

Présentation de la formation

Site : <http://www.alphorm.com>
Blog : <http://www.ConfigMgrdistrict.com>
Blog : <http://www.PowerShelldistrict.com>
Forum : <http://www.alphorm.com/forum>

Stéphane van Gulick

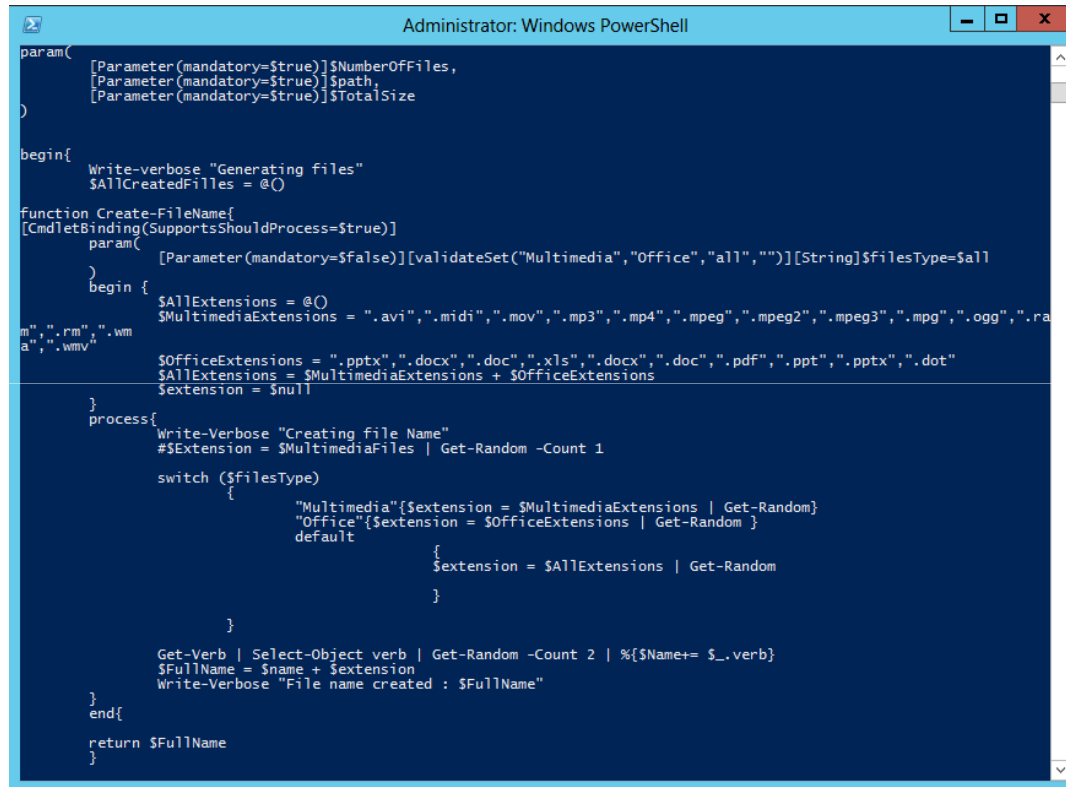
Consultant systèmes et automatisation

Certifications : MCT, MCITP, MCSA

Contact : svangulick@alphorm.com

Twitter : @Stephanevg

Introduction



```
param(
    [Parameter(mandatory=$true)]$NumberOfFiles,
    [Parameter(mandatory=$true)]$path,
    [Parameter(mandatory=$true)]$TotalSize
)

begin{
    Write-verbose "Generating files"
    $AllCreatedFiles = @()
}

Function Create-FileName{
    [CmdletBinding(SupportsShouldProcess=$true)]
    param(
        [Parameter(mandatory=$false)]$validateSet("Multimedia","Office","all","")$filesType=$all
    )
    begin {
        $AllExtensions = @()
        $MultimediaExtensions = ".avi",".midi",".mov",".mp3",".mp4",".mpeg",".mpeg2",".mpeg3",".mpg",".ogg",".ra"
        $OfficeExtensions = ".pptx",".docx",".doc",".xls",".docx",".doc",".pdf",".ppt",".pptx",".dot"
        $AllExtensions = $MultimediaExtensions + $OfficeExtensions
        $extension = $null
    }
    process{
        Write-Verbose "Creating file Name"
        # $Extension = $MultimediaFiles | Get-Random -Count 1
        switch ($filesType)
        {
            "Multimedia"{$extension = $MultimediaExtensions | Get-Random}
            "Office"{$extension = $OfficeExtensions | Get-Random }
            default
            {
                $extension = $AllExtensions | Get-Random
            }
        }

        Get-Verb | Select-Object verb | Get-Random -Count 2 | %{ $Name += $_.verb }
        $FullName = $name + $extension
        Write-Verbose "File name created : $FullName"
    }
    end{
        return $FullName
    }
}
```

Plan

- Présentation du formateur.
- Prérequis de cette formation.
- Quelques références.
- Présentation de la formation.



Présentation du formateur

- Présentation
- Stéphane van Gulick
 - Ingénieur Systèmes et automatisation
 - MCTS, MCITP EA, MCSA
 - Blog : <http://PowerShellDistrict.com>
 - Twitter : @stephanevg



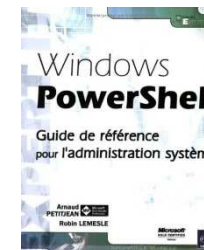
Prérequis de cette formation

- Connaissances des systèmes Windows
- Une expérience en Scripting est un plus. (Mais pas obligatoire !)
- Compréhension de principes algorithmiques (tel que les « IF, ELSE, FOREACH) etc..
- Être motivé et enthousiaste !
- Lab:
 - 1 machine Windows 7
 - (une deuxième machine windows 7 pour le chapitre sur le remoting).

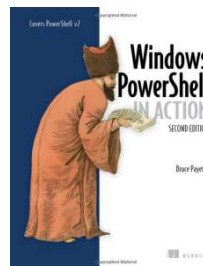
Quelques références

- Version CHM de l'aide PowerShell → <http://www.microsoft.com/en-us/download/details.aspx?id=10552>
- Script center → <http://technet.microsoft.com/en-us/scriptcenter/>
- Blog Scripting Guy → <http://blogs.technet.com/b/heyscriptingguy/>
- Livres ->

- Windows PowerShell Guide de référence →



- PowerShell in Action →



Présentation de la formation

3 grandes parties



Chapitres



Modules

Présentation de la formation

1. L'initiation au PowerShell 2.0

Chapitre 1 : Introduction à PowerShell

1. Présentation de la formation
2. Introduction générale à Windows PowerShell

Chapitre 2 : L'apprentissage de PowerShell

1. Introduction au langage
2. Opérateurs et expressions
3. La gestion du flux (l'utilisation du pipe)
4. Importer des données depuis des supports externes.

Présentation de la formation

- Partie 2 : L'utilisation avancé de PowerShell 2.0
 - Chapitre 3 L'utilisation avancé de PowerShell : Le scripting
 1. Les fonctions et les filtres
 2. La gestion d'erreur
 3. Les fonctions avancées
 - Chapitre 4 : La maitrise de PowerShell en travaillant à grande échelle
 1. Les modules
 2. L'exécution à distance
 3. L'exécution en tâche de fond : "Les jobs"

Présentation de la formation

Partie 3 : Maîtrise de PowerShell 2.0

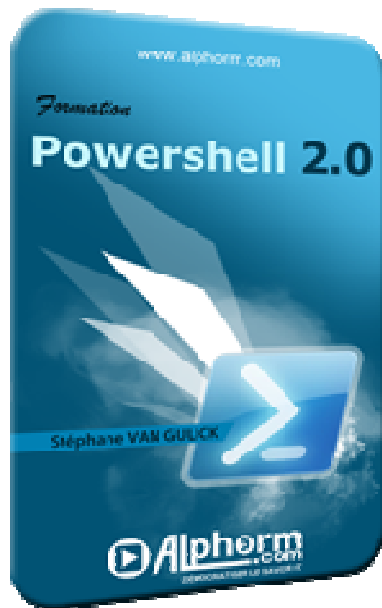
Chapitre 5: Créer ses propres cmdlets

- Le WMI
- L'automatisation d'applications (Com objects)
- L'exploitation du dotnet

Conclusion

 Are you ready ? 😊

Let's GO !



Initiation à PowerShell

Introduction à Windows PowerShell

Introduction générale à Windows PowerShell

Site : <http://www.alphorm.com>

Blog : <http://www.ConfigMgrdistrict.com>

Blog : <http://www.Powershelldistrict.com>

Forum : <http://www.alphorm.com/forum>

Stéphane van Gulick

Consultant systèmes et automatisation

Certifications : MCT, MCITP, MCSA

Contact : svangulick@alphorm.com

Twitter : @Stephanevg

Plan

1. Qu'est-ce PowerShell ?
2. Les différents versions existantes de PowerShell.
3. Prérequis nécessaire pour l'installation / l'utilisation de PowerShell.
4. Présentation du prompt, et de l'environnement de Scripting intégré.
5. Découverte des premières commandes.



Les avantages à utiliser Windows PowerShell ?

- Automatisation de tâches récurrentes / répétitives.
- Réduit le temps d'effort nécessaire pour l'accomplissement de cette tâche.
- Permet de paralléliser plusieurs tâches à la fois.
- Réduit la probabilité d'erreur.
- Possibilité de déléguer une tâche plus facilement.
- Est idéal pour la réutilisabilité.
- Permet d'aller au-delà des limites des interface graphiques.

