

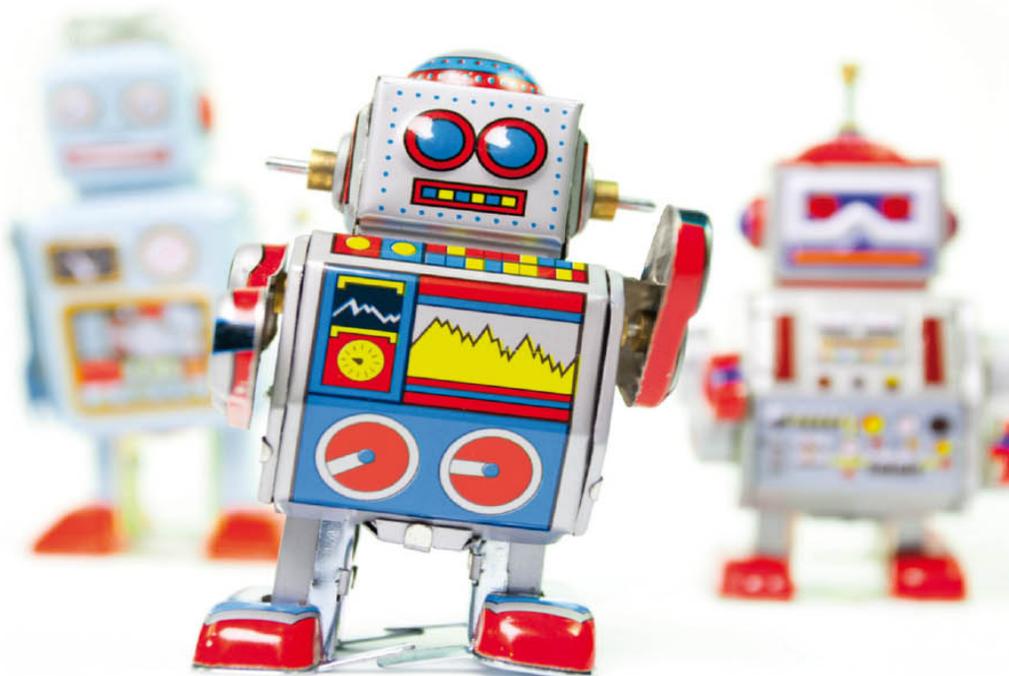
Mit  
vielen  
Beispielen  
aus der Praxis

# PowerShell 7 und Windows PowerShell

Das komplette Praxiswissen für  
Administratoren und IT-Profis  
Für Windows - Linux - macOS - Cloud



Dr. Tobias Weltner



**O'REILLY**<sup>®</sup>

Papier  
plus<sup>+</sup>  
PDF.

Zu diesem Buch – sowie zu vielen weiteren O’Reilly-Büchern – können Sie auch das entsprechende E-Book im PDF-Format herunterladen. Werden Sie dazu einfach Mitglied bei oreilly.plus<sup>+</sup>:

[www.oreilly.plus](http://www.oreilly.plus)

Dr. Tobias Weltner

# **PowerShell 7 und Windows PowerShell**

**Das komplette Praxiswissen für  
Administratoren und IT-Profis**

**O'REILLY®**

Dr. Tobias Weltner

Lektorat: Ariane Hesse

Lektoratsassistentz: Anja Weimer, Julia Griebel

Korrektorat: Sibylle Feldmann, [www.richtiger-text.de](http://www.richtiger-text.de)

Satz: Gerhard Alfes, mediaService, [www.mediaservice.tv](http://www.mediaservice.tv)

Herstellung: Stefanie Weidner

Umschlaggestaltung: Michael Oreal, [www.oreal.de](http://www.oreal.de), unter Verwendung eines  
Fotos von iStock by Getty Images von querbeet, Stock-  
Fotografie-ID184997148

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der  
Deutschen

Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über  
<http://dnb.d-nb.de> abrufbar.

ISBN:

Print 978-3-96009-163-9

PDF 978-3-96010-479-7

ePub 978-3-96010-480-3

mobi 978-3-96010-481-0

1. Auflage 2021

Copyright © 2021 dpunkt.verlag GmbH

Wieblinger Weg 17

69123 Heidelberg

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem  
Imprint »O'REILLY«. O'REILLY ist ein Markenzeichen und eine eingetragene  
Marke von O'Reilly Media, Inc. und wird mit Einwilligung des Eigentümers  
verwendet.



*Hinweis:*

Dieses Buch wurde auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie.

*Schreiben Sie uns:*

Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: [kommentar@oreilly.de](mailto:kommentar@oreilly.de).

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.

Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.

5 4 3 2 1 0

# Inhalt

## Vorwort

- Wie Sie dieses Buch nutzen
- Achtung
- Noch mehr Unterstützung

## 1 PowerShell: Erste Schritte

- PowerShell installieren

  - Windows-Betriebssystem
  - PowerShell nachrüsten: macOS und Linux
  - Kompatibilität der PowerShell

- PowerShell einrichten

  - Vorsicht mit Administratorrechten!
  - Interaktive Befehle eingeben
  - Autovervollständigung: Tippfehler vermeiden
  - Befehlszeilen erneut verwenden
  - Groß- und Kleinschreibung
  - Unvollständige und mehrzeilige Eingaben
  - PowerShell-Hilfe aus dem Internet nachladen
  - Skriptausführung erlauben
  - Weitere PowerShell-Einschränkungen

- Wichtige PowerShell-Werkzeuge

  - PowerShell-ISE-Editor
  - VSCoDe (Visual Studio Code)
  - Windows-Terminal

- Codebeispiele automatisch herunterladen

  - Befehl zum Herunterladen von Codebeispielen nachrüsten
  - Beispielcode in Zwischenablage kopieren

