REFERÊNCIA RÁPIDA DE POWERSHELL

Cmdlets, pipeline, objetos, scripting, módulos

Básico

Comandos e Ajuda

```
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
```

Aliases Comuns

`ls → Get-ChildItem`	Listar arquivos e diretórios
`cd → Set-Location`	Mudar diretório
`cp → Copy-Item`	Copiar arquivo ou diretório
`mv → Move-Item`	Mover ou renomear
`rm → Remove-Item`	Excluir arquivo ou diretório
`cat → Get-Content`	Ler conteúdo de arquivo
`echo → Write-Output`	Imprimir no pipeline
`cls → Clear-Host`	Limpar console

Variáveis

Noções Básicas de Variáveis

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

Variáveis Automáticas

`$_`	Objeto atual no pipeline
`$PSVersionTable`	Informações de versão do PowerShell
`$HOME`	Diretório pessoal do usuário
`$PWD`	Diretório atual
`$null`	Valor nulo
`$true / $false`	Constantes booleanas
`$Error`	Array de erros recentes
`$LASTEXITCODE`	Código de saída do último comando nativo

Variáveis de Ambiente

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

Operadores

Operadores de Comparação

`-eq  -ne`	Igual / diferente
`-gt  -lt`	Maior / menor que
`-ge  -le`	Maior/menor ou igual
`-like  -notlike`	Correspondência com curinga (`*`, `?`)
`-match  -notmatch`	Correspondência com regex
`-contains`	Coleção contém valor
`-in  -notin`	Valor está na coleção

Operadores Lógicos e Outros

`-and  -or  -not`	Operadores lógicos
`!`	NOT lógico (alias)
`-replace`	Substituição regex: `'hi' -replace 'h','b'`
`-split  -join`	Dividir / juntar strings
`..`	Intervalo: `1..5` → 1,2,3,4,5
`?:`	Ternário (v7+): `$x ? 'yes' : 'no'`

Controle de Fluxo

if / elseif / else

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

switch

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

Laços

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

Funções

Definindo Funções

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

Parâmetros Avançados

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

Objetos e Pipeline

Noções Básicas de 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 Principais de Pipeline

`Where-Object`	Filtrar objetos: `| Where {$_.CPU -gt 10}`
`Select-Object`	Escolher propriedades: `| Select Name, CPU`
`Sort-Object`	Ordenar: `| Sort CPU -Desc`
`ForEach-Object`	Transformar cada: `| ForEach {$_.Name}`
`Measure-Object`	Contar, somar, média, mínimo, máximo
`Group-Object`	Agrupar por valor de propriedade
`Format-Table`	Exibir como tabela: `| ft -Auto`
`Export-Csv`	Exportar para CSV: `| Export-Csv out.csv`

Arquivos

Operações com Arquivos

```
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 Caminho e Arquivo

`Test-Path $path`	Verificar se arquivo/diretório existe
`New-Item -Type File`	Criar arquivo
`New-Item -Type Directory`	Criar diretório
`Resolve-Path`	Obter caminho absoluto
`Join-Path`	Combinar segmentos de caminho
`Split-Path`	Obter pai ou folha
`Get-ItemProperty`	Atributos e metadados do arquivo
`Remove-Item -Recurse`	Excluir diretório recursivamente

Strings

Noções Básicas de Strings

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

Métodos de String

`.ToUpper() / .ToLower()`	Mudar capitalização
`.Trim()`	Remover espaços no início/fim
`.Split(',')`	Dividir em array
`.Replace('a','b')`	Substituir substring
`.StartsWith() / .EndsWith()`	Verificar prefixo / sufixo
`.Substring(0,5)`	Extrair substring
`.Contains('text')`	Verificar se contém
`-f operator`	`'{0} is {1}' -f 'sky','blue'`

Módulos

Gerenciamento de Módulos

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

Comandos de Módulo

`Get-Module`	Listar módulos carregados
`Import-Module`	Carregar módulo na sessão
`Remove-Module`	Descarregar módulo da sessão
`Update-Module`	Atualizar módulo instalado
`Get-Command -Module X`	Listar comandos no módulo
`$env:PSModulePath`	Caminhos de busca de módulos

Padrões Comuns

Tratamento de Erros

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

Política de Execução e Comunicação Remota

`Get-ExecutionPolicy`	Exibir política de script atual
`Set-ExecutionPolicy RemoteSigned`	Permitir scripts locais
`Enter-PSSession -Computer SRV1`	Sessão remota interativa
`Invoke-Command -Computer SRV1 -Script {}`	Executar bloco de script remotamente
`Start-Job { long-task }`	Executar tarefa em segundo plano
`Receive-Job -Id 1`	Obter saída de tarefa em segundo plano