D2C-IT
dare to change IT...

Introduction PowerShell – The Basics

Inleiding

Het doel van deze hand-out is om jullie een beknopte samenvatting te geven van de behandelde items. Het is bij PowerShell belangrijk om te beginnen bij de basis. Als je de basis items begrijpt en weet toe te passen, zal je ontwikkeling in PowerShell en het begrijpen van de “command-line” en scripts gemakkelijker vergaan.
Ondanks het feit dat sommige misschien al ervaring hebben met PowerShell kan het naar mijn mening geen kwaad om nog eens te beginnen bij de basis.
In deze hand-out zijn wat extra onderdelen toegevoegd. ”Functions” en “Profiles” worden niet behandeld in deze cursus maar in de praktijk zal je er zeker nodig hebben of tegen komen..

Wat is PowerShell?

PowerShell is een interactieve shell (en scripttaal). PowerShell maakt gebruik van een complete nieuwe architectuur. Het is gebouwd op het .net framework en heeft een krachtige interactie met .NET, COM en WMI objecten. Het emuleert vele legacy commando’s uit cmd.exe maar ook commando’s uit Linux en Unix shells. PowerShell is een revolutionaire shell. Het ziet er in eerste instantie uit zoals iedere andere shell maar onder de oppervlakte is PowerShell zeer krachtig. De rede dat PowerShell zo revolutionair is, is omdat het niet een tekst-based shell maar object-based shell is. Je zult zien dat je direct interactie zult hebt met objecten, waardoor je geen complexe dingen hoeft te bouwen om taken te kunnen uitvoeren. Op dit moment zegt je dat misschien nog weinig, maar als we verder in de cursus zijn, zal dat zeker op zijn plek vallen.

De GUI, zeer waardevol maar …

De GUI (Graphical User Interface) is uitermate krachtig. Zonder dat je kennis van alle functionaliteiten hebt, kun je zelf ontdekken wat allemaal mogelijk is. De wizards etc. helpen je flink op weg.
Als we bijvoorbeeld kijken naar Active Directory dan is het aanmaken van een gebruiker niet zo ingewikkeld.

Je krijgt een wizard, deze vul je in, je drukt op OK en de gebruiker is aangemaakt. Je kunt de gebruiker daarna openen en eventueel de ontbrekende gegevens aanvullen. Stel dat je dit moet doen voor 10 gebruikers, voor 100, of misschien wel voor een complete migratie? Buiten het feit dat dat niet echt uitdagend werk is en het erg tijd rovend en is de kans op fouten ook groot. Deze fouten kunnen misschien zelfs security issues met zich meebrengen.
In de huidige vorm van werken is de GUI niet weg te denken, maar Microsoft weet ook dat we dingen willen automatiseren. Hiervoor ontwikkelde Microsoft vroeger voor ons EXE files , WMI en COM objecten. In het voorbeeld van een AD user heeft Microsoft voor ons bijvoorbeeld de LDIF.exe ontwikkeld. Hiermee kon je in bulk gebruikers aanmaken vanuit een CSV file. Daarnaast was het ook mogelijk om gebruik te maken van LDAP en COM objecten zoals ADSI om d.m.v. met VBScripts dit proces te automatiseren. We noemen dit de “Old way”.

De “new way” is de PowerShell manier. Alle functionaliteiten van een product zitten nu in PowerShell i.p.v. in de MMC. Boven op PowerShell wordt nu de GUI gemaakt. Dit houdt in dat we nu eigenlijk alles kunnen automatiseren. De GUI zal hierdoor wel minder uitgebreid zijn en voornamelijk de basis elementen bevatten. Hieronder schematisch “ the Old Way” (links) versus “ the New Way”.

Auteursrecht © 2020 D2C-IT. Alle rechten voorbehouden.

Basis Concepten van PowerShell

PowerShell heeft nieuwe manier hoe je dingen kunt doen. Hieronder de basis tools en concepten.

