Powershell - Environnement et prise en main

PowerShell est aujourd'hui l'outil de référence pour l'administration Windows moderne. Il est utilisé aussi bien pour des commandes ponctuelles que pour l'automatisation, le DevOps et la gestion d'infrastructures. Avant d'écrire des scripts ou de manipuler des objets, il est essentiel de comprendre dans quel environnement PowerShell s'exécute et quelles règles de sécurité encadrent son utilisation.

Ce chapitre présente les **environnements d'exécution de PowerShell** et les **règles de sécurité liées aux scripts**. Il ne s'agit pas encore d'écrire des scripts complexes, mais de comprendre **où**, **comment** et **dans quel cadre sécurisé** PowerShell fonctionne.

Environnements PowerShell

PowerShell peut être utilisé à différents niveaux, selon le contexte et les besoins.

Console PowerShell

La **console** PowerShell est l'interface de base. Elle permet d'exécuter des commandes **une par une**, directement dans le terminal.

Caractéristiques principales :

- intégrée nativement à Windows
- idéale pour les commandes rapides et l'administration courante
- environnement minimaliste, orienté exécution immédiate

C'est le point d'entrée le plus simple vers PowerShell.

PowerShell ISE

PowerShell ISE est un ancien environnement graphique intégré à Windows PowerShell 5.1. Il n'est plus développé par Microsoft, mais reste présent sur de nombreux systèmes.

Intérêts pédagogiques :

- séparation claire entre console et zone de script
- autocomplétion des commandes
- exécution ligne par ligne

ISE peut être utile pour l'apprentissage, mais **ne doit plus être considéré comme un outil professionnel moderne**.

Visual Studio Code + extension PowerShell

Visual Studio Code est aujourd'hui **l'environnement recommandé** par Microsoft pour éditer des scripts PowerShell.

Il apporte :

- autocomplétion avancée
- analyse du code
- débogage

- intégration Git
- compatibilité Windows, Linux et macOS

C'est l'outil privilégié en entreprise, en automatisation avancée et en contexte DevOps.

NOTE: Le **terminal** de VSCode est en fait un terminal PowerShell.

Scripts PowerShell : vue d'ensemble

Un script PowerShell est un fichier texte portant l'extension `.ps1`. Il permet d'enchaîner des commandes, de structurer des actions et d'automatiser des tâches.

À ce stade, retiens simplement que :

- un script est un **fichier**
- un script PowerShell ne se lance jamais tout seul
- son exécution est soumise à des **règles de sécurité**

Pour qu'un script s'exécute, l'utilisateur doit exprimer clairement son intention, par exemple :

```
./script.ps1
```

La structure et l'écriture des scripts seront abordées dans un chapitre dédié.

Politique d'exécution des scripts

PowerShell applique une **politique d'exécution** destinée à protéger le système contre l'exécution de scripts malveillants.

Par défaut, Windows bloque l'exécution des scripts.

Consulter la politique actuelle

```
Get-ExecutionPolicy
```

Politiques principales à connaître

- **Restricted** Aucun script ne peut être exécuté (comportement par défaut).
- **RemoteSigned** Les scripts locaux peuvent être exécutés. Les scripts téléchargés doivent être signés.
- **Unrestricted** Tous les scripts peuvent être exécutés (fortement déconseillé).

Le mode **Undefined**, souvent rencontré, est équivalent à *Restricted*.

Politique recommandée pour l'apprentissage

```powershell
Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
```

Cette configuration :

- ne nécessite pas de droits administrateur
- ne modifie pas la sécurité globale du système
- permet de travailler sereinement en local

Exécution d'un script local et sécurité

PowerShell n'exécute jamais un fichier du dossier courant par défaut. Cela évite les exécutions accidentelles ou malveillantes.

Pour lancer un script situé dans le dossier courant, il faut préciser explicitement le chemin :

```powershell
./script.ps1
```

Le préfixe `./` signifie :

« exécute ce fichier depuis le dossier actuel »

C'est une règle de sécurité volontaire et fondamentale.

Conclusion

Dans ce chapitre, tu as compris :

- les différents environnements PowerShell (console, ISE, VS Code)
- le rôle général des scripts PowerShell
- pourquoi l'exécution des scripts est encadrée
- comment fonctionne la politique d'exécution
- pourquoi l'utilisation de `./` est obligatoire pour un script local

Ces notions constituent le **socle de sécurité et d'environnement** indispensable avant de passer à l'écriture de scripts et à l'automatisation proprement dite.
