Fiche de révision — PowerShell

I. Principe et syntaxe

PowerShell est un **environnement de commande**, comme l'invite de commandes MS-DOS (`cmd`), mais nettement plus riche et plus puissant. Là où `cmd` manipule du texte, PowerShell manipule des **objets** : le résultat d'une commande conserve ses propriétés et peut être passé tel quel à la commande suivante.

La composition d'une commande

Toute commande PowerShell, appelée **cmdlet**, suit la même structure :

```
Verbe-Nom -Parametre Valeur

New-Item      -ItemType Directory
Copy-Item     -Path "C:\Source\fichier.txt" -Destination "C:\Destination\"
Set-VMProcessor -VMName "SRVPA01" -Count 4
```

Le couple **Verbe-Nom** se traduit par **Action-Objet** : le verbe dit ce qu'on fait, le nom dit sur quoi. Les paramètres qui suivent se signalent par un tiret `-`, là où MS-DOS utilise un slash (`ipconfig /all`).

	MS-DOS / cmd	PowerShell
Séparateur de paramètre	`/`	`-`
Exemple	`ipconfig /all`	`Get-NetIPConfiguration -Detailed`
Ce qui circule	Du texte	Des objets, avec leurs propriétés

La vraie difficulté — la syntaxe s'apprend en dix minutes. Ce qui est difficile en PowerShell, c'est de trouver la bonne cmdlet. D'où l'importance de savoir chercher dans les commandes disponibles avant de chercher sur Internet.

Trouver la bonne commande

Cmdlet	Rôle
`Get-Command`	Liste toutes les commandes présentes sur la machine.
`Get-Command *VM*`	Filtre la liste sur un mot-clé. Le réflexe de base quand on cherche une cmdlet.
`Get-Verb`	Liste les verbes approuvés. Sert à deviner le nom d'une cmdlet inconnue.
`Get-Help New-VM -Full`	Affiche l'aide complète d'une cmdlet, ses paramètres et ses exemples.
`Get-Module -ListAvailable`	Liste les modules installés, donc les jeux de commandes disponibles.

PowerShell s'enrichit avec les rôles

PowerShell n'est pas figé. Chaque **rôle installé ajoute sa bibliothèque de commandes** : Active Directory, DHCP, DNS, Hyper-V, chacun apporte son **module**. Installer le rôle DNS sur un serveur, c'est aussi y installer les cmdlets `Get-DnsServerZone`, `Add-DnsServerResourceRecordA` et les autres.

Piège — il faut taper la commande sur la bonne machine. Une cmdlet Active Directory lancée depuis un poste client sans les outils d'administration renvoie « terme non reconnu » : ce n'est pas une faute de frappe, c'est le module qui n'est pas là.

Lire une erreur

Quand PowerShell affiche du **rouge**, on ne referme pas la fenêtre : on lit. De haut en bas et de gauche à droite. La première ligne donne la cmdlet fautive et le motif, les lignes suivantes donnent la catégorie et l'objet en cause.

> ***Exemple :*** *un `Copy-Item` dont le `-Path` ne correspond à aucun fichier réel renvoie une `ItemNotFoundException`. Le premier réflexe est de vérifier la **lettre de lecteur**, pas de réécrire la commande.*

II. Manipuler fichiers et dossiers

Cmdlet	Rôle	Paramètres clés
`New-Item`	Crée un élément : fichier, dossier, clé de registre.	`-Path`, `-Name`, `-ItemType`
`Copy-Item`	Copie un fichier ou un dossier.	`-Path`, `-Destination`, `-Recurse`
`Rename-Item`	Renomme un fichier ou un dossier.	`-Path`, `-NewName`
`Move-Item`	Déplace, et renomme en même temps si besoin.	`-Path`, `-Destination`
`Remove-Item`	Supprime un élément.	`-Path`, `-Recurse`, `-Force`

New-Item — le paramètre qui change tout

Valeur de -ItemType	Résultat
`Directory`	Crée un dossier.
`File`	Crée un fichier.
Omis	Un fichier vide est créé par défaut. C'est l'erreur classique quand on voulait un dossier.

```
New-Item -Path "D:\HYPER-V" -Name "SRVPA02" -ItemType Directory
```

> ***Précision :*** *`mkdir` fonctionne dans PowerShell, mais ce n'est pas une cmdlet native : c'est une **fonction** qui encapsule `New-Item -ItemType Directory`. L'alias `md`, lui, pointe vers cette fonction. À connaître pour ne pas la chercher dans la liste des cmdlets.*