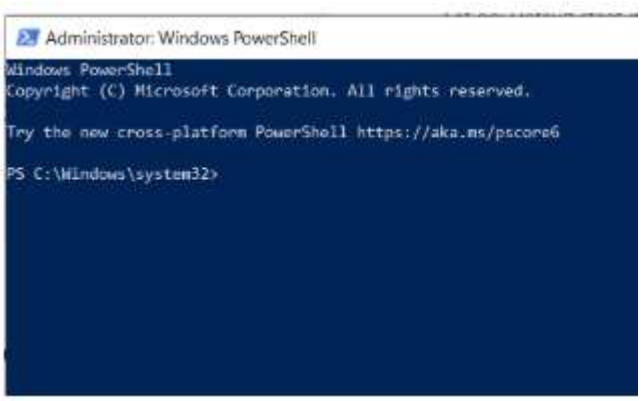
System requirements, PowerShell is beschikbaar voor alle operating systems vanaf windows 7. Mocht je nog een ouder OS hebben dan is het echt tijd om te gaan upgraden .

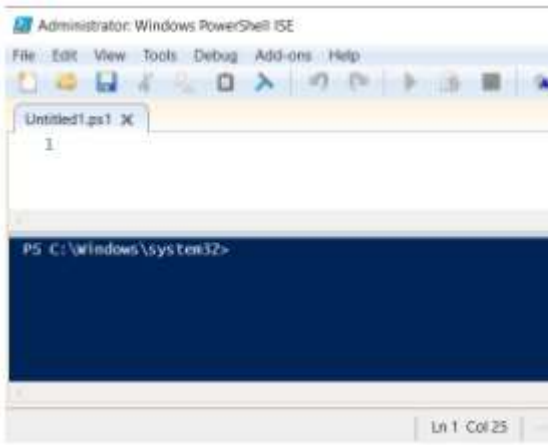
Console versus ISE

Er zijn 2 manieren hoe je default met PowerShell kunt werken. De commandline (links) en de ISE (Integrated Script Environment) . Bij zijn te vinden in de directory

```
# Check ExecutionPolicy
get-ExecutionPolicy
# Change ExecutionPolicy to RemoteSigned
set-ExecutionPolicy -ExecutionPolicy RemoteSigned
```

C:\Windows\System32\WindowsPowerShell\v1.0. PowerShell.exe en PowerShell_ISE.exe.
Let op: Default staat de ExecutionPolicy op restricted. Dit betekend dat je wel commando's kunt runnen in de commandline maar geen ps1 scripts.
Hieronder een voorbeeld om de ExecutionPolicy te bekijken en aan te passen





- simpel
- Quick
- Niet echt gebruiksvriendelijk

- Ontwikkelen van scripts
- Gebruiksvriendelijk

Cmdlet

Een Cmdlet is 'de kleinste functionele eenheid in een PowerShell. Zo klein als ze zijn, zo handig zijn ze ook. De Cmdlet, uit te spreken als 'command-let', voegt namelijk extra functionaliteit toe aan de Microsoft command-line. Cmdlet zijn een soort van functies die taken uitvoeren. De naam van een Cmdlet s over het algemeen zelf beschrijvend gekozen en

Auteursrecht © 2020 D2C-IT. Alle rechten voorbehouden.

bestaat uit een werkwoord (verb) gevolgd door een zelfstandig naamwoord (noun). Met het commando set-date, is het dus bijvoorbeeld mogelijk om de huidige datum in te stellen, terwijl get-process een lijstje met draaiende processen teruggeeft. Met de opdracht get-command krijgt men een overzicht van de beschikbare Cmdlets. Cmdlets zijn specifiek en speciaal voor PowerShell ontworpen. Wanneer je een parameter mee geeft aan een Cmdlet zijn er 2 methodes die je kunt gebruiken. **Named** parameters en **Positionele** parameters Wanneer je een "Named parameter" gebruikt dan is het karakter dat aangeeft dat het om een Named Parameter gaat "-" Dit geldt voor alle Cmdlets en zorgt voor consistentie in PowerShell. Bij een Named parameter moet je de parameternaam verplicht invullen.

```
# Examples van cmdlets
Get-Process
Get-service –name Netlogon
New-Item -Path 'C:\PowerShell' -ItemType Directory
```

Wanneer je positionele parameters gebruikt moeten de parameter waarden in de juiste volgorde staan maar de naam van de parameter kan weggelaten worden. Het reduceert het typen maar reduceert ook de leesbaarheid van het script.

```
# Example
Get-service Netlogon
New-Item 'D:\temp\Test Folder' -ItemType Directory
```

Get-Help

Get-Help is absoluut een essentiële cmdlet en geeft gedetailleerde informatie en voorbeelden over cmdlets. Get-help accepteert wildcards en kan gebruikt worden voor het zoeken naar cmdlets op bepaalde tekst.