Présence de Windows PowerShell

Système d'exploitation	PowerShell 2.0
Windows XP	PowerShell 2.0 (Facultatif)
Windows Server 2003	PowerShell 2.0 (Facultatif)
Windows Vista	PowerShell 2.0 (Facultatif)
Windows 7	PowerShell 2.0 (Natif)
Windows Server 2008	PowerShell 2.0 (Natif)
Windows 8	PowerShell 3.0 (Natif)
Windows Server 2012	PowerShell 3.0 (Natif)
Windows 8.1	PowerShell 4.0 (Natif)
Windows Server 2012 R2	PowerShell 4.0 (Natif)

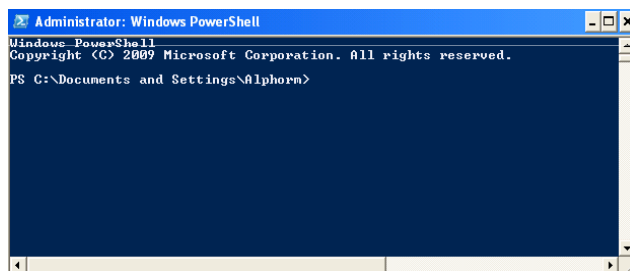


Prérequis à l'installation de Windows PowerShell

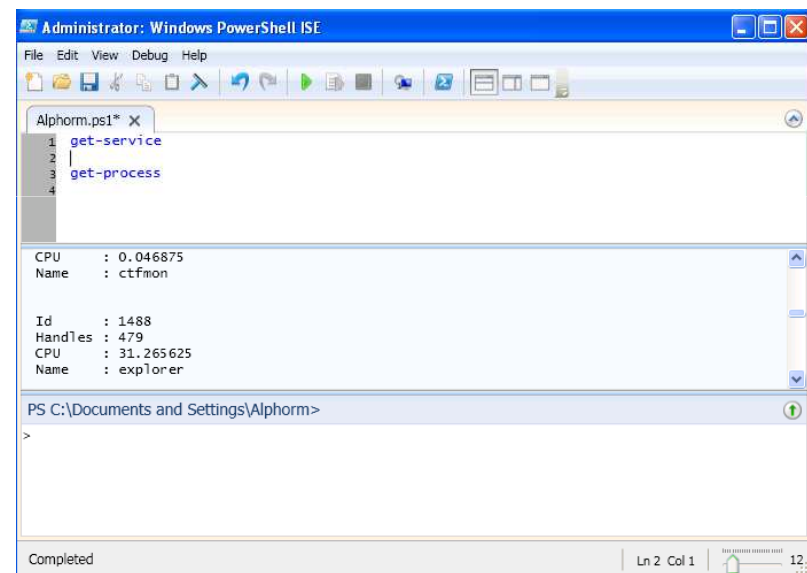
Powershell	Windows Management framework	Prérequis	Lien
2.0	WMF 2.0	Dotnet 2.0	http://support.microsoft.com/kb/968929/fr
3.0	WMF 3.0	Dotnet 3.5	http://support.microsoft.com/kb/2506143/fr
4.0	WMF 4.0	Dotnet 4.5	http://www.microsoft.com/fr-fr/download/details.aspx?id=40855

Découverte de la console PowerShell + ISE

Prompt PowerShell



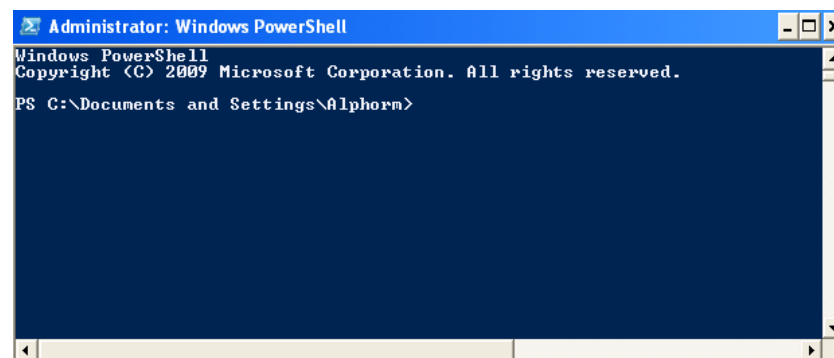
Console ISE



Prompt – Récapitulatif

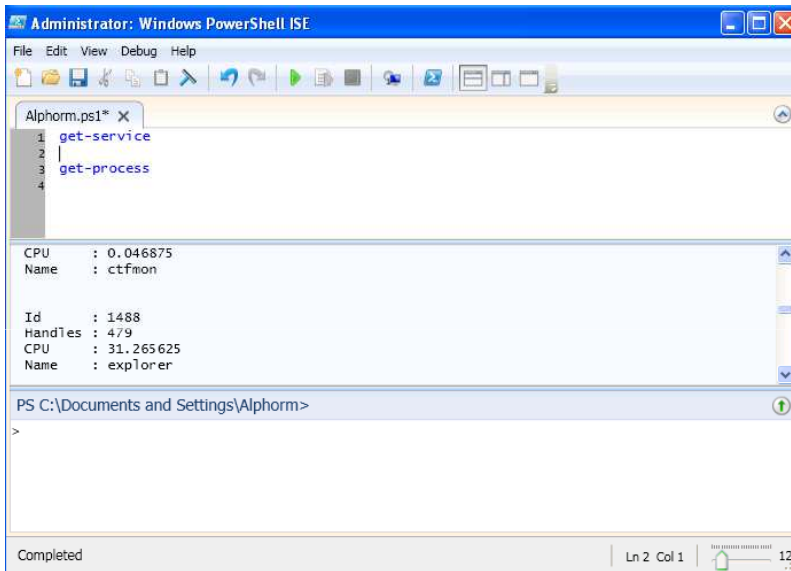
Commande	Résultat
F1	Réécrit la dernière commande lettre par lettre.
F3	Auto complète une commande.
F4	Supprime jusqu'au curseur.
F5	Remonte dans l'historique des commandes.
F7	Affiche l'historique des commandes. (CTRL+F7 l'efface)
F8	Auto complète votre ligne avec votre historique.
F9	Spécifie une ligne de l'historique précise (F7)
Tab / Shift + tab	Auto complète votre commande.
Flèche haut / bas	Navigue dans l'historique des commandes. (F7)
échappe	Efface la ligne entière.
CTRL + Flèche gauche / droite	Navigue sur la ligne du prompt mot par mot.

Commande	Résultat
Alt+ espace +E	Navigue vers menu édition.
CTRL+C	Cesse l'exécution de la commande en cours.
CTRL+S	Pause l'affichage en cours.
CTRL + end	Supprime tout depuis le curseur.



A screenshot of a Windows PowerShell console window titled "Administrator: Windows PowerShell". The window has a blue title bar and a dark blue background. The text inside the window reads: "Windows PowerShell", "Copyright (C) 2009 Microsoft Corporation. All rights reserved.", and "PS C:\Documents and Settings\Alphorm>". The cursor is at the end of the prompt line.

ISE – Récapitulatif



Administrator: Windows PowerShell ISE

File Edit View Debug Help

Alphorm.ps1* X

```
1 get-service
2
3 get-process
4
```

CPU : 0.046875
Name : ctfmon

Id : 1488
Handles : 479
CPU : 31.265625
Name : explorer

PS C:\Documents and Settings\Alphorm>

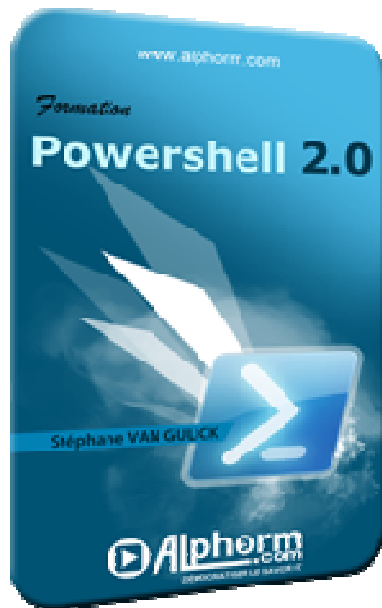
Completed | Ln 2 Col 1 | 12

Commande	Résultat
CTRL+O	Ouvre un script.
CTRL+N	Nouveau script.
CTRL+S	Sauvegarde le script.
CTRL+T	Nouvel onglet.
CTRL+W	Nouvel onglet réseau.
F5	Exécution du code en cours.
F8	Exécution d'une sélection du code.

Ce qu'on a couvert

1. Qu'est-ce PowerShell ?
2. Les différents versions existantes de PowerShell.
3. Prérequis pour nécessaire pour l'installation de PowerShell.
4. Présentation du prompt, et de l'environnement de Scripting intégré.
5. Navigation et raccourcis clavier.
6. Découverte des premières commandes de base.





Initiation à Windows PowerShell

Apprentissage de PowerShell

Introduction au langage

Site : <http://www.alphorm.com>

Blog : <http://www.ConfigMgrdistrict.com>

Blog : <http://www.PowerShelldistrict.com>

Forum : <http://www.alphorm.com/forum>

Stéphane van Gulick

Consultant systèmes et automatisation

Certifications : MCT, MCITP, MCSA

Contact : svangulick@alphorm.com

Twitter : @Stephanevg

Plan

- Vous connaissez déjà le PowerShell !
- Présentation de la structure et découverte premières commandes
- Comment trouver de l'aide ?
- Les variables



Le PowerShell vous connaissez déjà !

Les aliases
Dir
del
Pusd
Pwd
Cls

Structure des commandes

Verbe-nom
Get-help
Get-Command
Set-wmiObject
Move- Adcomputer
Read-host

Comment trouver de l'aide ?

- Get-help <nom de la commande>
- Get-command
- Get-member (gm)
- Internet : PowerShell + commande (ou action a réaliser)

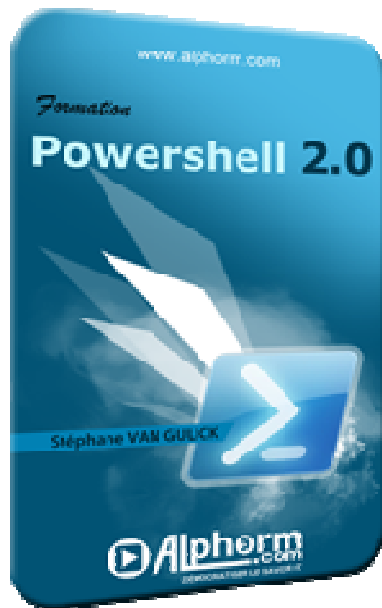
Les variables : Récapitulatif

assignation	Nom	Type	Exemple
<code>\$=<contenue></code>	String	[string] Texte	Bonjour
<code>\$=<commande></code>	Array	[array]Tableau	Bonjour, Bonsoir, salut
<code>\$=<Opération></code>	Integer	[int]Entier	1, 400, 1000, 14
Cmdlets	Double	[double]Double	14.456, 1.5, 78,4 etc...
Get-variable	Date	[date]Une date	11/10/2013, etc...
Set-variable	Hashtable	@{ }	Jean;25 Michel;23 Hamid;30
Clear-variable			
Remove-variable			
New-variable			

Ce qu'on a couvert

- Présentation de la structure et découverte des premières commandes
- Comment trouver de l'aide.
- Les variables.





