Einführung

In dieser Einleitung:

Diese Einführung enthält wichtige Informationen zur PowerShell und zum Gebrauch dieses Buchs, also sollten Sie diese vielleicht nicht überspringen.

Wer braucht eigentlich die PowerShell?

In einer Welt, die eigentlich schon alles hat, ist es höchst ungewöhnlich, dass ein Unternehmen wie Microsoft zig Millionen US-Dollar in Entwicklungsarbeit investiert, um eine vollkommen neuartige Skript- und Automationssprache zu entwickeln: *Windows PowerShell*. Dass dies dennoch geschehen ist, hat mit Ihnen zu tun. Wieso genau, offenbart ein Blick auf die PowerShell aus verschieden hohen Umlaufbahnen.

Moderne Lernkurve

Wer ganz nah um die PowerShell kreist, vielleicht gerade selbst vor der PowerShell-Konsole sitzt, sieht vor allem Befehle, die *Cmdlets*. Sie funktionieren in etwa so wie in anderen Konsolen (*Shells*). Allerdings unterstützt die PowerShell so viele davon und ist auf so einheitliche Weise erweiterbar, dass man mit ganz geringen Grundkenntnissen fast alles damit administrieren kann. Noch wichtiger: Weil alle Cmdlets genau denselben Grundregeln folgen, kann man das Wissen auch leicht auf andere Aufgabenbereiche und Cmdlets anwenden.

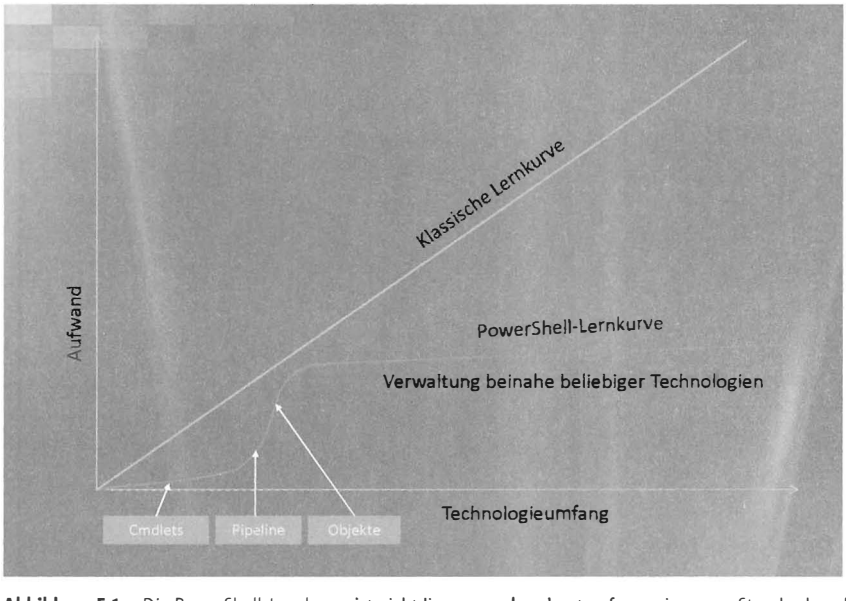

Abbildung E.1 Die PowerShell-Lernkurve ist nicht linear, sondern baut auf gemeinsamen Standards auf

Die PowerShell baut also auf Standards auf. Diese stellen in der PowerShell-Lernkurve zunächst eine kleine Steilwand dar, die Sie aber mithilfe dieses Buchs meistern. Danach flacht die Lernkurve stark ab und alles wird gut: Ob Sie mit Exchange ein Postfach anlegen, mit SharePoint eine Website veröffent-

lichen 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.

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

Entfernt man sich etwas von der PowerShell und betrachtet die Technik mit mehr Abstand, verschwimmen die einzelnen Befehle zu etwas Neuem: einer Sprache. Tatsächlich benimmt sich die 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. Und auch der Lernprozess, den Sie mit diesem Buch vor sich haben, verläuft ganz ähnlich.