- Beispielcode sofort ausführen
- Profilskripte: PowerShell dauerhaft anpassen
  - Einzelne Profilskripte verwenden
  - Fragen stellen
  - Alle Skripte herunterladen
- Zusammenfassung
  - Ausführungsrichtlinie festlegen
  - Hilfe nachrüsten
  - Windows PowerShell aktualisieren
  - Hilfsbefehle zum Download von Beispielcode
  - »Fehlende Befehle« verstehen
  - Zusätzliche Werkzeuge

## **2 Überblick: Was PowerShell leistet**

- Befehle lösen Aufgaben

- Literale

  - Text

  - Zahlen

  - Datum und Zeit

  - Kommentare

- Cmdlets: die PowerShell-Befehle

  - Nach Tätigkeitsbereich suchen (»Noun«)

  - Nach Tätigkeit suchen (»Verb«)

  - Nach Herkunft suchen

  - Standardisierte Verben

  - Cmdlets per Fenster suchen

  - Syntax verstehen

  - Ausführliche Hilfe und Beispiele

- Anwendungsprogramme (Applications)

  - Applications starten

  - Applications finden

  - Systembefehle nutzen

  - Hilfe für Applications abrufen

- .NET-Methoden

  - Methoden verwenden

  - Hilfe für Methoden

- Noch mehr Methoden
- Methoden: vielseitiger, aber kleinteiliger
- Operatoren
  - Operatoren nachschlagen
  - Vergleichsoperatoren
  - Textoperatoren
  - Zuweisungen und Pipeline
  - Zahlenreihen
- Befehle verbinden
  - Das Türchen-Modell
  - Normalfall: »Türchen 3«
  - Modernes Pipeline-Streaming: »Türchen 1«
  - Klassische Variablen: »Türchen 2«
- Neue Befehle nachrüsten
  - Beispiel 1: QR-Codes generieren
  - Beispiel 2: Automatische HTML-Reports
  - Beispiel 3: Musik spielen
- Zusammenfassung

### **3 Skripte und Funktionen**

- PowerShell-Skripte verstehen
  - Skriptcode eingeben
- Skripte mit dem ISE-Editor entwickeln
  - Neues Skript anlegen
- Skripte mit dem VSCode-Editor entwickeln
  - Neue Skripte anlegen
- Skripte ausführen
  - Skripte innerhalb von PowerShell starten
  - Skripte außerhalb von PowerShell starten
- Skripte automatisch starten
  - Skriptstart durch Task Scheduler
  - Profilskripte - die Autostartskripte
  - Profilskript anlegen und öffnen
  - Typische Profilskriptaufgaben durchführen
- Neue Befehle: Funktionen
  - Schritt 1: Nützlicher Code

Schritt 2: Als Funktion verpacken  
Schritt 3: Parameter definieren  
Schritt 4: Funktionen dauerhaft verfügbar machen  
Funktionen im Modul permanent verfügbar  
Die gemeinsame Grundlage: der Skriptblock  
Skripte sind gespeicherte Skriptblöcke  
Funktionen sind vorgeladene Skriptblöcke  
Module laden Funktionen bei Bedarf in den Speicher  
Skript oder Funktion?

#### **4 Cmdlets - PowerShell-Befehle**

Parameter steuern Cmdlets

Argumente an Parameter übergeben  
Parameter machen Cmdlets vielseitig  
Politisch inkorrekt: positionale Argumente  
Gratiszugabe: »Common Parameters«  
Auf Fehler reagieren

Vielseitigkeit durch Parametersätze

»Schmal, aber tief« - Cmdlets sind Spezialisten  
Mehrere Parametersätze: noch mehr Vielseitigkeit  
Praxis: Ereignisse aus dem Ereignislogbuch lesen  
»ISA/HASA« - wie Cmdlets in der Pipeline funktionieren  
Das »ISA/HASA«-Prinzip  
Praxisnutzen  
Vorteile des Pipeline-Streamings

#### **5 Die PowerShell-Pipeline**

Aufbau der PowerShell-Pipeline

Befehle reichen Ergebnisse weiter  
Pipeline steuert Befehle  
Prinzipieller Aufbau der Pipeline  
Die sechs wichtigsten Pipeline-Befehle  
Select-Object  
Detailinformationen festlegen  
Unsichtbare Eigenschaften sichtbar machen

- Eine bestimmte Eigenschaft auswählen: -
  - ExpandProperty
  - Selbst festlegen, welche Informationen wichtig sind
  - Weitere Informationen anfügen
  - First, -Last und -Skip
  - Berechnete Eigenschaften
- Where-Object
  - Clientseitiger Universalfilter
  - Leere Elemente aussortieren
  - Fortgeschrittene Syntax bietet mehr Möglichkeiten
  - Out-GridView: das »menschliche« Where-Object
- Sort-Object
  - Cmdlet-Ergebnisse sortieren
  - Sortierung mit anderen Cmdlets kombinieren
  - Datentyp der Sortierung ändern
  - Mehrere Spalten in umgekehrter Sortierung
- Group-Object
  - Häufigkeiten feststellen
  - Daten gruppieren
  - Berechnete Gruppierungskriterien
- Measure-Object
  - Statistische Berechnungen
  - Ordnergrößen berechnen
- Foreach-Object
  - Grundprinzip: eine Schleife
- Format-Cmdlets
  - Gefährlich: Format-Cmdlets verändern Objekte
  - Mehrspaltige Anzeigen
  - Tabellenausgabe mit Gruppierung

## **6 Arrays und Hashtables**

- Arrays verwenden
  - Auf Array-Elemente zugreifen
- Eigene Arrays (und einige Fallen)
  - Automatische Array-Erzeugung
  - Manuelle Array-Erzeugung

Hashtables - »sprechende Arrays«  
Hashtables in Objekte umwandeln  
Neue Objekte mit Eigenschaften initialisieren