Initiation à Windows PowerShell
Apprentissage de PowerShell

Opérateurs et expressions

Site : <http://www.alphorm.com>
Blog : <http://www.ConfigMgrdistrict.com>
Blog : <http://www.Powershelldistrict.com>
Forum : <http://www.alphorm.com/forum>

Stéphane van Gulick

Consultant systèmes et automatisation

Certifications : MCT, MCITP, MCSA

Contact : svangulick@alphorm.com

Twitter : @Stephanevg

Plan

- Opérateurs arithmétiques
- Opérateurs d'assignation
- Opérateurs de comparaison
- Opérateurs de traitement de texte
- Opérateurs logiques



Opérateurs Arythmétiques

Opérateur	Signification
+	Addition
-	Soustraction
/	Division
*	multiplication
%	Modulo

Opérateurs de d'assignation

Operateur	Signification
=	Assignation.
+=	Assignation en gardant la valeur precedente.
-=	Sous trait en reassignant le resultat.
/=	Divise en reassignant le resultat.
%=	Divise et assigne le modulo à la variable.

Opérateurs de comparaison

Opérateur	Signification	Variante sensible à la casse	Variante insensible à la casse
-eq	Equal	-ceq	-ieq
-ne	Not equal	-cne	-ine
-gt	Greather then	-cgt	-igt
-ge	Greather or equal	-cge	-ige
-lt	Less then	-clt	-ilt
-le	Less or equal	-cle	-ile

Les opérateurs de traitement de text

Opérateur	Signification	Variante sensible à la casse	Variante insensible à la casse
-like	comme	-clike	-ilike
-notlike	Pas comme	-cnotlike	-inotlike
-match	Correspond	-cmatch	-imatch
-notmatch	Correspond pas	-cnotmatch	-inotmatch

Opérateur	Signification
-replace	remplace
-join	concatène
-split	Sépare

Opérateur	Signification
-contains	Contient
-Notcontains	Ne contient pas

Les opérateurs logiques

Opérateur	Signification	Variante Bits
-and	et	-band
-or	ou	-bor
-not	non	-bnot
-xor	Ou exclusif	-bxor

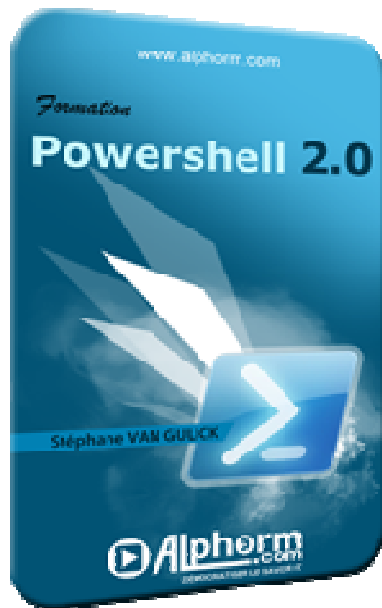
Ce qu'on a couvert

- Opérateurs arithmétiques
- Opérateurs d'assignation
- Opérateurs de comparaison
- Opérateurs de traitement de texte
- Opérateurs logiques

- Lien:

<http://www.cheatography.com/davechild/cheat-sheets/regular-expressions/>





Initiation à Windows PowerShell
Apprentissage de PowerShell
Les providers

Site : <http://www.alphorm.com>
Blog : <http://www.ConfigMgrdistrict.com>
Blog : <http://www.PowerShelldistrict.com>
Forum : <http://www.alphorm.com/forum>

Stéphane van Gulick

Consultant systèmes et automatisation

Certifications : MCT, MCITP, MCSA

Contact : svangulick@alphorm.com

Twitter : @Stephanevg

Plan

- Que sont les providers ?
- Pourquoi utiliser les providers ?
- Démonstration



Les providers

- Que sont les providers ?
 - C'est un adaptateur PowerShell. Cela permet d'utiliser certaines parties du système de la même manière que l'utilisation du FileSystem sous dos par exemple.
- Pourquoi utiliser les providers ?
 - Cela permet de naviguer et d'effectuer des opérations de base dans un système (comme le registre Windows) de la même manière que dans un système de fichier.
- Les commandes à utiliser dans les providers
 - Get-Item, Copy-Item, Remove-Item ...

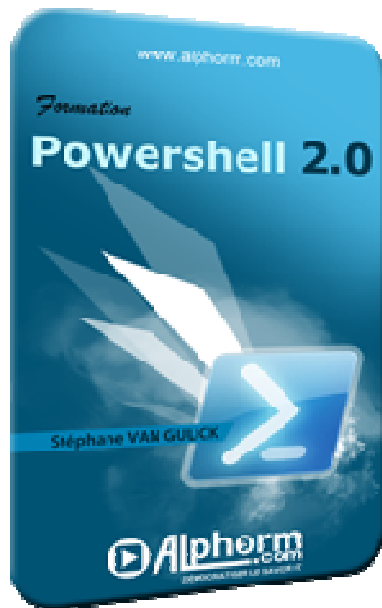
Les providers

- Démonstration :
 - Les providers disponibles
 - Les commandes à utiliser au sein des providers
 - Un exemple concret

Ce qu'on a couvert

- Pourquoi utiliser les providers
- Les providers disponibles
- Les Commandes
 - Get-Item, Clear-Item, Remove-Item
- Le chargement d'un module rajoute également un provider supplémentaire (Module Active Directory)





Initiation à Windows PowerShell
Apprentissage de PowerShell
La gestion du flux

Site : <http://www.alphorm.com>
Blog : <http://www.ConfigMgrdistrict.com>
Blog : <http://www.PowerShelldistrict.com>
Forum : <http://www.alphorm.com/forum>

Stéphane van Gulick

Consultant systèmes et automatisation

Certifications : MCT, MCITP, MCSA

Contact : svangulick@alphorm.com

Twitter : @Stephanevg

Plan

- Le pipe
 - La rédaction de « one liners » à l'aide du « Pipe »
- Les opérateur de conditions
- Les boucles
- Les switches
- Démonstration



Le pipe « | »

- A quoi ça sert ?
- Export vers des fichiers externes
- « Piper » des résultats vers d'autres commandes
 - (Permet la rédaction de « one liners »)
- Démonstration

Les opérateurs de condition

Les opérateurs de conditions	Signification
If	Si
Elseif	sinon si
else	sinon

- Démonstration

Les boucles

Les boucles	Signification
While	Tant que
Do while	Fait tantque
Do until	Fait jusqu'à
For	Pour
Foreach	Pour chaque

- Démonstration

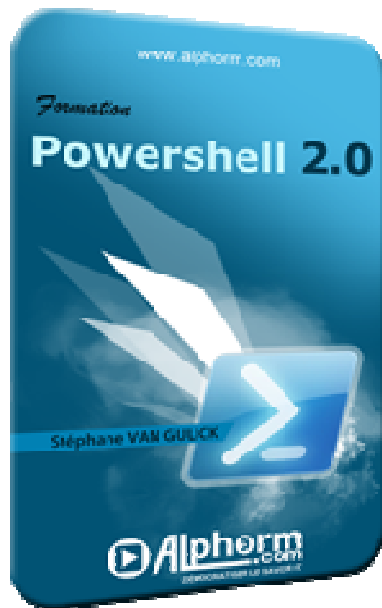
Les switches

- Dans quel cas utiliser un switch
- La structure du switch
- Démonstration

Ce qu'on a couvert

- Le pipe
 - La rédaction de « one liners » à l'aide du « Pipe »
- Les opérateur de conditions
- Les boucles
- Les switches





Initiation à Windows PowerShell

Apprentissage de PowerShell

Import de données externes

Site : <http://www.alphorm.com>

Blog : <http://www.ConfigMgrdistrict.com>

Blog : <http://www.PowerShelldistrict.com>

Forum : <http://www.alphorm.com/forum>

Stéphane van Gulick

Consultant systèmes et automatisation

Certifications : MCT, MCITP, MCSA

Contact : svangulick@alphorm.com

Twitter : @Stephanevg

Plan

- Pourquoi importer des données ?
- Import/export depuis différents supports:
- Fichiers textes
- Fichiers CSV
- Fichiers XML
- Export HTML



Pourquoi importer/exporter des données ?

- Import:
 - Pour faire un travail de moyen / grande échelle.
 - Traiter des données fournies par d'autres services.
 - Afin de garantir une base de travail stable.
- Export:
 - Exporter des rapports.
 - Une nouvelle base de travail.
 - Etc...

