IN DIESEM KAPITEL

Kapitel 1
PowerShell Backstage

Bevor Sie sich intensiv mit inhaltlichen Dingen in der PowerShell beschäftigen, erfahren Sie einige Dinge zu ihrem Hintergrund, die Ihnen einzuordnen helfen, was die PowerShell ist und warum sie sich so entwickelt hat, wie sie es getan hat.

Sie erhalten in diesem Kapitel kurze Einblicke zur Geschichte der PowerShell und zu den Absichten der Entwickler. Ich kläre, warum es die Windows PowerShell und PowerShell Core gibt und wie sie zusammenhängen. Letztlich erhalten Sie einen Ausblick auf zukünftige Versionen der PowerShell und eine Empfehlung, welche Sie einsetzen sollten.

Für Eilige, die bereits alle vorbereitenden Aufgaben erledigt haben und sofort mit dem Eintippen erster Befehle in der PowerShell beginnen wollen:

Beginnen Sie mit Teil II und schauen Sie in den Kapiteln des ersten Teils später wieder vorbei. In diesem Teil installieren Sie Ihre Arbeitsumgebung und richten sie so ein, dass Sie inhaltlich loslegen können.

Geschichte der PowerShell

PowerShell wurde speziell für die Systemverwaltung und -automatisierung entworfen. Sie erlaubt Zugriff auf WMI-Klassen, COM-Objekte sowie auf das gesamte .NET Framework. Die PowerShell verbindet die aus Unix-Shells bekannte Philosophie von Pipes und Filtern mit objektorientierter Programmierung.

Adressiert werden gleichermaßen Administratoren und Programmierer. Vornehmlich Administratoren können weiterhin einfache Befehle an einer Kommandozeile ausführen und miteinander verknüpfen. Vornehmlich Programmierer können komplexe Skripte mit der eigens dafür entwickelten *PowerShell Scripting Language* schreiben.

Ausgangssituation und Konzept

Um zu verstehen, welches Konzept hinter der PowerShell steht, schauen wir kurz auf die Situation unter Windows um die Jahrtausendwende:

- ✔ Es gibt eine bereits aus DOS-Zeiten bekannte Eingabeaufforderung (auch *Kommandozeile* genannt) mit etwas erweitertem Funktionsumfang. Sie bietet einige Standardbefehle und ermöglicht das Ausführen weiterer Konsolenanwendungen.
- ✔ Es existiert eine einfache, relative beschränkte Skriptsprache zur Automatisierung von Aufgaben. Über die Eingabeaufforderung sind jedoch nicht alle Funktionalitäten der grafischen Benutzeroberfläche erreichbar, sodass nicht alle Aufgaben automatisiert werden können.
- ✔ Mit Windows Server 2003 sind viele Funktionen auch per Eingabeaufforderung verfügbar, die Limitierung der Skriptsprache und Inkonsistenzen in der Bedienung verschiedener Konsolenanwendungen erschweren aber eine effektive Administration. Die Limitierung der Skriptsprache versuchte Microsoft durch Einführung von *Windows Script Host* zu überwinden, erreichte damit aber eher Skriptentwickler und kaum Administratoren.

In dieser Umgebung entwickelte Jeffrey Snover 2002 die Idee einer Plattform der nächsten Generation für die administrative Automatisierung. Traditionelle Verwaltungsprobleme sollten durch Nutzung der von Microsoft um das Jahr 2000 bereitgestellten .NET-Plattform gelöst werden.

Snover hatte nicht nur die Skriptentwickler im Blick, sondern erwartete erhebliche Vorteile auch für Tester, Power-User und Administratoren.

Das Produkt unter dem vorläufigen Namen *Monad* nutzt die .NET Common Runtime, um ein mächtiges, konsistentes, intuitives, erweiterbares und nützliches Werkzeug bereitzustellen, das die Administrationskosten senkt und das Leben von Nicht-Programmierern erheblich vereinfacht.

