PowerShell für Anfänger – Mehr erreichen in weniger Zeit

Du möchtest als Administrator mehr Geld verdienen? Dann solltest du dich mit der PowerShell beschäftigen. Mit dieser Kompetenz steigerst du einen Marktwert und kannst in weniger Zeit produktiver sein.

1. Kapitel: Einführung – Was ist PowerShell und warum solltest du sie lernen?

Was ist PowerShell?

PowerShell ist eine von Microsoft entwickelte Shell und Skriptsprache, die speziell für die Systemadministration und Automatisierung von Aufgaben in Windows-Umgebungen konzipiert wurde. Im Gegensatz zur klassischen Eingabeaufforderung (CMD) arbeitet PowerShell objektorientiert.

Das bedeutet: Statt einfachen Text zurückzugeben, liefert PowerShell strukturierte Objekte, mit denen du direkt weiterarbeiten kannst.

Du kannst PowerShell verwenden, um einzelne Befehle (Cmdlets) auszuführen oder vollständige Automatisierungsskripte zu schreiben – von der Benutzerverwaltung über Systemprüfungen bis hin zu Backup-Skripten.

Warum solltest du PowerShell lernen?

Als angehende IT-Fachkraft wirst du schnell feststellen, dass manuelle Konfigurationen zeitaufwändig und fehleranfällig sein können. PowerShell hilft dir dabei, wiederkehrende Aufgaben effizient und reproduzierbar zu automatisieren. Einige der wichtigsten Gründe, um PowerShell zu lernen sind u.a. folgende:

- **Automatisierung:** Spare Zeit, indem du Routineaufgaben wie das Erstellen von Benutzern oder das Überprüfen von Systemzuständen automatisierst.
- **Produktivität:** Führe komplexe Abläufe mit wenigen Zeilen Code durch.
- **Fehlervermeidung:** Reduziere manuelle Eingabefehler durch strukturierte Skripte.
- **Skalierbarkeit:** Verwende deine Skripte auf mehreren Systemen gleichzeitig – lokal oder remote.
- **Praxisrelevanz:** PowerShell-Kenntnisse sind bei Arbeitgebern im IT-Umfeld sehr gefragt.
- **Mehr Kompetenz** Mit dieser Fähigkeit machst du dich am Arbeitsmarkt attraktiver.

Die Rolle von PowerShell in modernen IT-Umgebungen

Mit der Weiterentwicklung von Windows und Cloud-Technologien hat sich PowerShell als zentrale Verwaltungsplattform etabliert – nicht nur lokal auf Windows-Systemen, sondern auch für Dienste wie Microsoft 365, Exchange Online, Azure oder Active Directory.

Im Bereich der Systemadministration von Windows-Betriebssystemen wie Windows Server, Windows 11 oder auch Exchange-Servern wirst du die PowerShell regelmäßig für folgende Aufgaben einsetzen:

- Verwaltung von Benutzerkonten und Gruppen
- Systemüberwachung und -wartung
- Softwareverteilung
- Netzwerkanalyse
- Zugriff auf APIs und Cloud-Services

PowerShell vs. CMD: Die wichtigsten Unterschiede

Merkmal	CMD (Eingabeaufforderung)	PowerShell
Datenverarbeitung	Textbasiert	Objektbasiert
Sprache	Batch	.NET-basierte Skriptsprache
Erweiterbarkeit	Eingeschränkt	Hoch (Module, Skripte, .NET-Integration)
Automatisierung	Grundlegend	Umfassend
Plattformen	Windows	Windows, Linux, macOS (mit PowerShell Core)

PowerShell oder PowerShell Core?

Standardmäßig arbeitest du unter Windows mit **Windows PowerShell** (Version 5.1). Daneben existiert **PowerShell Core** (ab Version 6), das plattformübergreifend (Windows, Linux, macOS) funktioniert und weiterentwickelt wird. Bei der Entstehung diese E-Books ist die aktuellste **`Windows PowerShell Version die 7.5.1 Stabile`**.

Für die Inhalte in diesem Buch verwenden wir **Windows PowerShell 5.1**, da sie in den meisten Unternehmensumgebungen weiterhin die Standardversion ist.

PowerShell ist ein mächtiges Werkzeug für jede IT-Fachkraft. Egal ob du im Helpdesk, in der Systemadministration oder im DevOps-Bereich arbeitest – mit PowerShell bist du besser vorbereitet, um IT-Aufgaben effizient, nachvollziehbar und skalierbar zu lösen.

