handboek

PowerShell

ISBN: 978-94-6356-357-4
NUR: 985
Trefw.: PowerShell, scripting, programmeren, besturingsprogramma

Omslag en opmaak: Van Duuren Media, Culemborg/Barcelona
Vormgeving (concept): Aad Metz, De Meern

Eerste druk: augustus 2024

Dit boek is gedrukt op een papiersoort die niet met chloorhoudende chemicaliën is gebleekt. Hierdoor is de productie van dit boek minder belastend voor het milieu.

Inhoud

Opmerkingen vooraf

- Dit boek bestaat uit meerdere delen, zodat je PowerShell op een gestructureerde manier kunt aanleren.
- De delen van het boek bestaan niet uit hoofdstukken, maar uit onderwerpen die elkaar logisch opvolgen. Het is belangrijk – indien PowerShell nieuw voor je is – om deze volgorde aan te houden.
- Tal van oefeningen komen aan bod aan de hand van instructies. Deze instructies kunnen één keer in een oefening voorkomen of in verschillende oefeningen gebruikt worden. Voldoende herhaling is belangrijk om PowerShell beter te begrijpen. Iedere instructie heeft zijn doel, waardoor het belangrijk is deze eventueel zelf kort te documenteren.
- Tijdens het doornemen zal het woord *cmdlet* heel veel gebruikt worden. Dit woord wordt in de PowerShell-scripttaal gebruikt om een commando aan te geven dat zich letterlijk als PowerShell-cmdlet of -functie zal gedragen.
- Wanneer alle oefeningen voltooid zijn en het behandelde begrepen is, ben je in staat om PowerShell te gebruiken om zaken te automatiseren in de verschillende toepassingen waar PowerShell voor gebruikt wordt.
- Deel 1 van het boek kan te allen tijde worden geraadpleegd indien er vragen zijn over hoe een bepaalde situatie het best wordt aangepakt.
- De focus wordt gelegd op instructies begrijpen om deze vervolgens handmatig uit te voeren of op te nemen in een bestaand script dat wordt bewerkt.
- In de verschillende delen wordt gebruikgemaakt van de besturingssystemen Windows Enterprise en Windows Server. Indien een ander besturingssysteem gebruikt wordt, wordt dit aangegeven. Een nieuwere Windows-versie zou geen probleem mogen vormen.

Van beide versies zijn voor dit boek de Engelse edities gedownload vanaf het Windows Evaluation Center (**www.microsoft.com/en-us/evalcenter/**). Hierbij zijn de versies niet geactiveerd, omdat ze enkel voor testdoeleinden zijn gebruikt. Windows kan ook zonder activering de noodzakelijke updates blijven ophalen, indien gewenst.

Let op: wanneer de opdrachten niet uitgevoerd worden met deze versies, kan dit leiden tot andere resultaten.

Vereiste kennis en benodigdheden

Extra studiemateriaal (zoals extra oefeningen en de benodigde bestanden) is te vinden op de website van uitgeverij Van Duuren Media, of via de snelle koppeling **mijn.cc/powershell3**.

Om dit boek volledig door te nemen, is de volgende basiskennis vereist:

- Een virtueel besturingssysteem kunnen installeren met virtualisatiesoftware. Indien deze kennis niet (voldoende) voorhanden is kan de instructie aan het begin van deel 1 uitkomst bieden.
- Kennis hebben van netwerkinstellingen, zoals IP-adressering, DNS, DHCP, DC enzovoort.

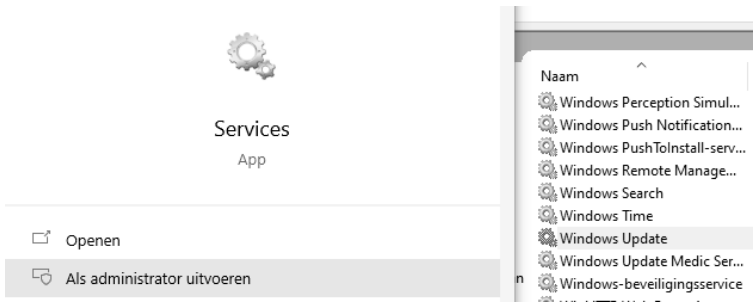
- De basis weten van het grafisch opzetten van een domein binnen Windows Server, waarbij een DC met een client kan communiceren.

Andere voorkennis is niet nodig.

Om dit boek volledig door te nemen, is een systeem met de volgende specificaties vereist:

- Virtualisatiemogelijkheid (eventueel in te schakelen in BIOS/UEFI).
- VMware, VirtualBox, Hyper-V of andere virtualisatiesoftware.
- Minimaal 8 GB RAM, bij voorkeur meer.
- Tussen de 60 GB en 100 GB vrije schijfruimte, bij voorkeur op een (externe) SSD.

Opmerking: om schijfruimte te beperken kan het handig zijn onmiddellijk na installatie de automatische updates op de te gebruiken virtuele machines uit te schakelen. Soms starten deze na een tijd toch nog op. Open daartoe de app Services als administrator:



Dubbelklik op Windows Update en stop de service eerst. Wijzig vervolgens de optie **Opstart-type** in **Uitgeschakeld**:

Vanzelfsprekend kan dit ook via PowerShell (open PowerShell als administrator):


```
PS C:\Windows\system32> stop-service wuauserv
PS C:\Windows\system32> set-service wuauserv -StartupType Disabled
```

Mocht Windows melden dat de testperiode van 90 dagen verlopen is, open PowerShell dan als administrator en voer uit: `slmgr -rearm`. Hierna wordt de pc herstart en geeft deze opnieuw een testperiode van meestal 90 dagen.

Merk je dat een bepaalde link niet meer werkt en je hebt deze zeker nodig (bijvoorbeeld als Microsoft Evaluation Center niet bereikbaar is, wat enkele jaren geleden is voorgekomen), gebruik dan **archive.org** om de link op te sporen.

Hoe gebruik je dit boek?

Dit boek kan op drie manieren doorgenomen worden, afhankelijk van welke PowerShell-kennis al aanwezig is.

Optie 1: onderwerp na onderwerp

Deel 1 is erop gericht onderwerp na onderwerp de PowerShell-kennis op te bouwen. Als beginner wordt het beste met dit deel gestart. Later zal deel 1 hoofdzakelijk gebruikt worden als naslagwerk tijdens het doorwerken van de overige delen van het boek, maar ook later bij het zelfstandig of in een team werken met PowerShell. Na deel 1 ga je aansluitend in volgorde verder met de resterende delen.

Optie 2: meteen starten met het uitbouwen van een domein

Deel 4 omvat het uitbouwen van een Windows-netwerk, waarbij zo veel mogelijk taken via PowerShell verricht worden. Na verschillende systemen on-premises te hebben opgezet, wordt overgestapt naar de cloud, waarbij onder andere Microsoft 365, Teams en SharePoint aan bod komen. Als extra wordt ook een introductie op Docker en Chocolatey behandeld.

Indien meteen gestart wordt met deel 4, is het handig toch deel 1 door te nemen tot en met het onderwerp *14. Opbouwen van een stevige instructie*.

Na het afronden van deze optie, kun je verder gaan met een van de andere delen naar keuze. Met deze optie ben je meteen zeer praktijkgericht gestart.

Optie 3: uit het boek halen wat nodig is

Bij deze optie wordt willekeurig geselecteerd wanneer wat doorgenomen wordt.

- Deel 1: onderwerp na onderwerp de PowerShell-kennis opbouwen.
- Deel 2: Visual Studio Code (VSC) met PowerShell Core leren gebruiken.
- Deel 3: PowerShell Core gebruiken in Linux met eventuele Windows-verbindingen. Zowel Debian als Red Hat Linux komen aan bod.
- Deel 4: uitbouwen van een Windows-domein met toepassing naar de cloud en als extra een introductie op Docker en Chocolatey.
- Deel 5: de bouwstenen van een eigen PowerShell-module met de verschillende test-mogelijkheden voor foutopvang.
- Deel 6: een groot aantal extra's die binnen PowerShell gebruikt kunnen worden, waarbij verwezen wordt naar de vele titels die bij dit deel in de inhoudsopgave staan.

Wat betekent dat kromme pijltje in de scriptcode?

Wanneer een coderegel vooraf wordt gegaan door het teken ➥, betekent dit dat de regel plus alles wat daaronder ingesprongen wordt weergegeven, op één regel achter elkaar moet worden doorgetypt. Druk aan het einde van de eerste regel dus niet op Enter, maar doe dat pas aan het eind van de instructie.

Tot slot

Alle oefeningen en voorbeelden in dit boek zijn meerdere malen getest om fouten tegen te gaan. Indien toch tegen een fout werd aangelopen, werd een remedie gezocht en deze beschreven bij de bijhorende instructies.