Zuerst steht Vokabelpauken auf dem Lehrplan, und schon nach wenigen Minuten werden Sie mit den ersten Vokabeln bereits einfache Aufgaben lösen. Tatsächlich werden einige Administratoren nie wirklich über diese Stufe hinauswachsen, weil es für sie vielleicht unökonomisch ist. Viele Urlauber kommen sehr gut mit ein paar Brocken Landessprache aus, um Wein, Käse und Baguette zu bestellen sowie nach Preis und Uhrzeit zu fragen. Nicht anders ist das in der IT. Hier zählt erst einmal nur, was essentiell ist, um eine Aufgabe zu lösen. Dieser erste Lernschritt bildet gleichzeitig den *Schwierigkeitsgrad 1* (für *Einsteiger*) und bildet *Teil A* dieses Buchs (Kapitel 1–5).

Nach dem Vokabelpauken folgen Grammatik und Satzbau, und so ist das auch bei der PowerShell. Sie lernen die PowerShell-Pipeline kennen, mit der Sie einzelne Wörter (Befehle) in geschliffene Ausdrücke verwandeln. Wer diese Grammatik beherrscht, kann plötzlich sehr viel differenziertere Aufgaben lösen und ist nicht mehr allein auf die Wortbrocken des Grundwortschatzes angewiesen. Auch hier gilt: Nicht jeder wird fließend die PowerShell sprechen lernen, doch werden Sie auf jeden Fall fließend die PowerShell verstehen lernen und dann die Ausdrücke anderer PowerShell-Sprechender mühelos nachvollziehen und für eigene Zwecke anpassen können. Der Satzbau entspricht *Schwierigkeitsgrad 2* (*Fortgeschrittene*) und bildet *Teil B* dieses Buchs mit den Kapiteln 6 und 7.

Abbildung E.2 Die PowerShell enthält verschiedene Schwierigkeitsgerade, von »Einsteiger« bis »Experte«

Nachdem Sie die Sprache beherrschen, wird es einige geben, die nicht nur konsumieren, sondern auch produzieren. Sie werden eigene Funktionen und Skripts schreiben und als Module exportieren, also den Literaturmarkt der PowerShell vergrößern. Damit helfen Sie anderen, die auf einer anderen Stufe der PowerShell-Lernkurve stehen geblieben sind, denn Ihre Module sind im Grunde nichts weiter als neue Vokabeln, die auch von denjenigen rasch einsetzbar sind, die über das erste Vokabellernen nicht hinausgehen wollten. Alles zu diesem Thema entspricht *Schwierigkeitsgrad 3* (*PowerShell-Autoren*) und findet sich in *Teil D* mit den Kapiteln 10–15.

Der Olymp der PowerShell-Sprache ist natürlich die Poesie, bei der Sie mit den Nuancen der Sprache spielen, ihre unterschwelligen Stärken betonen und Lösungen schaffen, die über das hinauswachsen,

was mit Alltagsausdrücken möglich ist. Nicht jeder wird Ihre Poesie verstehen, und manche PowerShell-Poesie liegt eigentlich im Grenzbereich zu anderen Sprachen und Lösungen. Dieses sanfte Grundverständnis für die Natur der PowerShell ist aber enorm wichtig, weil es auch bei ganz normalen und vollkommen unesoterischen Aufgabenstellungen den Kitt bildet, um mit den vorhandenen Befehlen die gesuchte Lösung zu erstellen. Deshalb befindet sich dieses Thema zwischen Satzbau und Skriptbau bereits in *Teil C* mit den Kapiteln 8 und 9. Der *Schwierigkeitsgrad 4* (*PowerShell-Profi*) ist für viele möglicherweise etwas harte Kost, doch muss man dieses Kapitel nicht von vorn bis hinten durchlesen oder gar auswendig kennen. Es genügt zu wissen, dass es diese Techniken gibt, falls Sie auf normalem Wege einmal nicht weiter kommen.

