

# Windows PowerShell

PowerShell s'est imposé comme l'outil central de l'**administration Windows** moderne. Pour des techniciens d'infrastructure, il constitue aujourd'hui un langage incontournable, à la fois simple pour les opérations courantes et suffisamment puissant pour automatiser à grande échelle, intégrer des pipelines DevOps, externaliser des configurations et interagir avec des API. Comprendre sa philosophie est indispensable pour maîtriser ce que sera le métier de technicien en informatique dans les années à venir.

🔥 PowerShell est un langage qui permet d'automatiser l'administration de Windows

## Evolution: du Batch à PowerShell

### L'approche historique : les fichiers Batch / CMD

Pendant plus de 30 ans, l'administration Windows s'est appuyée sur deux éléments essentiels :

- La console CMD (Command Prompt)
- Les scripts Batch (.bat / .cmd)

Ils permettent d'exécuter des commandes simples : copier des fichiers, lancer un programme, afficher un répertoire, écrire dans un fichier. Il s'agit d'une approche **séquentielle, texte-vers-texte**, conçue dans les années 80.

Forces :

- Très simple
- Présente partout
- Suffisant pour des automatisations basiques

Limites :

- Syntaxe vieillissante et peu expressive
- Pas d'accès structuré à des objets ou API modernes
- Maintenance difficile
- Peu adapté à l'administration de dizaines ou centaines de machines

### Le tournant PowerShell

Microsoft introduit PowerShell pour répondre aux besoins de l'administration moderne :

- piloter Windows Server
- automatiser Active Directory
- gérer des infrastructures distribuées
- interagir avec des API web
- produire des scripts maintenables et robustes

PowerShell devient rapidement l'outil recommandé par Microsoft pour toute opération avancée.

# Pourquoi PowerShell est devenu un standard

---

PowerShell combine un **langage moderne** et un environnement **pensé pour l'automatisation** et le DevOps. Quatre raisons clés expliquent son adoption massive.

## Un vrai langage de script moderne

PowerShell n'est pas une simple succession de commandes. Il propose :

- variables typées
- boucles et conditions avancées
- modules réutilisables
- gestion des erreurs (try/catch)
- paramètres et fonctions structurées

Cela en fait un langage à part entière, adapté aux grandes infrastructures.

## Une automatisation d'entreprise

PowerShell peut gérer :

- parc de postes
- serveurs Windows
- Active Directory
- Exchange / 365
- Hyper-V / VMware
- Azure
- tâches planifiées
- inventaires matériels et logiciels

Les **cmdlets** sont exhaustives et officielles, maintenues par Microsoft et les éditeurs tiers.

💡 Une **cmdlet** (prononcé *command-let*) est une **commande native PowerShell**, conçue pour exécuter une tâche précise dans l'environnement d'administration Windows ou multi-plateformes. Contrairement aux commandes classiques de CMD ou aux exécutables externes, une cmdlet est un composant .NET intégré au langage PowerShell, qui produit et manipule des objets, pas du texte.

## Une intégration naturelle dans le DevOps

PowerShell s'intègre parfaitement dans :

- scripts CI/CD
- pipelines de déploiement
- provisioning automatique
- Infrastructure as Code
- conteneurs (Base images PowerShell)
- gestion de configuration (DSC)

Il est aujourd'hui utilisé aussi bien par des administrateurs systèmes que par des ingénieurs DevOps.

## Cross-platform et open source

Depuis PowerShell Core, le projet est open source, disponible sur :

- Windows
- Linux
- macOS

L'écosystème est unifié, et les compétences acquises sont valables sur plusieurs OS.

---