Copy-Item et Rename-Item

Paramètre	Description
`-Path`	Chemin complet du fichier ou dossier source.
`-Destination`	Chemin complet de destination.
`-Recurse`	Copie un dossier avec tout son contenu.
`-NewName`	Nouveau nom, sans chemin. Uniquement le nom.

```
Copy-Item -Path "C:\MODELS\Model Server 2022.vhdx" -Destination "D:\HYPER-V\SRVPA01\"
Rename-Item -Path "D:\HYPER-V\SRVPA01\Model Server 2022.vhdx" -NewName "SRVPA01.vhdx"
```

> **À retenir — `-NewName` n'accepte que le nom, jamais un chemin complet. Pour déplacer et renommer en une seule opération, la cmdlet est `Move-Item`, pas `Rename-Item`.**

III. Exercice — créer une machine virtuelle

Étape 1 — créer le dossier de la VM

```
New-Item -Path "D:\HYPER-V" -Name "SRVPA02" -ItemType Directory
```

Étape 2 — copier le modèle dans le dossier

```
Copy-Item -Path "C:\MODELS\Model Server 2022.vhdx" -Destination "D:\HYPER-V\SRVPA02\"
```

Étape 3 — renommer le disque

```
Rename-Item -Path "D:\HYPER-V\SRVPA02\Model Server 2022.vhdx" -NewName "SRVPA02.vhdx"
```

Étape 4 — créer et configurer la VM

Cmdlet	Rôle
`New-VM`	Crée la VM et attache le .vhdx existant.
`Set-VMMemory`	Configure la mémoire dynamique.
`Set-VMProcessor`	Définit le nombre de cœurs.
`Connect-VMNetworkAdapter`	Raccorde la carte réseau virtuelle à un commutateur.
`Set-VM`	Règle les options générales, dont les points de contrôle automatiques.
`Start-VM`	Démarre la VM.

```
# 1. Creer la VM avec le disque attache
New-VM -Name "SRVPA01" -Path "D:\HYPER-V\SRVPA01\" -MemoryStartupBytes 4GB `
       -Generation 1 -VHDPath "D:\HYPER-V\SRVPA01\SRVPA01.vhdx"

# 2. Activer la memoire dynamique
Set-VMMemory -VMName "SRVPA01" -DynamicMemoryEnabled $true `
             -MinimumBytes 512MB -StartupBytes 4GB -MaximumBytes 4GB

# 3. Definir le nombre de coeurs
Set-VMProcessor -VMName "SRVPA01" -Count 4
```

Limite de New-VM — `New-VM` ne gère ni la mémoire dynamique ni le nombre de processeurs. Ces deux réglages passent obligatoirement par `Set-VMMemory` et `Set-VMProcessor` après la création.

Raccorder la carte réseau

```
Connect-VMNetworkAdapter -VMName "SRVPA01" -SwitchName "VSWITCH-PRIV02"
```

Paramètre	Description
`-VMName`	Nom de la VM ciblée.
`-SwitchName`	Nom du commutateur virtuel à connecter.

> ***Réflexe :*** *toujours faire un `Get-VMSwitch` avant, pour récupérer le nom exact du commutateur. Un nom approximatif produit une erreur, pas une correction automatique.*

Désactiver les points de contrôle automatiques

```
Set-VM -Name "SRVPA01" -AutomaticCheckpointsEnabled $false
```

Paramètre	Description
`-Name`	Nom de la VM ciblée.
`-AutomaticCheckpointsEnabled $false`	Désactive les points de contrôle automatiques.

Le script complet

```
New-Item -Path "D:\HYPER-V" -Name "SRVPA02" -ItemType Directory
Copy-Item -Path "C:\MODELS\Model Server 2022.vhdx" -Destination "D:\HYPER-V\SRVPA02\"
```

```
Rename-Item -Path "D:\HYPER-V\SRVPA02\Model Server 2022.vhdx" -NewName "SRVPA02.vhdx"
New-VM -Name "SRVPA02" -Path "D:\HYPER-V\SRVPA02\" -MemoryStartupBytes 4GB `
       -Generation 1 -VHDPath "D:\HYPER-V\SRVPA02\SRVPA02.vhdx" `
       -SwitchName "VSWITCH-PRIV02"
Set-VMProcessor -VMName "SRVPA02" -Count 4
Set-VM -Name "SRVPA02" -AutomaticCheckpointsEnabled $false
Start-VM -Name "SRVPA02"
```

> ***Note de saisie :*** *le caractère ` (accent grave) en fin de ligne est le* ***caractère de continuation*** *de PowerShell : il permet d'écrire une commande longue sur plusieurs lignes. Rien ne doit le suivre, pas même un espace.*

IV. Les variables

Une variable stocke une valeur pour la réutiliser plus loin dans le script. Elle se reconnaît au **signe dollar** et se déclare par simple affectation, sans annonce de type préalable.

```
$NomVM = "SRVPA02"
$Chemin = "D:\HYPER-V\$NomVM\"
New-VM -Name $NomVM -Path $Chemin -MemoryStartupBytes 4GB
```

L'intérêt est immédiat : pour créer une deuxième VM, on change **une seule ligne** au lieu de reprendre le nom à six endroits.

