

3 Windows Powershell

Inhoud

3.1	Intro.....	3
3.1.1	Windows Powershell	3
3.1.2	Powershell ISE	3
3.1.3	Windows PowerShell ISE gebruiken	4
3.1.4	Prompt in Powershell ISE	5
3.1.5	Script venster.....	5
3.2	Cmdlets.....	7
3.2.1	Eerste cmdlets.....	7
3.2.2	Get-Help	8
3.2.3	Enkele cmdlet groepen onder de loop	9
3.3	Parameters en parameterwaardes.....	11
3.3.1	Gebruik parameters:	11
3.3.2	Wildcards als waarde parameter	11
3.4	Aliassen	14
3.5	Pipes of pipelines	16
3.6	Operatoren.....	17
3.7	Variabelen.....	17
3.7.1	Speciale variabelen: Omgevingsvariabelen	18
3.8	Tips & tricks	18
3.9	Oefeningen	19
3.9.1	Oefening 1: 'Een bestand op het bureaublad'	19
3.9.2	Oefening 2: 'userprofiel'	19
3.9.3	Oefening 3: 'zoekfunctie'	19
3.9.4	Oefening 4: "info opvragen"	19
3.9.5	Oefening 5: 'test cmdlets'	20
3.9.6	Oefening 6: 'grid'	20
3.9.7	Oefening 7 'locatie wijzigen'	21
3.9.8	Oefening 8: 'ls'	21
3.9.9	Oefening 9	21
3.9.10	Oefening 10	22
3.9.11	Oefening 11 (Uitbreiding)	22

Doelstellingen:

1. De student weet het verschil tussen CMD en PowerShell ISE.
2. De student kan bestanden en mappen aanmaken met cmdlets.
3. De student kan navigeren door de mappen door gebruik te maken van relatieve, bijna-absolute en absolute paden.
4. De student kan in PowerShell cmdlets gebruiken om als administrator taken uit te voeren i.p.v. met de GUI.
5. De student kan PowerShell cmdlets gebruiken om als administrator bepaalde taken uit te voeren.
6. De student kan PowerShell cmdlets gebruiken in combinatie met omgevingsvariabelen om scripts te schrijven die op elke computer kunnen uitgevoerd worden.
7. De student kan in PowerShell aliassen voor cmdlets gebruiken, maar weet wat de cmdlet voor de alias is of weet deze te vinden.
8. De student kan gebruikers aanmaken en rechten toekennen met behulp van cmdlets
9. De student kan allerlei bronnen raadplegen en beoordelen op bruikbaarheid.



3.1 Intro

Windows GUI VS CLI

De meeste gebruikers van Windows zijn enkel vertrouwd met de GUI. Ze gebruiken grafische vensters voor de handelingen op het OS.

Een IT-professional werkt met de GUI, maar moet ook leren werken met de CLI (Command Line Interface). Vooral een netwerk administrator zal om taken te automatiseren en om *remote* (op afstand) computers in te stellen de opdrachtprompt gebruiken.

Routinetaken kan je '**scripten**'. Je maakt een script waarin commando's staan en deze commando's worden uitgevoerd bij het runnen van het script. Denk maar eens als je op 50 computers dezelfde taken moet uitvoeren. Wat heeft je voorkeur? Je voert de taken 50 keer uit, of je maakt 1 script met alle taken en je voert het script uit op 50 computers? Het script kan eenvoudig op afstand worden uitgevoerd.

Ook troubleshooting informatie opvragen over het OS is via de CLI meestal een snellere manier, zeker als administrator of netwerkbeheerder.

In Hoofdstuk 4 'Windows Administrator' zullen we zowel de Gui als de CLI behandelen.

CLI in Windows

In Windows 10 zijn verschillende CLI's aanwezig:

- Opdrachtprompt of Command Prompt : Hierin kan je DOS commando's ingeven.
Veelgebruikte commando's:
 - Ping
 - ipconfig
 - whoami
 - DIR
- Windows Powershell
- Windows Powershell ISE (Integrated Scripting Environment)

3.1.1 Windows Powershell

Windows PowerShell is een command-line en scripttaal die speciaal is ontworpen voor systeembeheer. (Het gelijkaardige in Linux wordt Bash Scripting genoemd.) Gebouwd op het .NET Framework, helpt Windows PowerShell IT-professionals om het beheer van het Windows-besturingssysteem en de applicaties die draaien op een Windows (Server) omgeving te controleren en te automatiseren.

Met Windows PowerShell-commando's, cmdlets genaamd, kan je computers vanaf de opdrachtregel beheren. Met Windows PowerShell providers heb je net zo gemakkelijk toegang Registry als tot het bestandssysteem. Daarnaast heeft Windows PowerShell een rijke 'expression parse' (= expresie ontleden) en een volledig ontwikkelde scripttaal. Dus in eenvoudige woorden kunt u alle taken voltooien die u met GUI en nog veel meer doet.

3.1.2 Powershell ISE

De Windows PowerShell Integrated Scripting Environment (ISE) is een hosttoepassing voor Windows PowerShell. In Windows PowerShell ISE kan je commando's uitvoeren en scripts schrijven, testen en debuggen in één enkele (Windows) grafische gebruikersinterface met multiline editing (meerlijn bewerking), tab completion (tabbladaanvulling), syntax coloring

(syntaxiskleuring), selective execution (selectieve uitvoering) en context-sensitive help (contextgevoelige hulp).

