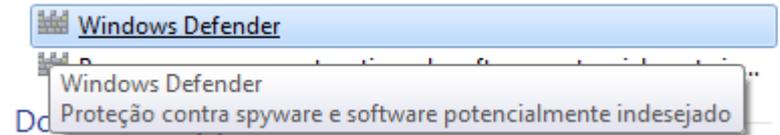


S.O. – Windows

Aula 07

Windows Defender



- ▶ **Proteção em tempo real.** O Windows Defender emite um alerta quando algum spyware tenta se executar ou se instalar no computador. Ele também emite um alerta quando alguns programas tentam alterar configurações importantes do Windows.
- ▶ **Opções de verificação.** É possível usar o Windows Defender para verificar se há spyware instalado no computador, agendar verificações regulares e remover automaticamente tudo o que for detectado em uma verificação.

Windows PowerShell

Programas (5)

Windows PowerShell

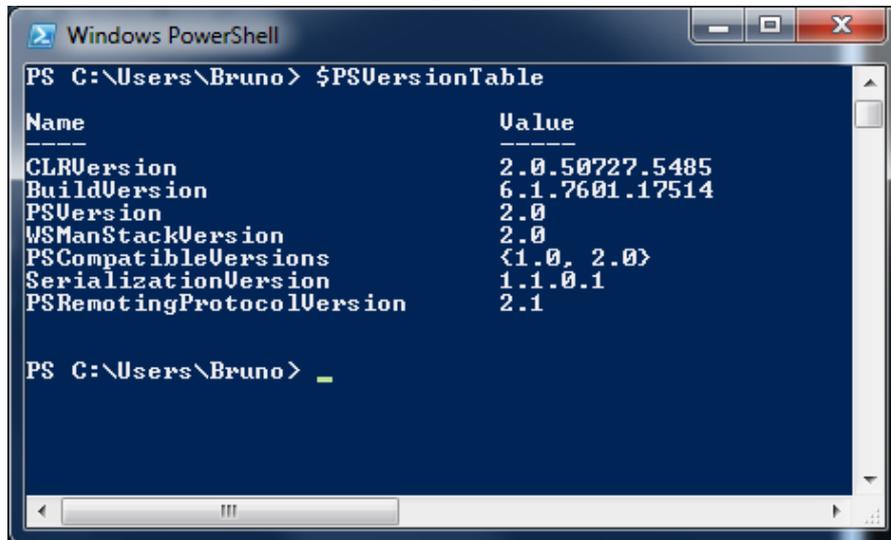
Realiza funções com base no objeto (linha de comando)

- ▶ Microsoft **Windows PowerShell** é um novo prompt de comando do **Windows**, muito mais poderoso que o `cmd.exe`, voltado à automatização, via scripts e canalização de objetos por uma sequência de comandos, para manutenção de sistemas por parte de administradores, além de um controle maior do sistema.

Windows PowerShell

Alterando algumas configurações da janela:

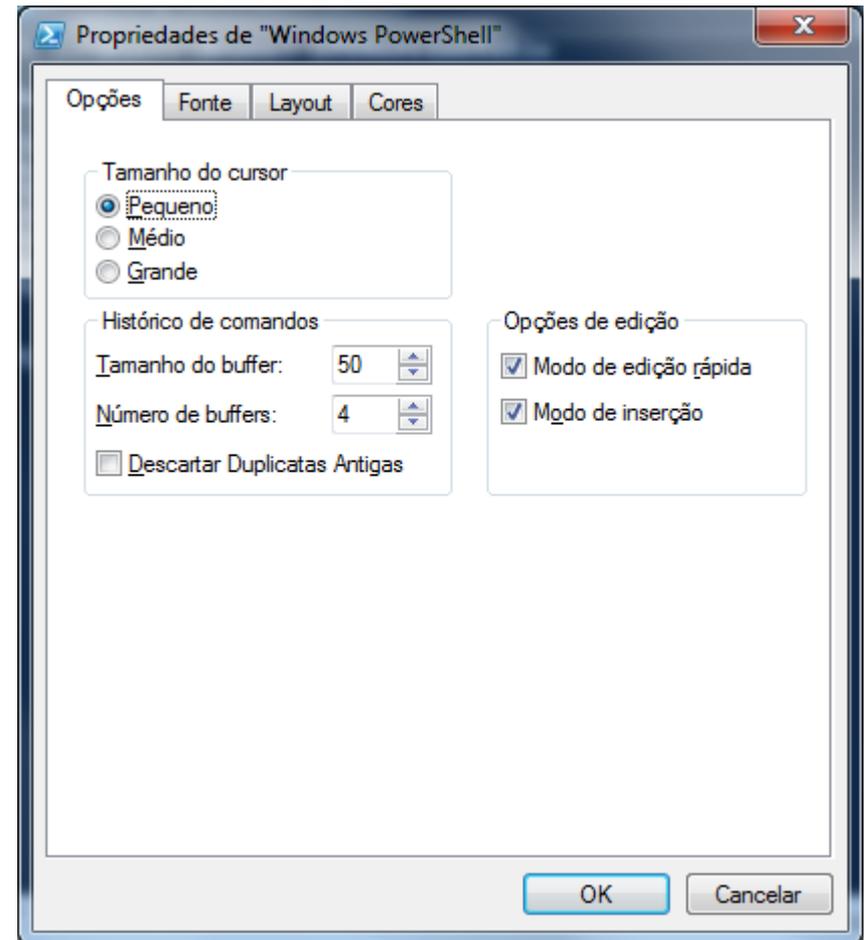
Verificando Versão:



```
Windows PowerShell
PS C:\Users\Bruno> $PSVersionTable

Name                           Value
----                           -
CLRVersion                     2.0.50727.5485
BuildVersion                   6.1.7601.17514
PSVersion                      2.0
WSManStackVersion              2.0
PSCompatibleVersions           <1.0, 2.0>
SerializationVersion          1.1.0.1
PSRemotingProtocolVersion      2.1

PS C:\Users\Bruno> _
```



Windows PowerShell

Comando HELP

```
Administrador: Windows PowerShell
PS C:\Users\Bruno> Get-Help Get-Event

NOME
    Get-Event

SINOPSE
    Obtém os eventos da fila de eventos.

SINTAXE
    Get-Event [-EventIdentifier] <int> [<CommonParameters>]
    Get-Event [[-SourceIdentifier] <string>] [<CommonParameters>]

DESCRIÇÃO
    O cmdlet Get-Event obtém eventos na fila de eventos do Windows PowerShell durante a sessão atual. Você pode obter todos os eventos ou usar o parâmetro EventIdentifier ou SourceIdentifier para especificar os eventos.

    Quando ocorre um evento, ele é adicionado à fila de eventos. A fila de eventos inclui eventos para os quais você se registrou, eventos criados usando o cmdlet New-Event e o evento emitido quando o Windows PowerShell é fechado. É possível usar Get-Event ou Wait-Event para obter os eventos.

    Esse cmdlet não obtém eventos dos logs do Visualizador de Eventos. Para obter esses eventos, use Get-WinEvent ou Get-EventLog.

LINKS RELACIONADOS
    Online version: http://go.microsoft.com/fwlink/?LinkID=113453
    Register-ObjectEvent
    Register-EngineEvent
    Register-WmiEvent
    Unregister-Event
    New-Event
    Remove-Event
    Wait-Event

COMENTÁRIOS
    Para ver os exemplos, digite:
    Para obter mais informações, digite:
    Para obter informações de sintaxe, digite:

PS C:\Users\Bruno>
```