Binnen het extra studiemateriaal is ook een .txt-bestand per boekdeel aanwezig met alle lange instructies die in dat deel voorkomen om typfouten tegen te gaan. Om eenvoudig te werken met dit bestand, knip je steeds de regels die je op dat moment nodig hebt en plak je deze binnen PowerShell (ISE of VS Code). Vind je niet meteen de goede instructieregel, gebruik dan Ctrl+F en geef een stukje uit de instructieregel op om de volledige regel op te zoeken. Werkt een gekopieerde instructie niet, dan zou dit kunnen liggen aan het formaat van de aanhalingstekens (“ ” versus " " en ‘ ’ versus ' '). PowerShell verwacht ‘rechte’ aanhalingstekens.

Veel succes!

1

Deel

Wegwijs in PowerShell

Introductie

PowerShell is een objectgeoriënteerde shell- en scripttaal voor Microsoft Windows met als doel taken te automatiseren. Stel dat een zelfde taak 200 keer per jaar moet worden uitgevoerd en dat die taak tien minuten kost. Wat is er dan beter? Ieder jaar die 200 taken handmatig uitvoeren of even tijd stoppen in een script om de taak te automatiseren? Dit script moet zowel voor mens als pc te begrijpen zijn en net hiervoor dient PowerShell. Het script omvat een uitlijning hoe een taak perfect moet worden uitgevoerd. Dit betekent dat die taak steeds opnieuw op dezelfde manier wordt uitgevoerd en dus veiliger is dan dat de mens dit handmatig zou doen.

PowerShell stamt uit 2006 en er zijn twee verschillende versies: 5.1 en 7.X. Deze twee versies slaan op de klassieke Windows PowerShell (5.1) en PowerShell (7.X). Deze laatste zal in de toekomst vooral gebruikt en als enige uitgebreid/vernieuwd worden, terwijl de klassieke enkel nog aanpassingen zal krijgen voor veiligheid (bijvoorbeeld kwetsbaarheden oplossen). Sommige oudere commando's zijn in de huidige PowerShell niet meer mogelijk, maar de basisprincipes van PowerShell zijn zo goed als onveranderd sinds versie 2.

De klassieke Windows PowerShell is gebaseerd op het .NET-Framework en PowerShell op .NET Core, waardoor beide steeds up-to-date moeten zijn.

.NET is een framework ontwikkeld door Microsoft en biedt een ontwikkelaarsplatform voor applicaties met bibliotheken en tools waarvan ontwikkelaars gebruik kunnen maken om services makkelijker, sneller en betrouwbaarder te gebruiken. .NET Framework is de oudste en is closed source. Het bestaat uit:

- de CLR (Common Language Runtime) die gebruikt wordt om apps uit te voeren en het geheugengebruik in de gaten te houden;
- een grote bibliotheek met betrouwbare, te hergebruiken code voor ontwikkelaars.

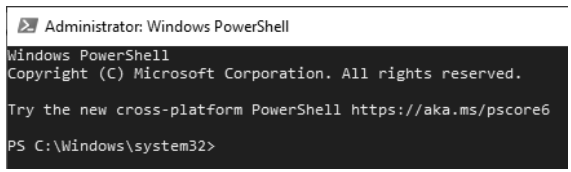
Omdat het closed source is, heeft Microsoft .NET Core opgericht als opensourcevariant, waardoor het veel meer interesse wekt bij ontwikkelaars en eveneens omdat het cross-platform gebruikt kan worden over meerdere besturingssystemen heen (Windows, Linux, Mac). Alles wordt ingezet op .NET Core, waardoor deze versie de nieuwste, maar ook de beste features van het .NET-framework moet omvatten. De nieuwe PowerShell had hierdoor ook eerst als naam PowerShell Core. En die naam wordt in dit boek aangehouden om geen verwarring te stichten met de klassieke Windows PowerShell.

PowerShell maakt gebruik van cmdlets (commandlets). De maker, Jeffrey Snover, heeft de term cmdlet in het leven geroepen zodat via een zoekmachine (bijvoorbeeld Google) steeds relevante PowerShell-informatie terug wordt gegeven.

PowerShell is zo ontworpen dat het makkelijk is om aan te leren en ook eenvoudig is om scripts te lezen vanwege het gebruik van cmdlets. Waar zit dan de valkuil? PowerShell kan in héél veel omgevingen gebruikt worden, bijvoorbeeld Azure en Microsoft 365. Het is net in die omgevingen dat PowerShell moeilijker wordt om te gebruiken en te begrijpen.

De cmdlets die in PowerShell gebruikt worden, hebben vaak een duidelijke naam die doen wat de naam zegt te doen. Met de komst van PowerShell Core zal een zelfde cmdlet tot dezelfde uitvoer komen, maar waarschijnlijk op een iets andere manier, afhankelijk van het besturingssysteem. Die iets andere manier maakt voor de gebruiker niet uit, aangezien de uitvoer hetzelfde blijft.

PowerShell is heel krachtig en kan zelfs meer dan wat handmatig met een pc ook zou kunnen. Het is belangrijk om PowerShell steeds als administrator te starten. Klik met de rechtermuisknop en kies **Als administrator uitvoeren**. Bij het openen wordt het woord Administrator in de titelbalk getoond. Sinds Windows 11 heeft PowerShell een zwarte achtergrondkleur.



PS staat voor PowerShell. Huidige locatie is hier C:.

De bestandslocatie van Windows PowerShell op een 64-bits pc:

- x86: %SystemRoot%\syswow64\WindowsPowerShell\v1.0
- x64: %SystemRoot%\System32\WindowsPowerShell\v1.0

De Windows-systeemmap syswow64 dient voor 32-bits systeembestanden.

Waarom 32- en 64-bits?

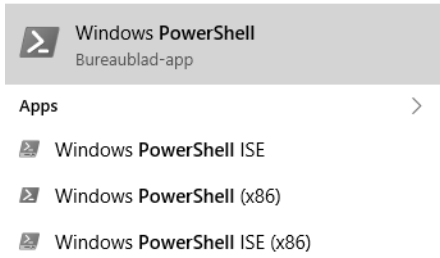
Beide versies zijn nodig. Stel dat een 32-bits besturingssysteem aangestuurd moet worden, is het belangrijk te werken met de 32-bits (x86)-versie. Het is zelfs noodzakelijk om na te gaan als een script niet werkt op een 32-bits systeem of dit niet ligt aan de gebruikte instructie. In een instructie worden cmdlets en commando's toegepast en het zou kunnen dat deze niet uit te voeren zijn via 32-bits.

Gelukkig wordt 32-bits op moderne systemen steeds minder gebruikt en zijn er maar heel weinig instructies die niet zullen slagen om uitgevoerd te worden op een 32-bits systeem, maar het is wel belangrijk hier rekening mee te houden als deze situatie zich voordoet.

Een voorbeeld is het cloudmanagementplatform voor Mobile Device Management (MDM) van Microsoft: Intune. Ondernemingen kunnen Intune gebruiken om onder andere alle Windows-systemen te beheren en zo ook PowerShell-scripts te activeren die moeten worden uitgevoerd. Hiertoe wordt op ieder toestel Intune Management Extension geïnstalleerd, wat een 32-bits Intune-hulpprogramma is. Hiertoe kan dit hulpprogramma PowerShell-scripts op een 32-bits systeem uitvoeren met de 32-bits versie van PowerShell. Bij een 64-bits systeem zal het hulpprogramma meestal de PowerShell-scripts uitvoeren via de 64-bits versie van PowerShell.

Via het menu Start wordt in Windows standaard vier maal PowerShell (twee keer 64-bits en twee keer 32-bits) gevonden met twee kerncomponenten:

- Windows PowerShell – CLI – Command Line Interface (= Shell)
- Windows PowerShell ISE – GUI – Graphical User Interface (= IDE-editor)



Naast deze zijn er ook door andere firma's uitgebrachte PowerShell-editors, waarvan één eveneens door Microsoft gemaakt: Visual Studio Code. Meer zelfs: Microsoft dwingt iedereen over te stappen naar Visual Studio Code, omdat de actieve ondersteuning voor PowerShell ISE is weggevallen en Visual Studio Code cross-platform werkt.

Surf naar de website die aangegeven staat bij het openen van PowerShell om PowerShell Core te installeren. Op deze website vind je de nodige info om PowerShell Core te installeren, maar ook om oudere PowerShell-versies voor Windows te installeren.

Zo wordt geleerd dat PowerShell sinds Windows 7 standaard mee geïnstalleerd wordt via een WMF-installatiepakket bij de Windows-installatie. WMF staat voor Windows Management Framework en bevat allerlei bronnen die als systeembeheerder geraadpleegd kunnen worden, zoals PowerShell, WinRM voor CLI op afstand, WMI- en CIM-bevragingen om onder meer hardware aan te spreken enzovoort.

De installatie van PowerShell Core en het gebruik van Visual Studio Code komt pas later aan bod. Eerst wordt alles uitgevoerd in Windows PowerShell 5.1 en Windows PowerShell ISE.