In dit hoofdstuk gebruiken we PowerShell ISE. Deze heeft IntelliSense (Ctrl + spatie), hiermee kan je cmdlets, paden ed. automatisch laten aanvullen, wat typefouten vermindert. Verder kan je hier ook Clipboard gebruiken (kopieëren en plakken) en je kan meerdere tabs gelijk open zetten. (File > New Powershell Tab)

Vanaf versie 5.0 is Powershell uitgebreid en krachtiger geworden. Je kan vanuit de terminal processen stoppen en starten en apps installeren en verwijderen. PowerShell maakt deel uit van het .net framework. Je kan alles uit het framework in PowerShell gebruiken, maar je kan ook in je .net programma's PowerShell scripts integreren.

Sinds de versie 6.0 maakt PowerShell deel uit van de .net Core, wat nog meer mogelijkheden biedt. Zoals scripts om in Linux te gebruiken. Dit ligt buiten de focus van deze cursus.

Pluralsight video's:

- [Powershell first day](#)
- [Powershell putting to work](#)
- [Powershell intro](#)

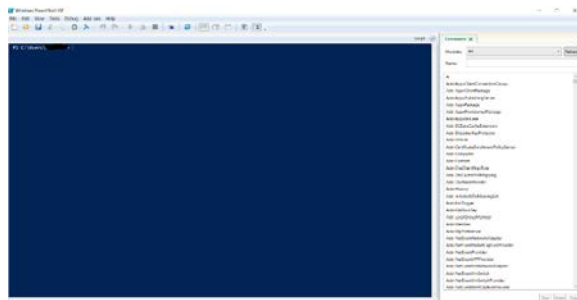
3.1.3 Windows PowerShell ISE gebruiken

Pluralsight video's:

- Powershell first day
- Powershell intro

Kennismaking met de terminal

Je kan verschillende deelvensters kiezen in View, die dan naast of boven de terminal verschijnen. Onderstaande figuur toont naast de terminal de lijst van cmdlets in alfabetische volgorde.



Hoewel PowerShell werkt met cmdlets kan je ook dos commando's gebruiken. Dit maakt dat je geen 2 verschillende terminals nodig hebt voor verschillende opdrachten. Er wordt gefluisterd dat CMD op termijn zal verdwijnen en Microsoft enkel nog PowerShell zal aanbieden met zijn nieuwe Windows versies.

In de hierboven vermeldden Pluralsight filmpjes worden veelal dos commando's (=aliassen, zie verder in de cursus). De bedoeling is dat jij met cmdlets leert werken.

Run as administrator

Je kan PowerShell als gewone gebruiker openen, maar voor specifieke taken open je PowerShell als administrator, omdat je anders onvoldoende rechten hebt. Dit zal vooral nodig zijn als je processen wil stoppen of starten. Als op PowerShell ISE rechtermuis klikt kan je kiezen om het programma ls administrator uit te voeren. Om veiligheidsredenen werk je meestal in de gewone PowerShell met gebruikers bevoegdheden.



Opdracht: Start powershell ISE en update de help functie

Start PowerShell ISE als administrator.

Geef het commando 'Update-Help' en voer deze cmdlet uit, dit duurt even!

Na installatie van de help, typ exit aan de prompt.

Vanaf nu kan je alle help gebruiken in de terminal als gewone gebruiker.

Hulp nodig? Zie [pluralsight 'powershell first day' item 4](#)

Met opmerkingen [LVH1]: Toevoegen: Reden van update-help (+mogelijke foutmelding NL)

3.1.4 Prompt in Powershell ISE

In de terminal verschijnt een knipperende cursor na een groter dan teken '>'. '>' noemen we de prompt. Hier kan je commando's typen.

Heel belangrijk is wat links van > staat. Het begint in powershell met PS.

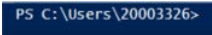
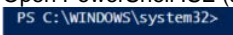
Na PS staat het absolute pad naar de **huidige map**. Dit duidt de plaats in de directory-structuur aan op de **schijf**.

Opmerking: Engelse namen gebruiken voor de standaard mappen van Windows! (Desktop, Documents, ...)

Opmerking: In het vorige hoofdstuk heb je gezien dat de 'homefolder' van een gebruiker altijd C:\users\Username is. Hierin worden alle documenten van de gebruiker opgeslagen. Hierin start PowerShell standaard op.

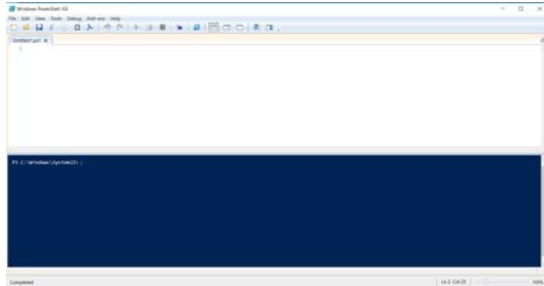


Opdracht powershell prompt

- Open PowerShell ISE (als gebruiker) en noteer de prompt van de terminal.

- Open PowerShell ISE (als admin) en noteer de prompt van de terminal.


3.1.5 Script venster

In PowerShell ISE is er een scriptvenster voorzien.



Dit verschijnt naar keuze boven of naast de terminal. In dit scriptvenster kunnen commandoregels gebouwd worden of meerdere regels onder elkaar die dan een script vormen.

Het script kan opgeslagen en uitgevoerd worden en met de knoppen in het lint(of F5 om het hele script te runnen) of met een commando in de terminal: `.\naamScript` + enter. (.\ -> Als het script zich in de huidige map bevindt. Anders moet je het pad meegeven.)

Bij markeren van 1 of meerdere commandoregels in het scriptvenster, kunnen deze uitgevoerd worden met F8 of in het lint het document met het groene pijltje op. Op deze manier moet een script niet opgeslagen worden.

3.2 Cmdlets

Pluralsight video: [Zie item 4 voor cmdlets](#)

Commando's in PowerShell krijgen de naam cmdlets. Dit zijn tekst opdrachten die via de terminal ingevoerd en door Windows 10 uitgevoerd.