Jeffery Snover hat sein Konzept unter dem Titel *Monad Manifesto* festgehalten. Allen, die am Hintergrund der Entwicklung der PowerShell und den Ideen des Entwicklers besonderes Interesse haben, sei das 16-seitige Papier zur Lektüre empfohlen. Sie finden es unter anderem unter folgender Adresse: `https://www.jsnover.com/Docs/MonadManifesto.pdf`.

Folgende Punkte sollten bei der Umsetzung der Ziele helfen:

- ✔ Administratoren können Befehle schneller und einfacher programmieren, da Monad viele Standardaufgaben von Befehlen übernimmt und durch einen einheitlichen Parser Konsistenz bietet.
- ✔ Strukturierte Daten (Objekte) werden anstelle von unstrukturierten Daten (Text) zur Weiterverarbeitung bereitgestellt.

- ✔ Skripte können nicht nur lokal, sondern auf einer Vielzahl von Remotecomputern ausgeführt werden.
- ✔ Anwender können auch eine grafische Oberfläche nutzen.

Sie vermuten es sicherlich bereits: Die Rede ist von der PowerShell. 2006 erfolgt die Umbenennung von Monad in PowerShell.

Die Windows PowerShell wird unentbehrlich

Im selben Jahr legt Microsoft fest, dass Microsoft Exchange Server 2007 per PowerShell administriert wird und auch die grafische Benutzeroberfläche auf PowerShell aufsetzt. Die Windows PowerShell Version 1 wird veröffentlicht und zum Download angeboten.

2007 wird die Windows PowerShell Teil von Microsofts Kriterienkatalog (*Common Engineering Criteria*), dem jedes Microsoft-Server-Produkt entsprechen soll. Jedes Server-Produkt von Microsoft soll von nun an PowerShell unterstützen.

Dies bedeutet ab 2009 für die Windows PowerShell ab Version 2 den endgültigen Durchbruch als zentrales Verwaltungs- und Automatisierungswerkzeug in Windows-Betriebssystemen. Sie wird ab sofort als fester Bestandteil von Windows-Betriebssystemen ausgeliefert.

Sie haben bereits jetzt den Begriff *PowerShell* in den Varianten *PowerShell*, *Windows PowerShell* und *PowerShell Core* kennengelernt. Das soll Sie nicht verwirren, sondern durch bedachte Wahl für Klarheit sorgen.

Im weiteren Verlauf des Buches werden Sie den Begriff PowerShell in verschiedenen Varianten für unterschiedliche Kontexte lesen:

- ✔ **PowerShell:**

  allgemein, bezeichnet das Gesamtprodukt und gilt für jede Version

- ✔ **Windows PowerShell:**

  bezeichnet die mit den Windows-Betriebssystemen ausgelieferte PowerShell, die ausschließlich unter Windows lauffähig ist

- ✔ **PowerShell Core:**

  das neue Open-Source-Projekt, das plattformunabhängig auf den Betriebssystemen Windows, Linux und macOS installiert werden kann

Ab Version 2 steht die Windows PowerShell als Bestandteil des Windows Management Frameworks mit der jeweils identischen Versionsnummer zum Download bereit. Damit kann eine neuere Version der Windows PowerShell auch auf älteren Betriebssystemversionen installiert werden.

Ab Windows PowerShell Version 3 können neuere Versionen der Windows PowerShell nachträglich auf Betriebssystemen ab Windows 7 beziehungsweise Windows Server 2008 (R2) installiert werden.

Bei der Windows PowerShell folgen Version 5 und Version 5.1 zeitlich relativ eng aufeinander. Das liegt daran, dass Windows Server 2016 erst später als Windows 10 erschien. In den Monaten zwischen Erscheinen von Windows 10 und Windows Server 2016 hat sich die Windows PowerShell weiterentwickelt. Letztlich hat man sich entschieden, Windows Server 2016 direkt mit PowerShell Version 5.1 auszuliefern und unter Windows 10 ein Update auf Version 5.1 vorzunehmen.