Voer uit:

```
$psversiontable
```

Deze uitvoer vertelt welke versie en versienummer gebruikt wordt.

Opmerking

Onderzoek wijst uit dat beginners en experts in PowerShell ongeveer dezelfde kennis hebben in theorie, maar in praktijk kunnen experts veel sneller problemen oplossen.

Opmerkingen bij de oefeningen (virtualisatiehulp op volgende pagina)

Maak twee nieuwe VM's (virtuele machines: client en server) in de door jou gekozen virtualisatiesoftware. Hiervoor worden de Evaluation-versies gebruikt die Windows gratis aanbiedt:

- De nieuwste versie van Windows Enterprise
- De nieuwste versie van Windows Server – Standard Evaluation Desktop Experience

Downloaden kan vanaf **microsoft.com/en-us/evalcenter**. Kies altijd het Engelstalige .ISO-bestand en zorg dat tijdens het downloaden en de installatie de internetverbinding actief blijft!

De VM's bevatten elk 50 GB (dynamisch) opslagruimte en de NAT-NIC-instelling. De server bevat 4 GB RAM met twee virtuele processorkernen en de client 2 GB RAM. Op beide machines mogen de additions/VMware Tools geïnstalleerd worden. Bij beide is de taal Engels (US). Time- & Currency en de keyboardoptie wordt volgens wat de gebruiker standaard gebruikt, ingesteld.

Bij de twee installaties wordt **Custom** gekozen en de volledige schijfruimte van 50 GB.

Het adminaccount op de server krijgt het wachtwoord Server* net na installatie en bij de blauwe balk van Network wordt **Yes** geselecteerd, zodat Network discovery in orde is. Na installatie wordt bij de client de gebruikersregio gekozen met de gepaste keyboardlay-out. Vervolgens: **domain join instead** linksonder waarbij gebruiker `ict` wordt aanmaakt met wachtwoord `Server*`. Beantwoord de drie vragen met willekeurig lange waarden voor de veiligheid.

Server*

Het wachtwoord Server* zal in dit boek vaak aan bod komen. Als opmerking hierbij kan gegeven worden dat * geen goed teken is om te gebruiken, omdat * ook als jokerteken dienstdoet, waardoor een onderliggend systeem dit teken verkeerd kan interpreteren. Maar hedendaagse systemen moeten hiermee om kunnen gaan en Server* heeft twee voordelen: het is snel te typen en makkelijk te onthouden als wachtwoord voor testscenario's.

Vervolgens wordt op alle vragen de tweede keuze gekozen (meestal **No**). De installaties zijn dan voltooid! Maar eerst: sluit beide machines af en maak een snapshot. Installeer hierna de VM-additions/VMware Tools (typical) op beide VM's, sluit opnieuw af en maak snapshots.

- PowerShell en PowerShell ISE worden altijd gestart als administrator.
- Indien een pc-naam, gebruikersnaam enzovoort getypt moet worden, wordt verwacht dat dit een logisch (= bestaand) item is. In dit boek wordt dat dan onderstreept weergegeven, bijvoorbeeld: pcnaam. Het is ook mogelijk dat een stuk *cursief* staat, wat wijst op optionele invoer. Bijvoorbeeld: -credential *werkgroep/gebruikersnaam*
- Alle oefeningen kunnen op de client of de server worden uitgevoerd, tenzij anders vermeld.
- Alle bestanden worden bewaard in de mappen aangegeven tijdens de oefeningen.
- Als iets uitgevoerd moet worden in PowerShell Core, dan zal letterlijk PowerShell Core in de opdracht staan.

Virtualisatiesoftware: Oracle VirtualBox

Download deze software vanaf **virtualbox.org**. VirtualBox is de ideale gratis virtualisatiesoftware om testen uit te voeren via virtuele machines. Bij deze versie kan makkelijk gebruik worden gemaakt van snapshots, waardoor de toestand van een VM (virtuele machine) kan worden opgeslagen.

Na installatie van VirtualBox, wordt ook de VM VirtualBox Extension Pack geïnstalleerd (eveneens te downloaden vanaf **virtualbox.org**), wat nog meer opties toevoegt aan het reeds geïnstalleerde pakket. Bij een update of andere versie-installatie van VirtualBox moet de Extension Pack opnieuw gedownload en geïnstalleerd worden naar de gewijzigde versie.

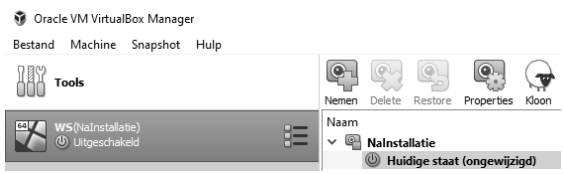
Bij het eerste maal openen van VirtualBox: klik op **Bestand**, **Voorkeuren** en kies de standaardmap waar de virtuele machines opgeslagen moeten worden (bij voorkeur op een SSD).

Na het installeren van een besturingssysteem op een VM, kunnen de Guest Additions toegevoegd worden aan de VM. Deze bieden tal van extra mogelijkheden voor de VM (guest) als onderdeel van de fysieke pc (host).

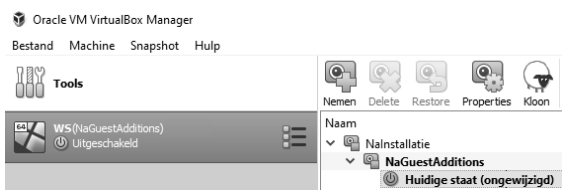
Werken met snapshots (momentopnames van de virtuele machine) wordt als volgt gedaan:

Klik op de knop en kies **Snapshots**. Vervolgens worden andere menuknoppen getoond. Vanaf nu kan gekozen worden de volgende instructies mee te doen of te begrijpen wat gebeurt aan de hand van de afbeeldingen. Klik op **Nemen** voor een snapshot van de huidige staat.

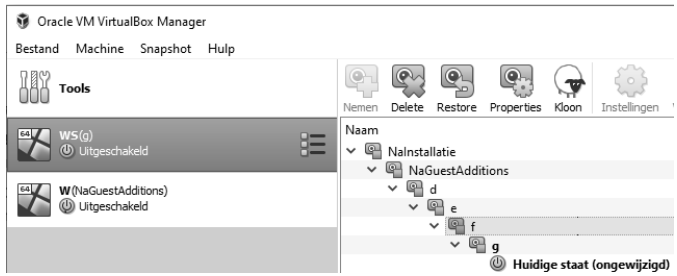
Huidige staat is geselecteerd. Klik op Nemen waarbij de snapshot NaInstallatie wordt genoemd.



De snapshot noemen we NaInstallatie.



Na GuestAdditions te hebben geïnstalleerd, wordt opnieuw een snapshot van de huidige staat gemaakt. Voordat een nieuwe snapshot wordt gemaakt, is het verstandig de VM te herstarten.



Om nu alle mogelijkheden uit te leggen, is er per snapshot één tekstbestand extra op het VM-bureaublad geplaatst met als titel de letter die hetzelfde is als de snapshotnaam.

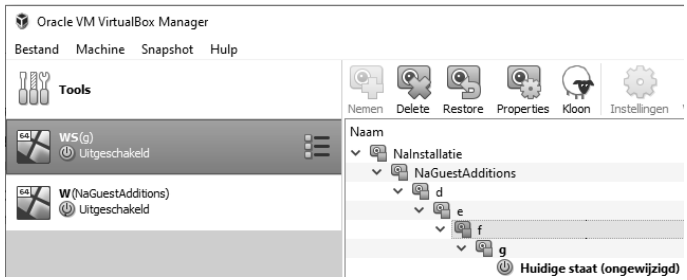
De huidige staat bevat dus momenteel een bureaublad met tekstbestanden d, e, f en g.txt. Klik met de rechtermuisknop op snapshot f en klik op **Restore**. Het volgende gebeurt:

Snapshot g bestaat nog, maar de huidige staat is aangepast naar de momentopname van snapshot f.

Snapshot g bestaat nog, maar de huidige staat is aangepast naar de momentopname van snapshot f. Na het starten vanaf de huidige staat, is tekstbestand g.txt verdwenen vanaf het VM-bureaublad. De VM wordt opnieuw afgesloten.

Er wordt nu hersteld vanaf snapshot g. Er wordt gevraagd of er nog eerst een snapshot genomen moet worden van de huidige staat (dit kan, maar is nu niet nodig).

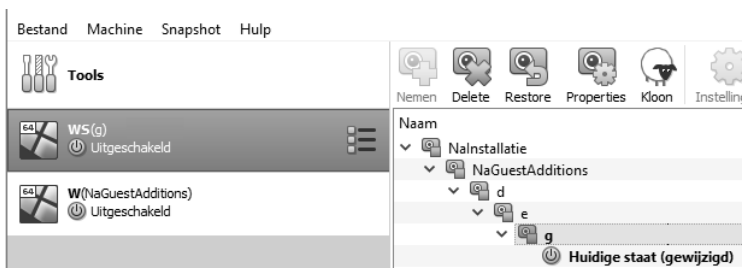
Schakel dit vinkje uit.