Fichiers textes:

- Import
 - Get-content
- Export
 - Out-file (>, >>)
 - Possibilité de lire le fichier pendant que Out-file l'utilise)
 - Set-content
 - Attendre la fin de set-content pour pouvoir lire le fichier
 - Add-content

Fichiers CSV:

- Import
 - Import-csv
- Export
 - Export-csv
- Dans le Shell
 - ConvertTo-CSV
 - ConverTFrom-CSV

Fichiers XML:

- Import
 - Import-CLIXML (Objets exportés avec Export-CliXml)
 - [xml]\$ContenueXml = Get-content -path \$FichierXml
- Export
 - Export-CliXML
 - \$variable.save(C:\alpormation\configuration\Datas.xml)
- Dans le Shell
 - ConvertTo-XML

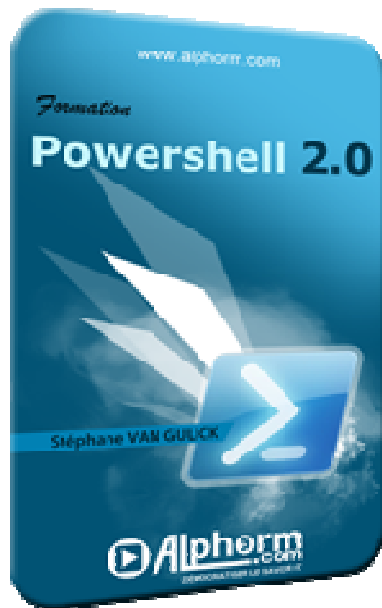
Export-html

- Export
 - ExportTo-HTML
- Modifier le style (Css)
 - Head
 - Body

Ce qu'on a couvert

- Import/export depuis différents supports:
 - Fichiers textes.
 - Fichiers CSV.
 - Fichiers XML
 - Fichiers HTML
- A retenir :
 - Caster : `[XML]$MaVariable = Get-Content -path c:\fichier.xml`
 - « Rapports » `ConvertTo-Html`





L'utilisation avancée de Windows PowerShell
Le scripting

Les fonctions et les filtres

Site : <http://www.alphorm.com>
Blog : <http://www.ConfigMgrdistrict.com>
Blog : <http://www.Powershelldistrict.com>
Forum : <http://www.alphorm.com/forum>

Stéphane van Gulick

Consultant systèmes et automatisation

Certifications : MCT, MCITP, MCSA

Contact : svangulick@alphorm.com

Twitter : @Stephanevg

Plan

- Qu'est ce qu'une fonction ?
- Qu'est ce qu'un filter ?
- Les différences entre une fonction et un filter.
- Quand utiliser une fonction, quand utiliser un filter ?



Les fonctions : La théorie

- Utilité:
 - S'exécute seulement une fois que tous les éléments du pipe sont chargés.
 - Permet d'écrire et de nommer un bout de code qui peut être rappelé à tout moment.
 - Permet d'effectuer une action qui va ou non, retourner une valeur / objet
 - Gain d'espace au sein du code.
 - Organisation du code (Clarté du code).
 - Gain de temps lors du codage (Réutilisabilité) .

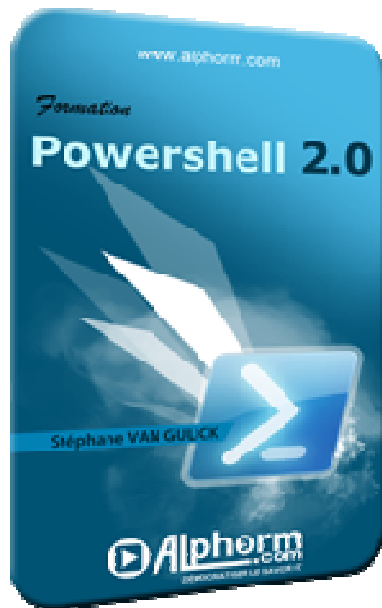
Les Filters : La théorie

- Utilité
 - Commence à exécuté du code dès qu'elle reçoit du contenu
 - S'utilise de la même manière qu'une fonction.
 - A exactement les même avantages qu'une fonction.
 - Structure du code
 - Gain de place, gain de temps, réutilisation de code etc..
- Démonstration

Ce qu'on a couvert

- Qu'est ce qu'une fonction ?
- Qu'est ce qu'un filter ?
- Les différences entre une fonction et un filter
- Quand utiliser une fonction, quand utiliser un filter ?





L'utilisation avancée de Windows PowerShell

Le scripting

La gestion d'erreur

Site : <http://www.alphorm.com>

Blog : <http://www.ConfigMgrdistrict.com>

Blog : <http://www.PowerShelldistrict.com>

Forum : <http://www.alphorm.com/forum>

Stéphane van Gulick

Consultant systèmes et automatisation

Certifications : MCT, MCITP, MCSA

Contact : svangulick@alphorm.com

Twitter : @Stephanevg

Introduction

```
Administrator: Windows PowerShell

PS E:\Users\Administrator> dir fichierexistepas
dir : Cannot find path 'E:\Users\Administrator\fichierexistepas' because it does not exist.
At line:1 char:1
+ dir fichierexistepas
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (E:\Users\Administrator\fichierexistepas:String) [Get-ChildItem], ItemNotFou
ndException
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.GetChildItemCommand

PS E:\Users\Administrator> copy fichierexistepas
copy : Cannot find path 'E:\Users\Administrator\fichierexistepas' because it does not exist.
At line:1 char:1
+ copy fichierexistepas
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (E:\Users\Administrator\fichierexistepas:String) [Copy-Item], ItemNotFou
ndException
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.CopyItemCommand

PS E:\Users\Administrator> gt-item
gt-item : The term 'gt-item' is not recognized as the name of a cmdlet, function, script file, or operable program.
Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ gt-item
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (gt-item:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS E:\Users\Administrator> get-contant fichier
get-contant : The term 'get-contant' is not recognized as the name of a cmdlet, function, script file, or operable
program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ get-contant fichier
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (get-contant:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS E:\Users\Administrator> del encoreunerreur!
del : Cannot find path 'E:\Users\Administrator\encoreunerreur!' because it does not exist.
At line:1 char:1
+ del encoreunerreur!
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (E:\Users\Administrator\encoreunerreur!:String) [Remove-Item], ItemNotFo
undException
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.RemoveItemCommand

PS E:\Users\Administrator>
```

Plan

- ExecutionPolicy
- Les différents types d'erreurs.
- Le comportement de PowerShell face à une erreur.
- Le détail d'une erreur.
- Prévoir les erreurs, et agir en conséquence.



ExecutionPolicy

Valeur	Comportement lors de la rencontre d'une erreur
Resitricted	Impossible d'exécuter des scripts. Possibilité d'utiliser le prompt uniquement (mode interactif).
AllSigned	Seulement les scripts qui sont signés peuvent être exécutés.
Remote signed	Les scripts téléchargés doivent être signés avant d'être exécutés.
unrestricted	Pas de restrictions. (demande de confirmation avant exécution d'un script).
Bypass	Aucune restriction, pas de demande de confirmation.

Les différent types d'erreurs

- Les erreurs non-critiques (Non-Terminating).
- Les erreurs critiques (Terminating).

Le comportement de PowerShell face à une erreur

- La variable `$erroractionpreference`

Valeur	Comportement lors de la rencontre d'une erreur
SilentlyContinue	Le script continue son exécution sans affichage.
Continue	Le script continue son exécution, tout en affichant l'erreur.
Stop	Le script s'interrompt (Erreur critique).
Inquire	Demande comment ce comporter.

Le détail d'une erreur.

- Le contenu de la variable `$error` .

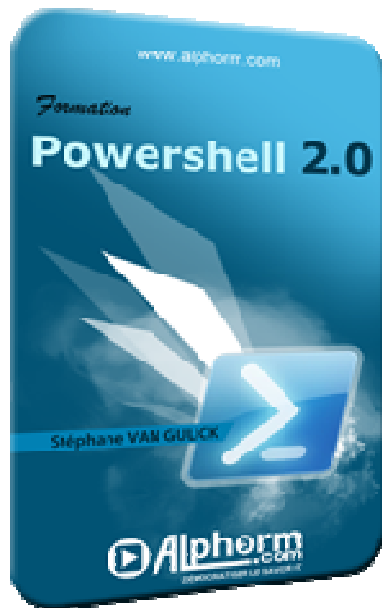
Prévoir les erreurs, et agir en conséquence

- \$?
- Trap {}
- Try {}, Catch {}, Finally {}
- Throw {}

Ce qu'on a couvert

- Les erreurs critiques et non-critiques
- Les différentes variables d'erreurs:
 - \$erroractionpreference
 - \$error
 - \$?
- Ce que contient finalement une erreur → c'est un objet !
- Comment prévoir une erreur
 - try, catch, finally
 - Throw





L'utilisation avancée de Windows PowerShell

Le scripting

Les fonctions avancées

Site : <http://www.alphorm.com>

Blog : <http://www.ConfigMgrdistrict.com>

Blog : <http://www.PowerShelldistrict.com>

Forum : <http://www.alphorm.com/forum>

Stéphane van Gulick

Consultant systèmes et automatisation

Certifications : MCT, MCITP, MCSA

Contact : svangulick@alphorm.com

Twitter : @Stephanevg

Plan

- Qu'est ce qu'une fonction avancé?
- Les « CmdletBinding » attributs
- Les « Parameter » attributs
- Documenter notre fonction



Les fonctions avancées: La théorie

- Qu'est ce qu'une fonction avancé?
 - Identique à une fonction « normal » sauf qu'elle permet plus de contrôle
 - Intégration de fonctionnalités natives tel que le –Verbose ou encore le -debug
 - Control des paramètres
 - Permet la validation des paramètres
 - Création d'aide
 - Intégration d'aliases
 - Permettre le paramètre de provenir du « pipe »

Les « CmdletBinding » attributs

- Utilité
 - Doit se mettre toujours avant la section Param()
 - Permet l'activation des fonctionnalités natives tel que le -verbose, -whatif, -debug, -confirm
- Démonstration.