Cmdlets zijn niet hoofdlettergevoelig, IntelliSense zal altijd elk deel met een hoofdletter laten beginnen. Parameters zijn wel hoofdlettergevoelig!!!

Een cmdlet of "Command let" is een lichtgewicht commando dat gebruikt wordt in de Windows PowerShell omgeving. De Windows PowerShell runtime roept deze cmdlets op bij de commando prompt. Je kunt ze programmatisch aanmaken en oproepen via Windows PowerShell API's.

3.2.1 Eerste cmdlets

PowerShell werkt met cmdlets! Deze hebben meer functionaliteiten en zijn krachtiger dan gewone commando's. De gewone commando's blijven werken in powershell!



Opdracht: De eerste cmdlets

Test volgende cmdlets. **De werking van deze cmdlets wordt later duidelijk**

- Get-help
- Get-command
- Get-service
- Start-service
- Help start-service
- Get-alias ls
- Get-childitem
- Show-command
- \$PSVersionTable

Op dit moment is de werking niet belangrijk. De opdracht is enkel om kennis te maken met Powershell en de Cmdlets.

Let wel op wat je ziet bij 'Get-help'!! Dit kan je helpen als je even niet meer weet :)

Probeer ook de volgende instructies:

- Pijltje omhoog
- Get-history
- CLS
- Exit

Wat valt er op aan de opbouw van de cmdlets?

Herken je overal een werkwoord en een zelfstandig naamwoord? (verb + noun)

Met opmerkingen [LVH2]: Opdracht vervangen door 'voorbeelden?'



- Als er iets dreigt fout te gaan: je kan steeds de uitvoer van een cmdlet onderbreken met Ctrl + s en definitief stoppen met Ctrl + c.
- Om PowerShell af te sluiten is het veiliger om steeds aan de prompt 'exit' te typen en dan enter. Zo worden lopende processen nog veilig afgewerkt.
- De meeste cmdlets bestaan uit een werkwoord en een zelfstandig naamwoord, of in het Engels: Verb-Noun.
- **Belangrijkste cmdlet : Get-Help**

Get-Help [cmdlet]
Get-Help [cmdlet] -examples

- Een eerder ingevoerd commando aan de prompt kan steeds terug opgehaald worden met het pijltje omhoog. Handig om fouten eruit te halen of verder aan te vullen, bespaart tijd en typewerk!

PowerShell is object georiënteerd

Dit wil zeggen dat als er output is na een commando deze output niet zomaar tekst is, maar een object. Dit wil dus zeggen dat tekstparsing (splitsen, ...) niet nodig is.

In een programmeertaal is een object iets dat bepaalde eigenschappen en gedragingen heeft. Een simpel voorbeeld van een object uit het dagelijkse leven: een auto: als eigenschappen heeft een auto het aantal deuren of de kleur en als gedrag bijvoorbeeld starten. Het werken met objecten is zeer flexibel, dit zal verder in deze cursus duidelijker worden.



Opdracht: PowerShell IntelliSense

- Type: get-c
Nu krijg je een lijst met mogelijke aanvullingen, als je even wacht krijg je info over de cmdlet die gemarkeerd is.
- Type: get-com
- Als je nu tab klikt, wordt je regel automatisch aangevuld tot Get-Command.
- Enter en het cmdlet wordt uitgevoerd.
- Get-command:
- Nu verschijnt een hele lijst commando's

Opmerking: Omdat de output een object is, kan je zeggen dat je enkel die commando's wil met een specifieke eigenschap, nl het commando type cmdlet. Om deze op te halen wordt een parameter meegegeven.

Weet na deze opdracht de IntelliSense te gebruiken!

Met opmerkingen [LVH3]: Vervangen door voorbeeld intelliSense
+ opmerking instellingen Powershell ISE

Intellisense oproepen met ctrl-spatiebalk

3.2.2 Get-Help

Get-Help is vergelijkbaar met de man pages bij Linux.

De help functie zal je begeleiden naar het juiste gebruik van cmdlets en de syntax ervan. Help geeft een beschrijving van de commando's, aliases, voorbeelden, syntax, opbouw, ...



Opdracht GET-HELP

1. Type in powershell 'GET-HELP
→ help geeft je een algemene help pagina over Windows Powershell cmdlets en concepten.
2. Type GET-HELP get-command.
→ Powershell geeft hulp voor het commando 'get-command'
3. Type help new-item -examples
→ Powershell toont je de help functie voor de cmdlet "new-item".
Met deze instructie worden enkel de voorbeelden getoond. Dit wordt weergegeven met de parameter -examples. Zie verderop in het werkboek voor meer verdieping voor het gebruik van parameters.

Met opmerkingen [LVH4]: Opmerking: In dit voorbeeld wordt gewerkt met paramteres -> zie ...verwijzing

4. Type help start-service -online

→ Powershell toont je de help functie voor de cmdlet "start service". Hier wordt specifiek de online help opgeroepen. (Dit wordt weergegeven met de parameter -online)



1. Snelle manier om reeds in de juiste map te starten: in verkenner de juiste map openen in de adresbalk. → je klikt in de adresbalk en typt Powershell ise en automatisch opent zich PowerShell ISE met voor de prompt de juiste map als huidige map.
2. Alle cmdlets en commando's van de huidige sessie opvragen: ook hiervoor bestaat er een cmdlet! Get-History

3.2.3 Enkele cmdlet groepen onder de loep

3.2.3.1 Set-

Deze cmdlets gaan iets instellen:

Bijvoorbeeld Set-Location. Hiermee bepaal je de plaats van de huidige map. Je geeft hier een pad aan mee; een absoluut pad (of een bijna absoluut pad) of een relatief pad.



Opdracht: zoeken naar cmdlets

Zoek uit hoe je een commando kan opstellen om enkel de cmdlets die beginnen met Set- te krijgen.