## **7 PowerShell-Laufwerke**

Dateisystemaufgaben meistern

Cmdlets für das Dateisystem finden

Erste Schritte

Ordner anlegen

Dateien anlegen und Informationen speichern

Encoding von Textdateien

Encodings sichtbar machen

Dateien finden

Dateien und Ordner kopieren

Dateien umbenennen

Dateien und Ordner löschen

Größe eines Laufwerks ermitteln

Größe eines Ordners ermitteln

Umgebungsvariablen

Alle Umgebungsvariablen auflisten

Auf einzelne Umgebungsvariablen zugreifen

Umgebungsvariablen ändern

Windows-Registrierungsdatenbank

Schlüssel suchen

Werte lesen

Neue Registry-Schlüssel anlegen

Registry-Schlüssel löschen

Werte hinzufügen, ändern und löschen

Virtuelle Laufwerke und Provider

Neue PowerShell-Laufwerke

Ohne Laufwerksbuchstaben direkt auf Provider zugreifen

-Path oder -LiteralPath?

Existenz eines Pfads prüfen

Pfadnamen auflösen

## **8 Operatoren und Bedingungen**

Operatoren – Aufbau und Namensgebung

Wie Operatornamen aufgebaut sind

Unäre Operatoren

Zuweisungsoperatoren

Vergleichsoperatoren

Ternary-Operator

Unterscheidung zwischen Groß- und Kleinschreibung

Unterschiedliche Datentypen vergleichen

Vergleiche umkehren

Vergleiche kombinieren

Vergleiche auf Arrays anwenden

Bedingungen

if-Bedingung

switch-Bedingung

Where-Object

Null-Koaleszenz-Operatoren

Pipeline-Verkettungsoperatoren

## **9 Textoperationen und reguläre Ausdrücke**

Texte zusammenfügen

Doppelte Anführungszeichen lösen Variablen auf

Der Formatierungsoperator -f

Array-Elemente in Text umwandeln

Textstellen finden und extrahieren

Texte splitten

Informationen in Texten finden

Reguläre Ausdrücke: Textmustererkennung

Erste Schritte: Textmuster basteln

Eigene reguläre Ausdrücke konzipieren

Textstellen ersetzen

Einfache Ersetzungen

Sichere Ersetzungen mit -replace

Mehrere Zeichen durch eins ersetzen

Split und Join: eine mächtige Strategie

Dateipfade ändern

X500-Pfade auslesen

## **10 Anwendungen und Konsolenbefehle**

Programme starten

Optionen für den Programmstart festlegen

Argumente an Anwendungen übergeben

Hilfe für Konsolenbefehle anzeigen

Beispiel: Lizenzstatus von Windows überprüfen

Ergebnisse von Anwendungen weiterverarbeiten

Error Level auswerten

Fragen an Benutzer stellen mit choice.exe

Rückgabertext auswerten

Laufende Programme steuern

Feststellen, ob ein Prozess läuft

Auf einen Prozess warten

Einstellungen laufender Prozesse ändern

Prozesse vorzeitig abbrechen

## **11 Typen verwenden**

Typumwandlungen

Geeignete Datentypen auswählen

Explizite Umwandlung

Deutsches Datumsformat mit dem Operator -as

Verkettete Umwandlungen

Typen: optimale Informationsbehälter

Implizite Umwandlung

Typisierte Variablen

Parameter und Argumente

Vergleichsoperationen

Statische Methoden eines Typs verwenden

Dateiextension ermitteln

Mathematische Funktionen

Zahlenformate konvertieren

DNS-Auflösung

Umgebungsvariablen

Pfade zu Systemordnern finden

- Konsoleneinstellungen
- Spezielle Datumsformate lesen
- Statische Eigenschaften verwenden
- Neue .NET-Typen finden
  - Type Accelerators untersuchen
- Typen nachladen
  - Assembly-Namen feststellen
  - Aktuell geladene Assemblies auflisten
  - Zusätzliche Assembly nachladen
  - Assembly aus Datei nachladen

## **12 Mit Objekten arbeiten**

- Objekte kennenlernen
  - Objekte funktionieren wie beschriftete Schubladen
  - Typisierte Variablen
  - Objekteigenschaften erforschen
- Eigenschaften lesen
- Eigenschaften ändern
- Methoden verwenden
- Zusammenfassende Systematik
  - Eigenschaften
  - Methoden
  - Hilfe finden
- Neue Objekte herstellen
  - Konstruktoren verstehen
  - Ein Credential-Object zur automatischen Anmeldung
  - Eigene Objekte erstellen

## **13 Eigene Typen und Attribute**

- Typen (Klassen) selbst herstellen
  - Einen eigenen Typ erfinden
  - Neue Objekte eines Typs generieren
  - Konstruktoren hinzufügen
  - Methoden zum Typ hinzufügen
- Vererbung von Typen
  - Konstruktoren werden nicht geerbt

- Philips Hue Smart Home mit Typen steuern
  - Kontakt zur Philips Hue Bridge herstellen
  - Lampen- und Schalterinventar
  - Lampen und Steckdosen ansprechen
- Attribute erstellen
  - SecureString-Transformationsattribute
- Auf API-Funktionen zugreifen
  - API-Funktion einsetzen
  - Wiederverwertbare PowerShell-Funktion herstellen