Bij het openen van de nieuwe huidige staat, staat g.txt opnieuw op het VM-bureaublad.

Nu wordt gekozen om snapshot f te verwijderen. Klik op **Verwijderen**. Eventueel kan het lang duren voordat het verwijderen is voltooid. In dit geval is dit niet zo, aangezien de snapshots amper iets bevatten.

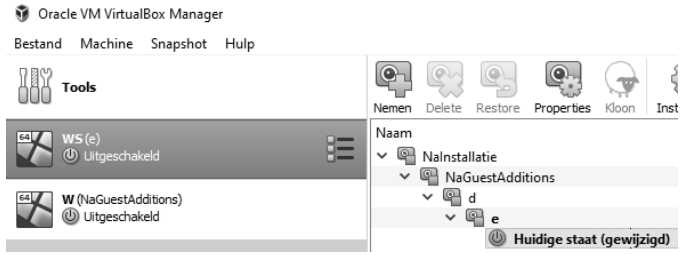
Klik op Verwijderen.



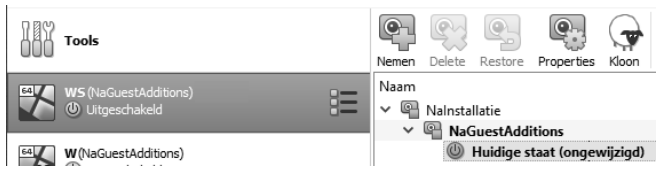
Snapshot f is nu verdwenen.

De huidige staat blijft hetzelfde: alle .txt-bestanden staan nog op het VM-bureaublad.

Stel dat nu snapshot g wordt verwijderd, dan blijft de huidige staat hetzelfde met alle .txt-bestanden op het VM-bureaublad, tot wanneer een restore naar een eerdere toestand heeft plaatsgevonden.



Nu is het tijd om terug te keren naar de oorspronkelijke staat: herstel vanaf NaGuestAdditions en verwijder daarna de overgebleven snapshots d en e, indien dit mee uitgevoerd werd.



De VM opnieuw starten kan enkel en alleen vanaf de huidige staat! Bij opstart zal gemerkt worden dat geen van de tekstbestanden nog aanwezig is.

Het kan handig zijn om eerst de volledige VM-map te back-uppen, voor terug te keren naar een bepaalde situatie. Ga hiervoor in Verkenner naar de locatie van de VM.

VirtualBox heeft soms de volgende eigenaardigheden:

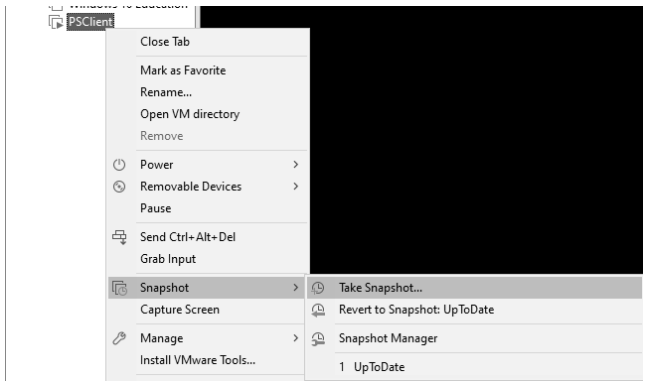
- na een tijd kunnen de VM's steeds trager werken;
- na een update van VirtualBox kunnen de VM's die al bestonden hallucinant traag worden.

Hopelijk zijn deze problemen met de nieuwere versies van VirtualBox verleden tijd. Eventueel kan het volgende worden toegepast: verwijder VirtualBox zonder de VM's en installeer de meest recente versie opnieuw. Daarna kun je een nieuwe VM maken en de virtuele vaste schijf van de oude VM hieraan koppelen.

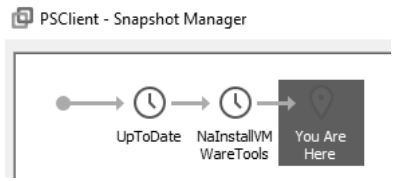
Virtualisatiesoftware: VMware

Naast VirtualBox is ook VMware bekende virtualisatiesoftware. VMware Workstation Pro heeft ook de mogelijkheid om snapshots te maken. In 2024 werd deze versie voor thuisgebruik gratis, waardoor je ook voor deze software kunt kiezen om te virtualiseren.

Voor VMware komt het gebruik van snapshots op hetzelfde neer. Klik met de rechtermuisknop op de naam van de virtuele machine en kies **Snapshot**.

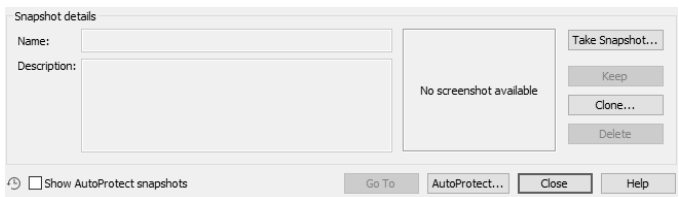


Hier kan een snapshot genomen worden of kan via de Snapshot Manager teruggekeerd worden naar een eerdere snapshot.



Snapshot Manager.

Onderaan staan enkele knoppen die zeer interessant zijn.



Opties voor snapshots.

- **Clone** Letterlijk een kopie maken vanaf een toestand.
- **Go To** Als geklikt wordt op een eerdere snapshot, is dit de knop om terug te keren naar die snapshottoestand.
- **AutoProtect** Dit is een functie die dagelijks het zelf ingestelde aantal snapshots automatisch aanmaakt. Een soort van back-up, die standaard niet wordt geactiveerd.

Virtualisatiesoftware: Hyper-V

Op een Windows-computer staat deze optie klaar om geactiveerd te worden. Aangezien Hyper-V geen software van derden is, zou de voorkeur hier naar uit moeten gaan. Waarom dan toch niet? Om Hyper-V correct met dit boek te gebruiken moet een eigen NAT-router aan-

gemaakt worden. Als je hier geen ervaring mee hebt, dan is het af te raden om met Hyper-V aan de slag te gaan. Daarom worden in het vervolg van het boek enkel acties binnen VMware en/of VirtualBox aangehaald.

Wil je toch aan de slag met Hyper-V? Open **Windows-onderdelen in- of uitschakelen** in de app Instellingen. Zet een vinkje bij **Hyper-V** en herstart de computer. Vervolgens kun je Hyper-V starten via Hyper-V-beheer. Ook met dit programma kun je snapshots (= controlepunten) maken.

Na de Hyper-V-activatie zul je een NAT-router moeten aanmaken alvorens een VM te starten.

Starten met virtualisatie

Mochten er intussen verschillende virtualisatieoplossingen op de computer staan, dan vormt dit geen probleem meer zoals vroeger. Loop je toch nog tegen problemen aan? Zet dan het vinkje aan bij **Windows Hypervisor-platform** in het configuratie-onderdeel Windows-onderdelen in- of uitschakelen. Herstart vervolgens de pc.

Open de gekozen virtualisatiesoftware en maak een nieuwe VM aan waarbij enkele keuzes worden gemaakt. De volgende keuzes zullen meestal in volgorde per virtualisatiesoftware verlopen:

- Eerst wordt de Windows-versie gekozen voor installatie. Staat hier geen 64-bits bij, dan moet waarschijnlijk nog de hardwarevirtualisatie aangevinkt worden in de BIOS- of UEFI-instellingen van de CPU bij het booten van de host (het fysieke toestel).
- Daarna moet de keuze in RAM gemaakt worden.
- Vervolgens moeten de keuzes betreffende de schijf worden gemaakt. Kies voor dynamische toewijzing. Dit betekent dat de schijfruimte op de fysieke schijf enkel dat zal innemen wat op de virtuele schijf is gebruikt. Stel dat dit 16 GB is na installatie van Windows op de VM, dan wordt er op de fysieke schijf maar 16 GB gebruikt. Via de andere optie wordt meteen gekozen om 50 GB van de fysieke schijf te benutten.