- Tip1: Gebruik het cmdlet 'get-command'.
- Tip2: Plaats na het commando set-*

Get-Command Set-* -CommandType cmdlet

Of Get-Command -Verb set

of gebruik het venster commands (name= get-command, verb: set → insert)

Meer info over het gebruik van '*' (=wildcart) verderop in het werkboek.

Met opmerkingen [LVH5]: Voorbeelden toevoegen.

3.2.3.2 Get-

Een object ophalen. Afhankelijk van de meegegeven parameters kan je de output inperken

Met opmerkingen [LVH6]: Voorbeelden toevoegen

3.2.3.3 New-

Zoals het woord het zegt wordt er iets nieuw aangemaakt:

Met opmerkingen [LVH7]: Voorbeelden toevoegen

Voorbeeld: New-Item



Opdracht New-Item

- Zoek uit welke parameters je kan meegeven. (Gebruik de help functie!)
Get-Help new-item
- Maak een nieuwe map aan op je bureaublad met de naam hoofdstuk3 met behulp van deze cmdlet.
New-Item .\Desktop\hoofdstuk3 -ItemType 'Directory'
- Zet in die map een tekstdocument (.txt) met de naam oplossingen
New-Item .\Desktop\hoofdstuk3\oplossingen.txt -ItemType 'File'

Met opmerkingen [LVH8]: Opmerking: paramters is 3.3 (verwijzing!) Niet als opdracht opmerking

- Geef het tekstdocument de inhoud: "Dit zijn de oplossingen van hoofdstuk 3".
(welke cmdlet zou je hiervoor gebruiken?)
Optie1: `New-Item -path .\Desktop\hoofdstuk3 -Name 'oplossingen.txt' -ItemType file -Value 'Dit zijn de oplossingen van hoofdstuk 3'`
Optie2: `Set-Content -Path .\Desktop\hoofdstuk3\oplossingen.txt -value 'Dit zijn de oplossingen van hoofdstuk 3'`
Optie3: `add-Content -Path .\Desktop\hoofdstuk3\oplossingen.txt -value 'Dit zijn de oplossingen van hoofdstuk 3'`
Opmerking: open het bestand via powershell door:
`.\Desktop\hoofdstuk3\oplossingen.txt` of
`invoke-item -path .\Desktop\hoofdstuk3\oplossingen.txt`

3.2.3.4 Out-

Met deze cmdlets bepaal je hoe de output gestructureerd gaan worden in je terminal of in je document als je de output naar een document wenst weg te schrijven. Hier komen we op terug in het onderdeel pipes.

3.2.3.5 -Item



Opdracht -Item

1. Zoek uit hoe je alle cmdlets die eindigen op een bepaald zelfstandig naamwoord kan ophalen.
Gebruik de cmdlet 'get-command' → Gebruik help-command voor meer informatie over de cmdlet!!! De parameter -noun geeft je de mogelijkheid te filteren op een bepaald zelfstandig naamwoord.
Bijvoorbeeld: `Get-command -noun 'process'`
Als je het antwoord weet op 1, kan je nu uitzoeken welke cmdlets er allemaal bestaan met -Item en wat ze doen.
`get-command -noun 'item'`

Met opmerkingen [LVH9]: voorbeeld

Met opmerkingen [LVH10]: voorbeeld! En omschrijving

3.3 Parameters en parameterwaarden

De output van bepaalde cmdlets is weer een object, dit wil zeggen, er kan weer een eigenschap aan meegegeven worden om de output weer te verkleinen. Of je kan de output opvangen en deze verder gebruiken (zie onderdeel pipes). Vergelijk deze manier van werken met het spel 'Wie is het?'. Door bepaalde kenmerken aan te geven, ga je steeds minder mogelijkheden hebben, maar wel veel specifiekere output krijgen.

Met opmerkingen [LVH11]: Objecten zijn nog niet gezien...

3.3.1 Gebruik parameters:

Er zijn 3 manieren om parameters te gebruiken:

1. Enkel de waarde meegeven: wordt gebruikt voor zaken die logisch zijn. Om dit duidelijk te maken is er het voorbeeld om de inhoud van een bestand op te halen:

Get-Content tekst.txt

Het is hier duidelijk dat je de naam van het bestand als parameter gaat meegeven, dus geef je enkel de waarde mee.
Hiervoor moet dit bestand wel in de huidige map staan!

2. Enkel de naam meegeven: Om extra opties aan of uit te zetten, geef je enkel de naam van de parameter mee. bijvoorbeeld het weergeven van voorbeelden hoe je een bepaalde cmdlet moet gebruiken:

Get-Help Get-Content -Examples

Examples is zo een typische parameter die zonder naam gebruikt wordt, om de output toch te filteren.

3. Naam en waarde meegeven: De derde optie is om een parameter zowel met naam als met waarde mee te geven. Dit wordt meestal gebruikt bij minder voorkomende parameters of als je een lijst van parameters wil meegeven.
Hiervoor het eerste voorbeeld, maar het bestand staat niet in de huidige map. Dan is het hele pad (absoluut of relatief) nodig om de inhoud van het bestand op te halen. Ervanuit gaande dat het tekst.txt bestand in de documentenmap van de gebruiker staat, is dit het voorbeeld:

Get-Content -Path .\Documents\tekst.txt

Opgelet: Eerst de naam van de parameter en dan de inhoud met een spatie tussen de 2.

Opmerkingen:

- Eén cmdlet kan meerdere parameters hebben. Er kunnen ook meerdere parameters gelijktijdig meegegeven worden. Hiervoor wordt na een parameter of na de waarde van een parameter een spatie gezet voor de volgende parameter.
- Eén parameter kan meerdere waardes hebben. Je kan aan 1 parameter 2 of meer waardes meegeven door deze te scheiden met een komma.

