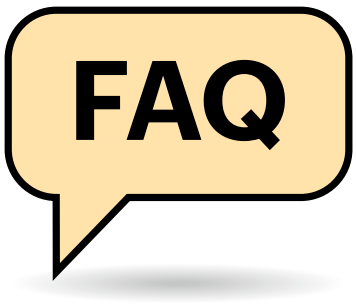


PowerShell

Unter Windows sind viele Aufgaben per Tastatur schneller erledigt als per Mausschubserei. Mit der veralteten Eingabeaufforderung muss man sich dazu schon lange nicht mehr herumplagen: Die PowerShell ist eine moderne und mächtige Alternative.

Von Hajo Schulz

Für wen?

Was genau ist eigentlich die PowerShell? Lohnt es sich, dass ich mich damit beschäftige?

Beliebt ist die PowerShell vor allem bei Systemadministratoren und Power-Usern, die damit zahlreiche Verwaltungs- und Wartungsaufgaben rund um das Betriebssystem schneller und effizienter erledigen als mit grafischen Programmen und Mausbedienung.

An der Oberfläche besteht die PowerShell aus einem Textfenster, in das man Befehle eintippt und deren Ergebnisse in Form von Text entgegennimmt. Der Geschwindigkeitsgewinn gegenüber GUI-Programmen resultiert unter anderem daraus, dass man damit in einem einzigen Fenster praktisch sämtliche anfallenden Admin-Jobs erledigen kann. Entscheidend ist außerdem, dass man sich ständig wiederholende Befehlsfolgen in Skripten zusammenfassen und so in einem Rutsch abarbeiten lassen kann.

Besser als Eingabeaufforderung?

Ihre Beschreibung der PowerShell erinnert mich sehr an die Eingabeaufforderung. Wo ist der Unterschied?

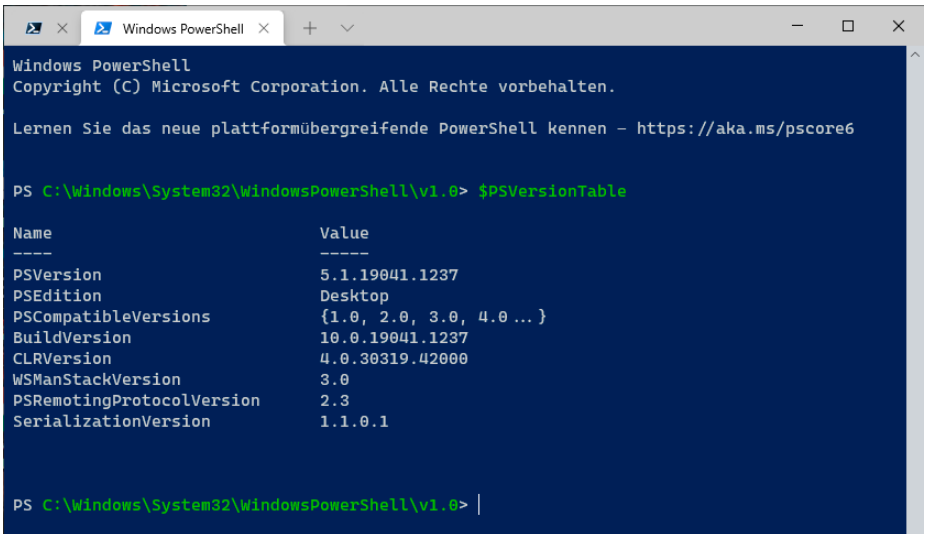
Tatsächlich ähnelt sich der grundsätzliche Umgang mit Eingabeaufforderung und PowerShell. Allerdings besitzt die PowerShell einen wesentlich größeren Befehlsumfang: Externe (Konsolen- und GUI-)Programme lassen sich wie gewohnt durch Eingabe ihres Namens aufrufen. Gängige Befehle zum Navigieren im Dateisystem oder zum Anlegen, Kopieren, Verschieben und Umbenennen von Dateien und Ordnern sind bis auf einige Optionen identisch mit denen in der Eingabeaufforderung. Darüber hinaus kennt die PowerShell Hunderte weiterer Befehle – sogenannte Cmdlets –, mit denen man nicht nur auf das lokale System, sondern auch remote auf andere Rechner im Netzwerk zugreifen kann.

