Przedmowa

Dziesięć lat po wydaniu programu Windows PowerShell 1.0, zespół PowerShell ogłosił wersję PowerShell Core 6. Prace nad PowerShell Core 6 rozpoczęły się kilka lat wcześniej i to właśnie wtedy zostałem kierownikiem inżynierii PowerShell. Na początku nie było łatwo, zwłaszcza w zakresie zgodności z Windows PowerShell, ale wraz z wydaniem PowerShell 7 rozpoczynamy oficjalnie nowy rozdział technologii PowerShell, w którym to PowerShell 7 może być wykorzystywany jako zamiennik dla (lub używany jednocześnie z) Windows PowerShell 5.1.

PowerShell 7 reprezentuje przyszłość powłoki PowerShell z uwagi na wprowadzenie trzech dużych zmian:

- **Jeden język automatyzacji dla Windows, Linux i macOS:** Windows PowerShell został dobrze przyjęty przez społeczność Windows. W miarę stopniowego przechodzenia branży IT w stronę chmury, pozwoliło to PowerShell stać się językiem pozwalającym korzystać z tej chmury. Usprawnienia poczynione w poleceniach sieciowych upraszczają wywoływanie API REST. Wczesne partnerstwa z Azure, Amazon Web Services, Google Compute Cloud i VMware zagwarantowały dostępność poleceń PowerShell dla dowolnego rozwiązania chmury na dowolnej platformie. Przy okazji PowerShell stał się świetną i użyteczną powłoką, bez względu na to, czy korzystamy z Windows, Linux czy macOS.
- **Przejście do open source:** Była to duża zmiana w sposobie pisania oprogramowania i angażowania się w społeczność. Możemy być teraz znacznie bardziej transparentni co do naszych planów, a także akceptować pomoc ze strony społeczności w celu rozwiązania problemów lub wprowadzania funkcji, które w innym przypadku nie byłyby priorytetem dla zespołu. Przy około 50 procentach żądań ściągnięcia pochodzących od społeczności, przyszłość PowerShella naprawdę jest projektem rozwijanym przez społeczność!

- **Wczesne wykorzystanie .NET Core:** Podczas przechodzenia z .NET Framework na .NET Core (a teraz po prostu .NET) nie zabrakło pewnych wyzwań. Zgodność z modułami programu Windows PowerShell była początkowo największym problemem. .NET przebył długą drogę w rozwiązywaniu problemu zgodności, wprowadzając z powrotem wiele API zapewniających zgodność PowerShell z istniejącymi modułami. Dodatkowo .NET Core zawiera pewne znaczące usprawnienia wydajności, które sprawiają, że istniejące skrypty i moduły PowerShell działają szybciej bez żadnych modyfikacji!

Misją powłoki PowerShell jest ułatwienie użytkownikom korzystania z zasobów obliczeniowych. W programie PowerShell 7 uwzględnia to różne platformy, takie jak Windows, Linux i macOS, ale również nowe architektury, takie jak ARM32 i ARM64. Przy modułach PowerShell dostępnych dla głównych chmur publicznych i prywatnych możemy uzyskać większą produktywność w scenariuszach hybrydowych lub korzystających z wielu chmur. Wciąż podtrzymujemy „świętą przysięgę" PowerShella – że nauka nowego języka jest trudna, ale dzięki inwestycji w naukę PowerShella będziemy nadal pomagać użytkownikom rozszerzać ich wpływ i podnosić swoją produktywność, jak choćby za pomocą bezserwerowych funkcji w formie usługi czy platformy Jupyter Notebooks. Jestem podekscytowany nową fazą powłoki PowerShell, którą rozpoczęliśmy wraz z wydaniem PowerShell 7, ale dla mnie osobiście jest to dopiero początek i wiele świetnych rzeczy dopiero przed nami! Ta książka autorstwa Thomasa jest świetnym sposobem na rozpoczęcie pracy z PowerShell 7 przy wykorzystaniu istniejącego doświadczenia zdobytego w pracy z programem Windows PowerShell.