Klik na het maken van de VM op de naam ervan en kies Instellingen, waarbij volgende zaken worden gecontroleerd (de meeste instellingen kunnen ook gewijzigd worden terwijl de VM in gebruik is):

- Instellen RAM.
- Processor: eventueel meer kernen geven, dit kan opgezocht worden via Windows Taakbeheer, Prestaties en bekijk hoeveel kernen er gebruikt worden (op de tab Processor). Deel het aantal kernen door 2 om hierop af te stemmen. Bijvoorbeeld vier kernen betekent maximaal twee virtuele CPU's uitdelen aan de VM.
- Voeg het .iso-bestand toe om te koppelen zodat het besturingssysteem wordt geïnstalleerd.
- Netwerk inschakelen op NAT of NAT-netwerk: deze optie zorgt ervoor dat de virtuele netwerkkaart een IP-adres zal krijgen van de zogeheten virtuele NAT-router, waardoor de VM zich niet in hetzelfde netwerk zal bevinden als de fysieke pc. Veiligheidsbewustzijn! Via een bridge zal de VM zich wel in het fysieke netwerk bevinden.

Voor moderne besturingssystemen moet misschien gekozen worden om een virtuele TPM aan te maken. Deze opmerking kun je krijgen bij het opstarten van de virtuele machine. In de virtualisatiesoftware zul je vervolgens de instelling om een virtuele TPM te gebruiken moeten activeren. De werking hiervan verschilt per virtualisatiesoftware. Dit zoek je het beste even online op.

Na dit te lezen en uit te voeren, staan de virtuele machines klaar om mee aan de slag te gaan! Na installatie van Windows Server wordt gevraagd om Ctrl+Alt+Del in te drukken:

- In VirtualBox is dit op te lossen door enkel de rechter-Ctrl-toets+Del in te drukken. De rechter-Ctrl-toets is de hosttoets. Deze moet bijvoorbeeld ook ingedrukt worden als de cursor niet buiten de VM geraakt of om volledig beeld te geven: Ctrl+F.
- In VMware is dit op te lossen door in de menubalk **VM** te kiezen en vervolgens **Send Ctrl+Alt+Del**. Om te schakelen tussen host- en guest-OS wordt gebruikgemaakt van linker-Ctrl+Alt.

Het is aan te raden om vanaf nu alle oefeningen in de virtuele machines te maken, zodat het fysieke toestel geen schade lijdt. Hiertoe kun je de bestanden met instructieregels uit de bestandsmap gebruiken. Bij VirtualBox kies je op elke VM **Apparaten**, **Drag-and-drop**, **Bidirectioneel** in het menu. Dit maakt het slepen van bestanden en mappen mogelijk. Kopiëren en plakken werkt vaak niet tussen fysieke en virtuele machine; vandaar slepen.

1 Aan de slag

Start PowerShell niet als administrator op de client en server. De huidige locaties zijn:

- Client: PS C:\users\ict>
- Server: PS C:\Windows\system32> of PS C:\Users\Administrator>

Typ `exit` in beide CLI-schermen en start PowerShell opnieuw als administrator. De locaties zijn nu:

- Client: PS C:\Windows\system32>
- Server: PS C:\Users\Administrator>

Typ `exit` in de CLI van de server en sluit de server volledig af, omdat de server pas later nodig is. De instructies die op de client worden uitgevoerd, zijn ook van toepassing op de server.

Voer vanaf nu alles uit op de client. Voer de cmdlet `get-childitem` uit. Deze cmdlet toont alle onderliggende mappen en bestanden.

Typ `powershell`, waardoor een nieuwe Windows PowerShell-sessie start in hetzelfde PowerShell-venster. Dit kan handig zijn om een volledige nieuwe sessie te starten.

Typ tweemaal `exit`, waardoor beide sessies in Windows PowerShell afsluiten.

2 Eerste instructie uitvoeren

Met instructie wordt het uitvoeren van een (reeks) opdracht(en) bedoeld vanaf een PowerShell-sessie. Binnen een opdracht kunnen één of meerdere commando's gebruikt worden. Zoals reeds bekend, wordt in PowerShell de naam cmdlet gebruikt in plaats van commando indien het gaat om een PowerShell-commando of -functie.

Open PowerShell op de client als administrator en voer uit:

```
rename-computer -newname "psclientInitialen"
```

Merk op dat gevraagd wordt om de pc opnieuw op te starten. Voer uit:

```
restart-computer
```

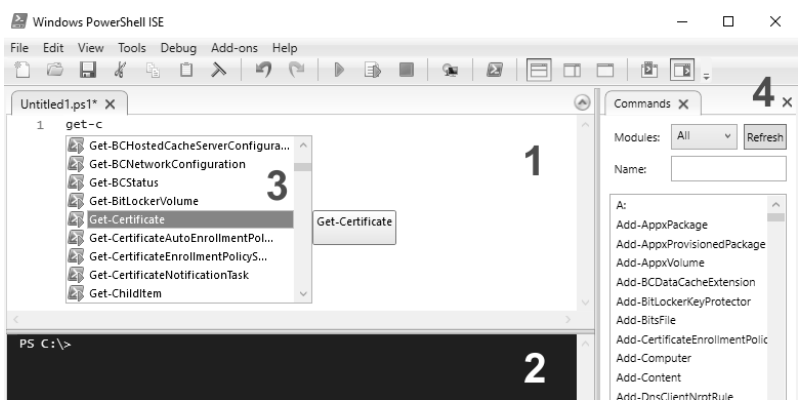
Controleer na de herstart zowel grafisch als in PowerShell of de naamswijziging is doorgevoerd. Start PowerShell en typ de globale variabele:

```
$env:computername
```

3 Gebruikmaken van Windows PowerShell ISE

Meestal wordt de CLI geopend voor het gebruik van makkelijke cmdlets of het uitvoeren van werkende scripts. In de ontwikkeling van Windows PowerShell werd de vraag steeds groter naar een IDE (Integrated Development Environment) om zelf gebruiksvriendelijke scripts te creëren. Dit werd mogelijk via de implementatie van Windows PowerShell ISE (Integrated Scripting Environment).

Windows PowerShell ISE bestaat hoofdzakelijk uit een scriptpaneel (1) en een consolepaneel (2), die onder meer gebruikmaken van IntelliSense (3; automatisch suggesties aanbieden wat bij de code kan horen) en een commandview (4), waardoor cmdlets snel en correct gevormd kunnen worden. De weergave kan aangepast worden via het menu **View** in de menubalk.



Daarnaast toont de ISE ook kleuren om een cmdlet, functie, variabele of ander type tekst aan te geven. Er kan meteen gestart worden met typen in ofwel het script- of het consolepaneel. Het verschil ligt hem in het gebruik van het aantal regels.

In het scriptpaneel kunnen enkele regels code onder elkaar geplaatst worden zonder uit te voeren. Druk, wanneer gereed, op F5 of op de knop **Run Script** en de output verschijnt in het consolepaneel.

Indien de uitvoer per regel moet worden getoond, wordt gebruikgemaakt van het consolepaneel, waarbij een regel getypt wordt en steeds bevestigd wordt met Enter. Dit komt op hetzelfde neer als gebruik van de CLI, met als voordeel het gebruik van IntelliSense.

Werkbalk.

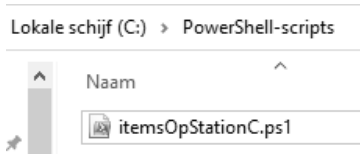
Knoppen op de werkbalk:

1. Maak consolepaneel leeg
2. Run Script (F5)
3. Run Selection (F8): voer geselecteerde regels uit
4. Stop Operation (Ctrl+spatie)
5. New Remote PowerShell Tab: activeer remote sessie met andere client
6. Start PowerShell.exe: Start CLI
7. Start command view in nieuw venster
8. Toon command view rechts in hetzelfde venster

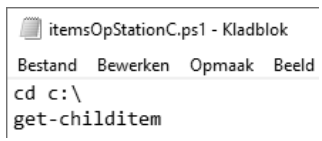
Oefening

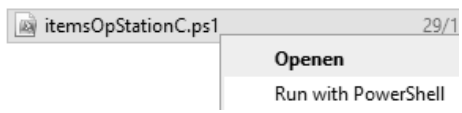
Start Windows PowerShell ISE en zorg dat het script- en consolepaneel en de command view zichtbaar zijn. Eventueel wordt op **View** geklikt om bepaalde vensters te tonen en/of rechtsboven om het scriptvenster open te klappen.

1. Typ in het scriptpaneel de tekst `cd c:\` en druk op Enter.
2. Typ de volgende regel: `get-childitem`.
3. Voer uit via F5 of de knop **Run Script** en bekijk de uitvoer in het consolepaneel.
4. Typ `exit` in het consolepaneel, waarbij gevraagd wordt om het script op te slaan als een .ps1-bestand. Klik op **Ja**, maak een nieuwe map `PowerShell-scripts` op het C:\-station en sla het bestand op met de naam `itemsOpStationC.ps1`.
5. Ga naar de map PowerShell-scripts en dubbelklik op itemsOpStationC.ps1, waarbij Kladblok opent. Tip: kijk ook even na of de .ps1-extensie zichtbaar is. Zo niet, maak ze dan zichtbaar (**Beeld**, **Bestandsnaamextensies** of **View**, **Extensions**).