Ein deutlicher Unterschied zeigt sich auch bei der eingebauten Skriptsprache: Während man der Batch-Sprache der klassischen Eingabeaufforderung ihr Alter mittlerweile doch deutlich anmerkt, kommt die PowerShell mit einer Skriptsprache, die diese Bezeichnung auch verdient. Übliche Kontrollstrukturen wie `if`-Abfragen und Schleifen gibt es ebenso wie etwa Zeichenkettenverarbeitung mit regulären Ausdrücken. Außerdem stehen der PowerShell sämtliche Klassen und Funktionen der Programmierumgebung .NET zu Gebote, über die sich sogar Windows-Systemfunktionen aufrufen lassen.

Zwei Versionen

Wenn ich die PowerShell starte, die in Windows eingebaut ist, weist sie mich jedes Mal darauf hin, dass ich doch mal die „neue plattformübergreifende PowerShell“ ausprobieren solle. Was hat es damit auf sich?

Von der PowerShell existieren zwei voneinander unabhängige Entwicklungszweige: Bestandteil von Windows ist die PowerShell 5; offiziell heißt sie bei Microsoft „Windows PowerShell“. Unabhängig davon ist – ebenfalls unter Microsofts Führung – eine komplett quelloffene Version der PowerShell entstanden, die anfangs „PowerShell Core“ hieß, mittlerweile aber einfach unter „PowerShell“ firmiert. Derzeit ist Version 7.1 aktuell. Diese Version der PowerShell läuft nicht nur unter Windows, sondern auch unter macOS und Linux.



Die schlichte Bedienoberfläche der PowerShell lässt kaum etwas von ihrer Mächtigkeit erkennen.

Microsoft hat angekündigt, die Windows PowerShell auf absehbare Zeit noch mit Windows auszuliefern und weiterhin mit Sicherheits-Updates zu versorgen. Neue Funktionen soll es dort aber nicht mehr geben; echte Weiterentwicklung findet nur noch in der Open-Source-Variante statt. Gerade wenn Sie sich neu in die PowerShell einarbeiten wollen, ist letztere also die bessere Wahl.

Woher nehmen?

Wie komme ich an eine aktuelle PowerShell?

Die Windows PowerShell ist schon seit etlichen Betriebssystemversionen in Windows enthalten, sowohl in den Desktop-Ausgaben als auch in Windows Server. Starten können Sie sie einfach über das Windows+X-Menü oder mit der Eingabe `powershell` im Windows+R-Dialog oder in das Suchfeld des Startmenüs.

Für die PowerShell 7 gibt es zwei Download-Quellen: Zum einen ist sie im Windows-eigenen Microsoft Store verfügbar, wo Sie sie einfach wie eine App herunterladen und installieren können. Zum anderen gibt es installierbare Pakete – auch für andere Plattformen als Windows – auf GitHub (siehe ct.de/yufx). Für Windows bekommen Sie unter den „Releases“ wahlweise ein MSI-Paket, das Pfade und Verknüpfungen automatisch einrichtet, oder ein Zip-Archiv, bei dem Sie alles von Hand einstellen können (und müssen). Insgesamt die wenigste Arbeit macht aber die Installation aus dem Store, weil sie sich auch automatisch um Aktualisierungen kümmert. Achtung: Sowohl im Store als auch auf GitHub gibt es neben der offiziell freigegebenen Version auch eine „PowerShell Preview“ zum Testen der allerneuesten Entwicklungen; für den produktiven Einsatz ist die aber nicht zu empfehlen. Der Befehl zum Starten der PowerShell 7 aus einer Eingabeaufforderung oder einem Suchfenster lautet `pwsh`.

Scripting kaputt?

Ich habe auf einer Webseite ein bisschen über PowerShell-Scripting gelesen. Allerdings gelingt es mir nicht, die dort beschriebenen Beispiele nachzuvollziehen: Der Versuch, selbst das einfachste Skript auszuführen, führt zu einer Fehlermeldung, die besagt, dass das Ausführen von Skripten auf diesem System deaktiviert sei. Was ist da los?

Dieses Problem tritt praktisch nur mit der Windows PowerShell auf: Microsoft konfiguriert sie aus Sicherheitsgründen im Auslieferungszustand so, dass sie das Ausführen jeglicher Skripte verweigert.