Die Windows PowerShell wird in der Windows-Welt zu **dem** Werkzeug für Administratoren und Programmierer. Fast alles ist bei der System- und Netzwerkverwaltung mit der Windows PowerShell möglich, Anwendungen bringen ihren eigenen Satz an PowerShell-Befehlen für die Verwaltung mit. Einige Einstellungen können ausschließlich über die PowerShell vorgenommen werden.

Im ersten Erscheinungsjahr 2006 ist die PowerShell als Produkt und Sprache zwar noch sehr jung, aber seit Version 5.1 gilt sie als fertig und abgeschlossen.

Die Windows PowerShell hat nachhaltig die Windows-Welt beeinflusst und beispielsweise die Arbeit von Windows-Administratoren deutlich verändert.

Sie lesen an verschiedenen Stellen, dass die Entwicklung der PowerShell unter Windows *eingestellt* oder *nicht weiterentwickelt* wird. Das könnte den Eindruck erwecken, dass ein unfertiges Produkt, das vielleicht nicht gut genug ist, aus dem Verkehr gezogen werden soll.

Das ist nicht korrekt. Bereits vor Fertigstellung der Version Windows PowerShell 5.1 äußerten sich Erfinder Jeffrey Snover und sein Projektteam auf verschiedenen Konferenzen dahin gehend, dass sie alles umgesetzt haben, was sie sich mit der Windows PowerShell vorgenommen haben (»*we're done!*«).

Es wird unter Windows keine neuen Versionen geben, da es sich um ein fertiges Produkt handelt. Mit einer weiteren Unterstützung und der Bereinigung auftretender Fehler ist zu rechnen.

PowerShell Core – plattformübergreifend und Open Source

2016 verkündet Microsoft, dass PowerShell unter der sogenannten MIT-Lizenz nunmehr Open Source und plattformübergreifend entwickelt wird. Das Produkt heißt jetzt PowerShell Core, die Versionszählung beginnt mit Version 6.0 und setzt damit die Versionszählung der Windows PowerShell fort. Veröffentlicht wird PowerShell Core auf einer Plattform für Projekte aus der Software-Entwicklung, dem Onlinedienst *GitHub*. Die erste Version von PowerShell Core (Version 6.0) erschien Anfang 2018, Version 6.1 bereits etwa neun Monate später.

Microsoft trägt damit seiner Cloud-Strategie (»Mobile first, Cloud first«) Rechnung und möchte PowerShell für diese heterogenen Umgebungen als universelles Werkzeug für alle Administratoren anbieten, gleich welches Betriebssystem oder welche Anwendung administriert werden soll.

Eine plattformübergreifende Bereitstellung der PowerShell hat auch zur Konsequenz, dass das .NET Framework nicht zur Verfügung steht, da es ausschließlich für Windows verfügbar ist. Um eine ähnliche Funktionalität auch für andere Betriebssysteme zur Verfügung zu stellen, entwickelte Microsoft die Frame-Variante *.NET Core*.

.NET Core wurde von Grund auf neu entwickelt und hatte zu Beginn des Jahres 2015 etwa ein Zehntel der Funktionalität des klassischen .NET Frameworks.

Durch eine leichtere Portierbarkeit auf Microsoft-fremde Plattformen sowie die Entwicklung als Open-Source-Projekt unter Beteiligung der weltweiten Entwicklergemeinschaft scheint .NET Core die ideale Basis für PowerShell Core zu sein.

Tabelle 1.1 gibt Ihnen einen Überblick über die PowerShell-Versionen und zeigt, wo und wann die einzelnen Versionen erschienen sind.