Voorbeelden	
Zoeken op cmdlets met wildcard (*)	Get-help *service*
Get-help van een specifiek cmdlet Get-help <cmdlet-name>	Get-help <cmdlet-name>
Get-help <cmdlet-name> -detailed Gedetaileerdere informatie dan alleen get-help. Geeft ook voorbeelden	Get-help get-childItem detailed
Get-help <cmdlet-name> -example Geeft alleen de voorbeelden weer	Get-help get-childItem example
Get-help <cmdlet-name> -full Volledige info , examples, uitleg over parameters etc	Get-help get-childItem -full

Auteursrecht © 2020 D2C-IT. Alle rechten voorbehouden.

Get-Member

Het Cmdlet Get-member is ook een essentieel commando. Met Get-Member kun je namelijk alle eigenschappen (properties) en Methods van het object opvragen.

```
# Example
Get-ChildItem | Get-Member
Get-ChildItem | gm      # gm : alias of get-member
Get-Service | Get-Member
```

Get-Command

Met de opdracht get-command krijgt men een overzicht van de beschikbare Cmdlets.

```
# Show available CMDlets
get-command
gcm
(get-command).count
get-command -verb get
get-command -noun netw*
get-command get-netw*
get-Command -noun *net
gcm -noun pssnapin
get-command -CommandType Alias
get-command -CommandType Function
get-command -CommandType Cmdlet

# Show Count Aliases , Functions and Cmdlets    => SubExpression Example)
write-host " Functions : $((get-command -CommandType Function).count)"
write-host " Aliases   : $((get-command -CommandType alias).count)"
write-host " Cmdlets   : $((get-command -CommandType Cmdlet).count)"
```

Objects

Om PowerShell goed te begrijpen moet je weten wat objecten zijn. Objecten zijn simpel gezegd een digitale representatie van iets, dat bestaat uit onderdelen en waar je iets mee kunt doen. Denk bijvoorbeeld aan een fiets. Een fiets bestaat uit onderdelen, wielen, pedalen, zadel, frame , stuur, etc. We gebruiken deze collectie van onderdelen als een geheel. We kunnen trappen, linksaf sturen en bijvoorbeeld remmen. De “dingen” die we dus kunnen doen met de fiets, worden “methods” genoemd (acties). In PowerShell is bijna alles

Auteursrecht © 2020 D2C-IT. Alle rechten voorbehouden.

een object. Of we nu werken met mailboxen, gebruikers, processen, eventlogs, netwerkkaarten, etc., Alles wordt gerepresenteerd als een object en heeft dus “properties” en “methods”. Objecten in PowerShell zijn vergelijkbaar alleen in plaats van onderdelen hebben we “properties”. In het voorbeeld van get-service is een property bijvoorbeeld de servicenaam of de status . Een ”method” kan zijn stop of start. Eerder hebben we het al gehad over get-member. Met behulp van get-member kun je de “properties” en “methods” opvragen


```
# Get Member (Object properties) van 1 service
Get-Service | Get-Member
Get-Service | where-object {$_.name -eq "winRM"} | Get-Member

# Waardes van de properties
Get-Service | where-object {$_.name –eq "winRM" } | Format-list
```

Voor meer informatie : get-help about_objects

The Pipeline

“The Pipeline” zorgt ervoor dat de output van een cmdlet wordt doorgezet als input naar het volgende cmdlet
De pipeline is een erg krachtig component in PowerShell. De pipeline is het karakter “|”. Het zorgt ervoor dat de output van een cmdlet wordt doorgezet als input naar het volgende cmdlet. Het piplineteken “|” word al jaren in vele Shells gebruikt. Maar PowerShell gebruikt het heel anders. De output van PowerShell zijn objecten i.p.v. tekst zoals bv in Linux/Unix.

Auteursrecht © 2020 D2C-IT. Alle rechten voorbehouden.

```
# Geeft een lijst met Service objecten terug.
Get-Service

# Als we deze bv willen filteren dan kunnen we deze output pipen
# Met dit commando filteren we uit alle services allen de services die running zijn.
Get-service | Where-object {$_.status –eq "running"}

# Met deze output kunnen we natuurlijk verder  gaan.
# Wat we nu doen is dat we de running commando's pipen naar het cmdlet Stop-services.
# Als extra geven we de optie –whatif mee om te kijken wat er gebeurd als!!.
# Zonder –whatif krijg je waarschijnlijk een blue screen en crashed je pc.
Get-service | Where-object {$_.status –eq "running"} | stop-service –whatif

# Van alle services willen we alleen de services met de status stopped en deze willen in lijst geformateerd hebben.
Get-service | Where-object {$_.status –eq "stopped"} | format-list
```