3.3.2 Wildcards als waarde parameter

In de opbouw van een commando kan als waarde van de parameter gebruik gemaakt worden van 1 of meerdere wildcard(s). Er zijn 4 mogelijke wildcards:

- * : komt overeen met 0, 1 of meerdere karakters
- ? : komt overeen met exact 1 karakter op de plaats waar dit teken staat

- [-] : bepaalt een range van karakters, waaruit het mogelijke karakter moet komen.
- [] : bepaalt uit welke karakters het karakter mag zijn.

Met opmerkingen [LVH12]: Voorbeelden toevoegen!

Meer info over wildcards? Type in Powershell: Get-Help about_Wildcards



Opdracht: Parameters en wildcards

1. Vul Get-Command met de juiste parameter aan, zodat je enkel cmdlets krijgt.
[Get-Command -CommandType Cmdlet](#)
(Gerbuik [help get-command ...](#))
2. Maak met behulp van een cmdlet een nieuwe map aan met de naam test in de documentenmap van de huidige gebruiker. Noteer alle stappen die je hiervoor doet, alsook de cmdlet met de juiste parameters (waarde, naam + waarde of enkel naam).
[New-Item .\Documents\test -ItemType 'Directory'](#)
3. Met welke cmdlet kan je een bestand en/of map kopiëren en/of verplaatsen naar een andere locatie (lees map). Opvragen, verwijderen? Welke parameters heb je nodig?

Met de cmdlet [Copy-Item](#) waarbij je de naam van het bestand of map meegeeft en de locatie naar waar je het wil verplaatsen.

Bijvoorbeeld de map test uit vorige opdracht kopiëren naar het bureaublad:

→ [Copy-Item .\Documents\test .\Desktop](#)

Als je het wil verplaatsen gebruik je

→ [Move-Item .\Documents\test .\Desktop](#)

Waar kan je de cmdlet vinden? [Get-command *copy*](#)

4. Zoek de werking en mogelijke parameters op van: → [Gebruik Get-help!!!](#)
 - a. [Set-Location](#) → [verandert de huidige map](#)
 - b. [Get-ChildItem](#) → [toont subitems](#)
 - c. [Get-Content](#) → [geeft de inhoud van een item](#)
 - d. [Set-Content](#) → [schrijft de inhoud van een item](#)

Opmerking:

- a. ook het venster [commands](#) kan je snel info verschaffen over parameters.
 - b. Online help via [docs.microsoft](#). (voorbeeld voor [set-location](#))
5. Bij [Get-childItem](#), [Remove-Item](#), [Copy-item](#), [Move-Item](#), [Copy_item](#), [New-Item](#), [Select_object](#) en [Sort-Object](#) vind je enkele belangrijke parameters, die de actie extra kracht of diepgang bieden. Hier is een lijstje met parameters. Zoek uit bij welke van bovenstaande cmdlets deze kunnen gebruikt worden en bepaal wat ze toevoegen aan de actie.
 - a. [-Recurse](#) → [ook submappen en bestanden](#)
 - b. [-Hidden](#) → [ook verborgen items.](#)
 - c. [-System](#) → [ook systeembestanden](#)

Met opmerkingen [LVH13]: Extra toevoegen:
Wat als een item moet overschreven worden. (Copy item naar dezelfde folder)
Wat als de inhoud van de map moet worden meegenomen.

- d. -File of -Directory → enkel bestanden of mappen (Bestaan niet bij *-Item commando's)
 - e. -Force
Get-Item -Force : ook verborgen of systeembestanden
New-Item -Force: schrijft deze over bestaande bestanden met dezelfde (idem voor Move-Item, Copy-Item,)
Remove-Item -Force: ook verborgen, systeem- of alleen-lezen bestanden worden verwijderd.
 - f. -Unique → ... die 1 keer voorkomen
 - g. -First <getal>, -Last <getal>, -Skip <getal>
6. Verklaar wat hier de bedoeling is: Get-Item ??[aeiouy]*
'get' Alle items, uit de huidige map, waarvan het derde teken een klinker is.

3.4 Aliassen

Omdat alle cmdlets uit tekstopdrachten bestaan is dit veel schrijfwerk en als je de IntelliSense niet gebruikt is dit ook zeer foutgevoelig (typefouten).

Op de vraag van ontwikkelaars om dit te vereenvoudigen heeft Microsoft ook aliassen voorzien voor de cmdlets.



Een alias is een andere (korte) naam voor een cmdlet.

Veel aliassen lijken op Linux commando's. Let wel op, de parameters zijn dat niet. Bij gebruik van een alias moet je de originele parameters van de cmdlet gebruiken.

Met opmerkingen [LVH14]: Voorbeelden



Opdracht: aliassen

1. Zoek uit hoe je kan zien van welke cmdlet een bepaalde alias is?
[Get-Alias \[alias\]](#)
2. Zoek uit hoe je van een cmdlet alle aliassen kan opvragen.
[Get-Alias -Definition \[cmdlet\]](#)
3. Maak zelf de alias "fy" aan van Get-Content en test de werking uit met een tekst bestand. → [Set-Alias -Name fy -Value Get-Content](#)
Zelf een alias maken kan met [Set-Alias \[alias\] \[cmdlet\]](#)
→ als je deze cmdlet nog niet kende kan je dit zoeken via de cmdlet 'get-command'. Bijvoorbeeld door `get-command set-*[ias]`. (Meer info over wildcards volgt nog).

Je kan aan de cmdlet 'set-alias' ook parameters aan meegeven. (zie help [set-alias](#) voor meer info!!)