Version	Vorinstalliert?	Erscheinungsjahr
1 *Windows PowerShell*	Nein, Download für Windows-Betriebssysteme ab Windows XP SP2	2006
2 *Windows PowerShell*	Ja, unter Windows 7 und Windows Server 2008 R2	2009
3 *Windows PowerShell*	Ja, unter Windows 8 und Windows Server 2012	2012
4 *Windows PowerShell*	Ja, unter Windows 8.1 und Windows Server 2012 R2	2013
5 *Windows PowerShell*	Ja, unter Windows 10	2015
5.1 *Windows PowerShell*	Ja, unter Windows 10 (Build 1607) und Windows Server 2016	2016
6 *PowerShell Core*	Nein, Downloadpaket für Windows, Linux und macOS	2018
6.1 *PowerShell Core*	Nein, Downloadpaket für Windows, Linux und macOS	2018

Tabelle 1.1: Die Versionen der PowerShell

Übersicht über die Versionen der PowerShell

Die Versionen 1 bis 5.1 markieren seit 2006 die Zeit der PowerShell als Komponente ausschließlich in Windows-Betriebssystemen mit tiefer Einbindung ins System durch die Basis .NET Framework.

2018 bedeutet den Wendepunkt für die PowerShell. Als PowerShell Core beginnt mit der Version 6.0 eine neue Zeitrechnung als plattformübergreifende Open-Source-Software. Durch das neue, noch nicht annähernd so mächtige .NET Core beginnt eine neue Reise mit momentan (noch) verminderter Leistungsfähigkeit.

Projekt »Glaskugel«: Zukünftige Versionen

Es ist sicher, dass es keine neuen Versionen der Windows PowerShell geben wird. Die Zukunft liegt bei der plattformübergreifenden Open-Source-Variante PowerShell Core.

Lassen Sie mich einen Blick in die Glaskugel wagen, welche Entwicklung die PowerShell in den nächsten Jahren wahrscheinlich nehmen wird.

- ✔ Da die Windows PowerShell die Windows-Welt der letzten Jahre nachhaltig geprägt hat, wird sie nicht aus Windows-Systemen verschwinden.
- ✔ Die Windows PowerShell stellt ein fertiges Produkt dar, es erscheinen nur noch Fehlerbehebungen und Sicherheitsupdates.
- ✔ Windows PowerShell und PowerShell Core finden mittelfristig unter dem Namen PowerShell zusammen.
- ✔ Durch die Einbindung der weltweiten Entwicklergemeinschaft des Open-Source-Projekts PowerShell Core entwickelt sich PowerShell Core schnell zu einem wichtigen Tool für alle beteiligten Betriebssysteme.
- ✔ .NET Core wird weiterentwickelt und liefert PowerShell Core deutlich mehr Funktionalität.
- ✔ Administratoren von Linux und macOS fragen sich immer weniger, warum sie sich mit PowerShell Core beschäftigen sollen.
- ✔ PowerShell entwickelt sich zu einem ernst zu nehmenden Werkzeug für beteiligte Betriebssysteme und die Cloud und hat ihre Stärken in heterogenen Umgebungen.

Sie können mich in einigen Jahren gern ansprechen, wie falsch ich doch gelegen hätte. Ich werde alles auf meine Glaskugel schieben …

PowerShell-Entwicklerteam: Ein Ausblick

Meine eigenen Einschätzungen in Ehren, ich möchte Ihnen allerdings aktuelle Einschätzungen des Microsoft-Entwicklerteams nicht vorenthalten. Die folgenden Aussagen stammen vom PowerShell-Erfinder Jeffrey Snover aus seiner Keynote zur europäischen PowerShell-Konferenz in Hannover im April 2018:

- ✔ PowerShell war noch nie so wichtig wie heute (2018).
- ✔ Die neue Mission für die PowerShell lautet: Unterstützung der digitalen Transformation. Verwalten Sie von *jedem* Client aus *jeden* Server oder Dienst, der in einer *beliebigen* Cloud ausgeführt wird. Oder bei Verwendung vor Ort: Verwalten Sie *jeden* Hypervisor, *jeden* Speicher und *jedes* Netzwerk.

✔ Was wird aus der Windows PowerShell?

- Die Entwicklung der Windows PowerShell ist abgeschlossen.
- Windows PowerShell bleibt vollständig unterstützt (wahrscheinlich für immer).
- Es gibt keine Entwicklung weiterer Features.
- Es gibt nur noch Sicherheitsupdates und Fehlerbehebungen.
- Alle Weiterentwicklungen erfolgen in PowerShell (Core).