Beheben lässt sich das, indem Sie die Windows PowerShell einmal mit Administratorrechten starten und den Befehl

```
Set-ExecutionPolicy RemoteSigned
```

ausführen. Er bewirkt, dass ab sofort alle Skripte erlaubt sind, sofern sie eine vertrauenswürdige Signatur tragen oder nicht als aus dem Internet stammend gekennzeichnet sind. Mit `Bypass` statt `RemoteSigned` schalten Sie die Skriptprüfung komplett auf Durchzug, mit `Restricted` stellen Sie den Auslieferungszustand wieder her. Dieselben Befehle und Optionen gelten auch für die PowerShell 7, aber dort ist `RemoteSigned` voreingestellt. Weitere Erklärungen zu dem Thema stehen im Artikel „about_Execution_Policies“ in der Online-Doku zur PowerShell (siehe ct.de/yufx).

Ist die PowerShell böse?

Ich habe gelesen, Hacker benutzen die PowerShell gerne als Werkzeug für Angriffe. Ist sie ein Sicherheitsrisiko?

Nein. Wie alle Tools, die eigenen Code ausführen, lässt sie sich natürlich auch für böse Absichten missbrauchen. Grundsätzlich ist sie aber nicht unsicherer als Batch-Dateien oder jegliche Programme aus dem Internet.

Wie bei Programmen gilt aber auch bei PowerShell-Skripten und -Modulen: Allzu sorglos heruntergeladen und – womöglich noch mit Administratorrechten – ausgeführt können sie durchaus eine Gefahr darstellen. Sie sollten nur verwenden, was aus vertrauenswürdiger Quelle stammt. Bei Skripten haben Sie sogar noch den Vorteil, dass es sich um ganz normale Textdateien handelt. Sie können sie also mit jedem Texteditor lesen und prüfen, ob sie das tun, was ihr Entwickler verspricht.

Erweitern

Ich vermisse einen Befehl, mit dem ich XY erledigen kann. Kennen Sie einen?

Für die Suche nach PowerShell-Befehlen ist das Kommando `Get-Command` oder kurz `gcm` zuständig. Außerdem ist hilfreich, dass Microsoft Cmdlets und Funktionen durchgängig nach dem Schema `Verb-Nomen` benennt. So liefert `Get-Partition` Informationen zu Festplatten- oder SSD-Partitionen und `Suspend-VM` hält eine (Hyper-V-)VM an. Mit Suchen nach dem Muster `gcm -Verb Enable` oder `gcm -Noun Certificate` suchen Sie nach bestimmten Verben beziehungsweise Nomen. Alle Varianten verarbeiten Jokerzeichen, wie Sie sie von Dateien her kennen, also etwa `gcm -Noun Bitlocker*`. Eine Liste der von Microsoft verwendeten Verben liefert `Get-Verb`.

In der Ausgabe von `Get-Command` gibt es eine Spalte „Source“, die das Modul benennt, aus dem der jeweilige Befehl stammt. So finden Sie verwandte Befehle zu einem Suchtreffer, indem Sie die Suche nach dem Muster `gcm -Module Storage` auf ein bestimmtes Modul einschränken.

Selbst wenn Sie auf diese Weise bei den eingebauten Befehlen nicht fündig werden, sollten Sie noch nicht aufgeben: Die PowerShell lässt sich um weitere Module erweitern und die Chancen stehen nicht schlecht, dass es so ein Modul auch für Ihr Problem gibt. Die größte Quelle für Module von Microsoft selbst oder aus der Benutzer-Community ist die PSGallery. Deren Inhalt können Sie wahlweise per Browser auf der Seite powershellgallery.com oder direkt in der PowerShell mit dem Befehl `Find-Module` durchsuchen. Letzterer verdaut ebenfalls Jokerzeichen: Wenn Sie etwa in der PowerShell Mails lesen wollen, lautet der passende Suchbefehl `Find-Module *imap*`. Ausführlichere Modulbeschreibungen und meist auch mehr Suchergebnisse liefert aber die Webseite. Haben Sie eine passende Erweiterung gefunden, können Sie sie mit `Install-Module Modulname` installieren. Wenn Sie noch `-Scope CurrentUser` an den Befehl anhängen, landet das Modul in Ihrem Benutzerprofil und Sie brauchen zur Installation keine Administratorrechte. *(hos@ct.de)*

Downloads und Dokumentation:
ct.de/yufx