PowerShell - Création de scripts PowerShell (.ps1)

PowerShell permet d'exécuter des commandes une par une, mais son véritable potentiel apparaît lorsqu'on regroupe ces commandes dans des **scripts (.ps1)** afin d'automatiser des tâches répétitives et fiables. La création de scripts PowerShell permet de **gagner du temps**, **réduire les erreurs humaines** et **standardiser les actions** sur un système. Un script peut être réutilisé, partagé et exécuté de manière identique sur plusieurs machines. C'est une compétence clé pour l'administration système, le support informatique et l'automatisation des tâches courantes (gestion de fichiers, diagnostics réseau, inventaires, sauvegardes).

Qu'est-ce qu'un script PowerShell ?

Un script PowerShell est un fichier texte avec l'extension `.ps1` qui contient une suite de commandes PowerShell exécutées automatiquement.

Exemple de fichier :

`analyse_fichiers.ps1`

Pour exécuter un script :

```
.\analyse_fichiers.ps1
```

Les variables

Déclaration d'une variable

En PowerShell, une variable commence toujours par `$`.

```
$nom = "Alice"
$age = 17
```

PowerShell est **faiblement typé** : il devine le type automatiquement.

Utilisation d'une variable

```
Write-Host "Nom : $nom"
Write-Host "Âge : $age"
```

Variables numériques

```
$a = 10
$b = 3
$somme = $a + $b
```

Les conditions (if / elseif / else)

Structure de base

```powershell
if (condition) {
    instructions
}
```

Exemple concret : vérifier la taille d'un fichier

```powershell
$fichier = Get-Item "test.txt"

if ($fichier.Length -gt 1MB) {
    Write-Host "Fichier volumineux"
}
else {
    Write-Host "Fichier de taille normale"
}
```

Comparateurs principaux

Opérateur	Signification
`-eq`	égal
`-ne`	différent
`-gt`	>
`-lt`	<
`-ge`	≥
`-le`	≤
`-and`	ET
`-or`	OU

Les boucles

Boucle `foreach` (la plus utilisée)

```powershell
$fichiers = Get-ChildItem -File

foreach ($fichier in $fichiers) {
    Write-Host $fichier.Name
}
```

Boucle `for`

```powershell
for ($i = 1; $i -le 5; $i++) {
    Write-Host "Compteur : $i"
}
```

Boucle `while`

```powershell
$compteur = 1

while ($compteur -le 3) {
```

```
    Write-Host $compteur
    $compteur++
}
```

Les fonctions

Définir une fonction

Une fonction permet de **réutiliser du code**.

```
function Dire-Bonjour {
    Write-Host "Bonjour !"
}
```

Appel :

```
Dire-Bonjour
```

Fonction avec paramètres

```
function Calculer-Somme {
    param (
        $a,
        $b
    )

    return $a + $b
}
```

Utilisation :

```
$resultat = Calculer-Somme -a 5 -b 7
```

Exemple de script complet (concret)

Objectif

Lister les fichiers d'un dossier, afficher leur taille, et signaler ceux de plus de 1 MB.

```
$fichiers = Get-ChildItem -File

foreach ($fichier in $fichiers) {

    if ($fichier.Length -gt 1MB) {
        Write-Host "$($fichier.Name) : gros fichier"
    }
    else {
        Write-Host "$($fichier.Name) : fichier normal"
    }
```

```
11
12  }
```

Bonnes pratiques de base

- Donner des **noms clairs** aux variables et fonctions
- Une fonction = **une tâche précise**
- Tester le script **dans un petit dossier** avant un vrai environnement
- Utiliser `Select-Object` et `Where-Object` avant les boucles quand c'est possible

Conclusion

Les scripts PowerShell permettent d'**automatiser des tâches système** de manière fiable et lisible.

En maîtrisant :

- les **variables**,
- les **conditions**,
- les **boucles**,
- les **fonctions**,

tu disposes déjà d'une base solide pour écrire de vrais scripts professionnels.