

## Must have commands

Command	Meaning
<code>gsv   clip</code>	Clips output of gsv to the clipboard

## Basic commands

Command	Meaning
<code>cls &amp; Clear-Host</code>	clear the screen
<code>cd</code>	change directory
<code>cd\</code>	go to root directory
<code>dir &amp; ls &amp; Get-Childitem</code>	shows whats in the directory
<code>cat &amp; Get-Content</code>	output the file
<code>Copy &amp; cp &amp; Copy-Item</code>	copy the file
<code>Measure-command {Get-Process}</code>	Shows you the execution time

## Help system

Command	Meaning
<code>Update-Help -Force</code>	Update the help
<code>Save-Help</code>	Save the help to a local file
<code>man</code>	Scrollable
<code>help</code>	Alias
<code>Get-Help "*keyword*"</code>	Get help
<code>Get-Help Get-*</code>	Get all commands which start with "Get-"
<code>Get-Help</code>	<code>-ShowWindow</code> <code>-Detailed</code> <code>-Examples</code> <code>-Full</code> <code>-Online</code> <code>-ShowCommand</code>
<code>Get-Help -Category Provider</code>	
<code>Get-verb</code>	Get the verbs used in PowerShell
<code>Get-Alias -Definition get-process</code>	Get all aliases for <code>Get-Process</code>
<code>Get-Help About_*</code>	Get all the about topics
<code>Get-Process   Get-Member</code>	Get all the methods, properties, ... of the object
<code>Get-Command -Module AD*</code>	Get all commands from modules which start with AD
<code>Get-Help about*   Out-GridView -PassThru   Get-Help -ShowWindow</code>	This will display a grid view with all about topics to ch

## Export

Command	Meaning
<code>Export-csv</code>	
<code>Export-Clixml</code>	
<code>Out-File</code>	
<code>Out-Printer</code>	
<code>Out-GridView</code>	
<code>Out-File</code>	
<code>ConvertTo-Csv</code>	

Command	Meaning
ConvertTo-Html	

## Pipeline

**Example: you want to check if a service is running on all the computers in the AD**

Command	Meaning
Get-ADComputer -Filter *   gm	Check what object you are working with (in this case ADComputer)
Get-Help Get-Service -ShowWindow	Search for -InputObject and see what it accepts (in this case ServiceController != ADComputer)
Get-ADComputer -Filter *	-ComputerName supports ByPropertyName Check for propertyname -Name gives you the name of the computer

```
Get-ADComputer -Filter * -Name | Select -Property @{n(ame)='ComputerName';e(xpression)={$_.name}} |
Get-Service -Name bits
```

**Example: command does not accept pipeline input (Get-WmiObject)**

```
Get-WmiObject -class win32_bios -ComputerName (Get-ADComputer -Filter *).Name
```

Or

```
Get-ADComputer -Filter * | Get-WmiObject win32_bios -Computername {$_.Name}
```

## Create new property

Command	Meaning
Select -Property @{n(ame)='ComputerName';e(xpression)={\$_.name}}	Creates a new property called ComputerName from the name
Select -ExpandProperty name	Return the property in a array

## WmiObject & CimInstance

## Remoting

Command	Meaning
Invoke-Command -ComputerName dc {Restart-Computer}	Execute the command on specific computers
icm dc {Restart-Computer}	Short version

## Execution Policy

Command	Meaning
Restricted	
Unrestricted	Run any script ( <b>not recommended</b> )
AllSigned	
RemoteSigned	Only run scripts that are remote signed
Bypass	
Undefined	

## Powershell Remote Session

Command	Meaning
Get-PSSession	Get open sessions
\$sessions = New PSSession -ComputerName dc	Open session
icm -Session \$sessions {...}	Execute command in script block to all sessions

## Parallelism

**Parallel:** `icm -Computername (Get-Content servers.txt) { ... }`

**Serial:** `foreach ($s in (Get-Content servers.txt) ) { icm -Computername $s { ... } }`

## Scripting

Command	Meaning
CTRL + J	Helps you build a script
\$var = 1	Declares new variable <b>var</b> with value 1
param(\$ComputerName)	Create new parameter
[ValidateSet("a","b","c")] [string]\$x = "a"	Value of \$x <b>must</b> be a value of the ValidateSet
"this is \$var"	" Resolves the value of \$var
'this is \$var'	' Does <b>not</b> resolve the value
"The value of &#96;\$var is \$var"	the ' (&#96;) prevents that the value is resolved
@' '@	Multiline string, usefull for New-Snippet
-ErrorAction	
-ErrorVariable	