## **14 Parameter für Fortgeschrittene**

- Argumentvervollständigung
  - Statische Autovervollständigung
  - Autovervollständigung über Enumerationsdatentyp
  - Eigene Enumerationsdatentypen erstellen
  - Autovervollständigung über ValidateSet
  - Dynamische Argumentvervollständigung
- Zuweisungen mit Validierern überprüfen
  - ValidateSet
  - ValidateRange
  - ValidateLength
  - ValidatePattern
  - ValidateCount
  - ValidateScript
  - Nullwerte und andere Validierer
- Parameter in ParameterSets einteilen
  - Gegenseitig ausschließende Parameter
  - Binding über Datentyp
  - Parameter in mehreren Parametersätzen
- Simulationsmodus (-WhatIf) und Sicherheitsabfrage (-Confirm)
  - Festlegen, welche Codeteile übersprungen werden sollen
  - Weiterleitung verhindern
  - Gefährlichkeit einer Funktion festlegen
- Dynamische Parameter

- Dynamische Parameter selbst definieren
- Splatting: Argumente an Parameter binden
- Splatting im Alltag einsetzen
- Übergebene Parameter als Hashtable empfangen
- Mit Splatting Parameter weiterreichen

## **15 Pipeline-fähige Funktionen**

- Anonyme Pipeline-Funktion

  - Prototyping

  - Pipeline-fähige Funktion erstellen

  - Benannte Parameter

  - Where-Object durch eine Funktion ersetzen

  - Kurzes Resümee

- Parameter und Pipeline-Kontrakt

  - ISA-Kontrakt: Pipeline-Eingaben direkt binden

- HASA-Kontrakt: Objekteigenschaften lesen

  - HASA und ISA kombinieren

  - CSV-Dateien direkt an Funktionen übergeben

  - Aliasnamen für Parameter

## **16 PowerShellGet und Module**

- PowerShellGet – die PowerShell-Softwareverteilung

  - Grundlage »PackageManagement«

  - PowerShellGet – Softwareverteilung per Cmdlets

- Wo PowerShell Module lagert

  - Unterschiede zwischen Windows PowerShell und PowerShell

  - Installierte Module untersuchen

- Module mit PowerShellGet nachinstallieren

  - Module finden und installieren

  - Modul herunterladen

  - Modul testweise ausführen

  - Modul dauerhaft installieren

  - Module aktualisieren

  - Side-by-Side-Versionierung

  - Eigene Module veröffentlichen

- Eigene Module herstellen
  - Neue Manifestdatei anlegen
  - Neue Moduldatei anlegen
  - Modul testen
  - Nächste Schritte
- Eigene Module verteilen
  - Netzwerkfreigabe
  - Modul in Repository übertragen
  - Modul aus privatem Repository installieren

## **17 Fehlerhandling**

- Fehlermeldungen unterdrücken
  - Bestimmen, wie Cmdlets auf Fehler reagieren
  - Fehler mitprotokollieren lassen
  - Erfolg eines Befehlsaufrufs prüfen
- Fehlerhandler einsetzen
  - Lokaler Fehlerhandler: try...catch
  - Globaler Fehlerhandler: Trap

## **18 Windows PowerShell-Remoting**

- PowerShell-Remoting aktivieren
  - Zugriff auch auf Windows-Clients
  - Remoting-Verbindung überprüfen
  - NTLM-Authentifizierung erlauben
  - PowerShell-Remoting überprüfen
- Erste Schritte mit PowerShell-Remoting
- Befehle und Skripte remote ausführen
- Kontrolle: Wer besucht »meinen« Computer?
- Remotefähigen Code entwickeln
  - Argumente an Remote-Code übergeben
  - Ergebnisse vom Remote-Code an den Aufrufer übertragen
  - Fan-Out: integrierte Parallelverarbeitung
  - ThrottleLimit: Parallelverarbeitung begrenzen
  - Double-Hop und CredSSP: Anmeldeinfos weiterreichen

- Eigene Sitzungen verwenden
  - Eigene Sitzungen anlegen
  - Parallelverarbeitung mit PSSessions
- SSH-basiertes Remoting
  - SSH-Remoting aktivieren
  - Enter-PSSession über SSH
  - Invoke-Command über SSH
  - Unbeaufsichtigte Ausführung und Parallelbearbeitung

## **19 Grafische Oberflächen gestalten**

- Eigene Fenster herstellen
- GUI mit Code definieren
- GUI mit XAML definieren
- Beispiel: Dienststopper-Anwendung

## **Index**

# Vorwort

PowerShell begann vor 15 Jahren als Windows Powershell. Heute ist diese Shell plattformübergreifend auf Windows, Linux und macOS verfügbar und wird begeistert von einer immer größeren Anwendercommunity genutzt.

In diesem Buch werden Sie Schritt für Schritt erfahren, was PowerShell eigentlich ist und was diese Shell alles für Sie tun kann. Ob Sie Administrator »on-premise« sind und lokale Server betreuen, Ihr Unternehmen in der Cloud verwalten oder ob Sie heterogene Umgebungen »unter einen Hut« bringen wollen - PowerShell bietet die Werkzeuge dafür.

Aber auch zu Hause und in der Schule lässt es sich hervorragend einsetzen: PowerShell ist es nämlich egal, ob Sie damit eine Unternehmens-IT administrieren, Ihre Philips-Hue-Homeautomation verwalten oder Sonnenkollektoren steuern. Ob Sie den eigenen NAS-Server auf dem Dachboden sichern, automatisiert Dateien aus dem Internet laden oder vielleicht bloß Matherätsel knacken wollen.

Und genau deshalb ist PowerShell auch in Ausbildung und Schulen spannend: Kaum eine andere kostenlose Programmierumgebung und Shell unterstützt so viele

Programmierkonzepte auf so vielen Plattformen – angefangen von Befehlsaufrufen über moderne Programmier Techniken wie den Einsatz von Funktionen und Objekten bis hin zu nativer Klassenunterstützung, Vererbung und universellem Netzwerk-Remoting. Kaum eine andere Umgebung bietet so unmittelbares Feedback wie PowerShell. Ideal also, um im Unterricht und in der Ausbildung anhand von nachvollziehbaren Praxisbeispielen die Funktionsweisen moderner IT auszuprobieren und zu vertiefen.