Thomas Lee jest członkiem społeczności PowerShell znacznie dłużej niż ja byłem kierownikiem inżynierii PowerShell. Niektórych rzeczy o PowerShell dowiedziałem się czytając jego artykuły i posty na jego blogu. W czasie gdy zespół PowerShell robił postępy w pracach nad wydaniem programu PowerShell 7, Thomas był z nami przez cały czas, promując, nauczając i informując społeczność o wszystkich tych świetnych rzeczach, które przynosi ze sobą PowerShell 7. Najważniejszym aspektem, który zdecydował o sukcesie PowerShell, była jego społeczność, a Thomas był jej nieodłączną częścią.

Steve Lee
Główny kierownik inżynierii oprogramowania
Zespół PowerShell

Wprowadzenie

Cześć i dziękuję za zakup tej książki. Gdy w 2003 roku Jeffrey Snover na konferencji Professional Developers Conference w Los Angeles przedstawił po raz pierwszy projekt Monad, który został później przemianowany na Windows PowerShell, siedziałem wówczas na widowni. Byłem podekscytowany tym, co wtedy zobaczyłem i usłyszałem. Był to przełomowy moment w mojej karierze.

Dzisiaj mamy nową wersję powłoki PowerShell, PowerShell 7, którą znowu możemy się ekscytować. Zespół rozwojowy PowerShell wraz z fantastyczną społecznością PowerShell wyniósł ten produkt na nowy poziom. Ja jestem nadal podekscytowany i mam nadzieję, że Ty również.

Chciałbym, abyś przed zagłębieniem się w treść tej książki poświęcił kilka minut na przeczytanie tego krótkiego wstępu, w którym to wyjaśniam moją motywację do napisania tej książki, jej strukturę oraz sposób korzystania z przedstawionych w niej skryptów PowerShell z użyciem maszyn wirtualnych Hyper-V.

Książka składa się 10 rozdziałów. Pierwszy rozdział poświęcony jest konfigurowaniu powłoki PowerShell 7 w naszym środowisku. Rozdział 2 omawia problem zgodności z Windows PowerShell i pokazuje, w jaki sposób PowerShell 7 stara się go rozwiązać. Pozostałych osiem rozdziałów dotyczy różnych funkcji systemu Windows Server i sposobu zarządzania nimi z poziomu powłoki PowerShell 7.

Rozdział 1: Konfigurowanie środowiska PowerShell 7: W tym rozdziale przyglądamy się sposobom instalacji powłoki PowerShell 7 i edytora VS Code. VS Code zastępuje starszą powłokę Windows PowerShell ISE. Przedstawione w tej książce zrzuty ekranu pokazują kod PowerShell uruchamiany w edytorze VS Code. W środowisku produkcyjnym moglibyśmy zrezygnować z używania na naszym serwerze VS Code (lub dowolnego innego narzędzia z graficznym interfejsem użytkownika) i zamiast tego korzystać z konsoli PowerShell 7 ze zdalnym edytowaniem tekstu.

Rozdział 2: Zgodność PowerShell 7 z Windows PowerShell: Zgodność z programem Windows PowerShell jest istotnym celem i znaczącym zadaniem inżynierskim. Ten

rozdział omawia ten problem zgodności, a przy tym dostarcza pewnych informacji na temat modułów. W dalszej części wyjaśniamy w nim sposób działania zgodności wstecznej i omawiamy niewielką część powłoki Windows PowerShell, z której nie można korzystać w PowerShell 7.

Rozdział 3: Zarządzanie Active Directory: Active Directory jest sercem sieci niemal każdej organizacji. Ten rozdział pokazuje, w jaki sposób możemy wdrożyć i zarządzać usługą Active Directory, wliczając w to tworzenie lasów i domen, jak również łączenie lasów za pomocą zaufania między lasami. Rozdział ten przedstawia również sposób zarządzania użytkownikami, komputerami, grupami i innymi obiektami Active Directory.

Rozdział 4: Zarządzanie siecią: Ten rozdział dotyczy zarządzania siecią w PowerShell 7. Sprawdzamy tu konfigurację kart sieciowych, a także instalujemy usługi DNS i DHCP.

Rozdział 5: Zarządzanie magazynem danych: Magazynowanie danych jest kluczowym aspektem każdego systemu komputerowego. Musimy gdzieś przechowywać nasze pliki i inne dane. Ten rozdział omawia zarządzanie dyskami i woluminami/partycjami, jak również korzystanie z modułu zewnętrznego do zarządzania uprawnieniami NTFS. Rozdział ten porusza temat funkcji Storage Replica do replikowania magazynu danych, także w kontekście odzyskiwania po awarii. Na koniec omawia on korzystanie z menedżera File Server Resource Manager do zarządzania przydziałami plików i osłoną plików.

