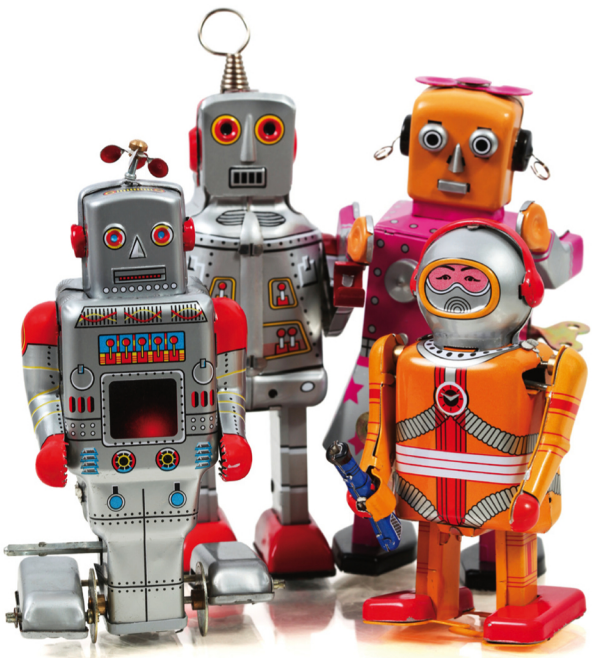
Das Standard-werk

Für PowerShell 2-5

# PowerShell 5

## Windows-Automation für Einsteiger und Profis

Dr. Tobias Weltner



O'REILLY®

# PowerShell 5 – Windows-Automation für Einsteiger und Profis

Dr. Tobias Weltner

# PowerShell 5 – Windows-Automation für Einsteiger und Profis

O'REILLY®

Dr. Tobias Weltner

Lektorat: Ariane Hesse
Korrektorat: Sibylle Feldmann
Satz: mediaService, www.mediaservice.tv
Herstellung: Susanne Bröckelmann
Umschlaggestaltung: Michael Oreal, www.oreal.de, unter Verwendung eines Fotos von Valerie Loiseleux / iStock. by Getty Images
Druck und Bindung: Druckerei C.H. Beck, www.becksche.de

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 978-3-96009-009-0
PDF 978-3-96010-033-1
ePub 978-3-96010-034-8
mobi 978-3-96010-035-5

1. Auflage 2016 Print 978-3-96009-009-0
1. korrigierter Nachdruck 2017
2. korrigierter Nachdruck 2018

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.



5 4 3 2

# Inhalt

# Einführung

## Wer braucht eigentlich PowerShell?

Jeder, der eine Aufgabe zweimal tut, sollte über Automation nachdenken. In der IT müssen viele Aufgaben nicht nur zweimal, sondern hundert- oder tausendfach erledigt werden. Und jeder kennt die wenig geliebten und immer wiederkehrenden Aufgaben: Monatliche Reports müssen erstellt oder Logbuchdaten durchforstet werden.

Was immer es ist und ganz gleich, ob beruflich oder privat motiviert: PowerShell ist die Sprache, mit der Sie solche Aufgaben beschreiben und danach so oft Sie mögen – und wann Sie mögen – von PowerShell ausführen lassen können. Das ist nicht nur bequemer, es kann auch im Not- oder Krisenfall wichtig sein. Wer dann die entsprechenden Notfallskripte zur Hand hat, lässt die in ruhigen Zeiten erarbeiteten Aufgaben blitzschnell abarbeiten.

Während Sie kleinere Aufgaben auf Ihrem eigenen Computer mit PowerShell automatisieren, ist diese Automationssprache auch für flächendeckende und unternehmensweite Automation hervorragend geeignet. Es gibt hier kaum Grenzen, und die PowerShell-Automation kann ganze Unternehmensstrukturen abbilden.

Über das sogenannte »PowerShell Remoting« (Kapitel 23) kommunizieren PowerShells über das Netzwerk miteinander, und dank PowerShell 6 steht PowerShell auch in heterogenen Landschaften zu Beispiel auf Linux und MacOS zur Verfügung.

Was die Frage aufwirft: Warum handelt dieses Buch eigentlich von PowerShell 5, wenn es offenbar auch ein PowerShell 6 gibt?