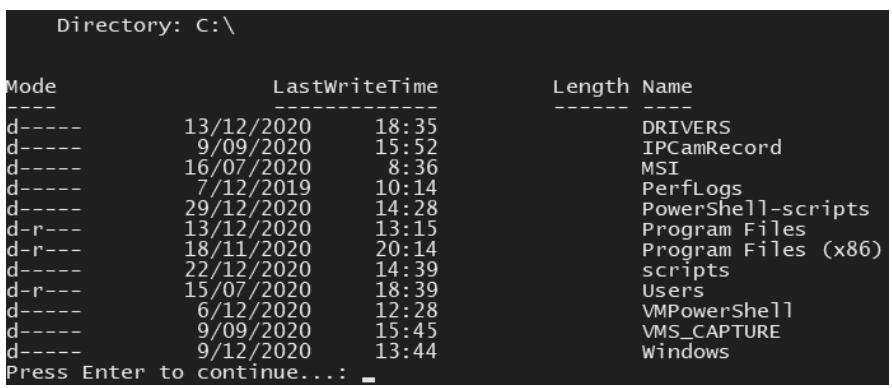
PowerShell is standaard beveiligd tegen het openen van bestanden met PowerShell-extensie. En dat is maar goed ook, want het woord Power verwijst naar krachtig en PowerShell kan inderdaad met één tot enkele cmdlets het besturingssysteem flink beschadigen. In plaats van PowerShell wordt Kladblok geopend, waarbij twee regels worden getoond. Merk op dat de regel met `exit` niet getoond wordt. Klik met de rechtermuisknop op itemsOpStationC.ps1 en voer uit in PowerShell.





Indien een melding wordt getoond betreffende Execution-Policy, is dit vrij normaal. Druk op A om het script uit te voeren. Op deze Execution-Policy wordt later teruggekomen, aangezien dit voor PowerShell zeer belangrijk is. Typ `set-executionpolicy remotesigned` bij problemen.

Het script wordt in een mum van tijd in PowerShell uitgevoerd, maar de uitvoer wordt niet getoond. Dat is niet fijn. Dubbelklik opnieuw op het bestand, voeg in Kladblok als laatste regel het woord `pause` toe en sla op met Ctrl+S. Klik opnieuw met de rechtermuisknop op het bestand en voer uit in PowerShell. Sluit af door op Enter te drukken. Het eerste PowerShell-script is een feit! Tip: vervang `pause` door `start-sleep 5`.



Uitvoer.

4 Visual Studio Code

Standaard wordt de IDE-omgeving Windows PowerShell ISE geïnstalleerd via een Windows-installatie. Dit gebeurt natuurlijk niet op andere besturingssystemen, waardoor een IDE voor PowerShell Core ontbreekt.

Via de installatie van Visual Studio Code wordt het mogelijk om PowerShell Extensions te activeren, waardoor Visual Studio Code gebruikt kan worden als IDE voor PowerShell op andere besturingssystemen. Belangrijk om te weten:

- Gebruik PowerShell ISE wanneer code op Windows uitgevoerd moet worden en deze nog voldoet voor PowerShell versie 5.1, maar weet dat deze ISE-versie geen optimale ondersteuning meer heeft, waardoor bekende bugs eventueel niet meer worden opgelost.
- Gebruik PowerShell ISE op Windows als een niet kant-en-klaar script speciaal voor Windows moet worden uitgevoerd, zodat fouten duidelijker aangetoond worden.
- Gebruik Visual Studio Code in alle andere gevallen.

De installatie van Visual Studio Code volgt in deel 2 van dit boek.

5 Typische PowerShell-opmerkingen

Vaak zal gebruik worden gemaakt van instructies die te maken hebben met services en processen, omdat die veel kunnen aantonen zonder de pc of de virtuele machine te beschadigen. Er zijn nog enkele aanvullende zaken die werken met PowerShell vergemakkelijken:

- PowerShell is niet hoofdlettergevoelig. De cmdlets en meer die gebruikt worden, zullen vaak met of zonder hoofdletter in de opgegeven instructies staan.
- PowerShell houdt rekening met wat de gebruiker al kent. Linux-gebruikers werken veel in een CLI (Command Line Interface) en maken daarvoor onder andere gebruik van UNIX-commando's. Windows-gebruikers die vroeger veel met CMD (Opdrachtprompt) werkten, gebruiken ook verschillende commando's.
  Al deze commando's zijn in PowerShell geïmplementeerd, waardoor heel wat cmdlets alternatieven (aliassen) hebben.
  - De cmdlet get-childitem kan bijvoorbeeld gebruikt worden door `dir` (commando in Windows voor directory) of `ls` (UNIX- en Linux-commando voor list) te typen.

Oefening

1. Start PowerShell en ga naar locatie C:\: `cd c:\` of `cd \` indien de C-partitie al actief is.
2. Voer achtereenvolgens de commando's `ls` en `dir` uit.
   Indien correct uitgevoerd, wordt tweemaal dezelfde output getoond.
3. Typ `get-ch` en druk daarna op de tabtoets. Get-Childitem wordt snel getoond. *Tab-completion* is een toevoeging om sneller binnen PowerShell te werken.
   Daarnaast kan als alias voor de cmdlet get-childitem, `gci` gebruikt worden, wat get-childitem afgekort is en in de lijst van aliassen is opgenomen: `get-alias gci`.

6 Houd rekening met iedereen

Start PowerShell ISE en voeg volgende code toe:

```
cd hkcu:\Software
new-item NSleutel
new-itemproperty -path NSleutel -name test -value "testwaarde"
```

Een toepasselijke naam voor dit script zou kunnen zijn: AanmaakRegistersleutel.ps1.

Sla op als .ps1-bestand met deze gekozen naam en bewaar in de map PowerShell-scripts.

Bovenstaand script zou voor de gebruiker duidelijker zijn met commentaar. Commentaar kan in PowerShell worden toegevoegd via # voor een enkele regel en <# #> voor meerdere regels.

Voeg bovenaan het volgende commentaar toe aan het script en daaronder een aparte commentaarregel:

```
<# ------------------------------------------------
Auteur: naam
Datum: datum
Functie: registersleutel en -waarde toevoegen aan HKCU – Software
------------------------------------------------ #>
cd hkcu:\Software
new-item NSleutel   #Toevoegen registersleutel
new-itemproperty -path NSleutel -Name test -value "Testwaarde"

#waarde toevoegen aan registersleutel
```

Sla het script opnieuw op en voer uit. Voor iedere gebruiker is het nu duidelijk wat het script doet. Open Regedit en ga op zoek naar de gemaakte sleutel. Verwijder deze handmatig.

7 Escapeteken

In PowerShell wordt ` (*backtick*) gebruikt als escapeteken. Zoek deze op het toetsenbord.

Indien het teken niet gevonden wordt, kan een kopie worden gemaakt van dit teken op een website waar het gevonden wordt via de zoekterm `escape character powershell` in Google.

Een escapeteken kan later in een instructie handig zijn. Enkele voorbeelden:

- `t = tab
- `n = nieuwe regel

Bijvoorbeeld: `"Nieuwe`nregel" > c:\powershell-scripts\escape.txt`. Open het bestand en merk op dat Nieuwe op de eerste regel staat en regel op de tweede. Toch kan het zijn dat

bepaalde tekstprogramma's dit nog op één regel weergeven. Om dit op te lossen moet `` `r `` toegevoegd worden net voor `` `n ``.

Oefening

Start PowerShell en klik met de rechtermuisknop in de titelbalk. Besteed ongeveer twee minuten aan de verschillende instellingen van de shell. De shell kan bijvoorbeeld aantrekkelijker worden gemaakt. Het is bijvoorbeeld belangrijk om een goed lettertype te kiezen om het verschil te zien tussen een accent ' en het escapeteken: `.

Het escapeteken kan ook gebruikt worden om een coderegel af te breken naar een volgende regel, wat het lezen van de code soms vergemakkelijkt. Dit wordt in de voorbeelden soms toegepast, om aan te geven dat een instructie op één regel hoort.

Pas geen kleuren aan, tenzij dit echt gewenst is. Het is namelijk niet makkelijk om terug te keren naar de standaardeigenschappen van PowerShell.

Niet alle instellingen voor het uiterlijk van de PowerShell-CLI kunnen grafisch aangepast worden; een voorbeeld is de titel wijzigen van de gestarte sessie. Voer daarvoor de volgende instructie in:

```
$host.ui.RawUI.WindowTitle = "Administrator: Nieuwe sessie van Naam Voornaam"
```

8 Cmdlets

Waarom get-childitem gebruiken als het ook korter kan met dir? Jeffrey Snover, de maker van PowerShell, wilde commando's implementeren die duidelijk waren in het gebruik van het systeem. Een cmdlet wordt steeds gevormd via *verb-noun*, waarbij het werkwoord (*verb*) algemeen vertelt wat het commando zal doen en het zelfstandig naamwoord (*noun*) daarna het commando specificeert. Vaak voorkomende verbs zijn `add`, `clear`, `convertFrom`, `convertTo`, `disable`, `enable`, `export`, `get`, `set`, `new`, `remove`, `import`, `start`, `stop`, `test`, `write`. Ze allemaal zien kan via `get-verb`.