Rozdział 6: Zarządzanie udostępnionymi danymi: Gdy mamy już skonfigurowane dyski jako woluminy i partycje, a przy tym ustawiliśmy odpowiednie uprawnienia, musimy udostępnić te dane w obrębie sieci. Ten rozdział pokazuje, w jaki sposób możemy skonfigurować serwer plików SMB, a także utworzyć i zabezpieczyć na nim udziały plików SMB. Omawia on również proces konfiguracji obiektu docelowego iSCSI i jego wykorzystanie w celu wdrożenia wysokoodpornego serwera plików skalowalnego w poziomie.

Rozdział 7: Zarządzanie drukowaniem: Drukowanie jest podstawową funkcją systemu Windows dostępną od samego początku jego istnienia. Ten rozdział pokazuje, w jaki sposób możemy skonfigurować i zarządzać serwerem wydruku. Dowiemy się z niego, jak dodać drukarkę i sterowniki drukarki, jak wydrukować stronę testową i jak skonfigurować pulę drukowania.

Rozdział 8: Zarządzanie Hyper-V: Hyper-V jest podstawowym produktem wirtualizacji firmy Microsoft. Ten rozdział omawia konfigurację i zarządzanie Hyper-V oraz tworzenie i zarządzanie maszynami wirtualnymi Hyper-V. Porusza on również temat maszyn wirtualnych, a także przenoszenia i replikacji magazynu maszyny wirtualnej – ważne tematy w dzisiejszym świecie ukierunkowanym na tego rodzaju maszyny.

Rozdział 9: Korzystanie z WMI z użyciem poleceń CIM: Windows Management Instrumentation jest funkcją systemu Windows dostępną od czasu wydania Windows NT 4.0. WMI zapewnia dostęp do informacji o naszym systemie i pozwala nam zarządzać

wybranymi aspektami tego systemu. WMI oferuje dostęp do funkcjonalności systemu Windows, której nie uzyskamy za pomocą poleceń PowerShell. Rozdział ten omawia komponenty WMI i pokazuje, w jaki sposób możemy dowiedzieć się czegoś więcej. Wyjaśniamy w nim również sposób zarządzania zdarzeniami WMI i konfigurowania trwałej obsługi zdarzeń w celu zarządzania krytycznymi zdarzeniami bezpieczeństwa.

Rozdział 10: Raportowanie: Wiedza na temat stanu naszej infrastruktury IT jest niezbędna do zarządzania naszym środowiskiem komputerowym. Ten rozdział demonstruje nam, w jaki sposób możemy używać PowerShell 7 do pozyskiwania informacji na temat naszej infrastruktury. Rozdział ten omawia proces tworzenia raportów o użytkownikach i komputerach Active Directory, raportów o systemie plików z użyciem menedżera FSRM, także raportów dotyczących użycia drukarek i hostów z maszynami wirtualnymi Hyper-V. W rozdziale tym omawiany jest również temat rejestrowania wydajności i tworzenie alertów w celu przechwytywania szczegółowych informacji o wydajności oraz tworzenia rozbudowanych raportów i wykresów pokazujących wydajność naszej infrastruktury.

Napisałem tę książkę, aby pokazać profesjonalistom IT, że przejście do PowerShell 7 jest łatwe i warte poświęconego czasu. Podobnie jak po przeprowadzce do nowego domu, w PowerShell 7 niektóre rzeczy będą nieco inne. Ale gdy się już przyzwyczaimy, raczej nie będziemy chcieli już wracać. Wraz z edytorem VS Code, powłoka PowerShell 7 jest po prostu lepsza. Mam nadzieję, że każdy rozdział tej książki będzie w stanie to pokazać.

Ta książka zakłada, że czytelnik jest profesjonalistą IT chcącym nauczyć się wykorzystywać PowerShell 7 w jak najlepszy sposób. Może to być czynny administrator, konsultant lub kierownik – wymagana jest jedynie wiedza o funkcjach systemu Windows Server, wiedza z zakresu własnej dziedziny, a także znajomość samego programu Windows PowerShell.