Les « Parameter » attributs

- Utilité
 - Permet de mettre de conditions de validation, et de control sur les différents paramètres de nos fonctions
- Mandatory
 - Force ou non qu'un valeur soit passé pour ce paramètre
- Position
 - Permet de définir une position fixe pour un certain paramètre
- ValueFromPipeLine
 - Permet de d'accepter ou non, qu'un paramètre provienne du pipe

Les « Parameter » attributs (suite)

- ValueFromPipeLineByPropertyName
 - Permet la de passer des paramètre via le pipe portant le même nom (exemple : « ComputerName »)
- ValueFromRemainingArguments
 - Permet d'utiliser ou non les arguments restant contenue dans le pipe
- HelpMessage
 - Permet de spécifier un message d'aide.
- ValidateSet
- ValidateScript

Documenter notre fonction

- Les commentaires
 - L'utilisation du signe « # »
 - L'utilisation d'un champ de commentaire plus large : « <# « code » #> »

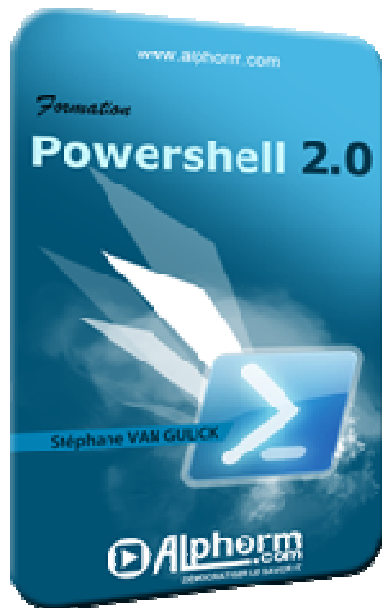
Le « comment-based help »

- Comment créer une aide « professionnel »
 - Il s'agit de mettre des tags à l'intérieur d'un champ de commentaire <# #>
 - .SYNOPSIS: Permet de spécifier en une phrase courte l'utilité du script ou de la fonction
 - .DESCRIPTION: La description est généralement plus long. Elle permet d'écrire ce que le script fait en détail
 - .PARAMETER NomDuParametre: Permet d'afficher des information concernant les paramètres de notre fonction (exemple : Computername)
 - .EXAMPLE: Permet de spécifier un exemple précis concernant l'utilisation de la fonction / script.

Ce qu'on a couvert

- Qu'est ce qu'une fonction avancé?
- Les « CmdletBinding » attributs
- Les « Parameter » attributs
- Documenter notre fonction
 - Les commentaires
 - Le comment-based help





L'utilisation avancée de Windows PowerShell

Le travail à grande échelle

Les modules

Site : <http://www.alphorm.com>

Blog : <http://www.ConfigMgrdistrict.com>

Blog : <http://www.PowerShelldistrict.com>

Forum : <http://www.alphorm.com/forum>

Stéphane van Gulick

Consultant systèmes et automatisation

Certifications : MCT, MCITP, MCSA

Contact : svangulick@alphorm.com

Twitter : @Stephanevg

Plan

- Qu'est-ce un module PowerShell ?
- Les différents types de modules existants.
- Comment créer son propre module.
- Les Modules manifest.
- Comment créer son propre manifest.



Qu'est ce un module ?

- Permet de regrouper un grand nombre de fonctions dans un même endroit.
- Facilite la distribution.
- Facilite l'accès à des nouvelles fonctionnalités qui ne sont pas toujours nécessaires.
- Facilite le partage.
- Permet la standardisation de fonctions de base.

Les différents types de modules existants

- Binary modules
 - Code écrit en C#
 - Extension .DLL
- Script modules
 - Basés sur des fichiers textes contenant des fonctions
 - Extension .PSM1

Les commandes de base

- Get-module
- Import-module
- Remove-module

- Démonstration

Comment créer un script module

- Fichier text
- Fonctions
- Extension .psm1

- Démonstration

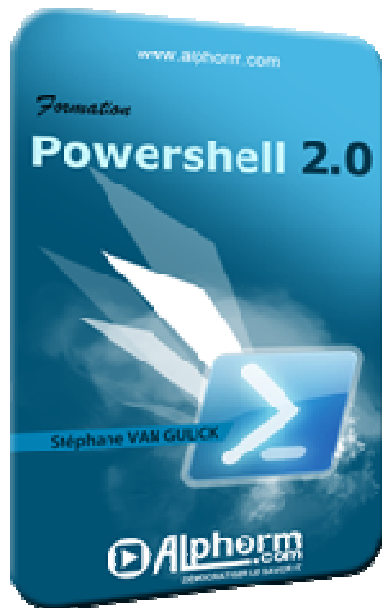
Les modules manifest

- Qu'est un module manifest ?
 - A été créer pour pouvoir créer de modules de manière pouvoir les distribuer a grande échelle
 - Contiendras les informations tel que : L'auteur, le copyright, Nom de l'entreprise, version de PowerShell a utiliser etc...
 - C'est un fichier avec l'extension .psd1
- Démonstration

Ce qu'on a couvert

- Qu'est-ce un module PowerShell ?
- Les différents types de modules existants
- Les Modules manifest





L'utilisation avancée de Windows PowerShell

Le travail à grande échelle

L'exécution à distance

Site : <http://www.alphorm.com>

Blog : <http://www.ConfigMgrdistrict.com>

Blog : <http://www.PowerShelldistrict.com>

Forum : <http://www.alphorm.com/forum>

Stéphane van Gulick

Consultant systèmes et automatisation

Certifications : MCT, MCITP, MCSA

Contact : svangulick@alphorm.com

Twitter : @Stephanevg

Plan: L'exécution à distance

- Les prérequis
- Les commandes offrant nativement des fonctionnalités d'exécution à distance.
- Les connexions Non-Persistantes
- Les connexions Persistantes Interactives
- Les connexions Persistantes Non-interactives



Les prérequis

- Windos remote management (WINRM) → Désactivée par défaut.
 - Web Services for management (WS-MAN)
 - HTTP, HTTPS (WINRM 2.0 → 5985)
- Par défaut, il est possible pour toute machine équipée de PowerShell 2.0 d'initier des connections distantes.
- Certaines conditions doivent être respectées pour pouvoir accepter ses connections.

Les prérequis – Mise en place

- Une machine en workgroup
 - Enable-PsRemoting
 - Active le service WinRM, Intègre Set-WSMANQuickConfig
- Déploiement en grande échelle(GPO)
 - Computer Configuration->Policies->Administrative templates->Windows Components
 - Configuration → Policies → WindowsSettings → SecuritySettings → SystemServices
Windows → Windows remote management = Automatic
- TOUJOURS ouvrir en tant que "Administrator"

Les commandes et le remoting

- Beaucoup de commandes permettent déjà de faire de l'exécution à distance via "-computename"
- Démonstration

Cmdlet avec -computerName

- Get-WinEvent
- Get-Counter
- Get-EventLog
- Clear-EventLog
- Write-EventLog
- Limit-EventLog
- Show-EventLog
- New-EventLog
- Remove-EventLog
- Remove-EventLog
- Remove-EventLog
- Get-WmiObject
- Get-Process
- Get-Service
- Set-Service
- Get-HotFix
- Restart-Computer
- Stop-Computer
- Add-Computer
- Remove-Computer
- Rename-Computer
- Reset-ComputerMachinePassword



Les connections Non-Persistantes Non-interactive

- Invoke-command
- L'exécution d'une seule commande voir d'un script à distance.
- Fonctionne en 3 étapes:
 - Établissement de le connexion
 - Envois de la commande / script
 - Fermeture de la connexion
- Démonstration



Les connections Non-Persistantes Non-interactives

- Les inconvénients de `invoke-command`
 - Est peut être simple d'utilisation mais la création de connexion, vérification de version etc est lourd pour l'os.
 - Ne permet pas de travailler de manière interactive
 - Impossibilité de garder des variables / fonctions chargées en mémoire
- Avantages
 - Permet de deployer un script /commandes sur plusieurs machines distantes

Connection Persistante interactive

- Possibilité de créer une session, executer des commandes, scripts, de quitter la session, puis de reprendre la session la ou elle s'était arrêter.
- Enter-PsSession
- Démonstration

Connexion Persistante Non-interactive

- Ouvrir et fermer la connexion après chaque exécution de commande / script est lourd.
- Permet de créer une connexion persistante
- Elle est non-interactive (ne permet pas de travailler sur le retour des commandes)
- New-PsSession
- Démonstration

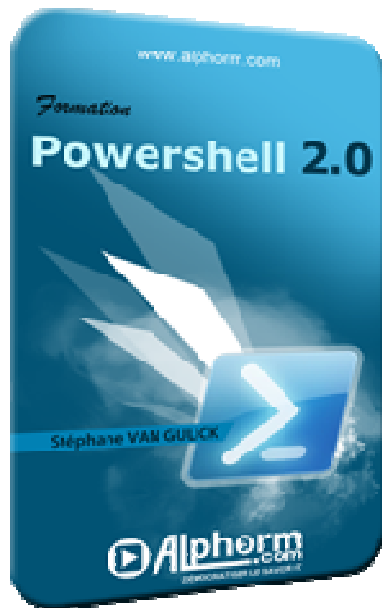
Troubleshooting

- Service WINRM toujours sur automatic
 - Par défaut pour OS serveurs.
 - Pas pour OS clients.

Ce qu'on a couvert

- Les prérequis
- Les commandes offrant nativement des fonctionnalités d'exécution à distance.
- Les connexions Non-Persistantes
- Les connexions Persistantes Non-interactive
- Les connexions Persistantes Interactive





L'utilisation avancée de Windows PowerShell

Le travail à grande échelle

L'exécution en tâche de
fond : Les jobs

Site : <http://www.alphorm.com>

Blog : <http://www.ConfigMgrdistrict.com>

Blog : <http://www.PowerShelldistrict.com>

Forum : <http://www.alphorm.com/forum>

Stéphane van Gulick

Consultant systèmes et automatisation

Certifications : MCT, MCITP, MCSA

Contact : svangulick@alphorm.com

Twitter : @Stephanevg

Plan

- Introduction
- Un « job », c'est quoi ?
- Les commandes
 - Commandes natives
 - Cmdlets PowerShell



Introduction : Dir C:\ -recurse

```
Windows PowerShell

Mode                LastWriteTime         Length Name
-----
d-----          13.09.2013         14:57      Files
d-----          13.09.2013         14:57      Folder

Directory: C:\Program Files\Idera\PowerShellPlus\Internal\Snippets\EditorSnippets\Files and Folders\Files

Mode                LastWriteTime         Length Name
-----
d-----          13.09.2013         14:57      Text
-a---          18.01.2013         15:55    1066 zip1.snippet

Directory: C:\Program Files\Idera\PowerShellPlus\Internal\Snippets\EditorSnippets\Files and Folders\Files\Text

Mode                LastWriteTime         Length Name
-----
-a---          18.01.2013         15:55      828 appendtext.snippet

Directory: C:\Program Files\Idera\PowerShellPlus\Internal\Snippets\EditorSnippets\Files and Folders\Folder

Mode                LastWriteTime         Length Name
-----
-a---          18.01.2013         15:55      818 renamefolder.snippet

Directory: C:\Program Files\Idera\PowerShellPlus\Internal\Snippets\EditorSnippets\Functions

Mode                LastWriteTime         Length Name
-----
d-----          13.09.2013         14:57      Prompt
-a---          18.01.2013         15:55      764 funcbasic.snippet
-a---          18.01.2013         15:55      784 funclistsourcesnippet

Directory: C:\Program Files\Idera\PowerShellPlus\Internal\Snippets\EditorSnippets\Functions\Prompt

Mode                LastWriteTime         Length Name
-----
-a---          18.01.2013         15:55      782 promptadvanced1.snippet
-a---          18.01.2013         15:55      625 promptcomputer.snippet
```

Un job, c'est quoi ?

