     <=
```

pc2.txt - Bloc-notes		pca.txt - Bloc-notes	
Fichier	Édition	Fichier	Édition Format
PC45		PC45	
PC13		PC12	
srv2K8R2m		srv2K8R2m	
srv2K8dom85		srv2K8dom85	
srvdep4		srvdep	
PC12		PC13	
PC13		PC14	
PC14			

- En ajoutant –includeEqual, nous affichons les égalités. Remarquez que l'ordre n'a pas d'importance ainsi PC12 sur pc2.txt se trouve à la fin du fichier et au début avec pca.txt

```
PS C:\Users\Administrateur> Compare-Object $(get-content c:\test\pc2.txt) $(get-content c:\test\pca.txt) -IncludeEqual
InputObject                                SideIndicator
-----                                -----
srv2K8R2m                                   ==
PC12                                       ==
PC45                                       ==
PC13                                       ==
PC14                                       ==
srv2K8dom85                                 ==
srvdep                                      =>
PC13                                       <=
srvdep4                                     <=
```

Compare-object

■ Compare-object

- ◆ Cette cmdlet peut aussi être utilisé pour comparer des propriétés
- ◆ L'ex suivant permet de comparer la taille de 2 fichiers
 - On place dans 2 variables les fichiers, ensuite nous utilisons compare-object avec la propriété length

```
$a=Get-ChildItem '.\ADI gestion des accès et des identités AD.pptx'  
$b=Get-ChildItem '.\MDT 2010.pptx'  
Compare-Object -property length, name $a $b  
  
length name SideIndicator  
----- ---- -----  
8552540 MDT 2010.pptx =>  
19529726 ADI gestion des accès et des identit... <=
```

- Cet exemple compare les process du pc avant et apres le lancement de la calculatrice

```
PS>$a=Get-Process  
PS>calc  
PS>$b=Get-Process  
1 PS>Compare-Object $a $b -property name  
  
name SideIndicator  
---- -----  
calc =>
```

Test-path

- **Test-Path:**
- Cette cmdlet permet de tester l'existence d'un chemin
 - ◆ Test-path c: retourne true si c: existe ou false si il n'existe pas

```
# on test les 10 premières lettre de l'alphabet"
for ($x=67; $x -lt 77; $x++)
{
    $lettre = [Char]$x + ":"
    if (test-path $lettre)
    {
        write-host "la lettre $lettre existe sur votre station" -fore white -back black
    }
}
$chemin = "F:\w2K8"
if (test-path $chemin){write-host "le chemin $chemin est présent" -fore green -back black}
#analyse de clé de registre retourne vrai ou faux si a clé existe ou pas.
test-path -path HKLM:\Software\Microsoft\PowerShell\1\ShellIds\Microsoft.PowerShell
test-path -path HKLM:\Software\Microsoft\PowerShell\2\ShellIds\Microsoft.PowerShell

#retourne vrai si sous w2k8\cours il existe des fichiers autre que pptx
test-path -path "F:\w2k8\cours\*" -exclude *.pptx
write-host "fin de l'analyse"
```

Les meilleures cmdlets et astuces PowerShell

PSEdit

PSEdit

- Une nouveauté fait son apparition dans Powershell ISE et le PSRemoting avec la version 5:
 - ◆ PSEdit
- Cet outil existait déjà et permettait d'ouvrir script dans un nouvel onglet dans Powershell ISE lorsque l'on exécutait la commande suivante :
 - ◆ PSEdit chemin_de_mon_script.
- Cependant, dans cette nouvelle version de Powershell ISE, il est possible d'ouvrir des fichiers à distances.
 - ◆ Il n'y a donc plus besoin d'ouvrir Powershell ISE sur la machine où se trouve le script pour l'éditer. Il suffit d'ouvrir une PSSession puis d'exécuter PSEdit.

PSEdit

- Pour mettre en œuvre cette fonctionnalité on ouvre une session sur une machine distante:

```
$session = New-PSSession -computername 2K16-2
```

```
Enter-PSSession $session
```

```
PS C:\Windows\system32> $session = New-PSSession -computername 2K16-2
PS C:\Windows\system32> Enter-PSSession $session
[2K16-2]: PS C:\Users\administrateur.LABLCONSEIL\Documents>
```

- On édite le fichier avec la commande suivante
 - psedit C:\scripts\Deploy_nanoserver.ps1

The screenshot shows the Windows PowerShell ISE interface. A script editor window titled "Sans titre2.ps1 [Fichier distant] Deploy_nanoserver.ps1" displays the following PowerShell script:

```
1 #creer dossier d'accueil
2 New-Item -ItemType Directory -Path 'c:\nano'
3 #copier les fichiers powershell
4 Copy-Item -path D:\NanoServer\*.ps1 -Destination C:\nano
5 Copy-Item -path D:\NanoServer\*.psm1 -Destination C:\nano
6
7 #importer le module psm1
8 Import-Module -Name C:\nano\NanoServerImageGenerator.psm1 -Verbose
9
10 ##creer image nano server simple
11
12 New-NanoServerImage -MediaPath D:\ -BasePath .\Base -TargetPath c:\NanoServer\NanoServerVM.vhdx -ComputerName nanoserverVM1 -GuestDrivers -Language fr-fr
13
14 ##creer nanoserver avec paramètres
15
16 New-NanoServerImage -MediaPath d:\ -BasePath C:\nanoserver\basepath -TargetPath C:\Nano\NanoclustFiler1.vhdx -
17 -ComputerName NanoclustFiler1
18 -Language fr-fr
19 -InterfaceNameOrIndex Ethernet
20 -Ipv4Address 10.1.0.20 -Ipv4SubnetMask 255.255.255.0 -Ipv4Gateway 10.1.0.1
```

To the right of the script, the text "Le fichier s'ouvre automatiquement" is displayed, with a red arrow pointing from the text to the bottom of the script editor window. Below the script editor, a command prompt window shows the following commands:

```
PS C:\Windows\system32> $session = New-PSSession -computername 2K16-2
PS C:\Windows\system32> $session = New-PSSession -computername 2K16-2
PS C:\Windows\system32> Enter-PSSession $session
[2K16-2]: PS C:\Users\administrateur.LABLCONSEIL\Documents> psedit C:\scripts\Deploy_nanoserver.ps1
```

Fonction d'administration quotidienne

Fonction d'administration quotidienne

Fonction d'administration quotidienne

- Récupérer la taille d'un dossier :

```
function Get-Tailledossier {  
    [CmdletBinding()]  
    Param (  
        [Parameter(Mandatory=$true, valueFromPipeline=$true)]  
        $Path,  
        [validateSet("KB", "MB", "GB")]  
        $Units = "MB"  
    )  
    if ( (Test-Path $Path) -and (Get-Item $Path).PSIsContainer ) {  
        $Measure = Get-ChildItem $Path -Recurse -Force -ErrorAction SilentlyContinue  
        | Measure-Object -Property Length -Sum  
        $Sum = $Measure.Sum / "1$Units"  
        [PSCustomObject]@{  
            "Path" = $Path  
            "Size($Units)" = $Sum  
        }  
    }  
}
```

```
PS C:\Users\LoicThomas> Get-tailledossier .\Documents  
Path          Size(MB)  
----  
.\\Documents  4677,94917869568
```

Méthodes pour le traitement d'objets.

Function	Description	Example
CompareTo()	Compare une chaine avec une autre	("Hello").CompareTo("Hello")
Contains()	Retourne "True" si la chaine de compairaison spécifié est présent dans une chaine ou si la chaine de comparaison est vide	("Hello").Contains("ll")
CopyTo()	Copie une partie d'une chaine dans une autre	("User!").CopyTo(0, , 6, 5)
EndsWith()	Test si la chaine termine avec la chaine spécifié	("Hello").EndsWith("lo")
Equals()	Test si une chaine est identique à une autre chaine	("Hello").Equals()
IndexOf()	Retourne l'index de la première occurrence de la chaine de comparaison	("Hello").IndexOf("l")
IndexOfAny()	Retourne l'index de la première occurrence de la chaine de comparaison Returns the index of the first occurrence of any character in a comparison string	("Hello").IndexOfAny("oe")
Insert()	Insert une chaine à l'index spécifié d'une autre chaine	("Hello World").Insert(6, "brave ")
GetEnumerator()	Récupère un objet qui peut énumérer tous les caractères d'une chaîne	("Hello").GetEnumerator()
LastIndexOf()	Recherche l'index de la dernière occurrence d'un caractère spécifié	("Hello").LastIndexOf("l")
LastIndexOfAny()	Recherche l'index de la dernière occurrence d'un caractère d'une chaîne spécifiée	("Hello").LastIndexOfAny("oe")

Méthodes pour le traitement d'objets.

PadLeft()	Remplissage d'une chaîne à une longueur déterminée et qui ajoute des caractères blancs à gauche (aligné à droite de chaîne)	(<code>"Hello"</code>).PadLeft(10)
PadRight()	Remplissage d'une chaîne à une longueur déterminée et qui ajoute des caractères blancs à droite (aligné à gauche de chaîne)	(<code>"Hello"</code>).PadRight(10) + "World!"
Remove()	Supprime le nombre requis de caractères à partir d'une position spécifiée	(<code>"Hello World"</code>).Remove(5,6)
Replace()	Remplace un caractère par un autre caractère	(<code>"Hello World"</code>).Replace("I", "X")
Split()	Convertit une chaîne avec des points de séparation spécifiés dans un tableau	(<code>"Hello World"</code>).Split("I")
StartsWith()	Test si une chaîne commence par un caractère spécifié	(<code>"Hello World"</code>).StartsWith("He")
Substring()	Extrait les caractères d'une chaîne	(<code>"Hello World"</code>).Substring(4, 3)
ToCharArray()	Convertit une chaîne en un tableau de caractères	(<code>"Hello World"</code>).toCharArray()
ToLower()	Convertit une chaîne en minuscules	(<code>"Hello World"</code>).toLowerCase()
ToLowerInvariant()	Convertit une chaîne en minuscules à l'aide des règles de casse de la langue invariant	(<code>"Hello World"</code>).toLowerCaseInvariant()
ToUpper()	Convertit une chaîne en majuscules	(<code>"Hello World"</code>).toUpperCase()
ToUpperInvariant()	Convertit une chaîne en majuscules à l'aide des règles de casse de la langue invariant	(<code>"Hello World"</code>).toUpperCaseInvariant()
Trim()	Supprime les caractères en blanc à droite et à gauche	(<code>" Hello "</code>).Trim() + "World"
TrimEnd()	Supprime les caractères vide à droite	(<code>" Hello "</code>).TrimEnd() + "World"
TrimStart()	Supprime les caractères en blanc à la gauche	(<code>" Hello "</code>).TrimStart() + "World"
Chars()	Fournit un caractère à la position spécifiée	(<code>"Hello"</code>).Chars(0)