## »Windows PowerShell« 5 vs. »PowerShell« 6

Seit der ersten »Windows PowerShell«-Version in 2006 ist inzwischen »Windows PowerShell« 5 auf allen Windows-Systemen ab Windows 7 und Server 2008R2 der empfohlene Standard. Sie müssen allenfalls noch Ihre »Windows PowerShell«-Version auf die aktuellste Version bringen, wobei Ihnen Kapitel 1 zur Seite steht. »Windows PowerShell« ist nun »fertiggestellt« und wird sich auch nicht mehr wesentlich ändern.

Stattdessen arbeitet Microsoft jetzt an »PowerShell« 6 weiter.

Huh? Ist es also doch nicht fertig? Die Wahrheit ist: »PowerShell« 6 ist nicht der Nachfolger von »Windows PowerShell« 5. Lassen Sie uns also kurz ein wenig Klarheit in die PowerShell-Versionen bringen, damit Sie wissen, welche PowerShell für Sie richtig ist.

## »Windows PowerShell« …

Die fest in Windows integrierte PowerShell nennt Microsoft »Windows PowerShell«. Sie wird auch künftig in Windows integriert sein und basiert auf dem vollständigen .NET Framework, also den vollständigen Programmierschnittstellen des Windows-Betriebssystems. »Windows PowerShell« kann deshalb alle Aspekte und Funktionen von Windows verwalten und automatisieren.

Diese »Windows PowerShell« gilt nun in Version 5 als ausgereift und fertiggestellt. Sie haben auf Jahre Planungssicherheit.

## … und »PowerShell«

»PowerShell« 6 ist eine Neuentwicklung, die auf einem eingeschränkten und portablen .NET Framework basiert (dem sogenannten .NET Core). Es ist auch auf Linux und MacOS lauffähig und steht ausschließlich als optionaler Download von *https://github.com/PowerShell/PowerShell* zur Verfügung.

Der Vorzug von PowerShell 6 ist nicht etwa ein noch größerer Funktionsumfang. Wegen des eingeschränkten .NET Framework ist eher das Gegenteil der Fall. Der Vorzug von PowerShell 6 ist seine universelle Einsetzbarkeit. Es ist nicht mehr auf Windows beschränkt.

Zwar ist es technisch möglich, aber derzeit nicht besonders sinnvoll, auf Windows-Systemen die »PowerShell« 6 einzusetzen. Greifen Sie besser zur eingebauten »Windows PowerShell«. Sie kann deutlich mehr. Nur wenn Sie Nicht-Windows-Systeme in Ihre Automationspläne integrieren möchten, verwenden Sie dort »PowerShell« 6.

Über das »PowerShell Remoting« aus Kapitel 23 können »Windows PowerShell« und »PowerShell« 6 quer über das Netzwerk miteinander kommunizieren.

## Warum PowerShell lernen?

PowerShell-Analphabetismus konnte man als Windows-Administrator in den ersten Jahren noch gut kaschieren, doch inzwischen lässt sich dieses Defizit kaum noch verbergen. Zu allgegenwärtig ist PowerShell im Betriebssystem und in vielen Softwareprodukten, und wer es nicht beherrscht, kann häufig schlicht einen Teil seines Jobs nicht mehr erledigen.

Umgekehrt wird ein angenehmerer Schuh daraus: Wer sich bislang erfolgreich vor PowerShell »gedrückt« hat, kann sich mit einigen Wochen des Lernens (natürlich verbunden mit den üblichen Frustrationen, die zu jedem Erlernen dazugehören) das gesamte Funktionsspektrum der aktuellsten Windows-PowerShell-Version 5 zu eigen machen. Seit es PowerShell 6 auf Linux und MacOS gibt, können Sie Ihr PowerShell-Wissen sogar ohne großen Aufwand auch auf andere Betriebssysteme übertragen, und wer PowerShell beherrscht, wird für sein Unternehmen ein immer wertvollerer Mitarbeiter.

## Moderne Lernkurve

Wer vor der PowerShell-Konsole sitzt, sieht vor allem Befehle – die *Cmdlets*. Sie funktionieren etwa so wie Befehle in anderen Konsolen (*Shells*). Allerdings unterstützt PowerShell so viele davon und ist auf so gleichartige Weise erweiterbar, dass man mit ganz geringen (und vor allen Dingen vollkommen einheitlichen und konsistenten) Grundkenntnissen fast alles damit administrieren kann. Weil alle Cmdlets auf denselben Grundregeln basieren, kann man das Wissen auch leicht auf andere Aufgabenbereiche und Cmdlets anwenden.