✔ Vorhersagen:

- PowerShell Core wird in PowerShell umbenannt.
- Linux-Distributionen enthalten zukünftig PowerShell.

Die Folien des Vortrags, aus dem ich zitiert habe, finden Sie unter der Adresse `https://github.com/psconfeu/2018/blob/master/Jeffrey%20Snover/Keynote_psconfeu_2018.pdf`. Wenn Sie den Vortrag als Video anschauen wollen, besuchen Sie die Adresse `https://www.youtube.com/watch?v=zy4fDSdrM7M`.

Just zum Redaktionsschluss dieses Buchs veröffentlicht Microsoft die Mitteilung, dass mit der noch für 2019 geplanten Version 7 von PowerShell Core die erwartete Umbenennung in PowerShell vorgenommen werden soll. Hinsichtlich des Funktionsumfangs sind keine großen Neuerungen geplant.

Mittelfristig soll PowerShell 7 neben Windows PowerShell 5.1 Bestandteil von Windows werden.

Für Sie bedeutet das bei der Lektüre:

✔ Alle beschriebenen Inhalte sind selbstverständlich nach wie vor gültig.

✔ Achten Sie auf die PowerShell-Versionsnummern: Versionen ab 6 sind PowerShell (Core), Versionen bis 5.1 Windows PowerShell.

✔ Da zukünftige Versionen nicht »PowerShell Core«, sondern »PowerShell« heißen werden, verstehen Sie den Zusatz »Core« bitte als Hinweis, dass es sich hierbei um die plattformübergreifende moderne PowerShell handelt und nicht um die auf Windows beschränkte Windows PowerShell.

Welche PowerShell-Version Sie verwenden sollten

Wenn Sie ein aktuelles Windows-Betriebssystem verwenden, auf dem die Windows PowerShell vorinstalliert ist, spricht momentan alles dafür, diese zu verwenden. Durch die tiefe Einbettung in das Windows-System durch das .NET Framework stehen Befehle

und Funktionen für alle denkbaren Aufgaben zur Verfügung. Zudem unterscheiden sich die Windows PowerShell und PowerShell Core syntaktisch nicht. PowerShell Core kann momentan nur einen deutlich kleineren Befehlssatz verwenden, der erst nach und nach erweitert werden wird. Wieso also momentan auf die vorhandenen Möglichkeiten in Windows-Systemen verzichten?

Wenn Sie PowerShell Core kennenlernen, mit der Windows PowerShell vergleichen oder sich mit der Administration heterogener Umgebungen vertraut machen möchten, können Sie PowerShell Core auf Windows-Systemen parallel zur Windows PowerShell installieren und betreiben.

Als Nutzer von Linux oder macOS führt kein Weg an PowerShell Core vorbei, da Ihnen die Windows PowerShell dort nicht zur Verfügung steht. Auch wenn PowerShell Core momentan noch nicht an die Funktionalität der Windows PowerShell heranreicht, können Sie sehr gut die Syntax und das Konzept der PowerShell kennenlernen und die stetige Funktionserweiterung begleiten.

Übrigens: Alle verwendeten Befehle und Beispiele im Buch funktionieren sowohl unter den Versionen der Windows PowerShell als auch der PowerShell Core.

Kurz und knackig

Die PowerShell hat mit ihrer Kombination aus konsistentem Werkzeug für Administrations- und Automatisierungsaufgaben in Windows sowie eigener Skriptsprache eine große Lücke in der Windows-Welt gefüllt und den Arbeitsalltag von Windows-Administratoren und -Entwicklern verändert.

Bis zu Windows PowerShell Version 5.1 basiert die PowerShell auf .NET Framework und ist ausschließlich für Windows-Betriebssysteme verfügbar.

Ab Version 6.0 basiert die PowerShell Core auf .NET Core, ist Open Source und auch für Linux und macOS verfügbar.

Im nächsten Kapitel erfahren Sie, wo Sie in Windows die PowerShell finden und wie Sie sie auf den verschiedenen Systemen installieren.