- Cela va nous permettre de faire d'exécuter des commandes en arrière plan, et nous libérer le prompt immédiatement.
- Très utile lorsqu'on utilise des commandes / scripts qui ont besoin de beaucoup de temps pour finir

.

Les commandes

- 3 façons de créer un 'job'
 - Avec `-AsJob`
 - `Get-WmiObject -Asjob` (Execute toujours la commande l'un apres l'autre)
 - `Invoke-command -AsJob`
- Cmdlet job
 - `Start-job -ScriptBlock{...}`

Utilisation:

- Récupération des résultats:
- Get-job
 - Retourne le statut actuel des jobs.
- Receive-job
 - Récupère le résultat du job.
 - L'option `-keep` Permet de garder le retour de la console (Par défaut le retour est supprimé).

Utilisation:

- Actions sur un job :
 - Get-job
 - Remove-job
 - Supprime un job.
 - Wait-job
 - Attends que le job ce termine.
 - Receive-job
 - Récupère le résultat.
 - Démonstration

Les jobs enfants

- `Get-job -id <JobId> | select-object -expand childjobs`
- Démonstration

Gérer les jobs

- Il est possible de gerer les jobs en identifiant les jobs parmi une de identification suivantes :
 - N° ID
 - Nom du job
 - Piper directement le job vers une des commande de gestion
- Commandes disponibles
 - Remove-job
 - Stop-Job
 - Wait-job

Ce qu'on a couvert

- Synchrone / Asynchrone
 - Différences
- Les jobs fonctionnent en 3 étapes
 - Lancement du job
 - Start-job
 - Récupérer les résultats / les états des jobs
 - Receive-job
 - Gérer les jobs
 - Get-job, wait-job, remove-job, Stop-Job





La maîtrise de Windows PowerShell
Créer ses propres cmdlets

Le WMI

Site : <http://www.alphorm.com>
Blog : <http://www.ConfigMgrdistrict.com>
Blog : <http://www.Powershelldistrict.com>
Forum : <http://www.alphorm.com/forum>

Stéphane van Gulick

Consultant systèmes et automatisation

Certifications : MCT, MCITP, MCSA

Contact : svangulick@alphorm.com

Twitter : @Stephanevg

Plan

- Qu'est ce que le WMI ?
- A quoi ça vous servir ?
- Quels sont les prérequis ?
- Quelques outils indispensables pour gérer le WMI
- L'architecture WMI en quelques mots
- PowerShell et le WMI



Qu'est ce que le WMI ?

- Windows Management Interface
 - Est l'implémentation Microsoft du modèle de données CIM (Common information Model)
 - C'est une modèle de données qui a été créer dans le but de pouvoir manager de manière universelle des systèmes divers (PC, Servers, switch, applications system d'exploitations etc..)
 - WMI (CIM) permet d'interagir avec n'importe quel système de manière standardisé.
 - Existe depuis l'implémentation depuis Windows 2000 sp2
 - Est devenue très facilement accessible avec l'arrivé de PowerShell

A quoi ça vous servir ?

- Avoir un control sur les éléments du systèmes offert par des 'providers' tel qu'une application, la base de registre, ou encore le système d'exploitation.
- Avoir accès a des informations / méthodes supplémentaires de l'OS.
- Exemple :
 - Access informations du BIOS
 - Possibilité créer de partages

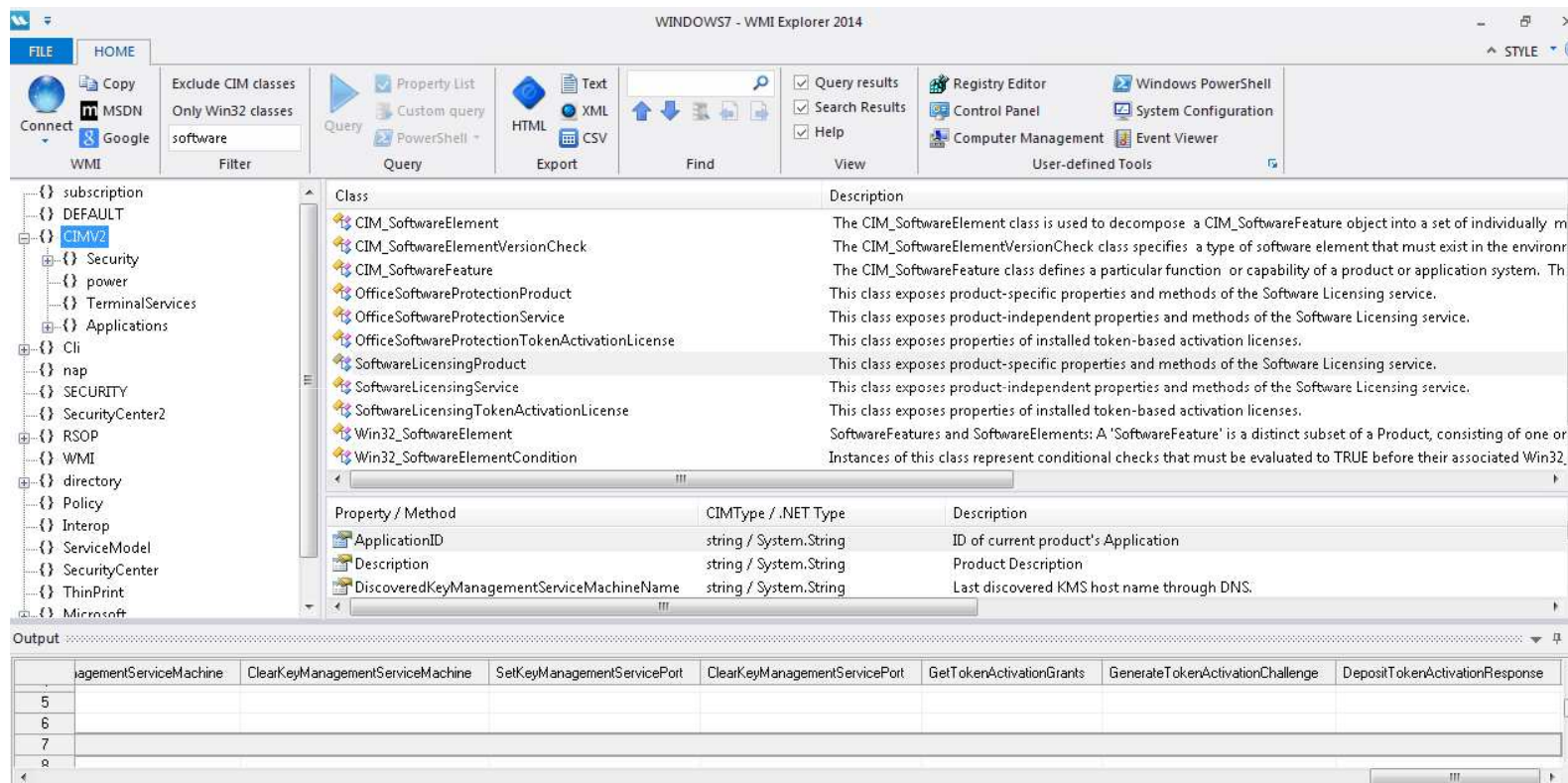
Quels sont Les prérequis ?

- Aucun !
 - Si ce n'est qu'il faut avoir les bons outils pour pouvoir explorer WMI

Les outils pour explorer le WMI

- Ils ont tous des noms surprenant !
 - WMI explorer
 - <http://www.ks-soft.net/hostmon.eng/downpage.htm>
 - WMI explorer
 - Sapien
 - WMI explorer... (fait par The powerShell Guy (script powerShell))
 - <http://jdhitsolutions.com/blog/2013/03/wmi-explorer-from-the-powershell-guy/>
 - Ou encore... WMI explorer
 - SolarWinds http://www.solarwinds.com/products/freetools/wmi_monitor/

WMIExplorer2014 → www.sapien.com



L'architecture WMI en quelques mots

- Les 'providers' offrent des 'classes' dans les quels ce trouvent des 'instances'.
 - Chaque instance a des propriétés (Nom, Description)
 - Et peut avoir des 'méthodes' tel que Stop, start delete etc..
- Démonstration de WMI explorer et de l'architecture WMI

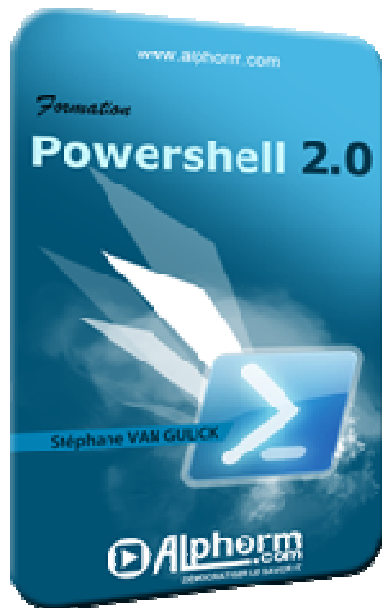
PowerShell et le WMI

- Les commandes PowerShell disponibles permettant de gérer le WMI
 - Get-WMIObject
 - Retourne des objects de WMI
 - Set-WmiInstance
 - Permet de mettre des valeurs a des propriétés WMI
 - Invoke-WMIMethod
 - Permet d'appeler une méthode (WMI)
 - Remove-WmiObject
 - Permet de supprimer une instance créer de la base WMI
 - **Démonstration**

Ce qu'on a couvert

- Qu'est ce que le WMI ?
- A quoi ça vous servir ?
- Quels sont Les prérequis ?
- Quelques outils indispensables pour gérer le WMI
- L'architecture WMI en quelques mots
- PowerShell et le WMI
 - Les commandes WMI
 - .put()





La maîtrise de Windows PowerShell
Créer ses propres cmdlets

L'automatisation d'applications :
Les objets COM

Site : <http://www.alphorm.com>
Blog : <http://www.ConfigMgrdistrict.com>
Blog : <http://www.PowerShelldistrict.com>
Forum : <http://www.alphorm.com/forum>

Stéphane van Gulick

Consultant systèmes et automatisation

Certifications : MCT, MCITP, MCSA

Contact : svangulick@alphorm.com

Twitter : @Stephanevg

Plan

- Qu'est ce qu'un objet COM ?
- A quoi ça vous servir ?
- PowerShell et les objets COM



Qu'est ce qu'un objet COM ?