Ipv een commandlet kan je ook een uitvoerbaarprogramma instellen in een alias, en zelf een cmdlet met parameters, maar dan moet je er accolades rond zetten: `Set-Alias [alias] [{cmdlet paramter}]`

Opmerking: als de alias al bestaat wordt die nu overschreven, een cmdlet kan meerdere aliassen hebben, maar een alias kan maar naar 1 cmdlet verwijzen.

Veiliger is de cmdlet [New-Alias](#), uitvoeren met een cmdlet of een cmdlet met parameter(s), maakt een nieuwe alias.

→ `New-Alias -name 'fy2' -Value Get-Content`

Zelfgemaakte aliassen zijn wel enkel geldig gedurende de huidige sessie.

Moeten ze langer blijven, moet je ze toevoegen aan de \$profile van PowerShell. Maar dit ligt buiten de scope van deze cursus.

Enkele veel gebruikte aliassen:

Cmdlet	alias
Set-location	Cd
Get-childItem	Ls
Get-content	Cat
Copy-item	Cat

Help (= functie)

man

Al de aliases zien in Powershell? Type `Get-command -CommandType alias`



Opdracht: LS (Linux)

Probeer het veelgebruikte commando `ls` uit Linux met de bijhorende Linux parameter `-l`. Wat is het resultaat?

Dezelfde output in PowerShell is er niet omdat permissies anders worden weergegeven in Windows, nl met de ACL. Hiervoor heb je volgend onderdeel, pipes nodig. Je haalt eerst alle items op, geeft deze door en vraagt per item de ACL op:

`ls | get-acl`

Zoek uit hoe dit in PowerShell moet.

`Get-Alias ls` → Dit levert je de cmdlet van de alias `ls` op... `ls` -> `Get-ChildItem`

3.5 Pipes of pipelines

Meer info: [zie pluralsight powershell-putting-to-work, item 5](#)

Zoals reeds aangegeven is de output van een cmdlet niet gewoon tekst, maar een object. Tot nu is de output telkens in de terminal getoond, maar het is ook interessant om deze naar een bestand weg te schrijven, deze verder uit te filteren

Om een object door te geven aan een volgende opdracht wordt een verticale streep gebruikt, het pipe-teken: | (op de laptops van school: 'alt gr + 1').

Dit wil zeggen dat de output van de ene opdracht als input voor de volgende opdracht gegeven wordt. Dit biedt vele mogelijkheden, maar maakt een opdrachtlijn ook langer. Bij variabelen later in deze cursus worden ook andere mogelijkheden aangereikt. Maar met pipes werken is de meest volwassen manier.

Deze pipe is handig als de output op een andere manier moet getoond worden dan het standaard formaat.

Met opmerkingen [LVH15]: Voorbeelden!!



Opdracht = pipelines

1. Zoek de verschillende format cmdlets en test ze eens uit op het cmdlet Get-Command.
 - a. Dus: Get-Command | en dan de gekozen output cmdlet.
 - b. Bijvoorbeeld: Get-command | format-list
 - c. Bijvoorbeeld: Get-command | format-list Name, Status
 - [Format-List](#)
 - [Format-Table](#)
 - [Format-Wide](#)
 - [Format-Custom](#)
2. Wanneer de output meer dan scherm vullend is, kan je dit in pagina's opdelen door het commando 'more'. Let wel, dit commando werkt wel in de gewone PowerShell, maar niet in de PowerShell ISE. Om naar de volgende pagina te gaan, gebruik je de spatiebalk.
 - a. Test onderstaand commando in PowerShell en in PowerShell ISE. Get-Command | more
Zie je het verschil? → [Inpowershell ISE werkt dit niet. In Powershell worden de resultaten per pagina getoond.](#)

Verder is er reeds eerder sprake geweest van het filteren van de output, door gebruik te maken van door de parameter -CommandType met waarde cmdlet bij Get-Command worden per cmdlet alle eigenschappen getoond.

Nu is de enige eigenschap die getoond moet worden de naam van de cmdlet. Dit kan door het cmdlet Select-Object te gebruiken en dan te bepalen wat van dat object getoond moet worden. Volgens alle bovenstaande wordt dit dan het volgende commando:

Get-Command -CommandType cmdlet | Select -Object name



Opdracht: filteren van output met pipelines

1. Test dit uit en gebruik ook eens de alias.
Alias zoeken voor cmdlet get-command?
→ [Get-Alias -definition Get-Command](#)
→ [gcm -commandType cmdlet | Select -Object name](#)
Opmerking: opzoek naar de juiste alias? → [Get-Alias -definition Get-Command](#)
2. Gebruik eens de wildcard * om te controleren of je oorspronkelijk alle eigenschappen getoond krijgt.
→ [gcm -commandType cmdlet | Select -Object *](#)
Is er een verschil in het standaard formaat:
Get-Command -CommandType cmdlet
Get-Command -CommandType cmdlet | Select-Object *
[Bij de eerste krijg je een lijst met beperkte eigenschappen, bij de tweede ziet de output er heel anders uit, omdat je nu alle eigenschappen krijgt van elk cmdlet.](#)
3. Een andere manier om data te filteren is Where-Object. Zoek de werking hiervan uit. → [Help where-object](#) → [help Where-Object -Examples](#)

Je kan output ook wegschrijven naar een file. Hier kan je gebruik maken van pipe en set-context filename ofwel gebruik je het > teken.

Bij > wordt het bestand overschreven, bij >> wordt de tekst op het einde toegevoegd.



Opdracht: output wegschrijven met pipelines

1. Test volgende commando's:
2. `Get-Command -CommandType Cmdlet | Set-Content test.txt`
3. `Get-Command -CommandType Cmdlet > test1.txt`
4. Bekijk de 2 files of ze hetzelfde zijn.
5. In oefening 6 ga je op zoek naar een manier hoe je zelf die output format kan bepalen.