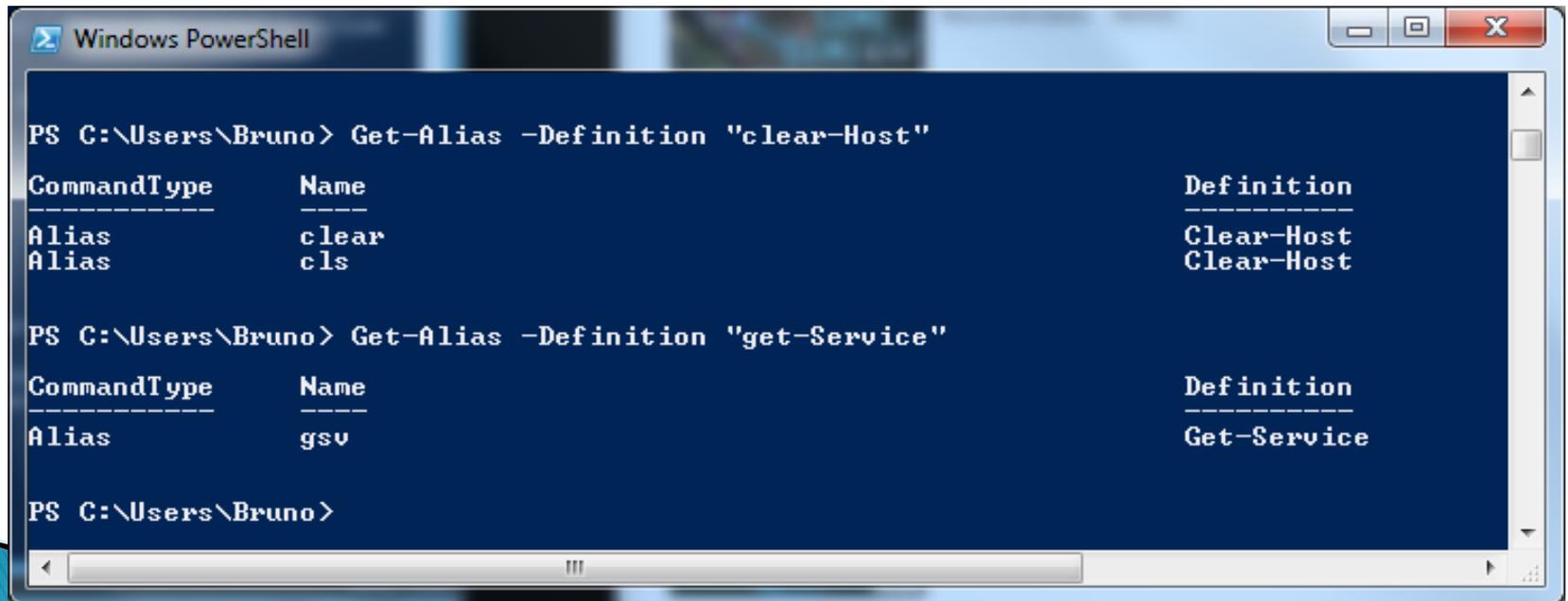
```
Administrador: Windows PowerShell
PS C:\Users\Bruno> help *dir*

Name                Category  Synopsis
-----
rmdir               Alias    Remove-Item
chdir               Alias    Set-Location
dir                 Alias    Get-ChildItem
mkdir              Function
about_Redirection  HelpFile  Descreve como redirecionar a saída do Windows PowerShell para

PS C:\Users\Bruno> _
```

Windows PowerShell

- ▶ Comandos Legados (aliases):
 - cls = Clear-Host
 - dir = Get-ChildItem
 - Get-Alias (para verificar demais comandos)



```
Windows PowerShell

PS C:\Users\Bruno> Get-Alias -Definition "clear-Host"

CommandType      Name              Definition
-----
Alias             clear            Clear-Host
Alias             cls              Clear-Host

PS C:\Users\Bruno> Get-Alias -Definition "get-Service"

CommandType      Name              Definition
-----
Alias             gsv              Get-Service

PS C:\Users\Bruno>
```

Windows PowerShell

- ▶ **cmdlets** do PowerShell
 - Sintaxe: Estilo Verbo-Substantivo
- ▶ Anatomia de um comando:

Comando	Parâmetro 01	Parâmetro 02
Get-EventLog	-LogName Security	-ComputerName Bruno

- ▶ Parâmetros podem se apresentar de várias formas:
 - Parâmetro / Valor
 - Parâmetro / Valores múltiplos
 - Parâmetro sem valores