- Un objet COM est une interface offerte par les fabricant de logiciels afin d'offrir une possibilité d'automatisation sur leurs produits

A quoi ça vous servir ?

- Les objets COM va permettre d'automatiser des actions au sein de programmes connues tel que Word, Internet explorer ou bien encore de l'explorateur de fichier.

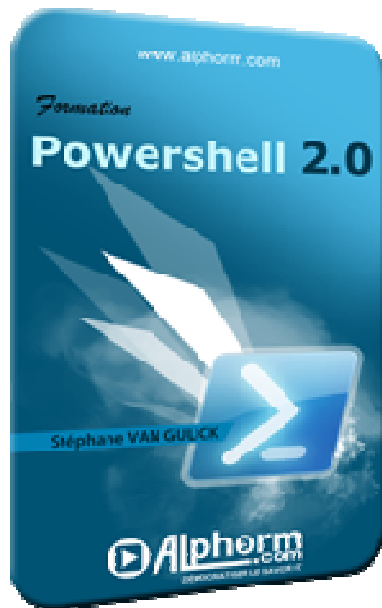
PowerShell et les objets COM

- Comment identifier les objets COM existants ?
 - Registre
 - WMI
- Comment utiliser un objet COM ?
 - `$Variable = New-Object -ComObject <NomComObjet>`
 - `$Variable | Get-Member`
- Démonstration

Ce qu'on a couvert

- Qu'est ce qu'un objet COM ?
- A quoi ça vous servir ?
- PowerShell et les objets COM





La maîtrise de Windows PowerShell
Créer ses propres cmdlets
L'exploitation du DotNet

Site : <http://www.alphorm.com>
Blog : <http://www.ConfigMgrdistrict.com>
Blog : <http://www.Powershelldistrict.com>
Forum : <http://www.alphorm.com/forum>

Stéphane van Gulick

Consultant systèmes et automatisation

Certifications : MCT, MCITP, MCSA

Contact : svangulick@alphorm.com

Twitter : @Stephanevg

Plan

- Introduction
- Pourquoi utiliser le .NET
- Le vocabulaire
- Comment instancier des objets
- Notre premier fonction basée sur DotNet



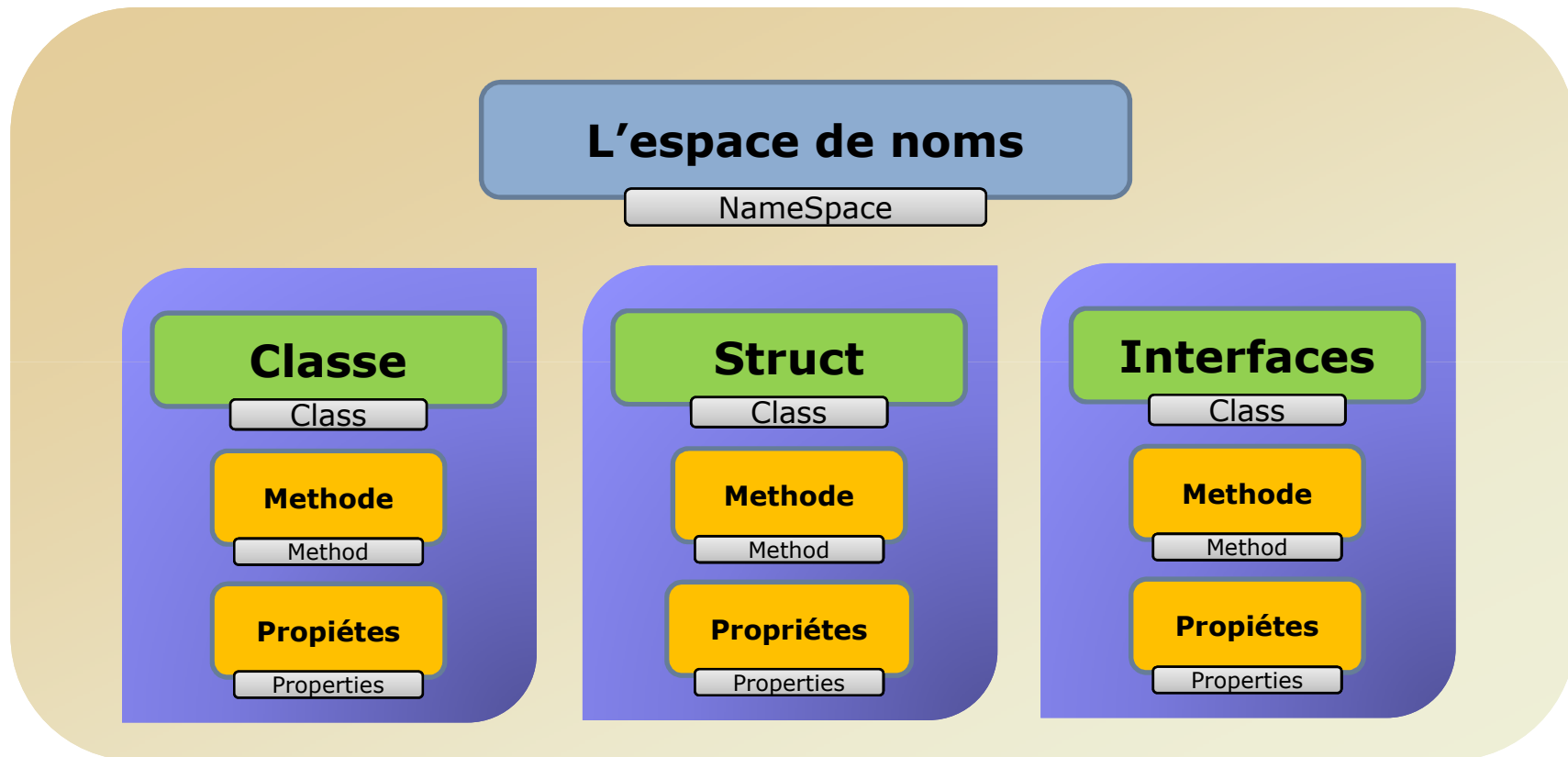
Introduction

- Module difficile
- Il y a beaucoup de vocabulaire liée au développement. Ceci dépasse un peu les compétences de base d'un simple « scripteur »

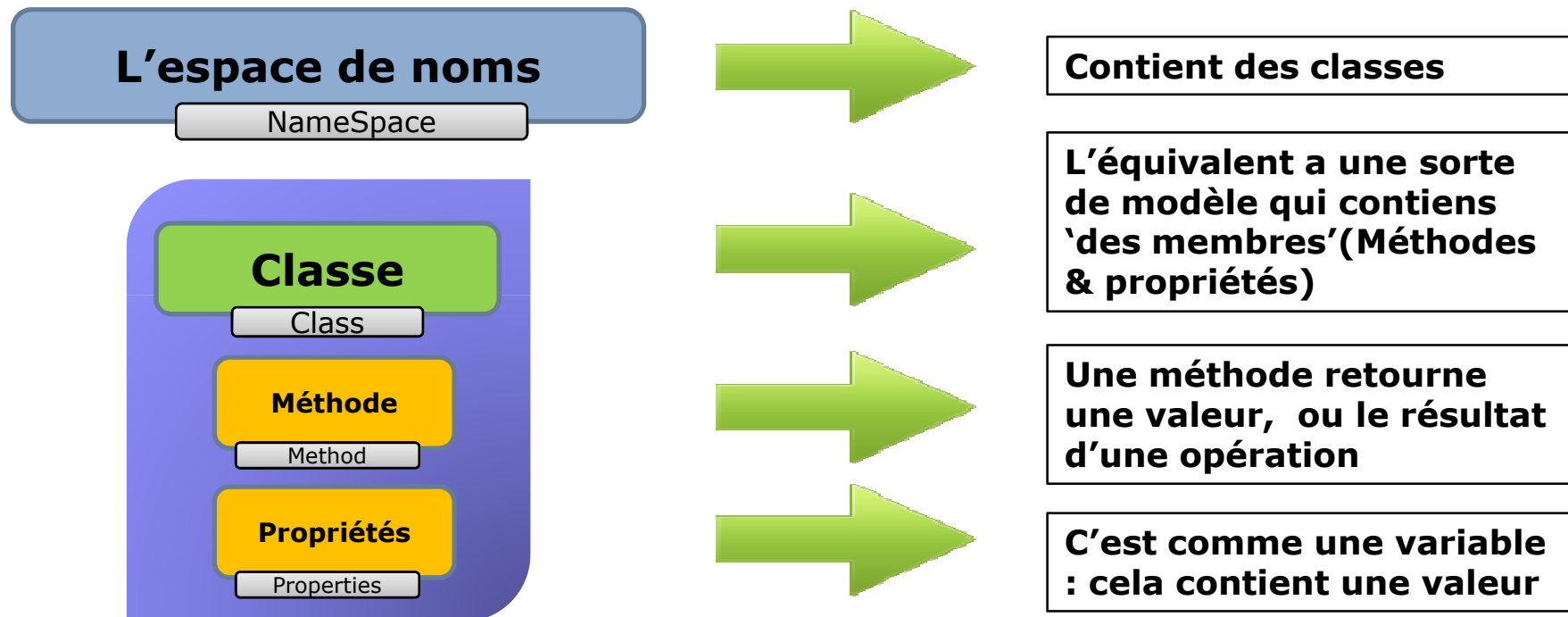
Pourquoi utiliser le .NET ?