Les portées de variable

Portée	Étendue
Global	Disponible partout dans la session PowerShell en cours, y compris hors du script.
Script	Disponible dans tout le script, mais pas au-delà.
Local	Portée par défaut. Disponible dans le bloc courant uniquement.

Les variables automatiques

PowerShell alimente lui-même certaines variables. Elles ne se déclarent pas, elles se lisent.

Variable	Contient
`$_`	L'objet courant dans un pipeline.
`$?`	Le résultat de la dernière commande : $true si elle a réussi, $false sinon.
`$PSScriptRoot`	Le dossier dans lequel se trouve le script en cours d'exécution.
`$null`	L'absence de valeur.
`$true / $false`	Les deux valeurs booléennes.

V. Les booléens

George Boole, mathématicien du 19e siècle, a formalisé une algèbre reposant sur deux états seulement : **vrai** ou **faux**. C'est l'**algèbre de Boole**. En informatique tout repose là-dessus : un bit vaut 0 ou 1, faux ou vrai. Pas de milieu.

Valeur	Signification	Bit
`$true`	Vrai	1
`$false`	Faux	0

C'est pour cette raison que, pour activer ou désactiver quelque chose en PowerShell, on passe un booléen. C'est un interrupteur : allumé ou éteint, rien d'autre.

```
Set-VM -Name "SRVPA01" -AutomaticCheckpointsEnabled $false
Set-VMMemory -VMName "SRVPA01" -DynamicMemoryEnabled $true
```

Les opérateurs logiques

Opération	PowerShell	Python	Excel (français)
ET	`-and`	`and`	`=ET(A1>0; B1>0)`
OU	`-or`	`or`	`=OU(A1>0; B1>0)`
NON	`-not`	`not`	`=NON(A1>0)`

```
$true -and $false     # False
$true -or  $false     # True
-not $true            # False
```

Culture informatique — la logique booléenne est un fondamental commun à tous les langages et à tous les outils : PowerShell, Python, SQL, JavaScript, C#, jusqu'aux formules Excel. Boole a posé les bases au 19e siècle, et ça tourne encore partout en 2026. Pas mal pour un type sans ordinateur.

VI. Les fichiers de script

Extension	Nature	Exécuté par
`.bat`	Script MS-DOS / CMD	L'interpréteur de commandes Windows
`.reg`	Fichier de modification du registre	L'éditeur du registre
`.ps1`	Script PowerShell	PowerShell

PowerShell ISE

PowerShell ISE (Integrated Scripting Environment) est l'éditeur intégré : il permet d'écrire un script complet, de l'exécuter ligne par ligne ou en totalité, et d'enregistrer le tout en `.ps1`. C'est là qu'on assemble les commandes testées une à une dans la console.

Stratégie d'exécution — un `.ps1` ne se lance pas par double-clic, et PowerShell refuse par défaut d'exécuter les scripts. La stratégie d'exécution se consulte avec `Get-ExecutionPolicy` et se modifie avec `Set-ExecutionPolicy`, en session administrateur.

Mémo — à connaître par cœur

Question	Réponse
Structure d'une commande	`Verbe-Nom -Parametre Valeur`
Traduction de Verbe-Nom	Action-Objet.
Séparateur de paramètre	`-` en PowerShell, `/` en MS-DOS.
Nom d'une commande PowerShell	Une cmdlet.
Lister les commandes	`Get-Command`
Lister les verbes approuvés	`Get-Verb`
Créer un dossier	`New-Item -ItemType Directory`
Créer un fichier	`New-Item -ItemType File`
Oublier -ItemType	Un fichier vide est créé par défaut.
Copier un dossier entier	`Copy-Item -Recurse`
Renommer	`Rename-Item -NewName` (nom seul, pas de chemin)
Déplacer et renommer	`Move-Item`
Créer une VM	`New-VM`

Question	Réponse
Mémoire dynamique	`Set-VMMemory`, jamais `New-VM`
Nombre de cœurs	`Set-VMProcessor -Count`
Raccorder au commutateur	`Connect-VMNetworkAdapter -SwitchName`
Couper les points de contrôle auto	`Set-VM -AutomaticCheckpointsEnabled $false`
Démarrer une VM	`Start-VM`
Déclarer une variable	`$NomVM = "SRVPA02"`
Les deux booléens	`$true` et `$false`
Extension d'un script PowerShell	`.ps1`
Devant une erreur rouge	On lit. De haut en bas, de gauche à droite.