Windows PowerShell

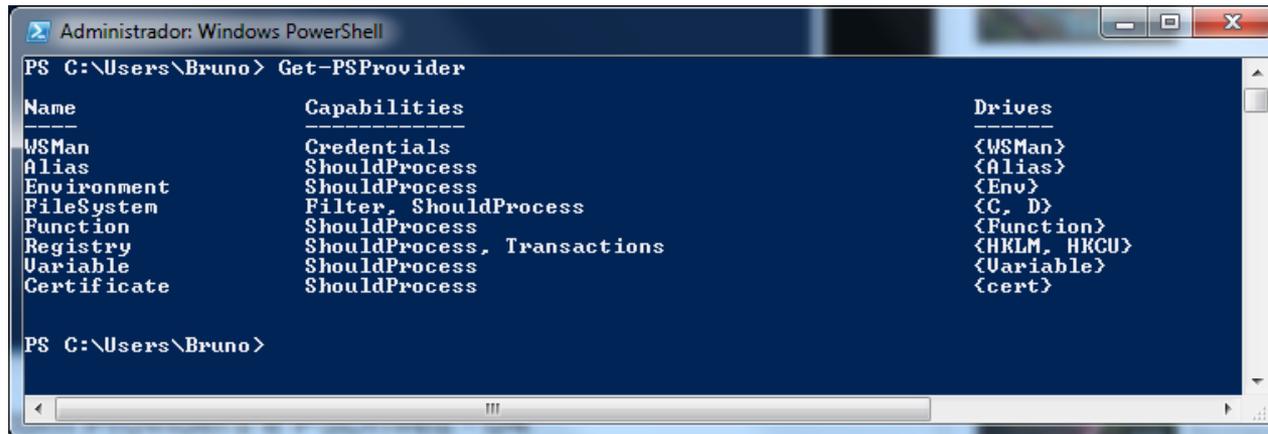
- ▶ Visualizando a lista de verbos aprovados
 - Get-Verb
- ▶ Usando Aliases
 - Alias = Nome alternativo de um cmdlet
- ▶ Exemplo:
 - cmdlet = Get-ChildItem
 - Alias = dir
 - Alias = ls

Windows PowerShell

- ▶ Criando seu próprio Alias
 - Exemplo:
 - `New-Alias -Name serv -Value Get-Service`
- ▶ Todos comandos criados com `New-Alias` são perdidos quando se fecha o Shell, para se manter tem que adicionar os Alias no script do perfil do usuário que está executando o PowerShell.

Windows PowerShell

- ▶ Providers do PowerShell
 - PSPProvider = adaptador.
- ▶ Get-PSPProvider



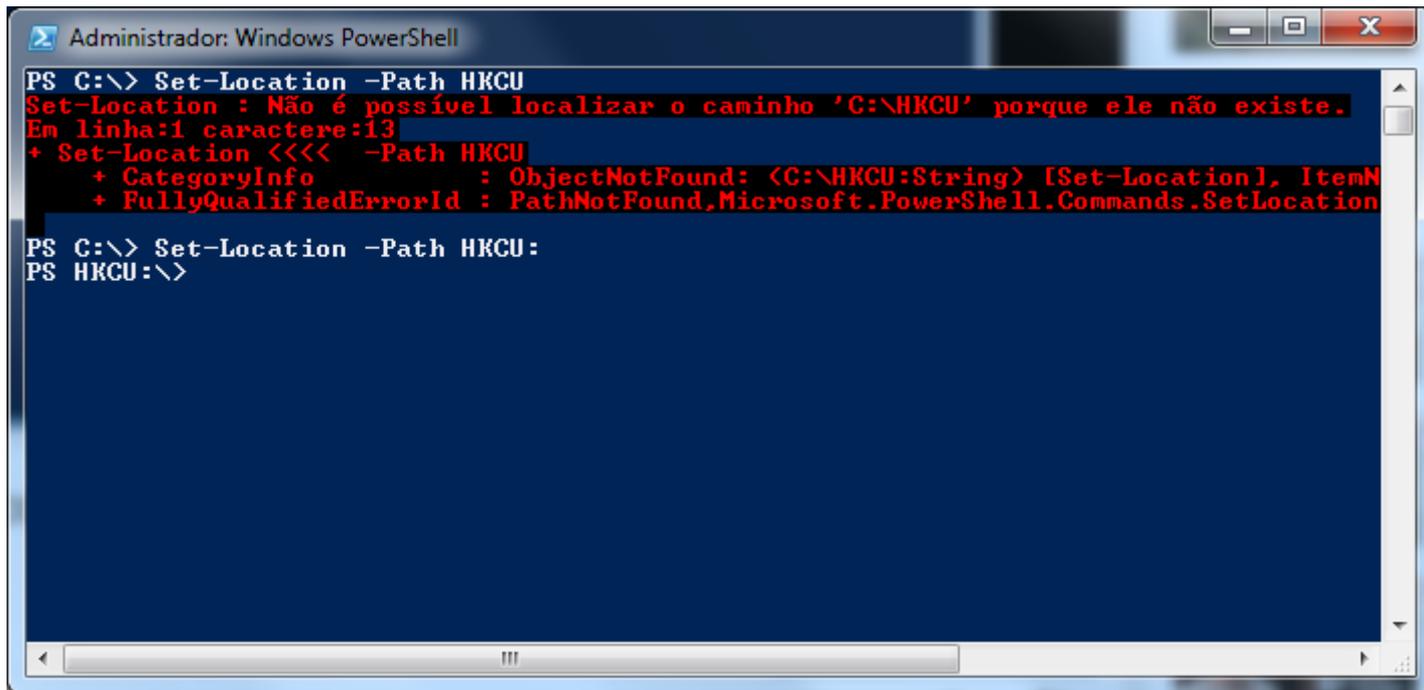
```
Administrador: Windows PowerShell
PS C:\Users\Bruno> Get-PSPProvider

Name                Capabilities                Drives
-----                -
WSMan                Credentials                  <WSMan>
Alias                ShouldProcess                <Alias>
Environment          ShouldProcess                <Env>
FileSystem           Filter, ShouldProcess       <C, D>
Function             ShouldProcess                <Function>
Registry             ShouldProcess, Transactions <HKLM, HKCU>
Variable             ShouldProcess                <Variable>
Certificate          ShouldProcess                <cert>

PS C:\Users\Bruno>
```