`invoke` is een andere veelgebruikte en speciale verb, die vooral wordt gebruikt bij sessies op afstand om een instructie uit te voeren. Denk bij `invoke` aan de Nederlandse vertaling "beroep doen op".

Enkele basis-cmdlets zijn `compare-object`, `sort-object`, `select-object`, `convertFrom-SecureString`, `Get-Alias`, `New-Alias`, `Get-Command`, `Get-Credential`, `Set-Date`, `Get-ExecutionPolicy`, `Get-Host`, `Set-Location`, `Stop-Process`, `Remove-PSDrive`, `Get-Service`, `Set-Variable`, `Read-Host`.

Uit bovenstaande voorbeelden kun je al snel afleiden waarvoor de verb of de cmdlet staat.

Merk op dat cmdlets altijd in het enkelvoud eindigen: bijvoorbeeld `get-service` en niet `get-services`.

9 Snel werken met PowerShell

Start PowerShell en voer de volgende cmdlet uit:

```
get-host
```

Dit toont onder meer de versie en diverse configuratie-instellingen specifiek voor deze versie. Voer nu volgende cmdlet uit:

```
get-command
```

Dit toont de verschillende typen commands: cmdlet, functies en aliassen. Merk op dat de output maar bleef doorstromen. Indien de output blijft komen, kan deze gestopt worden door Ctrl+C in te drukken.

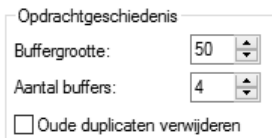
Vervang de cmdlet door het volgende:

```
get-command | format-wide -column 3
```

Hierbij worden alle mogelijke commando's in drie kolommen getoond, zonder info.

In de opstelling van de vorige cmdlet werd de eerste maal gewerkt met het pipeteken | dat ervoor zorgt dat onder andere cmdlets na elkaar worden uitgevoerd, waarover later meer.

Door op Pijl-omhoog te drukken worden in dezelfde of een nieuwe PowerShell-sessie de vorige commandoregels getoond die via PowerShell eerder werden uitgevoerd.



Opdrachtgeschiedenis.

Opdrachtgeschiedenis maakt dit mogelijk. Vroeger (voor versie 5) was dit in te stellen via de eigenschappen in PowerShell, waarbij een buffergrootte (aantal gebruikte cmdlets) en een aantal buffers (cmdlets van aantal PS-shells tegelijk) worden bijgehouden. Dit is nog steeds zo in te stellen, maar intussen worden de laatste 4096 instructies bewaard, zodat deze ook op te vragen zijn in nieuwe PowerShell-sessies.

Voer volgend cmdlet uit:

```
get-command | more
```

`More` stopt de uitvoer als deze langer is dan één scherm. Gebruik:

- Enter om een volgende regel op het volgende scherm op te roepen
- Spatie om een volgend scherm op te roepen

Een andere handige functionaliteit die vaak nodig is, gaat om kopiëren en plakken binnen de CLI. Dit is mogelijk door tekst binnen de CLI te selecteren en vervolgens één keer met de rechtermuisknop op de te kopiëren tekst te klikken. Hierdoor wordt het geselecteerde gekopieerd en kan dit opnieuw door een rechtermuisklik op een andere plaats geplakt worden.

10 Parameters bij cmdlets

Vaak moeten cmdlets ook parameters meekrijgen om een instructie correct uit te voeren. Parameters zijn optioneel, tenzij een parameter noodzakelijk is of deze toegevoegd moet worden voor correcte uitvoer. Een noodzakelijke parameter is een *mandatory* of verplichte parameter.

Start PowerShell en voer de volgende cmdlet uit:

```
get-service
```

Deze cmdlet wordt uitgevoerd en toont alle services met status, naam en displaynaam. Voer nu de volgende cmdlet uit:

```
get-eventlog
```

De cmdlet wordt niet uitgevoerd door het ontbreken van de verplichte parameter `logname`. Geef als `logname` de volgende waarde op: `security`. Hierdoor worden alle gebeurtenissen getoond op het gebied van beveiliging op het systeem. Voer nu opnieuw de cmdlet correct uit met parameter:

```
get-eventlog -logname security
```

Let op: een parameter start altijd met - (dash), de waarde van de parameter niet. Veel parameters kunnen leiden tot een instructie met meerdere regels. Om zelf een instructie te bouwen die meerdere regels beslaat, kan het volgende worden aangehouden:

- Alles op één regel blijven typen. Binnen de shell zal automatisch op een nieuwe regel gestart worden als het einde van het venster in zicht komt.
- Binnen PowerShell ISE kan ` (backtick) geplaatst worden op het einde van de eerste regel om zo verder te typen op een volgende regel. Dit brengt overzicht en brengt meerdere regels samen tot één instructieregel.

De beste manier is om via PowerShell ISE instructies van meerdere regels te vormen. Binnen de shell is er nog de optie als een open haakje nog niet is afgesloten. Bijvoorbeeld:

```
invoke-command -scriptblock {
```

Als nu op Enter wordt gedrukt zal >> verschijnen om het vervolg te typen (met } om te eindigen).

Naast de specifieke parameters per cmdlet zijn er ook enkele algemene parameters met of zonder waarde(n) die bijna bij iedere cmdlet kunnen gebruikt worden. Hieronder is een opsomming te vinden. Deze parameters kunnen tijdens oefeningen aan bod komen:

Parameter	Betekenis
`-confirm`	Gebruiker moet bevestigen bij uitvoer cmdlet (waarde `suspend` = beslissing afwachten, eerst controleren en beslissing pas na `>> exit` typen.) Test met `restart-computer -confirm` en druk op Enter. Druk op S. Nu kunnen extra instructies worden ingevoerd. Typ `exit` en druk daarna op N.
`-erroraction`	Parameter die optreedt bij fouten met als mogelijke waarden: `SilentlyContinue` (laat de fout voor wat het is en vervolg), `Continue` (toon de fout en vervolg), `Inquire` (toon de fout en vraag om vervolgbevestiging) en `Stop` (toon de fout en ga niet verder).
`-errorvariable`	Sla alle foutmeldingen op in een variabele.
`-verbose`	Toon extra informatie tijdens de uitvoering (kan zéér handig zijn).
`-warningaction`	Zelfde als `-erroraction`, maar voor waarschuwingen.
`-warningvariable`	Zelfde als `-errorvariable`, maar voor waarschuwingen.
`-whatif`	Zeer handig en zeer belangrijk. Indien onzeker over een cmdlet, voer deze dan uit met `-whatif` als dit mogelijk is. Dit toont vervolgens wat allemaal zal veranderen, terwijl zaken in de praktijk nog niet veranderd worden. Een aanrader om te onthouden!
`-force`	Indien een cmdlet zeker moet uitgevoerd worden, is de kans groot dat dit lukt door achteraan deze parameter toe te voegen. Sommige cmdlets werken hier niet mee en zullen een foutmelding opleveren. Eveneens wordt `-force` gebruikt om geen vragen te krijgen in de trend van 'ben je zeker dat dit uitgevoerd mag worden?'

Wat volgt, komt later aan bod: via de helpfunctie kunnen alle mogelijke switchen en parameters opgevraagd worden. Bij een switch (zoals `force`, `whatif`) wordt er geen waarde na geplaatst, tenzij het een boolean($true of $false) is.

11 Anatomie van PowerShell-commando

Cmdlet	Parameter 1		Parameter 2		Parameter 3
`Stop-Service`	`-Name`	`wuauserv`	`-Computername`	`Client, server`	`-Force`
	Parameternaam	Parameter-waarde	Parameternaam	Parameter-waarde	Switch zonder waarde

12 Bouwstenen

Bij scripting worden aan de computer opdrachten gegeven in de vorm van een instructie. Een computer begrijpt deze instructie niet, omdat een computer werkt op basis van het binair talstelsel (1 en 0). De instructie moet hierdoor omgezet worden naar enen en nullen. Dit noemt men compileren. Een compiler zal het originele programma volledig omzetten naar machinetaal.

Binnen de opdrachten in de instructie worden onder meer variabelen en datatypen gebruikt. Een variabele is een plek waar informatie wordt opgeslagen. Een variabele komt overeen met één of meerdere geheugenplaats(en) in het werkgeheugen van de computer, waarin die informatie wordt opgeslagen.

Zo'n geheugenplaats heeft een bepaald adres, waarnaar een link gelegd wordt via de variabelenaam die wordt opgegeven. Als een variabele wordt aangesproken, zal de inhoud worden opgeroepen van één of meer geheugenplaatsadressen.