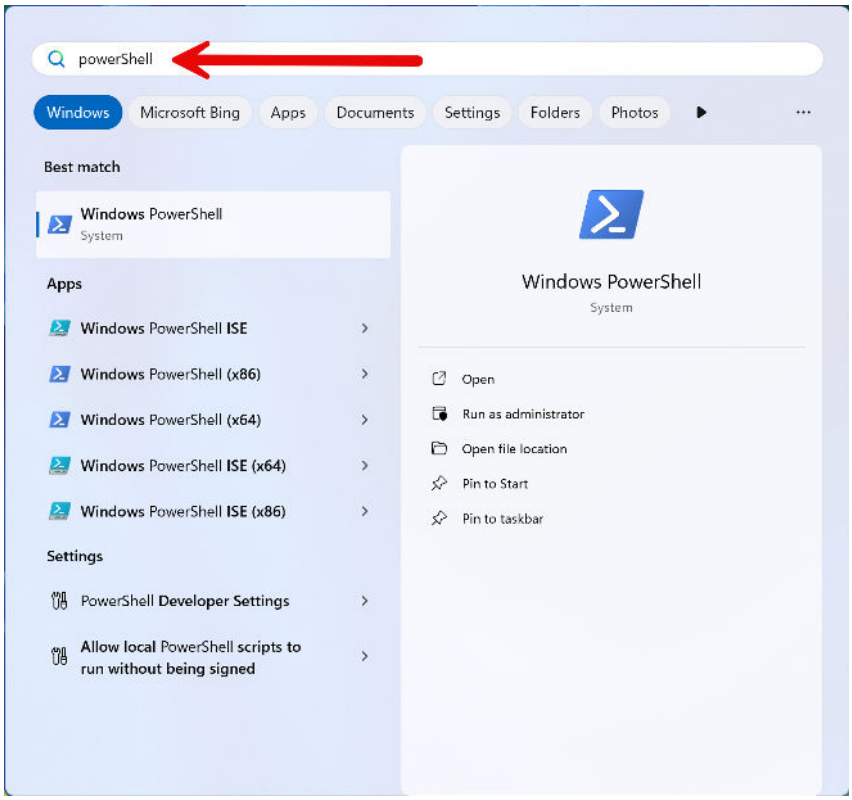
Im nächsten Kapitel lernst du, wie du PowerShell auf deinem System startest und erste einfache Befehle ausprobierst.

2. Kapitel: PowerShell starten und verstehen

PowerShell öffnen – So findest du den Einstieg

Um PowerShell zu nutzen, brauchst du keine zusätzliche Software – die Standardversion 5.1 ist bereits auf jedem modernen Windows-Betriebssystem vorinstalliert. Nur die PowerShell Version 7 musst du gesondert installieren.

Um die PowerShell zu starten, tippst du in das Windows Startmenü **PowerShell** ein. Jetzt erhältst du verschiedene Optionen:



PowerShell starten (Windows 10/11)

Startmenü öffnen

- Tippe „PowerShell“ ein.
- Klicke auf **„Windows PowerShell“** oder **„PowerShell (x86)“**.
- Für administrative Aufgaben: Rechtsklick → **„Als Administrator ausführen“**.

Tastenkombination

- Windows + R → Eingabe von powershell → Enter.
- Damit öffnet sich in neuen Systemen wie Windows 11 direkt auch das Windows Terminal

Windows Terminal

- Ab Windows 10/11 verfügbar.
- Unterstützt mehrere Shells (PowerShell, CMD, WSL).
- Vorteil: moderne Oberfläche mit Tabs.
- Um ein Terminal zu öffnen, musst du in das Windows Startmenü nur **Terminal** eingeben.

PowerShell ISE (Integrated Scripting Environment)