Die ersten Schritte mit Cmdlets sind einfach und benötigen kaum PowerShell-Kenntnisse. Auch der Umgang mit der PowerShell-Pipeline ist zunächst recht einfach. Dann aber steht man vor einer Steilwand, die autodidaktisch oft nur schwer zu meistern ist. Hier werden viele wichtige Grundlagen gelegt. Man beschäftigt sich mit Operatoren, Objekten und der wahren Natur von PowerShell. Diese Steilwand meistern Sie mit diesem Buch.

Sobald die Grundlagen einmal gelegt sind, flacht die Lernkurve stark ab, und alles wird gut: Ob Sie mit Microsoft Exchange ein Postfach anlegen, mit SharePoint eine Site veröffentlichen oder einfach nur Protokolldateien parsen oder Server verwalten wollen – Ihr PowerShell-Wissen befähigt Sie dazu, über den Tellerrand zu blicken und Ihre PowerShell-Erfahrung auch in ganz anderen IT-Bereichen einzusetzen. Entscheidend ist nun nur noch Ihr Fachwissen im jeweiligen Bereich.

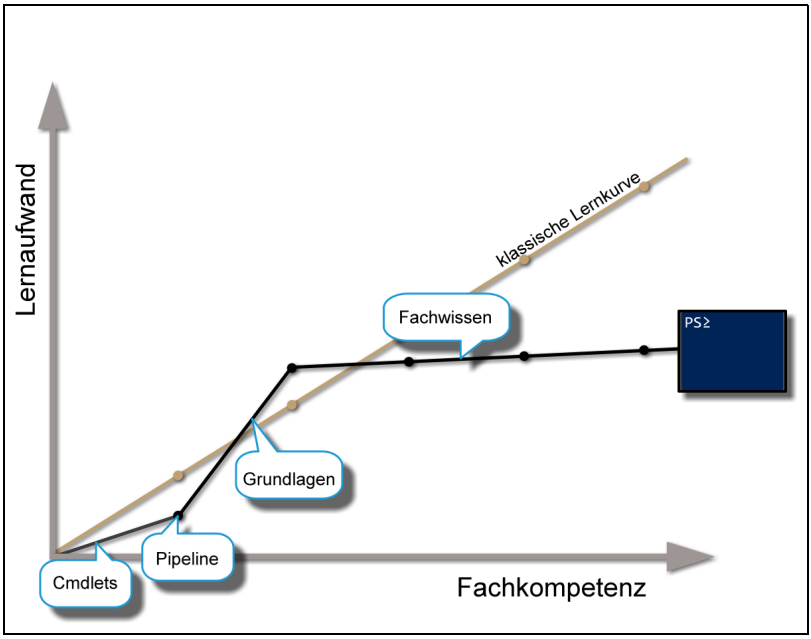



*Abbildung E.1: Die PowerShell-Lernkurve ist nicht linear, sondern baut auf gemeinsamen Standards auf.*

Sie müssen zwar schon selbst wissen, »warum« Sie ein Exchange-Postfach anlegen oder eine SharePoint-Site umbenennen wollen – »wie« das geschieht, ist aber dank der gemeinsamen PowerShell-Standards nun keine Frage mehr. `Get-ADUser`, `Get-Mailbox`, `Get-SPSite` und `Get-VM` beschaffen Ihnen mit denselben PowerShell-Grundkenntnissen Active-Directory-Benutzerkonten, Exchange-Server-Mailboxen, SharePoint-Sites oder VMware virtuelle Maschinen. Kennen Sie erst einmal ein Cmdlet, dann kennen Sie alle.

## Computer – ich/ich – Computer: PowerShell als Fremdsprache

Entfernt man sich ein wenig von PowerShell und betrachtet die Technik mit mehr Abstand, verschwimmen die einzelnen Befehle zu etwas Neuem: einer Sprache. Tatsächlich benimmt sich PowerShell in vielen Aspekten genau wie eine Fremdsprache, wie Spanisch oder Italienisch etwa, nur dass Sie sich diesmal nicht mit dem Eisverkäufer unterhalten, sondern mit Ihrem Computer. Auch der Lernprozess, den Sie mit diesem Buch vor sich haben, verläuft ganz ähnlich.