Met opmerkingen [LVH16]: Test = voorbeeld =/
opdracht

3.6 Operatoren

Er zijn basis operatoren die werken meestal zoals je zou verwachten. Je kan een opdracht gewoon aan de prompt typen en krijgt onmiddellijk de output in de terminal. Toch krijg je soms onverwachte output. Steeds testen is de boodschap.



Opdracht operatoren

Test eens en leg de output uit.

- | | |
|---|----------------|
| 1. <code>(5 + 3) * 2</code> | 1. 16 |
| 2. <code>"Hello" + " " + "World"</code> | 2. Hello World |
| 3. <code>10mb + 50kb</code> | 3. 10536960 |

Er zijn ook logische operatoren, deze ben je tegen gekomen in de opdracht Where-Object onderzoeken.

3.7 Variabelen

[Pluralsightlink](#)

Bij het onderdeel Pipes hebben we output onmiddellijk doorgegeven als input voor een andere cmdlet of commando. De output kan ook voor later gebruik bewaard worden.

Hiervoor worden variabelen gebruikt: dit wil zeggen er wordt een geheugenplaats gereserveerd en daar wordt de waarde, die we aan de variabele toekennen, opgeslagen.

Een variabele moet steeds met '\$' beginnen. Declaratie is niet nodig, type moet niet meegegeven worden.

Voorbeelden:

- \$bestandsNaam = "data.txt"
- \$bestandsInhoud = Get-Content \$bestandsNaam
- \$bestandsInhoud = Get-Item \$bestandsNaam | Get-Content

3.7.1 Speciale variabelen: Omgevingsvariabelen

Bij paden in hoofdstuk 2 is er reeds gesproken over de omgevingsvariabelen die ingesteld staan in Windows 10. In de opdrachtprompt kon je deze omgevingsvariabelen gebruiken om een pad samen stellen. Je gebruikte dan bijvoorbeeld %UserProfile% om in de gebruikersmap van de huidige gebruiker te komen.

In PowerShell werkt dit niet op deze manier. In bovenstaande wordt er verteld dat variabelen in PowerShell worden aangeduid met een \$ teken. Een omgevingsvariabele is een speciale soort variabele, maar ook deze begint met een \$. Omgeving in het Engels is: environment, hier gebruik je nu de afkorting env met een :

De gebruikersmap wordt dus: \$env:UserProfile



Opdracht: Omgevingsvariabelen

Type aan de prompt in PowerShell: \$env: en gebruik dan IntelliSense om te kijken welke mogelijkheden je allemaal krijgt.

→ [Een hele lijst met omgevingsvariabelen](#)

3.8 Tips & tricks

Om te testen wat je kennis is van PowerShell kan je in Pluralsight gebruik maken van de skills meter: [Pluralsightlink](#)

Zo kan je gericht op zoek naar extra studiemateriaal op Pluralsight.

3.9 Oefeningen

3.9.1 Oefening 1: ‘Een bestand op het bureaublad’

Situatie: Op het Bureaublad van de huidige gebruiker staat het bestand foto.gif.

Vraag: Geef het commando dat dit bestand als resultaat geeft.

Let op:

- Het commando moet werken los van de huidige map.
- Het commando moet werken hoe de gebruiker ook heet.

Oplossing:

Get-Item \$env:UserProfile\Desktop\foto.gif

Bestand openen: invoke-item \$env:UserProfile\Desktop\foto.gif

3.9.2 Oefening 2: ‘userprofiel’

Wat doet: Get-Item \$env:UserProfile\Documents*.docx

⇒ Geef alle Word bestanden uit de map Documents van de huidige gebruiker.

Zorg dat in je antwoord volgende deelvragen zijn beantwoord:

- Wat is *.docx?
→ .docx = Bestandsformaat/bestandsextensie voor een opgemaakte tekst en wordt het meest gebruikt door Microsoft Office Word)
* = wildcard
*.docx matcht alle items die eindigen op .docx, Op Windows is de Documenten map steeds Documents, los van de taalinstellingen
- Wat is \$env:UserProfile?
(Dit geeft het pad naar de profielfirectory)
→ \$env:UserProfile is hetzelfde als %UserProfile% in Windows paden, een omgevingsvariabele die verwijst naar de locatie van de eigen map van de huidige gebruiker, bijv. C:\Users\jansmit.
- Moet het Documenten of Documents zijn op een Nederlandstalige Windows 10?
→ Op Windows is de Documenten map steeds Documents, los van de taalinstellingen

3.9.3 Oefening 3: ‘zoekfunctie’

Vraag: Geef het commando om het volledige pad te tonen van alle gif-bestanden die zich bevinden in een submap; de naam van de submap moet eindigen op s of n.

→ (Get-Item *[sn]*.gif).FullName

→ Get-Item *[sn]*.gif | Select-Object FullName

Er is een verschil in de format van de output.

3.9.4 Oefening 4: ‘info opvragen’

Vraag: Geef de commando's om:

- info.txt van de huidige map
 - te kopiëren naar een USB-stick op F:
 - te verplaatsen naar de documenten-map

- oldmeme.gif te verwijderen

Oplossing:

Copy-Item info.txt F:\

Move-Item info.txt \$env:UserProfile\Documents\

Rename-Item dieren.search-ms dieren.xml

Remove-Item oldmeme.gif

3.9.5 Oefening 5: 'test cmdlets'

Wat is het resultaat van deze opeenvolgende commando's?

Wat doet elk commando?