Dieses Buch bietet zahlreiche Praxisbeispiele aus den unterschiedlichsten Einsatzbereichen und demonstriert schrittweise die vielfältigen Möglichkeiten der PowerShell.

So schlüpfen Sie zu Anfang des Buches in die Rolle des einfachen PowerShell-Anwenders. Sie generieren mit Einzeilern Excel- und HTML-Reports und können mit wenigen Schritten Cloud-Systeme steuern oder Notenblätter aus dem Internet herunterladen und als PDF exportieren.

Von diesen ersten Fingerübungen inspiriert lernen Sie den Minimal-Wortschatz der PowerShell kennen und fügen mehrere Befehle zu eigenen größeren Automationslösungen zusammen.

Die Anwendungsbeispiele werden immer komplexer, und Sie lernen schrittweise alle weiteren wichtigen Konzepte der PowerShell kennen. Dazu gehören Remotezugriffe, direkte Betriebssystemaufrufe, das Erstellen grafischer Oberflächen sowie eigene PowerShell-Befehle und -Module.

Am Ende dieses Buches beherrschen Sie dann eine der modernsten Automationssprachen, die für Windows, Linux und macOS kostenfrei zur Verfügung stehen, und haben

sich privat wie beruflich vom einfachen Anwender zum versierten IT-Automatisierer qualifiziert.

Damit Sie auf dieser Reise nicht allzuviel eintippen müssen, automatisiert PowerShell in diesem Buch auf Wunsch natürlich auch das Eintippen der Beispiele für Sie. Schon im ersten Kapitel lernen Sie den passenden Befehl kennen: durch Eingabe der jeweiligen Listingnummer fügt PowerShell den Quellcode aus dem Buch automatisch ein.

## **Wie Sie dieses Buch nutzen**

Dieses Buch setzt keinerlei Grundkenntnisse voraus, wenn Sie von vorn zu lesen beginnen - und das ist auch empfehlenswert. Die Kapitel bauen aufeinander auf. Am Anfang jedes Kapitels finden Sie eine kurze Zusammenfassung, falls es einmal eilig ist.

Die PowerShell-Beispiele im Buch sind jeweils in einer anderen Schriftart formatiert. Damit Sie leichter erkennen, welche Eingaben von Ihnen erwartet werden, wird bei allen Eingaben die PowerShell-Eingabeaufforderung `PS>` (einschließlich der Leerstelle hinter dem `>`) vorangestellt. Diese Eingabeaufforderung kann bei Ihnen auch anders aussehen und sollte in den Beispielen natürlich nicht mit eingegeben werden.

## **Achtung**

Bitte verwenden Sie die Begleitmaterialien immer im Kontext des entsprechenden Buchkapitels. Viele der Beispiele funktionieren nur, wenn Sie die entsprechenden Vorarbeiten im Kapitel beachtet haben, oder können auch unerwartete Resultate liefern, wenn man die Beispiele aus dem Zusammenhang des Kapitels reißt.

## **Noch mehr Unterstützung**

Falls trotz aller Sorgfalt einmal Fragen offenbleiben oder Sie weitere Ideen und Themenwünsche haben, besuchen Sie einfach das interaktive Leserforum zu diesem Buch:

[\*https://github.com/TobiasPSP/OReilly/discussions\*](https://github.com/TobiasPSP/OReilly/discussions)

Damit bleibt mir an dieser Stelle nur noch, Ihnen viel Spaß zu wünschen bei der Lektüre dieses Buchs und bei der Arbeit mit der faszinierenden PowerShell! Ich würde mich freuen, von Ihnen im Diskussionsforum zu hören.

Herzlichst,

Tobias Weltner

# Kapitel 1

## PowerShell: Erste Schritte

**In diesem Kapitel:**

**PowerShell installieren**

**PowerShell einrichten**

**Wichtige PowerShell-Werkzeuge**

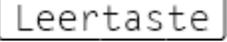
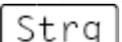
**Codebeispiele automatisch herunterladen**

**Profilskripte: PowerShell dauerhaft anpassen**

**Zusammenfassung**

**Ausführlich werden in diesem Kapitel die folgenden Aspekte erläutert:**

- **Windows PowerShell:** In Windows integrierte PowerShell, die auf dem klassischen *.NET Framework Version 4.5* oder höher basiert und auch künftig für Automationsaufgaben im Windows-Umfeld eingesetzt werden kann. Der Startbefehl lautet `powershell.exe`, und die aktuelle Version ist 5.1.

- **PowerShell:** Neuartige, plattformunabhängige PowerShell, die als portable Anwendung bei Windows parallel zur *Windows PowerShell* verwendet werden kann und auch auf Linux, macOS und weiteren Betriebssystemen zur Verfügung steht. Der Startbefehl lautet `pwsh.exe`. Diese PowerShell beruht auf dem plattformunabhängigen neuen *.NET Framework Core*, das weitgehend kompatibel zum klassischen *.NET Framework* ist. Die *PowerShell* ist im Gegensatz zur *Windows PowerShell* quelloffen (Open Source).
- **Autovervollständigung:** Mit einigen Tastendrücken kann man sich bei der Eingabe von Befehlen, Parametern und Argumenten Tipparbeit sparen:  vervollständigt Eingaben. Drücken Sie die Taste mehrmals, zeigt PowerShell bei jedem Drücken einen weiteren Vorschlag.  +  geht in der Reihenfolge wieder einen Schritt zurück. Mit  +  werden Eingabevorschläge als vollständige Auswahlliste präsentiert. Editoren zeigen dazu ein IntelliSense-Menü an. In der Konsole erscheint ein textbasiertes Auswahlfeld. Unabhängig von den Autovervollständigungsvarianten können Sie mit  und  frühere Eingaben aus Ihrer Befehlshistorie anzeigen lassen. Diese Liste ist anfangs leer und wächst mit der Verwendung der Konsole.
- **Zeilen löschen und Befehlsabbruch:** Um die gesamte aktuelle Zeile zu löschen, drücken Sie . Um einen Befehl abzubrechen, drücken Sie  + .
- **Groß- und Kleinschreibung:** PowerShell selbst unterscheidet bei Befehlsnamen und Parametern nicht zwischen Groß- und Kleinschreibung. Ob die