- ▶ PSDrive - Conexões a vários tipos de storage
- ▶ Get-PSDrive = mostra os drives conectados atualmente ao PowerShell.

Windows PowerShell



```
Administrador: Windows PowerShell
PS C:\> Set-Location -Path HKCU
Set-Location : Não é possível localizar o caminho 'C:\HKCU' porque ele não existe.
Em linha:1 caractere:13
+ Set-Location <<<< -Path HKCU
+ CategoryInfo          : ObjectNotFound: (C:\HKCU:String) [Set-Location], ItemN
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.SetLocation

PS C:\> Set-Location -Path HKCU:
PS HKCU:\>
```

Windows PowerShell

▶ Variáveis

- `$Nome = "Bruno"`
- `$Idade = 32`
 - Se quiser especificar o tipo de variável:
 - `[String]$Nome = "Bruno"`

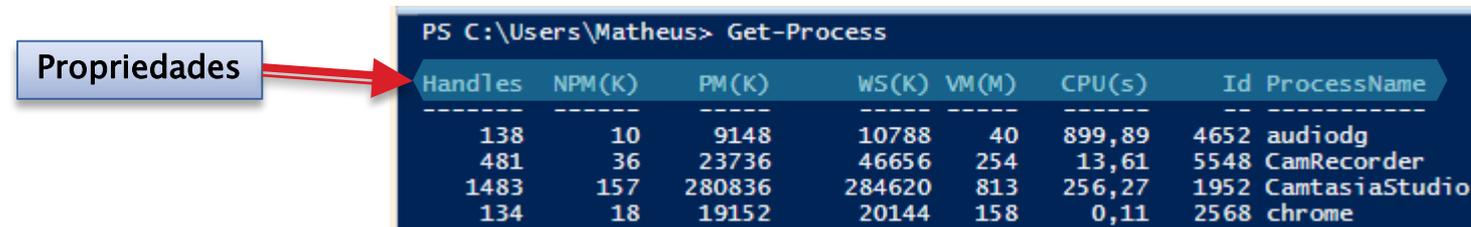
▶ Vetores

- `$Nomes = "Bruno", "Vinícius", "Roberto"`
- `$Nomes[2]` (Exibirá o nome Roberto)
 - `$Nomes += "Adão"` (aumentando um vetor existente)

Windows PowerShell

▶ Objetos

- São entidades que possuem propriedades e métodos.



```
PS C:\Users\Matheus> Get-Process
```

Handles	NPM(K)	PM(K)	WS(K)	VM(M)	CPU(s)	Id	ProcessName
138	10	9148	10788	40	899,89	4652	audiodg
481	36	23736	46656	254	13,61	5548	CamRecorder
1483	157	280836	284620	813	256,27	1952	CamtasiaStudio
134	18	19152	20144	158	0,11	2568	chrome

