RÉFÉRENCE RAPIDE POWERSHELL

Cmdlets, pipeline, objets, scripts, modules

Bases

Commandes et aide

```
Get-Help Get-Process         # show help for cmdlet
Get-Help Get-Process -Online # open online docs
Get-Command *service*        # find commands by name
Get-Alias ls                 # show alias target
```

Alias courants

`ls → Get-ChildItem`	Lister les fichiers et répertoires
`cd → Set-Location`	Changer de répertoire
`cp → Copy-Item`	Copier un fichier ou répertoire
`mv → Move-Item`	Déplacer ou renommer
`rm → Remove-Item`	Supprimer un fichier ou répertoire
`cat → Get-Content`	Lire le contenu d'un fichier
`echo → Write-Output`	Afficher dans le pipeline
`cls → Clear-Host`	Effacer la console

Variables

Bases des variables

```
$name = "Alice"          # string
$count = 42              # integer
$pi = 3.14               # double
$list = @(1, 2, 3)       # array
$hash = @{a=1; b=2}      # hashtable
```

Variables automatiques

`$_`	Objet courant du pipeline
`$PSVersionTable`	Informations de version PowerShell
`$HOME`	Répertoire personnel de l'utilisateur
`$PWD`	Répertoire courant
`$null`	Valeur nulle
`$true / $false`	Constantes booléennes
`$Error`	Tableau des erreurs récentes
`$LASTEXITCODE`	Code de sortie de la dernière commande native

Variables d'environnement

```
$env:PATH                # read env var
$env:MY_VAR = "value"    # set env var
Get-ChildItem Env:       # list all env vars
```

Opérateurs

Opérateurs de comparaison

`-eq  -ne`	Égal / différent
`-gt  -lt`	Supérieur / inférieur
`-ge  -le`	Supérieur ou égal / inférieur ou égal
`-like  -notlike`	Correspondance avec joker (`*`, `?`)
`-match  -notmatch`	Correspondance regex
`-contains`	La collection contient la valeur
`-in  -notin`	Valeur dans la collection

Opérateurs logiques et autres

`-and  -or  -not`	Opérateurs logiques
`!`	NON logique (alias)
`-replace`	Remplacement regex : `'hi' -replace 'h','b'`
`-split  -join`	Diviser / joindre des chaînes
`..`	Intervalle : `1..5` → 1,2,3,4,5
`?:`	Ternaire (v7+) : `$x ? 'yes' : 'no'`

Flux de contrôle

if / elseif / else

```
if ($age -ge 18) {
    "Adult"
} elseif ($age -ge 13) {
    "Teen"
} else {
    "Child"
}
```

switch

```
switch ($color) {
    "red"   { "Stop" }
    "green" { "Go" }
    default { "Unknown" }
}
```

Boucles

```
foreach ($item in $list) { $item }
for ($i=0; $i -lt 5; $i++) { $i }
while ($x -lt 10) { $x++ }
1..5 | ForEach-Object { $_ * 2 }
```

Fonctions

Définir des fonctions

```
function Get-Greeting {
    param([string]$Name = "World")
    "Hello, $Name!"
}
Get-Greeting -Name "Alice"
```

Paramètres avancés

```
function Copy-SafeFile {
    [CmdletBinding()]
    param(
        [Parameter(Mandatory)][string]$Path,
        [Parameter(Mandatory)][string]$Dest
    )
    Copy-Item $Path $Dest -WhatIf:$WhatIfPreference
}
```

Objets et pipeline

Bases du pipeline

```
Get-Process | Sort-Object CPU -Desc | Select-Object -First 5
Get-Service | Where-Object Status -eq "Running"
Get-ChildItem | Measure-Object -Property Length -Sum
```

Cmdlets de pipeline clés

`Where-Object`	Filtrer les objets : `| Where {$_.CPU -gt 10}`
`Select-Object`	Sélectionner des propriétés : `| Select Name, CPU`
`Sort-Object`	Trier : `| Sort CPU -Desc`
`ForEach-Object`	Transformer chaque objet : `| ForEach {$_.Name}`
`Measure-Object`	Compter, sommer, moyenner, min, max
`Group-Object`	Regrouper par valeur de propriété
`Format-Table`	Afficher en tableau : `| ft -Auto`
`Export-Csv`	Exporter en CSV : `| Export-Csv out.csv`

Fichiers

Opérations sur les fichiers

```
Get-Content log.txt                    # read file
Set-Content out.txt "hello"            # write (overwrite)
Add-Content out.txt "more"             # append
Get-Content log.txt | Select-String "error"  # grep
```

Cmdlets de chemin et fichier

`Test-Path $path`	Vérifier si fichier/répertoire existe
`New-Item -Type File`	Créer un fichier
`New-Item -Type Directory`	Créer un répertoire
`Resolve-Path`	Obtenir le chemin absolu
`Join-Path`	Combiner des segments de chemin
`Split-Path`	Obtenir le parent ou la feuille
`Get-ItemProperty`	Attributs et métadonnées du fichier
`Remove-Item -Recurse`	Supprimer un répertoire récursivement

Chaînes

Bases des chaînes

```
"Hello, $name"               # interpolation (double quotes)
'Hello, $name'               # literal (single quotes)
"Length: $($list.Count)"      # expression in string
@"
Multi-line
here-string with $name
"@
```

Méthodes de chaîne

`.ToUpper() / .ToLower()`	Changer la casse
`.Trim()`	Supprimer les espaces de début/fin
`.Split(',')`	Diviser en tableau
`.Replace('a','b')`	Remplacer une sous-chaîne
`.StartsWith() / .EndsWith()`	Vérifier le préfixe / suffixe
`.Substring(0,5)`	Extraire une sous-chaîne
`.Contains('text')`	Vérifier si contient
`-f operator`	`'{0} is {1}' -f 'sky','blue'`

Modules

Gestion des modules

```
Get-Module -ListAvailable          # installed modules
Find-Module -Name Az*              # search gallery
Install-Module -Name Pester        # install from gallery
Import-Module ActiveDirectory      # load module
```

Commandes de module

`Get-Module`	Lister les modules chargés
`Import-Module`	Charger un module dans la session
`Remove-Module`	Décharger un module de la session
`Update-Module`	Mettre à jour un module installé
`Get-Command -Module X`	Lister les commandes dans un module
`$env:PSModulePath`	Chemins de recherche de modules

Motifs courants

Gestion des erreurs

```
try {
    Get-Item "C:\missing" -ErrorAction Stop
} catch {
    "Error: $_"
} finally {
    "Cleanup here"
}
```

Politique d'exécution et remoting

`Get-ExecutionPolicy`	Afficher la politique de script actuelle
`Set-ExecutionPolicy RemoteSigned`	Autoriser les scripts locaux
`Enter-PSSession -Computer SRV1`	Session distante interactive
`Invoke-Command -Computer SRV1 -Script {}`	Exécuter un bloc de script à distance
`Start-Job { long-task }`	Exécuter une tâche en arrière-plan
`Receive-Job -Id 1`	Obtenir la sortie d'une tâche en arrière-plan