Groß- und Kleinschreibung bei Befehlsargumenten (wie zum Beispiel Pfadnamen oder anderen Angaben) wichtig ist, hängt vom jeweiligen Befehl und dem verwendeten Betriebssystem ab. Bei Kennworteingaben beispielsweise kommt es natürlich immer auf die richtige Groß- und Kleinschreibung an.

---

## **Achtung**

Wenn Sie es eilig haben und die Grundlagen der PowerShell schon kennen, dürfen Sie diesen Einleitungsteil selbstverständlich überspringen. Sie sollten aber in jedem Fall wenigstens den Abschnitt »[Zusammenfassung](#)« am Ende dieses Kapitels beachten. Dort werden wichtige Grundeinstellungen besprochen, die die Voraussetzung für viele Beispiele in den folgenden Kapiteln sind.

---

PowerShell ist eine verblüffend flexible und machtvolle plattformunabhängige Automationsprache, die mit geringem Aufwand ein enormes Spektrum von Aufgaben automatisieren kann.

Dazu zählen typische IT-Administrationsaufgaben ebenso wie völlig andere Einsatzbereiche aus Mathematik, Forschung und Lehre, in der Büroautomation und nicht zuletzt in Hobby und Tüftelei: Schon im nächsten Kapitel werden wir uns kurz der Musikkomposition und Steuerung von MIDI-Musikinstrumenten widmen, und in *Kapitel 13* erfahren Sie zum Beispiel, wie PowerShell sogar Lampen und Steckdosen in Ihrem Zuhause fernsteuert. Die Grundlagen und Strategien sind dabei indes immer dieselben.

Bei all diesen Beispielen geht es also ausschließlich um zweierlei: kurzweilig und verständlich möglichst viele

Einsatzszenarien der PowerShell vorzustellen, um Ideen zu wecken und eine breite Leserschicht anzusprechen, und die stets gleichen allgemeinen Mechanismen zu verstehen, die dabei zum Einsatz kommen.

Denn mit dem Wissen, das Sie beispielsweise in der Musikkomposition oder auch bei der Fernsteuerung von Elektrogeräten benötigen, können Sie natürlich auch Server aufsetzen oder Drittanbietersoftware für Backup-Lösungen steuern - und umgekehrt. Hier wird deutlich, dass PowerShell ebenfalls eine ideale Plattform für Ausbildung und Schule ist, denn alle modernen IT-Grundlagen lassen sich damit anschaulich und unterhaltsam vermitteln.

Zunächst aber muss PowerShell vollständig eingerichtet werden, und genau darum geht es in diesem ersten Kapitel. Dabei werden auch einige wichtige Sicherheitseinstellungen besprochen, und der Unterschied zwischen *PowerShell* und *Windows PowerShell* wird Ihnen schnell klar werden.

Danach folgt ein kleiner Exkurs zu den wichtigsten kostenlosen PowerShell-Tools, mit denen Sie Ihren Computer in eine moderne PowerShell-Entwicklungsumgebung verwandeln.

Zum Abschluss des Kapitels erfahren Sie, wie die vielen Hundert Skriptbeispiele in diesem Buch ohne Tipparbeit direkt von PowerShell aus dem Internet heruntergeladen und ausgeführt werden und wie offengebliebene Fragen unmittelbar an den Autor gerichtet werden können.

---

## **Hinweis**

In diesem Buch wird das Wort »PowerShell« immer dann kursiv gesetzt, wenn spezifisch die neue

plattformübergreifende *PowerShell* gemeint ist. Der nicht kursiv gesetzte Begriff »PowerShell« bezieht sich allgemein auf die Sprache PowerShell und gilt sowohl für *Windows PowerShell* als auch für *PowerShell*.

Schon in diesem Kapitel wird Ihnen bereits PowerShell-Code begegnen, der aber hier als reines »Werkzeug« für die Einrichtung Ihres Computers dient. Sie können den Code natürlich bereits neugierig mustern, aber bitte brüten Sie nicht zu lange über einzelnen Codebeispielen. Es kommt in diesem Kapitel nur auf die Ergebnisse der Codebeispiele an, nicht auf den Code selbst und seine Mechanismen. Diese stehen ab dem zweiten Kapitel im Vordergrund und werden dort dann auch systematisch besprochen.

---

## **PowerShell installieren**

PowerShell ist plattformunabhängig und kann auf verschiedenen Betriebssystemen ausgeführt werden, zum Beispiel Windows, Linux oder macOS.

Historisch ist PowerShell allerdings als Teil des Windows-Betriebssystems entstanden und dort als sogenannte *Windows PowerShell* immer automatisch enthalten. Seit 2016 ist diese Fassung in Version 5.1 fertiggestellt, wird auf diesem Stand gehalten und weiter von Microsoft gewartet.

Alle Entwicklungsarbeit geht seit 2016 in die neue, plattformunabhängige *PowerShell*, bei Drucklegung dieses Buchs in Version 7.2, die bei allen Betriebssystemen zuerst nachinstalliert werden muss. Wie das geschieht, erfahren Sie jetzt, und zwar in separaten Abschnitten für die jeweiligen Betriebssysteme.