Schließlich finden Sie in *Teil E* mit den Kapiteln 16 bis 19 noch ganz spezielle erwähnenswerte PowerShell-Techniken wie Remotezugriffe oder Hintergrundjobs, die für Benutzer aller Schwierigkeitsgrade gleichermaßen spannend und interessant sind.

Eine strategische Plattform

Entfernt man sich noch etwas weiter von der PowerShell, rücken plötzlich andere Prioritäten in den Vordergrund und die PowerShell erscheint wiederum in neuem Licht. Die PowerShell ist lösungsorientiert, ja – aber es ist auch eine strategische Plattform, deren Bedeutung andere Automationssprachen weit überstrahlt. Tatsächlich ist die PowerShell eine neuartige zweite Benutzeroberfläche, wenn auch so andersartig als die gewohnte Maus-Fenster-Klick-Oberfläche, dass man dies zuerst gar nicht wahrnimmt. Microsoft hat nicht nur eine Skript- und Automationssprache entwickelt, um VBScript ins .NET-Zeitalter zu überführen oder die antiquierte Befehlszeile zu renovieren, sondern um eine langfristige Lösung zu schaffen für ein Problem, das viele heute vielleicht noch gar nicht wahrnehmen:

Unsere IT-Landschaft wird immer komplexer und muss sich immer höheren Anforderungen stellen, ökonomischen genau wie regulatorischen. Gleichzeitig steht nicht zu erwarten, dass qualifiziertes Fachpersonal in gleichem Maße aus dem Boden sprießt wie die Nachfrage danach steigen wird. Administratoren müssen also vielseitiger werden. Versuchen Sie heutzutage, einen SQL Server-Spezialisten mit der Administration von Microsoft Exchange Server zu betrauen, dürfte die Katastrophe absehbar sein. Da aber beide Produkte über die PowerShell verwaltbar sind, ist es mit der PowerShell sehr viel leichter, auch in anderen IT-Bereichen Verantwortung zu übernehmen.

Hier werden die Parallelen zur grafischen Benutzeroberfläche besonders deutlich: Während man Maus, Fenster und Menüs als Selbstverständlichkeit intuitiv bedient und sich auf die Eigenarten der Programme konzentriert, gilt Ähnliches für die PowerShell: Der Einsatz der Sprache selbst – der Cmdlets, Pipeline und Erweiterungsmodule – gehört zu den Selbstverständlichkeiten, über die man bald nicht mehr bewusst nachdenkt.

Administratoren müssen nicht nur vielseitiger werden, sondern sich auch die Arbeit erleichtern, um die nötigen Freiräume dafür zu schaffen. Das bei Kuckucksuhren und Schokoladenkonfekt begehrte Prädikat »handgemacht« verliert in der IT langsam, aber sicher seinen Charme. Alles händisch zu administrieren kostet zu viel Zeit, ist zu fehleranfällig und entspricht nicht modernen juristischen und regulatorischen Dokumentationsansprüchen.

Da die Aufgaben in der IT andererseits aber vielerorts noch nicht einmal annäherungsweise standardisiert sind, werden viele Aufgaben nur mit maßgeschneiderten Skripts automatisierbar sein. Auch

hier bildet die PowerShell einen ökonomischen Standard. PowerShell-Skripts sind für alle IT-Bereiche einsetzbar, lösen also auf Wunsch das bunte Sammelsurium aus PerlScript, VBScript, KIXX, Batch etc. ab, konsolidieren also die Automation. PowerShell-Skripts sind einheitlich dokumentierbar, signierbar und integriert in die Standardsicherheitsmechanismen von Windows.

Nicht zu unterschätzen ist dabei die Modularisierbarkeit. Während früher Berater und externe Firmen Skriptauftragslösungen erstellt haben, welche die Aufgaben zwar erledigten, aber kaum verständlich, nachvollziehbar, wartbar oder weiterverwertbar waren, bestehen solche Auftragslösungen bei der PowerShell aus Erweiterungsmodulen mit den gewünschten neuen »Vokabeln«. Der eigentliche Workflow ist davon getrennt, und Ihr Unternehmen kann diese Vokabeln künftig auch in ganz anderem Zusammenhang nutzen, um mit dem Computer zu sprechen und Aufgaben zu lösen.