Zuerst steht Vokabelpauken auf dem Plan, und schon nach wenigen Minuten werden Sie mit den ersten Vokabeln bereits einfache Aufgaben lösen können. Dieser erste Lernschritt bildet den Teil **»Interaktive Befehlskonsole«** dieses Buchs **(Kapitel 2–7)**. Hier lernen Sie den Umstieg von der klassischen *cmd.exe*-Befehlskonsole zur PowerShell, und für viele Administratoren reicht das Wissen aus diesem Teil bereits.

Dass PowerShell indes nicht eine klassische Textkonsole ist, sondern in Wirklichkeit eine hochmoderne objektorientierte Shell, entdecken Sie im Teil **»Objektorientierte Shell« (Kapitel 8–11)**.

## PowerShell als Ersatz für VBScript und Batch

Nachdem Sie die Sprache beherrschen, wird bei Ihnen mit einiger Wahrscheinlichkeit der Wunsch aufkommen, PowerShell nicht nur interaktiv zu verwenden, sondern auch als vollwertige Programmier- und Skriptsprache, um damit einfache und auch komplexe Automationsaufgaben zu lösen. Dies erwartet Sie im Teil **»Automationssprache« (Kapitel 12–21)**: Hier geht es zum einen darum, wiederkehrende Aufgaben mit PowerShell-Lösungen ein für alle Mal zu lösen.

Zum anderen geht es aber vor allem darum, wie Sie in PowerShell selbst automatisieren, also dafür sorgen, dass Sie nicht ständig ein Rad neu erfinden müssen. Sie erfahren, wie sich Ihre Automationslösungen in neue eigene PowerShell-Befehle kapseln lassen und wie Sie diese neuen Befehle für sich selbst und Ihre Kollegen bereitstellen können. Mit jeder neuen Lösung, die Sie entwickeln, wächst so Ihr Lösungsspektrum.

## Grenzenloses PowerShell

PowerShell wird zwar auf lokalen Computern ausgeführt, kennt aber kaum Grenzen. Es steht inzwischen nicht nur auf Windows-Betriebssystemen zur Verfügung, sondern auch auf Linux und MacOS. Alle PowerShells können über das Netzwerk miteinander sprechen, sich abstimmen und so die Grundlage für übergreifende und unternehmensweite Automationslösungen sein. Lesen Sie dazu mehr im Teil **»Remoting und Parallelverarbeitung« (Kapitel 22–25)**.

Neben der »räumlichen« Unabhängigkeit enthält PowerShell auch Wege, um die Sicherheitskontexte und Identitäten zu wechseln, in denen Befehle ausgeführt werden. Auch diese Technik beruht im Kern auf dem Remoting und bildet die Grundlage für JEA (*Just Enough Admin*), mit der eine rollenbasierte Administration möglich wird, zum Beispiel für Self-Servicing-Portale.

## Strategische Plattform und Orchestrierung

Wer seine Infrastruktur »via Code« definieren und ausrollen möchte, findet in PowerShell auch hierzu machtvolle Lösungen.

Hierbei trennt PowerShell das »Was« vom »Wo«. Das »Was« könnte die Definition eines Webservers oder Active Directory sein. Das »Wo« entspräche den Konfigurationsdaten, also den jeweiligen Server- und Webnamen. Das »Wann« kann wiederum beliebig gewählt werden, denn sobald Infrastruktur als Code definiert ist, kann man die dahinterliegenden Automationsskripte jederzeit und beliebig häufig einsetzen. Mehr dazu zeigt Ihnen **»DevOps und Enterprise« (Kapitel 26/27)**.

## Anwendungsentwicklung

Zwar war PowerShell ursprünglich als Automationssprache gedacht, die unbeaufsichtigt Aufgaben durchführt. Inzwischen wird PowerShell allerdings häufig auch zur Anwendungsentwicklung eingesetzt: Administratoren schreiben sich maßgeschneiderte Tools selbst oder stellen Tools dem Helpdesk zur Verfügung.

Wie man solche Oberflächen in PowerShell gestaltet, ist ebenso Thema des Teils **»Spezielle Techniken« (Kapitel 28–34)** wie innovative neue Wege, um Skripte zuverlässig zu entwickeln und weiterzupflegen. Mit dem *Test-Driven Development* (TDD), das ebenfalls aus der modernen Softwareentwicklung stammt, lassen sich PowerShell-Skripte äußerst zuverlässig und mit eingebauter Qualitätskontrolle herstellen.