## Windows-Betriebssystem

In allen Windows-Betriebssystemen ab Server 2008 und Windows 7 ist die PowerShell bereits fester Bestandteil – genauer gesagt die *Windows PowerShell*. Hier könnten Sie also sofort loslegen, sollten aber wenigstens prüfen, ob Ihre *Windows PowerShell* auf dem aktuellen Stand ist (dazu gleich mehr).

Noch empfehlenswerter ist es, zusätzlich zur integrierten *Windows PowerShell* auch die neue *PowerShell* hinzuzustallieren. Beide laufen friedlich parallel nebeneinander, und so können Sie sich ein eigenes Bild von den Unterschieden machen.

### Immer vorhanden: Windows PowerShell

Sie starten die *Windows PowerShell* mit dem Befehl `powershell.exe` (oder kurz `powershell`) – wie jedes andere Programm auch. Drücken Sie zum Beispiel  + , um das Dialogfeld *Ausführen* zu öffnen (siehe [Abbildung 1.1](#)), und geben den Befehl `powershell` ein. Dann klicken Sie auf *OK*.

In der Taskleiste erscheint ein blaues Symbol, das Sie per Rechtsklick und *An Taskleiste anheften* am besten dauerhaft dort anpinnen. Ein Rechtsklick auf das Symbol öffnet die sogenannte Sprungliste, sozusagen das Cockpit der *Windows PowerShell* (siehe [Abbildung 1.2](#)).

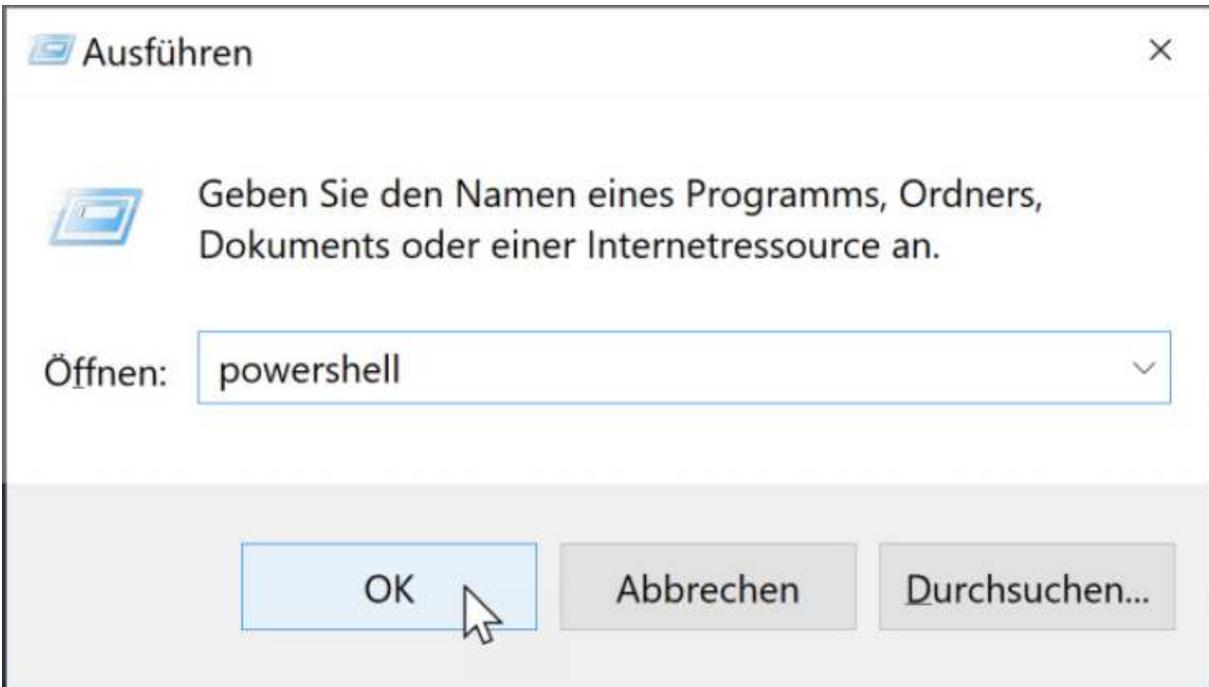


Abbildung 1.1: Windows PowerShell starten

Von hier aus können Sie die PowerShell zum Beispiel mit vollen Administratorrechten starten, und auch der in *Windows PowerShell* integrierte Editor *ISE* (*Integrated Script Environment*) steht hier bereit.