- Set-Location \$env:UserProfile\Documents
- Set-Content 'een regel tekst' data.txt
- Set-Location ..
- Get-Content data.txt
- Get-ChildItem Documents

Oplossing:

1. Verandert de huidige map in de map Documenten van de huidige gebruiker, bijv. pad: C:\Users\mijnnaam\Documents

2. Zet data.txt als de inhoud van het bestand genaamd 'een regel tekst' (zonder quotes) en maakt dit bestand indien het nog niet bestaat. Bijv. pad: C:\Users\mijnnaam\Documents\een regel tekst

3. Verander de huidige map in de bovenliggende map; de huidige map wordt bijv. pad C:\Users\mijnnaam

4. Leest de inhoud van het bestand data.txt en toont deze als resultaat, bijv. pad: C:\Users\mijnnaam\data.txt

5. Toont de inhoud van de map Documents, dus bijv. de inhoud van het pad: C:\Users\mijnnaam\Documents. Eén van de bestanden hierin zal 'een regel tekst' zijn.

Analyse. Mogelijk heeft de maker van deze commando's twee fouten gemaakt: 1 'een regel tekst' en data.txt omgewisseld in het Set-Content commando; 2 in het Get-Content commando niet Documents\ voor data.txt gezet

3.9.6 Oefening 6: 'grid'

- Er is een commando met Grid in de naam. Zoek het. Kennen!
`Get-command *grid*`
- Bekijk de help.
`Help Out-GridView`
- Voer uit: `Get-Item * | <hier het commando>`
`Get-Item | Out-GridView`

Hiermee kan je:

- zoeken, filteren en sorteren
- kolommen verbergen en toevoegen (RMK op kolom)
- selecteren en kopiëren naar Excel, kladblok, ...

Oplossing:

- ⇒ `Get-Commmand -Noun *Grid*` geeft Out-GridView
- ⇒ "Sends output to an interactive table in a separate window."
- ⇒ `Get-Item * | Out-GridView`

3.9.7 Oefening 7 'locatie wijzigen'

- Voer uit: `Get-ChildItem`
- Voer uit: `Set-Location ..` en dan `Get-ChildItem`
- Geef het commando om de huidige map te wijzigen in C:\ [Set-Location C:](#)
- Geef het commando om de huidige map te wijzigen van C:\ naar de map Bureaublad
[Set-Location \\$env:UserProfile\Desktop](#)
- Voer uit: `Set-Content powershell.txt 'mijn tekst'`
→ [Het bestand powershell.txt wordt gemaakt in mijn bureaublad.](#)
- Voer uit: `Get-Content powershell.txt`
[De inhoud wordt getoond: mijn tekst](#)

3.9.8 Oefening 8: 'ls'

- Voer uit: `(ls).CreationTime`
⇒ Niet nuttig: de naam wordt niet getoond.
- Voer uit: `ls | Select-Object Name, CreationTime`
⇒ `Select-Object` laat toe om zelf de eigenschappen te kiezen
⇒ Alias: `select`
- Voer uit: `ls | select Name, CreationTime | ft -AutoSize`
⇒ `ft` is een alias voor `Format-Table`
- Alternatief: `ls | ft Name, CreationTime -AutoSize`

3.9.9 Oefening 9

- Voer uit: `Set-Content test.txt ".."`
- Voer uit: `$files = ls -File` (`ls` = alias voor `get-childitem` → `get-childitem -file`)
 - `-File` zorgt dat enkel bestanden worden teruggegeven
Test ook=
 - ⇒ `Get-childitem -filter 'test*'`
 - ⇒ `Get-ChildItem -attributes hidden`
 - ⇒ `Get-ChildItem -directory`

- C. Voer uit: `$files = $files | sort CreationTime`
 - `sort` is een alias voor `Sort-Object`
- D. Voer uit: `$file = $files[$files.Count - 1]`
 - Alternatief: `$file = $files | select -Last 1`
- E. Voer uit: `Get-Item (cat $file)`

3.9.10 Oefening 10

- A. Voer uit: `ls -Force | select -Unique Extension`
- B. Voer uit: `ls -Recurse | ft -GroupBy Extension`
- C. Wat doet:
`man ls -Parameter * | select -Last 3 -Skip 2`

3.9.11 Oefening 11 (Uitbreiding)

Geef het commando om:

1. Een tekstbestand te maken op het Bureaublad met erin het volledige pad van de 5 laatst gewijzigde mappen in %AppData%. Het pad van de laatste gewijzigde map komt op de eerste regel te staan.
2. De naam van het bestand is de naam van de huidige gebruiker (automatisch) gevolgd door spatie, appdata.txt
 Bijv. Gebruikersnaam Jeff -> Bestandsnaam: Jeff appdata.txt
3. Geef de commando's om:
4. Een map genaamd kopie te maken op het Bureaublad.
5. De mappen, opgesomd met hun pad in het bestand appdata.txt, met hun inhoud te kopiëren naar de map kopie.

Mogelijke oplossing:

1. `(ls $env:APPDATA -Directory | sort LastAccessTime -Descending | select -first 5).FullName | Set-Content "$env:USERPROFILE\Desktop\$env:UserName appdata.txt"`
 Of `(Get-ChildItem $env:APPDATA -Directory | sort-object LastAccessTime -Descending | select-object -first 5).FullName | Set-Content "$env:USERPROFILE\Desktop\$env:UserName appdata.txt"`
4. `New-Item -Type Directory $env:USERPROFILE\Desktop\kopie`
5. `Get-Content "$env:USERPROFILE\Desktop\$env:UserName appdata.txt" | Copy-Item -Recurse -Destination $env:USERPROFILE\Desktop\kopie`

```
Get-ChildItem -Path $env:APPDATA | sort-object -Descending -Property LastWriteTime
| Select-Object -First 5 -Property name, fullname, lastwritetime | Set-Content -
Path "$env:USERPROFILE\Desktop\$env:UserName appdata.txt"
```

Met opmerkingen [LVH17]: Toevoegen: extra oefeningen en voorbeelden van scripts

|