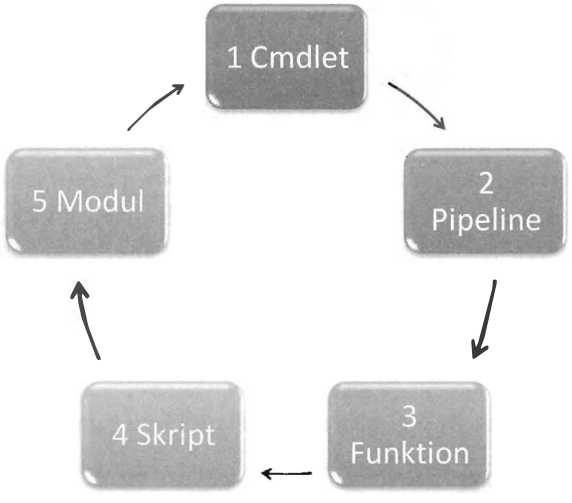

Abbildung E.3 PowerShell-Zyklus: Selbst komplexe Skripts werden über Module wieder zu einfachen Cmdlets

Und schließlich lebt die PowerShell längst nicht nur in der interaktiven PowerShell-Konsole, sondern steht als direkte Programmierschnittstelle zur Verfügung. Viele Softwarehersteller nutzen die PowerShell bereits als interne Skriptsprache, und künftig wird die Automationsoberfläche PowerShell immer stärker mit der grafischen Oberfläche Windows verschmelzen. Schon heute liefern viele Dialogfelder und Assistenten zum Schluss den PowerShell-Code, der das bewerkstelligt, was Sie im Dialogfeld zusammengeklickt haben.

Plattformübergreifende Technik

Entfernt man sich noch einmal ein Stück weiter von der PowerShell, treten plötzlich auch andersartige Systeme in den Blickwinkel. Was Sie nun verschwommen sehen, ist zwar heute noch weitgehend Vision, aber die ersten Grundzüge der Umsetzung sind bereits erkennbar. Während die PowerShell zurzeit noch eine reine Windows-Technik ist, wird es damit mittelfristig auch möglich, plattformübergreifend andere Systeme zu verwalten und einzubinden. Schon heute ist dies mit einer Technik namens *WSMan* möglich.

Persönliche Entwicklung

Schließlich kann die PowerShell auch 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 auch, 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. Zwar wird es immer gute Gründe für Klicklösungen geben, aber sobald Sie eine Aufgabe mehr als einmal durchführen müssen, sollten Sie über die PowerShell nachdenken.

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 die PowerShell am Ende fließend spricht, ist allerbestens aufgestellt für moderne IT-Landschaften. Und falls doch mal etwas schief geht, mag es Sie trösten, dass das wiederholte Schlagen mit der Stirn auf die Schreibtischoberfläche pro Stunde immerhin 68 Kalorien verbraucht.

Gerade falls Sie vorher noch nie geskriptet haben, sehen Sie die PowerShell als Chance: Wer den Zug vielleicht zu VBScript-Zeiten vor zehn Jahren verpasst hat und sich nun etwas abgehängt vorkommt, kann heute auf einen neuen Zug aufspringen. Mit diesem Buch haben Sie alles, was Sie wissen müssen, und können sich natürlich auch zurückgezogen im stillen Kämmerlein und bei eigenem Tempo in die PowerShell einarbeiten, um dann plötzlich und unerwartet als neuer Skriptguru das Rampenlicht zu betreten.

