Powershell

Allgemein

- Powershell ist eine Interpretierte sprache die Comands in CMDlets umwandelt
- muss nicht compiled werden
- live runtime

`cd` ==Interpreter==> `set-location`

CMC befehle zu Powershell

- `cd` => `Set-Location`
- `cls` => `Clear-Host`
- `copy` => `Copy-Item`
- `dir` => `Get-ChildItem`
- `echo`=> `Write-Output`
- `rmdir` => `Remove-Item`
- `ipconfig` => `Get-Host`

CMD befehle die es nicht in Powershell gibt

- `ping` => `Test-Connection`
- `ipconfig` => `Get-NetIPConfiguration`
- `tracert` => `Test-NetConnection`
- `nslookup` => `Resolve-DnsName`
- `arp`=> `GetNetneighbour`

Aliase

Aliase sind eine kürzere Eingabe von Cmdlets. Dadurch kennt und versteht PowerShell CMD befehle. `dir`wird duch `Get-ChildItem` ersetzt.

Umgebungsvariablen

- Speichert Infos zur Betreibssystem umgebung
- Infos über Betriebssystempfand
- Variablen die Systemweit abgespeichert werden
- Liste der Umgebungsvariablen: `Get-ChildItem env:`
- Zeige den Aktuellen Umgebungsvariablen von path an: `Get-ChildItem env:path`

Cmdlets

```
Test-Connection -count 2 127.0.0.1
```

Verb-Objekt Prameter Option

Verben:

- `Get`
- `Set`
- `Add`

Pipeline |

Bsp: `Get-Comand-Name where-Object Name -like write` (Filtert die Ausgabe nach Get-Comand-Neme nach write)

Vergleichsoperatorn

Typ	Operator	Vergleichstest
Gleichheit	-eq	equals
	-ne	ungleich
	-gt	Größer als
	-ge	Größer als oder gleich
	-lt	Kleiner als
	-le	Kleiner als oder gleich
Matching	-like	Zeichenfolge stimmt mit Platzhaltermuster überein
	-notlike	Zeichenfolge stimmt nicht mit Platzhaltermuster überein
	-match	Zeichenfolge stimmt mit regex-Muster überein
	-notmatch	String stimmt nicht mit regex-Muster überein
Containment	-contains	-Auflistung enthält einen Wert
	-notcontains	-Auflistung enthält keinen Wert
	-in	value befindet sich in einer Auflistung.
	-notin	value befindet sich nicht in einer Auflistung.

Ausgabemanipulation

- `Where-Object`
- `Select-Object`
- `Sort-Object`
- `Measure-Object`

Powershell Scripting

Kommentare

#

Variablen

Variablen sind Dynamisch. Typ muss nicht deklariert werden.

`$x = 1` oder `$y = "Hello World"`

Eingabe

```
$Anzahl = Read-Host -Prompt "Zahl Eingeben"
```

Ausgabe

```
Write-Host "Hello World"
```

Arrays

```
$array = @{"one", "two", "three"}
```

`Write-Host $array[0]` => one

Länge der arrays: `array.length` => 3

If

```
$var = 4

if ($var -eq "1"){
    Write-Host "var is 1"
}

else{
    Write-Host "var is not 1"
}
```

For

```
for ($=0; $i -lt 4; $i++){
    Write-Host $i
}
```

Foreach

```
foreach($i in $array){
    Write-Host $i
}
```

While

```
$i = 0

while ($i -lt 4){
    Write-Host $i
    i++
}
```

Import an .csv

Importing Csv: `$array = Import-Csv -Path c:\datei.txt -Delimiter ","`

Using the Data: `$array[0].Port` oder `$array[0].Protokol`

RegEx

Select-String

Selectiere einen String: `Select-String -pattern '<RegEx>'`

RegEx Ausdrücke

- mindestens eine Zahl: `[0-9]`
- mindestens N Zahlen: `[0-9]{N}`
- mit einem Buchstaben beginnend: `^[a-z A-Z]`
- mit einer Zahl zwischen N und M: `[N-M]`
- mit einer einzelnen Zahl beginnend, worauf ein Buchstabe folgt: `^[0-9][a-z A-Z]`
- mit einem m oder r beginnend: `^[mr]`
- mit einem m oder r endent: `[mr]$`
- nur aus Kleinbuchstaben: `[mr] -casesensitiv`

Ausdruck	Beispiel	Erklärung
[]	[egh]	eines der Zeichen `e`, `g` oder `h`
	`[0-6]`	eine Ziffer von `0` bis `6` (Bindestriche sind Indikator für einen Bereich)
	`[A-Za-z0-9]`	ein beliebiger lateinischer Buchstabe oder eine beliebige Ziffer
^	`[^a]`	ein beliebiges Zeichen außer `a` (`^` am Anfang einer Zeichenklasse negiert selbige)

Ausdruck	Erklärung
\s	Leerzeichen und die Klasse der Steuerzeichen `\f`, `\n`, `\r`, `\t` und `\v`
\S	ein Zeichen, das kein Whitespace ist, also `[^\s]`
\d	digit; eine Ziffer, also `[0-9]`
\D	no digit; ein Zeichen, das keine Ziffer ist, also `[^\d]`
\w	word character; ein Buchstabe, eine Ziffer oder der Unterstrich, also `[a-zA-Z_0-9]`
\W	no word character; ein Zeichen, das weder Buchstabe noch Zahl noch Unterstrich ist, also `[^\w]`
.	jedes beliebige Zeichen

Ausdruck	Erklärung
^	Anfang einer Zeile (Zeichen nicht mit dem Negierer verwechseln – dieser steht nicht am Anfang)[2]
$	Ende einer Zeile
\b	Wortgrenze
\B	Nichtwortgrenze (also das Pattern darf nicht an einer Wortgrenze stehen)
\A	Anfang des untersuchten Strings
\G	Ende des vorausgegangenen Matches
\Z	Ende des untersuchten Strings, ohne dem finalen Terminator, sofern er existiert
\z	Ende des untersuchten Strings

Ausdruck	Erklärung
{min,max}	Der voranstehende Ausdruck muss mindestens *min*-mal und darf maximal *max*-mal vorkommen.
?	Der voranstehende Ausdruck ist optional, er kann einmal vorkommen, braucht es aber nicht, das heißt, der Ausdruck kommt null- oder einmal vor. (Dies entspricht `{0,1}`)
+	Der voranstehende Ausdruck muss mindestens einmal vorkommen, darf aber auch mehrfach vorkommen. (Dies entspricht `{1,}`)
*	Der voranstehende Ausdruck darf beliebig oft (auch keinmal) vorkommen. (Dies entspricht `{0,}`
{n}	Der voranstehende Ausdruck muss exakt *n*-mal vorkommen. (Dies entspricht `{n,n}`)

{0,max}	Der voranstehende Ausdruck darf maximal *max*-mal vorkommen.