- Em cima dessas propriedades podemos aplicar os operadores comparativos e condicionais.
 - `$Process = Get-Process`
 - `$Process[0]`
- Podemos listar todas propriedades e métodos de um objeto utilizando o comando: **Get-Process | Get-Member** que nos retorna a lista das propriedades e métodos desse objeto, se quisermos acessar uma propriedade ou um método do objeto pode fazer isso dessa maneira.
 - `$Process[0].Name`

Windows PowerShell

▶ Operadores Comparativos

- **-eq**, Igual a, utilizado para comparar números e textos.
- **-lt**, Menor que, Utilizado para comparar números.
- **-gt**, Maior que, utilizado para comparar números.
- **-ge**, Maior ou igual a, utilizado para comparar números.
- **-le**, Menor ou igual a, utilizado para comparar números.
- **-ne**, Não igual a, utilizado para comparar números e textos.
- **-contains**, Contém, utilizado para comparar vetores.
- **-like**, Igual, utilizado para comparar textos ou números.
- **-notlike**, Não igual a.
- **-is**, é, utilizado para comparar tipos de objetos.
- **-isnot**, não é, utilizado para comparar tipos de objetos

Windows PowerShell

▶ Operadores Condicionais

- **if**, verifica se a condição for verdadeira.
- **else**, utilizado após o **if**, verifica se a condição não foi verdadeira.
- **elseif**, utilizado após um **if** ou outro **elseif**, verifica se a condição anterior não for verdadeira e uma nova condição for.
- **while**, utilizado para repetir um bloco enquanto uma condição for verdadeira.
- **for**, utilizado para repetir um bloco um determinado número de vezes.
- **foreach**, utilizado para repetir um bloco uma vez para cada item em uma lista.
- **where**, utilizado para filtrar itens em uma lista.
- **switch**, utilizado para executar múltiplas verificações substituindo uma cadeia de ifs.

Windows PowerShell

▶ Exercício

- Se eu quisesse exibir em tela de 0 a 19 eu faria o seguinte script:

```
PS C:\Users\Bruno> for<$i=0;$i -lt 20; $i++> {Write-Host "$i subindo..." }  
0 subindo...  
1 subindo...  
2 subindo...  
3 subindo...  
4 subindo...  
5 subindo...  
6 subindo...  
7 subindo...  
8 subindo...  
9 subindo...  
10 subindo...  
11 subindo...  
12 subindo...  
13 subindo...  
14 subindo...  
15 subindo...  
16 subindo...  
17 subindo...  
18 subindo...  
19 subindo...
```

- Escreva um script no PowerShell que exiba os números **pares** de 2 a 20 na tela. Dica: $\$i+=2$

Windows PowerShell

▶ Redirecionando Saídas dos cmdlets

- Por exemplo, queremos matar o processo do Notepad que esteja rodando em nosso computador, podemos executar:
 - `Get-Process -Name "Notepad" | Stop-Process`
- Quanto utilizamos a pipeline para redirecionar a saída de um comando para outro, ele passa um objeto por vez, sendo o objeto passa pela pipeline referenciado como `$_` do outro lado, sendo assim se você quiser aplicar um filtro utilizando o **Where-Object**, você pode fazer dessa maneira.
 - `Get-Process | Where-Object { $_.Name -like "Notepad" }`

Windows PowerShell

▶ Redirecionando Saídas dos cmdlets

- Outro comando muito utilizado após a pipeline é o **ForEach-Object**, que serve caso nos queiramos executar uma ação para cada um dos objetos passados na pipeline, por exemplo.
 - `Get-Process | Where-Object { $_.Name -like "N*" } | ForEach-Object {Write-Host $_.Name" começa com N"}`

Windows PowerShell

▶ Encontrando os cmdlets

- Se eu quero pegar alguma informação já sei que devo procurar um comando que comece com **Get**, se quero alterar alguma coisa procuro por **Set**, se quero criar algo novo procuro por **New** ou **Add**, se quero remover **Remove**.
- Queremos encontrar algum comando para acharmos algumas informações do disco, então executamos o comando:
 - **Get-Command "Get-*Drive*"**