Wie Sie dieses Buch nutzen

Dieses Buch ist in mehrere Teile gegliedert, von denen Sie ja schon gehört haben. Es setzt keinerlei Grundkenntnisse voraus, zumindest wenn Sie von vorn zu lesen beginnen. Wer unter Zeitdruck steht, kann aber auch quer einsteigen, und wer noch weniger Zeit hat, findet in jedem Kapitel Kästen mit der Überschrift *Kurz und knapp…*, wo die jeweils wichtigsten Inhalte für Krisenzeiten zusammengefasst sind.

Namentlich ist dieses Buch ein *Einsteigerworkshop*, also eine Trainingsgrundlage, mit der Sie sich Schritt für Schritt PowerShell-Kompetenz aneignen. Das Wort *Einsteiger* bezieht sich allerdings vor allem auf die Tatsache, dass alle Kapitel didaktisch aufeinander aufbauen und bei null beginnen. Es bedeutet jedoch keineswegs, dass dieses Buch bloß Einsteigerthemen abdecken würde. Tatsächlich werden Sie auf dem internationalen Markt kaum ein Buch finden, das die PowerShell umfassender und tiefgreifender erklärt.

Weil die Vorkenntnisse unterschiedlich sind, finden Sie am Ende jedes Kapitels einen Nachschlag. Dieser kann aus kniffligen Rätseln und Aufgaben bestehen oder Randaspekte des Kapitelthemas aufnehmen. Sind Sie sich nicht sicher, ob Sie ein Kapitel vielleicht überspringen können, schauen Sie sich diesen Teil an. Ist das, was Sie dort lesen, für Sie ein alter Hut, dann dürfen Sie gern zum nächsten Kapitel überholen. Meistens jedoch werden Sie verblüfft sein, was es zu den einzelnen Themen alles zu entdecken gibt.

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> « vorangestellt. Diese Eingabeaufforderung kann bei Ihnen auch anders aussehen und sollte in den Beispielen natürlich nicht mit eingegeben werden. In den ersten beiden Kapiteln wird außerdem angezeigt, wann Sie ↵ oder andere Tasten drücken müssen. In den folgenden Kapiteln trauen wir Ihnen das dann auch ohne speziellen Hinweis zu.

Die meisten PowerShell-Codebeispiele sind sehr kurz und können mit geringem Aufwand schnell eingetippt werden. Manche Beispiele sind allerdings auch länger, und deshalb finden Sie auf der Begleit-CD für jedes Kapitel eine Datei mit allen Beispielen. Dies sind also keine eigenständigen PowerShell-Skripts, sondern sozusagen Mitschnitte der Eingaben, aus denen Sie den Code, den Sie ausprobieren wollen, auswählen und dann in der PowerShell-Konsole einfügen.

Noch mehr Unterstützung

Falls bei der Arbeit mit diesem Buch Fragen auftauchen oder Sie Anregungen haben, besuchen Sie mich unter *http://www.powershell.com*. Oder senden Sie mir eine E-Mail-Nachricht an *tobias@powershell.com*.

Dieses Buch ist stark geformt worden durch meine jahrelange Arbeit als PowerShell-Trainer für Unternehmen im Mittelstand und Großkundensegment. Dass ich so etwas sehr gern und sehr gründlich tue, werden Sie bestimmt schnell an diesem Buch erkennen. Falls Sie Interesse haben, an einem PowerShell-Training mit mir teilzunehmen, beispielsweise einem Bootcamp, einem Umsteigerseminar oder bei einer In-house-Veranstaltung direkt vor Ort, dann mieten Sie mich einfach. Sie erreichen mich für Anfragen und Details unter *tobias.weltner@email.de*.

Und falls Sie besonders faul sind (oder ökonomisch denken), testen Sie doch mal die (kommerzielle) Entwicklungsumgebung *PowerShellPlus* (*http://www.powershellplus.com*). Sie entstand als Reaktion auf PowerShell-Trainings und ermöglicht Ihnen mit Intellisense-artigen Codevervollständigungen, Debuggern und Lerncentern, die PowerShell

Jetzt wünsche ich Ihnen aber viel Spaß mit der PowerShell!

Herzlichst, Ihr

Dr. Tobias Weltner