De variabelen die worden gebruikt, moeten herkenbaar gemaakt worden aan de compiler. Daarom bestaan er variabelen van verschillende soorten datatype: bijvoorbeeld een *string* voor tekst en een *int* voor gehele getallen. Een datatype toont aan welk soort informatie een variabele zal bevatten en hoeveel plaats deze zal innemen in het werkgeheugen. Een *integer (int)* kan bijvoorbeeld gebruikt worden om te rekenen en zal hiervoor vier bytes gebruiken.

Enkel om aan te tonen, bevat onderstaande tabel de meest gebruikte gegevens-/datatypen:

Gegevenstype	Bereik	Geheugen	.NET Framework*
Gehele getallen			
Byte	0 – 255	1 byte	System.Byte
Short	-32768 – 32767	2 bytes	System.Int16
Int	-2 miljard – +2 miljard	4 bytes	System.Int32
Long	-2^{63} – 2^{63}-1 (19 cijfers)	8 bytes	System.Int64
Reële getallen of kommagetallen			
Float	7 cijfers	4 bytes	System.Single
Double	15 cijfers	8 bytes	System.Double
Decimal	28 cijfers	16 bytes	System.Decimal
Tekst			
Char	1 teken	2 bytes	System.Char
String	tekenreeks	2 bytes/teken	System.String
Logisch			
Bool	True/False	1 bit	System.Boolean

* Klasse System in .NET.

13 Waarde toekennen aan een variabele

Een variabele krijgt een naam door er een $-teken voor te plaatsen. Aan de variabele zal normaal een waarde worden toegevoegd. Welke naam de variabele draagt maakt niet uit (afgezien van enkele uitzonderingen die online terug te vinden zijn). Dit betekent dat deze constructies beide kunnen:

- $getal = 4
- $woord = 4

Beide zijn correct, want de naam van de variabele maakt niet uit. Vanaf het moment dat de variabele bestaat, moet deze wel in het vervolg van de code met diezelfde naam worden aangesproken.

Vaak zal gebruik worden gemaakt van korte namen: $a, $x, $y. Belangrijk: het is beter om dit niet te doen en eerder te kiezen voor lange goede namen, bijvoorbeeld $computernaam. Dit maakt het duidelijker wanneer de code later wordt herlezen of wanneer een andere persoon de code leest. Hierdoor hoeft er ook minder commentaar te worden opgenomen.

De waarde die aan de variabele wordt toegevoegd zal PowerShell bekijken en er zelf een datatype aan toekennen. Indien dit om tekst gaat, zal dit door de gebruiker tussen aanhalingstekens (“ ” of ‘ ‘) geplaatst zijn, waardoor PowerShell er het datatype string aan zal geven. Gaat dit om een geheel getal, dan zal dit een integer (int32) worden en bij een kommagetal een double. Dit is makkelijk op te vragen via `variabelenaam.gettype()`.

14 Opbouwen van een stevige instructie

Waartoe PowerShell in staat is, wordt met dit eerste voorbeeld aangetoond. Dit voorbeeld is bedoeld om dit te begrijpen en naar mate meer kennis in PowerShell wordt opgedaan dit zelfstandig uit te voeren.

```
New-LocalUser
   [-AccountExpires <DateTime>]
   [-AccountNeverExpires]
   [-Description <String>]
   [-Disabled]
   [-FullName <String>]
   [-Name] <String>
   -Password <SecureString>
   [-PasswordNeverExpires]
   [-UserMayNotChangePassword]
   [-WhatIf]
   [-Confirm]
   [<CommonParameters>]
```

Op de client wordt een nieuwe gebruiker toegevoegd, waarvoor de cmdlet enkele waarden toegekend moet krijgen via de parameters die in de afbeelding weergegeven worden.

De parameters zonder [ ] hebben een verplichte waarde nodig om het cmdlet te kunnen uitvoeren. In dit geval gaat dit om `-Name` en `-Password`, waarbij vermelding van `-Name` niet hoeft.

We maken een gebruiker student met wachtwoord Server*:

```
new-localuser student -password "Server*"
```

Helaas werkt dit niet omdat Server* niet toegevoegd kan worden als securestring. Dit wordt in de foutboodschap gemeld.

```
New-LocalUser : Cannot bind parameter 'Password'. Cannot convert the "Server*" value of type
"System.String" to type "System.Security.SecureString".
```

Foutmelding.

Wat wel werkt is `new-localuser student`. Nadat op Enter wordt gedrukt zal het wachtwoord gevraagd worden en wordt `Server*` opgegeven. Nu zal PowerShell dit afhandelen, waardoor Server* wel als securestring toegevoegd kan worden.

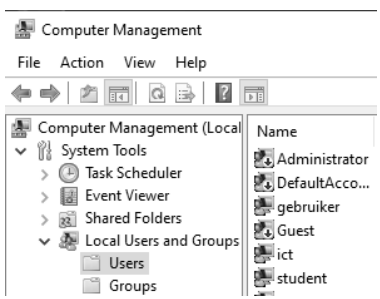
Eveneens was het mogelijk om de gebruiker student te maken zonder wachtwoord, door gebruik te maken van de andere versie die `new-localuser` aanbiedt. Van ieder cmdlet kunnen meerdere versies bestaan.

```
New-LocalUser
   [-AccountExpires <DateTime>]
   [-AccountNeverExpires]
   [-Description <String>]
   [-Disabled]
   [-FullName <String>]
   [-Name] <String>
   [-NoPassword]
   [-UserMayNotChangePassword]
   [-WhatIf]
   [-Confirm]
   [<CommonParameters>]
```

Versie 2: we maken gebruiker `gebruiker` zonder wachtwoord:

```
new-localuser gebruiker -nopassword
```

Via Computerbeheer kunnen vervolgens de lokale gebruikers worden opgevraagd.



Bekijk nu de andere parameters die meegegeven kunnen worden bij het toevoegen van de nieuwe gebruiker. Vergelijk hierbij de twee afbeeldingen die beide versies opleveren om een lokale gebruiker toe te voegen. De volgende instructie kan bijvoorbeeld worden opgesteld. Voer deze uit:

```
➥ new-localuser -name beheerder -accountneverexpires -description "Beheerder van fictieve
    organisatie" -Fullname 'Beheerder_Organisatie'
```

Wat betekent dat pijltje?

Wanneer een coderegel vooraf wordt gegaan door het teken ➥, betekent dit dat de regel plus alles wat daaronder ingesprongen wordt weergegeven, op één regel achter elkaar moet worden doorgetypt. Druk aan het einde van de eerste regel dus niet op Enter, maar doe dat pas aan het eind van de instructie (in bovenstaand voorbeeld aan het eind van de tweede regel).

```
PS C:\Windows\system32> new-localuser -name beheerder -accountneverexpires -description "Beheerder van
fictieve organisatie" -Fullname "Beheerder_Organisatie"

cmdlet New-LocalUser at command pipeline position 1
Supply values for the following parameters:
Password: *******

Name      Enabled Description
----      ------- -----------
beheerder True    Beheerder van  fictieve organisatie
```

Na het bevestigen van bovenstaande instructie wordt opgemerkt dat opnieuw een wachtwoord moet worden ingevoerd. Geef Server* op of indien geen wachtwoord nodig is, bevestig met Enter. Dit heeft dezelfde werking als de parameter `-NoPassword` toevoegen, waardoor de eerste versie van gebruikersaanmaak dus eigenlijk wel kon. Aangezien dit om een beheerdersaccount gaat, wordt Server* opgegeven.

Merk op dat in deze instructie waarden aan de parameters werden toegevoegd met geen of verschillende soorten aanhalingstekens. In dit geval moet enkel `description` over aanhalingstekens beschikken, omdat er spaties gebruikt worden. Of dit dubbele of enkele aanhalingstekens zijn, maakt in dit geval weinig uit. Later wordt duidelijk welke wanneer gebruikt moeten worden.

Maar wat als er meerdere gebruikers in één keer moeten worden toegevoegd en dit op verschillende machines? Maak dan in Kladblok een bestand met de namen apart en voeg er extra info aan toe, telkens gesplitst door een komma:

```
name, username, password, expirationInYears, description
Toon Hoeikens, toon, THServer*, 2, Financieel directeur
Karen Blaai, karen, KBServer*, 1, CEO
Wesley Doets, wesley, WDServer*, 2, Administratief directeur
```

Maak het bestand zo na dat op iedere regel een nieuwe gebruiker wordt opgegeven.

Sla het bestand vervolgens op als een .csv-bestand `users.csv` in de map c:\powershell-scripts. Ga naar deze map in PowerShell ISE via het consolepaneel: `cd c:\powershell-scripts` (zorg er altijd voor dat de locatie juist is). Neem het .csv-bestand op in de variabele `$csv`:

```
$csv = import-csv .\users.csv
```

Typ `$csv` om de inhoud van `$csv` te bekijken.