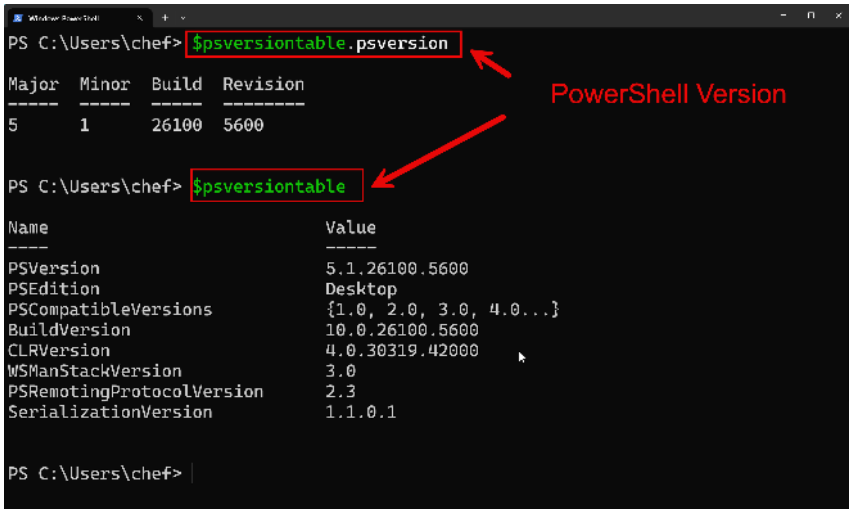
- Eine grafische Entwicklungsumgebung für PowerShell-Skripte.
- Ideal für Einsteiger, da Syntax-Highlighting, IntelliSense und Debugging enthalten sind.
- Starte sie über das Startmenü mit „PowerShell ISE“.
- Die PowerShell ISE gibt es sowohl in der x86, sowie auch in einer x64 Version.

💡 **Hinweis:**

Die PowerShell ISE wird nicht mehr aktiv weiterentwickelt, ist aber für Einsteiger nach wie vor gut geeignet. Für Fortgeschrittene empfiehlt sich später ***`Visual Studio Code`***.

Windows PowerShell Version anzeigen:

Um die installierte Version der Windows PowerShell anzuzeigen, öffnest du ein PowerShell-Terminal. Gib dort den Befehl **$PSVersionTable.PSVersion** ein.



Dieser Befehl zeigt dir die genaue Versionsnummer der PowerShell-Installation an. Die Ausgabe enthält Haupt-, Neben- und Build-Nummern, die dir Auskunft über die verwendete Version geben.

Sicherheitskonzept: Execution Policy

PowerShell ist ein äußerst mächtiges Automatisierungs- und Konfigurations-Framework von Microsoft, das es ermöglicht, komplexe Aufgaben effizient zu Skripten und auszuführen.
Diese Leistungsfähigkeit birgt jedoch auch Sicherheitsrisiken, da bösartige Skripte unautorisierten Zugriff auf Systeme und Daten verschaffen könnten.

Um diese Risiken zu minimieren, implementiert PowerShell die sogenannte "**Execution Policy**". Diese Sicherheitsfunktion legt fest, welche Skripte ausgeführt werden dürfen und welche nicht.
Standardmäßig ist PowerShell so konfiguriert, dass es die Ausführung von Skripten blockiert, die nicht explizit autorisiert wurden. Dies verhindert, dass unbeabsichtigt Schadsoftware oder unsichere Codes ausgeführt werden.

Es gibt verschiedene Stufen der Execution Policy, von strikten Einstellungen wie **Restricted** (nur interaktive Befehle, keine Skripte erlaubt) bis hin zu **Unrestricted** (alle Skripte können ausgeführt werden, jedoch mit Warnhinweisen).

Administratoren können diese Richtlinien je nach Sicherheitsanforderungen anpassen,

um ein ausgewogenes Verhältnis zwischen Flexibilität und Schutz zu gewährleisten. Dabei kann man sich verschiedenen Methodeni bedienen. So lässt sich z.B. die **PowerShell komplett über die Gruppenrichtlinien verbieten**.

Durch diese Mechanismen stellt PowerShell sicher, dass nur vertrauenswürdige Skripte ausgeführt werden, und hilft so, potenzielle Sicherheitslücken zu schließen.

Execution Policy	Beschreibung	Skriptausführung erlaubt?
Restricted	Standardwert – keine Skriptausführung erlaubt.	❌ Nein
AllSigned	Nur signierte Skripte von vertrauenswürdigen Herausgebern werden ausgeführt.	✅ Ja, aber nur signierte Skripte
RemoteSigned	Lokale Skripte dürfen ausgeführt werden, heruntergeladene müssen signiert sein.	✅ Ja, mit Einschränkungen
Unrestricted	Alle Skripte können ausgeführt werden, evtl. mit Warnhinweis bei Remote-Dateien.	✅ Ja
Bypass	Keine Einschränkungen und keine Warnungen oder Meldungen.	✅ Ja
Undefined	Keine Einstellung gesetzt, es gilt die Gruppenrichtlinie oder der Standardwert.	➖ Abhängig vom Kontext

Die Execution Policy anzeigen und bearbeiten

Wie bereits erläutert, legt die **Execution Policy** fest, ob und welche Skripte ausgeführt werden dürfen.

Welche Richtlinie aktuell aktiv ist, lässt sich wie folgt abfragen: