Veel gebruikte Commando's:

Get-Verb → om alle mogelijke werkwoorden te zien

Get-Alias → geeft een overzicht van alle aliassen die je kan gebruiken

Show-Command Get-ChildItem → hiermee krijg je een venster waarin je de parameters en hun waarden kunt invullen

Get-PSProvider → geef je een lijst van alle providers

Get-Service –Name naam_service → een service te selecteren, zonder de optie –Name of –Value wordt er poging gedaan om alle services te selecteren

Stop-Service → om een service te stoppen

Start-Service → om een service te starten

WhatIf → kan je gebruiken om vooraf te bepalen wat er zou gebeuren indien je dit commando uitvoert, je kan hiervoor ook **–Confirm** gebruiken

Get-ChildItem → hiermee worden items opgehaald

update-help → hiermee wordt de laatste content van het internet gehaald

help → hiermee krijg je meer info over het commando

- met de optie **–Full** → krijg je veel meer info, waaronder ook voorbeelden
- met optie **–ShowWindows** → wordt de help in een apart venster getoond

Get-commando → hiermee krijg je een lijst van alle beschikbare Cmdlets en functies van alle geladen modules

Get-History → hiermee kan je kijken welke commando's je in de huidig sessie hebt gebruikt

Invoke-History → hiermee kun je het commando uit je geschiedenis herhalen (pijltje omhoog werkt ook)

Get-Item → hiermee kunnen bestanden en mappen op de logische schijven worden opgehaald

Set-ACL → hiermee kunnen rechten op files en shares aangepast worden

Set-ADGroup → hiermee pas je AD-groepen aan of voeg je deze toe aan AD.

Set-ADUser → hiermee kan je AD-users aanpassen of toevoegen

Set-Alias → hiermee kan je aliassen aanmaken of wijzigen

Set-Date → hiermee kan je de datum en tijd aanpassen

Export-Csv → hiermee stuur je iets door naar een csv (comma-seperated value)- bestand

Notepad → hiermee kan je een bestand openen in notepad

Export-Clixml → hiermee stuur je iets naar een xml-bestand

Out-File → hiermee genereer je de output naar een nieuw venster

Out-GridView → hiermee krijg je een grid waarin er kan worden gesorteerd met behulp van een venster

ConvertTo-Html → hiermee wordt naar een .html-file weggeschreven

Get-Content → hiermee lees je de inhoud van bestanden

Compare-Object → hiermee vergelijk je bestanden en/of output met elkaar

New-Item → hiermee kun je een nieuwe file aanmaken

Remove-Item → hiermee kan je een bestand of een map weghalen

Set-Location → hiermee kan je veranderen van drive

Powershell.exe –noexit → hiermee kan je een script uitvoeren in de command line

. → hiermee kan je een script uitvoeren in de PowerShell zelf

Get-ExecutionPolicy → hiermee kan je de policy achterhalen die ingesteld is, de mogelijkheden zijn Restricted, Allsigned, **RemoteSigned**, Unrestricted

Read-Host → hiermee kan je input vragen van een gebruiker

→ hiermee zet je commentaarlijnen in je script

cls → hiermee maak je je scherm leeg

Function { } → hiermee beschrijf je een functie

Start iexplore → hiermee kan een bepaalde website met Internet Explorer geopend worden

Write-Host → hiermee kan je iets op het scherm laten verschijnen

$variabele = waarde → hiermee ken je een bepaalde waarde toe aan een variabele

‘ ‘ (enkele aanhalingstekens) → alles wat hiertussen staat wordt letterlijk gereproduceerd op het scherm

“ “ (dubbele aanhalingstekens) → hier worden de waarden van de variabele eerst ingelezen en daarna niet geüpdatet

`-teken (backtick) → als je dit gebruikt als escape teken, vertel je PowerShell dat de variabele niet als variabele gelezen moet worden, maar als exacte tekst

Dir Variable → geef je een lijst van de standaard aanwezige en zelfgemaakte variabelen

Del Variable | Remove-Variable -Name → hiermee verwijder je een variabele

[declaratie]$variabele → declareren van een variabele, mogelijke declaraties zijn Int, Long, String, Byte, Bool, Decimal

If(conditie){if actie}[optioneel]elseif(conditie){elseif actie}[optioneel]else{else actie} → met deze conditie kan je keuzes maken in je commando’s en scripts. Mogelijke condities: -eq → = ; -lt → < ; -gt → > ; -le → =< ; -ge → => ; -like → & of *

help about-if → hiermee krijg je hulp ivm de if-conditie

Switch ($variabele) → een constructie om controles uit te voeren, **break** kan gebruikt worden om uit

{ de switch te komen zonder alle andere waarden te evalueren

1 {statement1}

2 {statement2}

…

default {als geen van vorige mogelijkheden}

}

Do {actie} until (conditie) → hiermee kan je een gepast aantal uitvoeringen doen ; een actie wordt uitgevoerd totdat de conditie is bereikt

Do {actie} while (conditie) → hiermee wordt een actie uitgevoerd zolang de conditie waar is

; → hiermee kan je expressie na elkaar zetten

For (conditie){actie} → hiermee zal de actie uitgevoerd worden zolang de conditie waar is

Foreach (item IN set){actie} → hiermee wordt voor elke individueel item van de set een iteratie uitgevoerd

Test-Path → hiermee kan je testen of een bestand bestaat

Function naam($arg1,$arg2){actie;return $output} → hiermee kan je een stukje code met de naam laten uitvoeren, waarbij je 2 argumenten meegeeft waarop de code zal toegepast worden en het resultaat in $output terug zal gestuurd worden

$variabele = New-Object –comobject Comapp.application → hiermee definiëren we een COM-object

New-Object → hiermee creeër je nieuwe objecten

-comobject → duid aan dat het om een COM(Component Object Model)-object gaat

Get-WMIObject –class Win32_class → hiermee kan je WMI(Windows Management Instrumentation)-objecten gebruikten

Get-Wmiobject –List –Namespace root\wmi → hiermee worden alle WMI-objecten weergegeven

Get-WMIObject –List | Where-object `

{$_.name –match 'Win32'} → hiermee worden alle win32 Cmdlets weergegeven