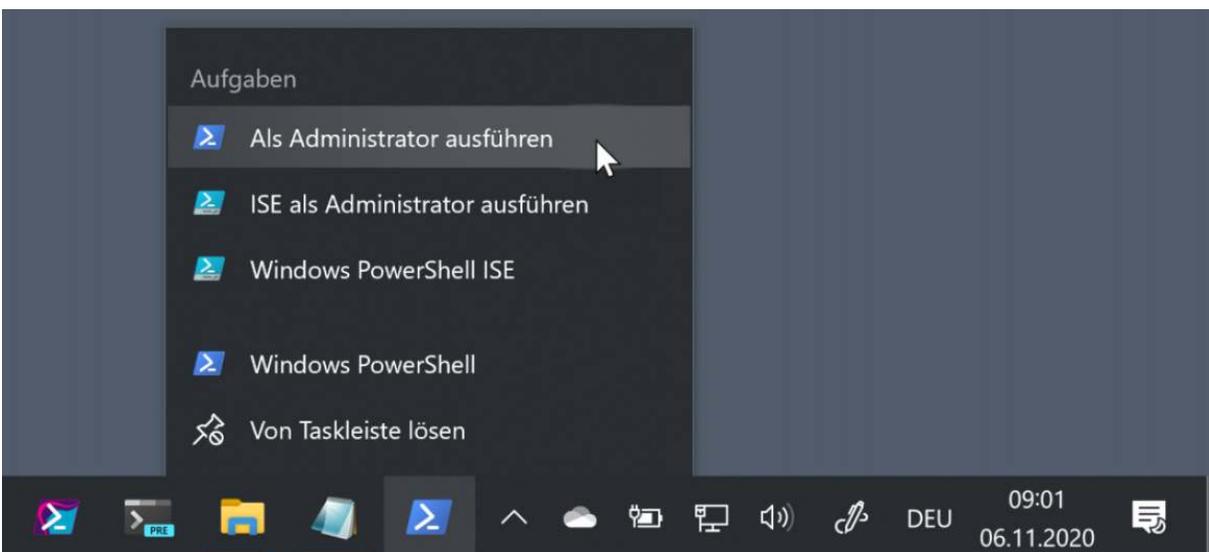
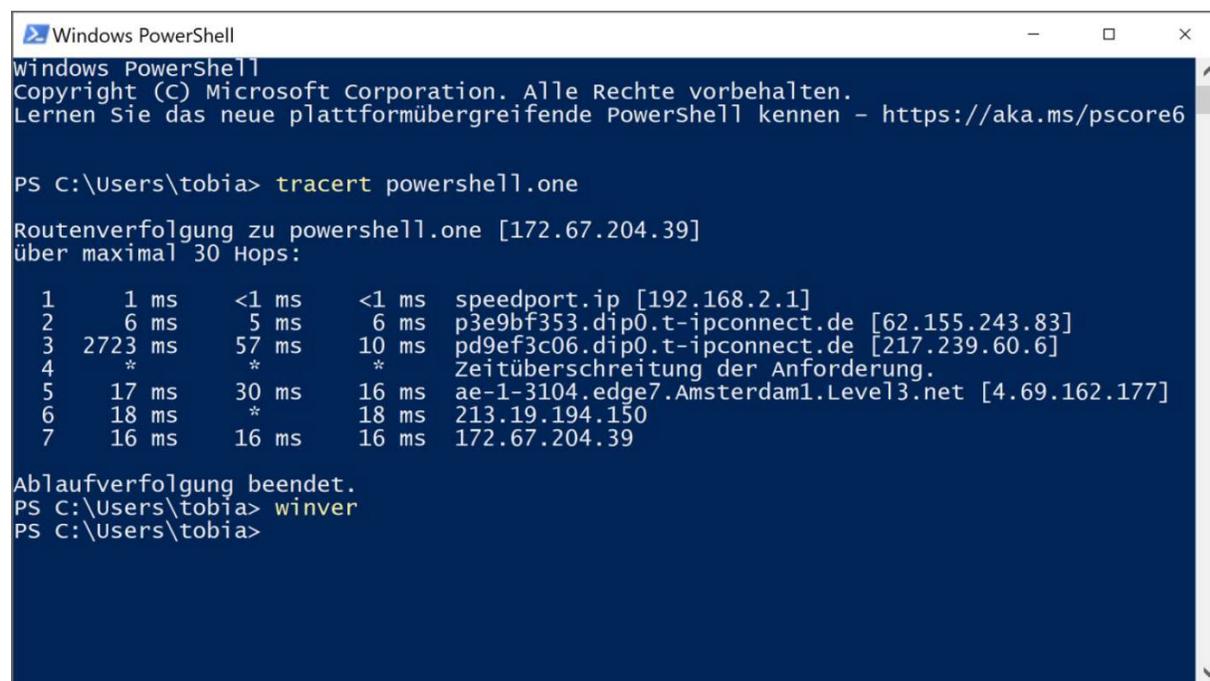


Abbildung 1.2: Sprungliste der Windows PowerShell

Die *Windows PowerShell* selbst erscheint als blaue oder schwarze Konsole, in die Sie Befehle eingeben können (siehe [Abbildung 1.3](#)). Auch wenn es Ihnen bereits in den Fingern juckt: Verschieben Sie Ihre Experimente bitte noch einen Moment – was Sie alles mit PowerShell tun können, veranschaulicht der Rundkurs im nächsten Kapitel.

Lassen Sie uns zunächst alles einsatzbereit machen.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Alle Rechte vorbehalten.
Lernen Sie das neue plattformübergreifende PowerShell kennen – https://aka.ms/pscore6

PS C:\Users\tobia> tracert powershell.one

Routenverfolgung zu powershell.one [172.67.204.39]
über maximal 30 Hops:

 1    1 ms    <1 ms    <1 ms    speedport.ip [192.168.2.1]
 2    6 ms     5 ms     6 ms    p3e9bf353.dip0.t-ipconnect.de [62.155.243.83]
 3 2723 ms   57 ms   10 ms   pd9ef3c06.dip0.t-ipconnect.de [217.239.60.6]
 4      *      *      *      Zeitüberschreitung der Anforderung.
 5   17 ms   30 ms   16 ms   ae-1-3104.edge7.Amsterdam1.Level3.net [4.69.162.177]
 6   18 ms   *      18 ms   213.19.194.150
 7   16 ms   16 ms   16 ms   172.67.204.39

Ablaufverfolgung beendet.
PS C:\Users\tobia> winver
PS C:\Users\tobia>
```

Abbildung 1.3: Die in Windows integrierte Windows PowerShell

### **Ist Ihre Windows PowerShell noch aktuell?**

Falls Sie Windows 10 oder Server 2016 (und besser) verwenden, ist alles gut: Diese Betriebssysteme enthalten *Windows PowerShell* in Version 5.1, der letzten und ausgereiften Version. Diese Version wird über die gesamte Laufzeit von Windows 10 weitergepflegt und automatisch aktualisiert.

Wer ein älteres Windows-Betriebssystem nutzt, muss *Windows PowerShell* manuell aktualisieren. Das ist sinnvoll, weil die Codebeispiele dieses Buchs die