Aliassen

Aliassen zijn verkorte namen (afkortingen) voor een commando/cmdlet en kan gebruikt worden i.p.v. de volledige naam. Als je niet goed kunt typen is dit een uitkomst. In scripts raden we het af om geen aliassen te gebruiken. Dit om de leesbaarheid van het script te bevorderen. Tip: Als je de shell afsluit zullen de zelf aangemaakte aliassen verloren gaan

```
# Show all Aliases
   get-alias
# Show cmdlet of alias
   get-alias dir
   get-alias ls
   get-alias cd
 # Show Aliases of Cmdlet
   get-alias -Definition cd   # this will fail because cd is not a cmdlet but an alias
   get-alias -Definition get-childitem
   get-alias -Definition copyitem
 # New Alias
   New-Alias -name gh -Value get-help -Description "Alias for get-help"
```

Auteursrecht © 2020 D2C-IT. Alle rechten voorbehouden.

Get-help *alias*

- Get-alias get-command (-shows existing alias)
- New-alias gh Het-help (-aanmaken nieuwe alias) 9̶9̶ 9̶

Functions

Functies, in de simpelste vorm, zijn vergelijkbaar met aliassen, maar kunnen complete script Blocks bevatten. Bekijk de voorbeelden hieronder.

Get-help *function*

```
# Function Example 1
Function scriptdir
{
   set-location "c:\mystuff\ps scripts"
}

# Function Example 2
Function Get-BigFiles
{
   dir –recurse | where-object {}$.Length –get 100mb
}

# Function Example 3
function get-BigFiles {
   <#
   .Synopsis
     get-BigFiles
   .DESCRIPTION
     get-BigFiles wills search for big file
   .EXAMPLE
      $BigFiles = get-bigfiles -path C:\PowerShell -size 100mb
      $BigFiles  | select fullname,name,Length | fl
   #>

   [CmdletBinding()]

   param(
      # Param1 help description
      [Parameter(Mandatory=$false)]
      $Path = "C:\" ,

      # Param1 help description
      [Parameter(Mandatory=$false)]
      $size = "100mb"
   )
```

Auteursrecht © 2020 D2C-IT. Alle rechten voorbehouden.

```
  Begin{}
  Process{
    if( test-path $Path )
    {
      $result = get-childitem -path $path –recurse |
          where-object {$_.Length –gt  $size}
    }
    else
    {
      write-warning "Could not find path $path"
    }#EndIf
  }
  end{
    return $result
  }
}#end function
```

“Functions” worden opgeslagen in de functie provider (zie provider) en zullen net zoals bij aliasen verloren gaan als je de shell afsluit.

Voor meer informatie :

- Get-help function
- Get-Help about_function

Profiles

Profiles zijn simpel gezegd een “startup script” voor PowerShell. Elke keer als je PowerShell start wordt dit gestart. Het is te vergelijken met bijvoorbeeld autoexec.bat. Een PowerShell profile bevat simple scriptregels, Functions, aliassen en soms erg complexe codes . $profile is een speciale variabele die de locatie van het profile(s) bevat. Het bevat niet het profile zelf, maar de locatie. $profile bestaat altijd, maar vanwege security redenen bestaat het actuele profile niet en zal je deze zelf aan moeten maken.

De snelste manier om je profile te maken, is door de volgende regel te runnen:

```
# Create Profile
New-item –path $profile –type File –force
```

Auteursrecht © 2020 D2C-IT. Alle rechten voorbehouden.

Providers

Een provider is een gemeenschappelijke interface naar verschillende datastores. Denk bij een datastore aan het “file system”. Een PowerShell provider is een adapter, deze zorgt er voor dat een bepaald data storage medium te zien is als een diskdrive

Voor meer informatie : get-help about_provider

```
Get-Psdrive

# Create a new driveletter to a folder with New-PSDrive
New-PSDrive -Name W -PSProvider FileSystem -Root \\localhost\c$
New-PSDrive -Name Powershell -PSProvider FileSystem -Root \\localhost\c$
Get-psdrive -name Powershell | remove-psdrive

# Connect to registry
Set-location HKLM:
Get-ChildItem -Path HKCU:\ | Select-Object Name
```

Auteursrecht © 2020 D2C-IT. Alle rechten voorbehouden.