## Persönliche Entwicklung

PowerShell kann eine Karriereentscheidung sein. Wann immer Sie eine Aufgabe lösen, stehen dahinter unsichtbare Motivationen. Die offensichtlichste ist natürlich, die Aufgabe gut zu erledigen, denn dafür werden Sie (wahrscheinlich) bezahlt. Ebenso wichtig ist aber, was die Lösung dieser Aufgabe sonst noch für Sie bedeutet und wie sie in Ihre Lebensbilanz einfließt. Wer sich Tag für Tag durch Dialogfelder klickt, kann zwar enorm erfolgreich Aufgaben lösen, entwickelt seine Fähigkeiten aber nicht weiter, und wenn das Dialogfeld eines Tages nicht mehr da ist, gilt das vielleicht auch für den eigenen Arbeitsplatz.

Es mag Sie anfangs etwas mehr Zeit kosten, die Lösung damit zu automatisieren, aber bedenken Sie: Jede Extraminute, die Sie hier investieren, investieren Sie eigentlich in Ihre persönliche Fortbildung. Auch ein Arbeitgeber sollte dies als Chance verstehen und Freiräume dafür gestatten. Denn mit jeder erfolgreich gemeisterten PowerShell-Lösung wächst Ihre Sprachfertigkeit. Wer PowerShell am Ende fließend spricht, ist bestens aufgestellt für moderne IT-Landschaften.

Gerade wenn Sie vorher noch nie geskriptet haben, sehen Sie PowerShell als Chance: Wer den Zug zu VBScript-Zeiten vor zehn Jahren vielleicht verpasst hat, kann heute auf einen neuen Zug aufspringen.

# Wie Sie dieses Buch nutzen

Dieses Buch setzt keinerlei Grundkenntnisse voraus, zumindest wenn Sie von vorn zu lesen beginnen. Wer unter Zeitdruck steht, kann auch quer einsteigen, und wer noch weniger Zeit hat, findet am Anfang jedes Kapitels eine Zusammenfassung, in der die jeweils wichtigsten Inhalte für Krisenzeiten zusammengefasst sind.

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.

Viele PowerShell-Codebeispiele sind sehr kurz und können mit geringem Aufwand schnell eingetippt werden. Umfangreichere Beispiele sind mit einer Listing-Unterschrift gekennzeichnet. Unter dem dort genannten Dateinamen finden Sie die Codebeispiele auch in den Begleitmaterialien, die Sie hier herunterladen können: *http://downloads.oreilly.de/9783960090090*.

Sollte es zu diesem Buch Errata geben, finden Sie sie ebenfalls hier: *http://downloads.oreilly.de/9783960090090*.

---

**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

Inzwischen gibt es in fast jedem Land eine starke PowerShell-Community. Auch in Ihrer Nähe finden sich »PowerShell User Groups«, in denen man kostenlos viel dazulernen und persönliche Kontakte aufbauen kann. Jährlich finden zudem internationale PowerShell-Konferenzen wie die »PowerShell Conference Europe« (*http://www.psconf.eu*) statt.

Falls bei der Arbeit mit diesem Buch Fragen auftauchen oder Sie Anregungen haben, besuchen Sie mich: *http://www.powertheshell.com*. Oder senden Sie mir eine Nachricht an meine Mailadresse *tobias.weltner@email.de*.

Bevor ich Ihnen jetzt viel Spaß mit PowerShell wünsche, geht noch ein großes Dankeschön an meine Lektorin Ariane Hesse und die Korrektorin Sibylle Feldmann, die dieses Buch mit allergrößtem Sachverstand und mit Sorgfalt begleitet haben.

Herzlichst Ihr

Dr. Tobias Weltner

# Kapitel 1
# PowerShell kennenlernen

**In diesem Kapitel:**



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

- **PowerShell-Host:** PowerShell ist Bestandteil von Windows und kein einzelnes Programm. Programme, die den Zugriff auf PowerShell ermöglichen, werden »Host« (»Gastgeber«) genannt. PowerShell liefert zwei Hosts mit: die PowerShell-Konsole (*PowerShell.exe*) und den komfortableren ISE-Editor (*PowerShell_ise.exe*).
- **Groß- und Kleinschreibung:** Die Groß- und Kleinschreibung wird bei Befehlen und Parameternamen nicht unterschieden.
- **Farbcodierung während der Eingabe:** Ab PowerShell 5 färbt nicht nur der ISE-Editor, sondern nun auch die PowerShell-Konsole Eingaben ein. Die Farben unterscheiden zwischen Befehlen, Parametern und Argumenten. So kann man Eingaben mit einem kurzen Blick auf die Farben auf Richtigkeit überprüfen. Enthält die Eingabe Syntaxfehler, also formale Fehler wie fehlende Anführungszeichen, kennzeichnet ISE diesen Teil mit einer roten Wellenlinie. Die Konsole zeigt eine rote spitze Klammer am Ende des Eingabeprompts an.