- PowerShell offre beaucoup de cmdlets (236 en PowerShell 2.0)
- On peut augmenter ce nombre grâce aux différents modules disponibles (Active Directory, Module Exchange etc..)
- Des fois, ce n'est pas suffisant ☹
- PowerShell est basé sur du .NET, donc tout ce qui est faisable en .NET, on peut le faire en PowerShell.

Le vocabulaire



Le vocabulaire

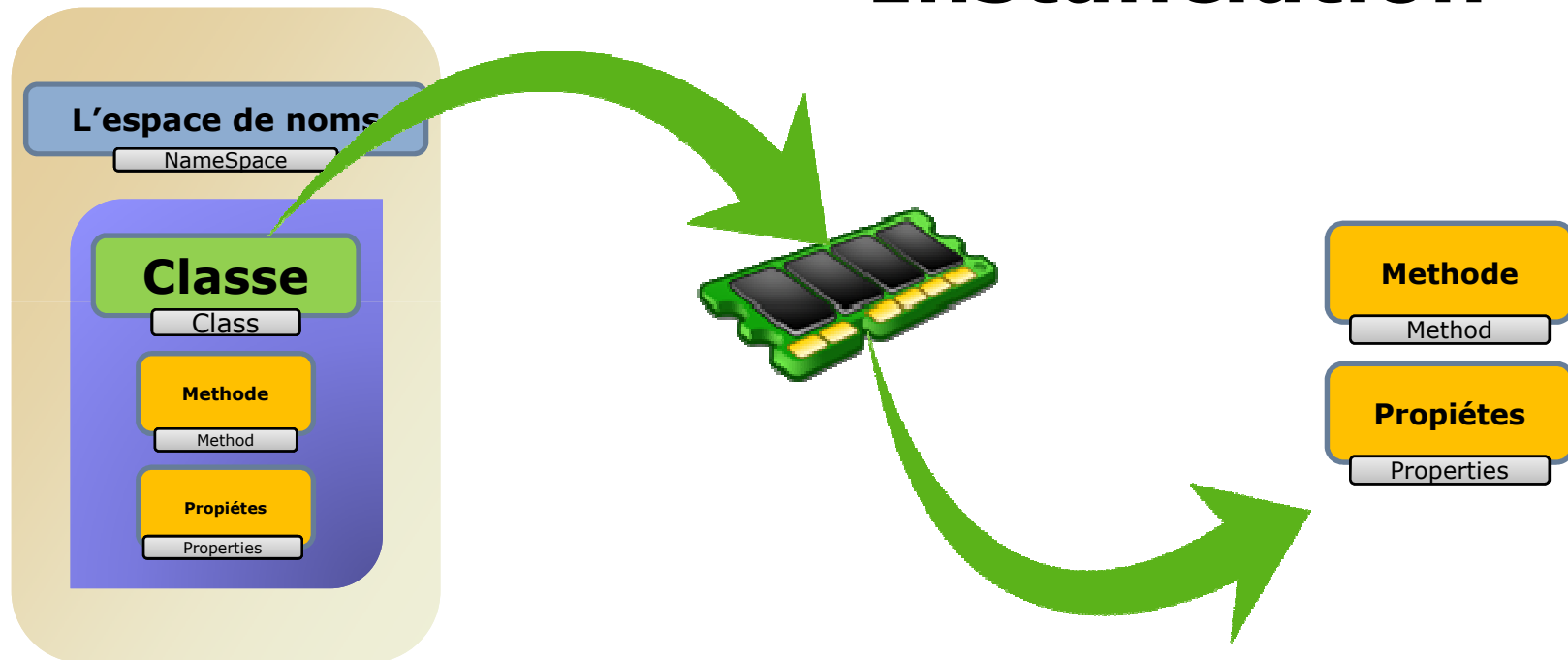


Les classes statiques

- Les classes statiques n'ont pas besoin d'être 'instanciées'. (Et ne peuvent pas l'être).
- Méthodes retournant simplement un résultat.
- C'est la façon 'facile' d'utiliser le .NET au sein de PowerShell
- Démonstration

Comment s'en servir ?

Instanciación



La jungle des classes

- Comment trouve-t-on les classes / Méthodes / propriétés .NET
 - Notre moteur de recherche favoris + msdn (ou .NET)
 - Demonstration

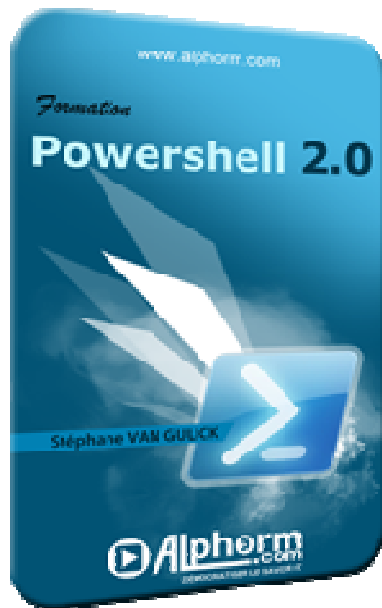
Les assemblies

- Une assembly c'est un fichier.DLL qui va contenir des classes, qui vont pouvoir être chargés dans .NET. Ceci dans un but de gain d'espace disque, et de réutilisabilité du code.
- PowerShell charge beaucoup de composants du framework .NET par défaut, mais certains éléments le sont pas. Il est cependant possible de les charger afin d'avoir accès à leur Classes, méthodes et propriétés via powershell.

Ce qu'on a couvert

- Introduction
- Pourquoi utiliser le .NET
- Le vocabulaire
 - Le framework
 - Les espaces de nom
 - Les classes (static et non static)
 - Méthodes & propriétés
- Comment instancier des objets
- Notre premier fonction basée sur DotNet





Conclusion

Site : <http://www.alphorm.com>
Blog : <http://www.ConfigMgrdistrict.com>
Blog : <http://www.PowerShelldistrict.com>
Forum : <http://www.alphorm.com/forum>

Stéphane van Gulick

Consultant systèmes et automatisation

Certifications : MCT, MCITP, MCSA

Contact : svangulick@alphorm.com

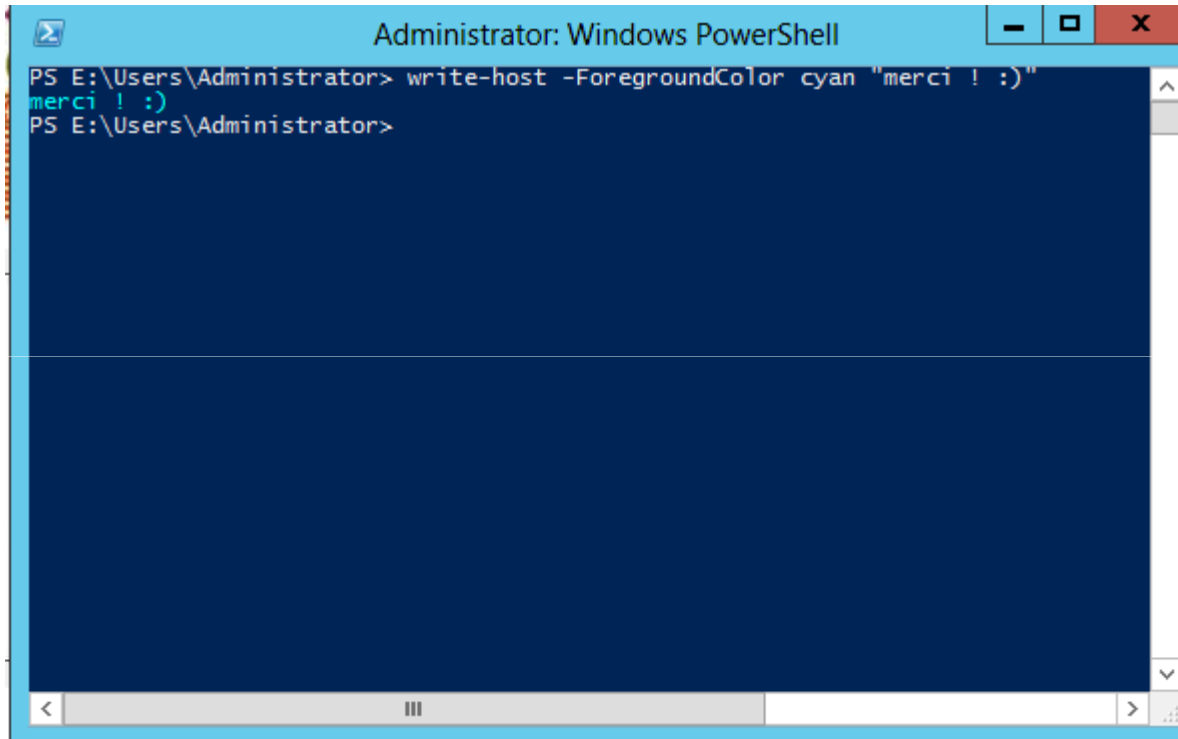
Twitter : @Stephanevg

Conclusion

- Regard en arrière
 - Initiation, approfondissement, specialisation
- Prochaine étapes
 - PowerShell 3.0
 - PowerShell lié aux composants MS (Active directory, Exchange, System Center)
 - Scripter le plus possible



Merci !



```
PS E:\Users\Administrator> write-host -ForegroundColor cyan "merci ! :)"
merci ! :)
PS E:\Users\Administrator>
```