- **Ausgabebefehle und Umwandlungsabkürzungen:** PowerShell gibt Resultate sofort aus. Ein spezieller Ausgabebefehl wie `echo` ist nicht nötig. Auch unterstützt PowerShell einfache Rechenaufgaben, bei denen die in der IT üblichen Größenordnungen wie KB oder GB direkt (ohne Leerzeichen) an eine Zahl angefügt werden können. Mit dem Präfix `0x` werden hexadezimale Zahlen markiert, und `..` liefert einen Zahlenbereich, zum Beispiel `1..49`.
- **Autovervollständigung:** Mit ↑ und ↓ gelangen Sie zurück zu Befehlsfolgen, die Sie schon einmal eingegeben haben. Möchten Sie einen Befehl nachträglich ändern oder erweitern, verwenden Sie die Pfeiltasten, um, anstatt den gesamten Befehl neu einzugeben, zu dem jeweiligen Befehl zurückzukehren und ihn zu ändern. Mit ⇆ aktivieren Sie die eingebaute Autovervollständigung. Diese kann Befehlsnamen, Pfadnamen und andere Eingaben für Sie vervollständigen. Drücken Sie die Taste mehrmals, zeigt PowerShell bei jedem Druck einen anderen Vorschlag. In ISE steht außerdem das IntelliSense-Menü zur Verfügung, das über Strg+Leertaste Eingabevorschläge nicht sofort einfügt, sondern zuerst in einem Kontextmenü anbietet.
- **Zeilen löschen und Befehlsabbruch:** Wollen Sie die gesamte aktuelle Zeile löschen, drücken Sie Esc. Möchten Sie im Mehrzeilenmodus die aktuelle Zeile zwar nicht ausführen, aber auch nicht verlieren, drücken Sie Strg+C.
- **Skriptausführung:** Anfangs kann PowerShell nur interaktive Befehle ausführen, aber keine Skripte. Mit `Set-ExecutionPolicy` sollte die Skriptausführung so bald wie möglich aktiviert werden, weil viele interaktive Befehle aus Skriptdateien geladen werden und andernfalls nicht funktionieren.
- **Hilfe zu PowerShell-Befehlen:** PowerShell-Befehle sind gut dokumentiert, aber die Dokumentation muss zunächst mit `Update-Help` aus dem Internet heruntergeladen werden.
- **Unterschiedliche PowerShell-Versionen:** Es gibt aktuell fünf PowerShell-Versionen, die alle aufeinander aufbauen. Die aktuelle PowerShell-Version erfährt man zum Beispiel über den Befehl `$host.Version`.

---

Häufig wird PowerShell gleichgesetzt mit der typisch blauen Textkonsole, die kryptische Befehlseingaben von Ihnen erwartet. Dieses Bild wäre aber falsch. PowerShell ist keine Textkonsole.

PowerShell ist eine Automationssprache und vollkommen unabhängig von bestimmten Programmen. Sie ist als Teil des »Windows Management Framework« (WMF) fest in das Betriebssystem Windows integriert. Die PowerShell-Konsole (*PowerShell.exe*) ist also lediglich ein Programm, mit dem man Kontakt zur PowerShell aufnehmen kann. Andere Programme können das auch, und so ist die PowerShell-Konsole längst nicht das einzige Programm, in dem Ihnen die Sprache »PowerShell« begegnet.

Bettet ein Programm – so wie die PowerShell-Konsole – die PowerShell-Sprache ein, wird es »PowerShell-Host« oder einfach nur »Host« genannt. Dieses englische Wort steht für »Gastgeber«, und so sind PowerShell-Programme Gastgeber für die PowerShell, bieten also die Bühne und Ihnen die Möglichkeit, Befehle an PowerShell zu senden und Ergebnisse zu empfangen. Ein weiterer solcher Host ist ebenfalls Bestandteil von Windows: der modernere »ISE«-Editor (»Integrated Script Environment«, *PowerShell_ise.exe*).