Książka omawia różne podstawowe funkcje systemu Windows, wliczając w to Active Directory, File Services Resource Manager, WMI, drukowanie i więcej. Każdy rozdział opisuje obszar danej funkcji oraz komponenty, z którymi wchodzimy w interakcję. Następnie rozdział pokazuje nam, w jaki sposób możemy użyć PowerShell 7 do wdrożenia, zarządzania i korzystania z danej funkcji.

Nie jest możliwe omówienie w tej książce (a w zasadzie w żadnej książce o PowerShell) każdego aspektu każdej funkcji systemu Windows. Jak mawia Jeffrey Snover, „Aby dostarczyć, trzeba wybrać", i mam nadzieję, że wybrałem mądrze. Zawarłem również odnośniki do stron, na których można znaleźć więcej informacji. Bez problemu można do mnie napisać maila i przesłać mi swoją opinię (`DoctorDNS@Gmail.Com`).

Ta książka zawiera różne skrypty PowerShell 7, których możemy użyć do zarządzania pewnymi aspektami systemu Windows. Skrypty dostępne są do pobrania ze strony wydawnictwa Wiley lub z mojego repozytorium GitHub pod adresem *github.com/doctordns/Wiley20*. Gdyby ktoś odkrył jakiś problem z dowolnym z tym skryptów lub

znalazł błąd w dokumentacji, proszę o zgłoszenie tego na moim repozytorium GitHub (*github.com/doctordns/Wiley20/issues*).

Głównym celem podczas tworzenia tej książki było pokazanie, jak łatwo możemy używać PowerShell 7 do zarządzania infrastrukturą Windows Server. Istnieją pewne różnice w sposobie instalacji tej powłoki, a do tego musimy przywyknąć do pracy z edytorem VS Code, który zastąpił powłokę ISE. W czasie pisania odkryłem kilka problemów związanych ze zgodnością z Windows PowerShell i omawiam je w rozdziale 2. Czas już przejść do PowerShell 7.

Skrypty i treść tej książki stworzyłem przy wykorzystaniu maszyn wirtualnych Hyper-V z systemem Windows Server 2019 Datacenter. Aby z tej książki i jej skryptów można było wynieść jak najwięcej, warto samemu zbudować wszystkie prezentowane tu maszyny wirtualne i używać ich do testowania skryptów. Oczywiście zamiast maszyn wirtualnych można używać hostów fizycznych, ale maszyny wirtualne są łatwiejsze w użyciu. Dla osób, które mogą nie mieć pod ręką wymaganego sprzętu, zawarłem w tej książce zrzuty ekranu pokazujące wyniki uzyskiwane w każdym z kroków każdego skryptu. Stworzyłem również odpowiedni zestaw skryptów ułatwiających budowę tych maszyn wirtualnych. Można je pobrać z mojego GitHuba. Więcej informacji na temat tych skryptów i sposobów ich pozyskiwania można znaleźć w rozdziale 1.

Od samego początku jednym z najbardziej imponujących aspektów produktu PowerShell była jego rozbudowana i aktywna społeczność. Tysiące ludzi z całego świata, którzy kochają PowerShell i dostarczyli nam już wiele wspaniałych rzeczy: tweety, wpisy na forach, artykuły na blogach, skrypty, moduły, strony internetowe i wiele więcej. Spora liczba funkcji dostępnych w PowerShell 7 pochodzi właśnie od społeczności.

Jeśli ktoś ma jakikolwiek problem z dowolnym aspektem dowolnego komponentu tej książki – lub dowolnym aspektem systemu Windows – pomoc w rozwiązaniu każdego z nich będzie mógł znaleźć w internecie.

Praktycznie każda strona mediów społecznościowych, na której gromadzą się osoby techniczne, będzie oferować jakiś rodzaj treści lub pomocy związanej z PowerShell. Zapraszam do odwiedzenia forum PowerShell na stronie Spiceworks, gdzie sam jestem moderatorem (*community.spiceworks.com/programming/powershell*).

Ciesz się tą książką i ciesz się PowerShell 7.

Fare thee well now,
Let your life proceed by its own design.
Nothing to tell now,
Let the words be yours; I'm done with mine.
„Cassidy", John Barlow/Robert Weir

Thomas Lee
Czerwiec 2020
Cookham, Anglia

ROZDZIAŁ

1

Konfigurowanie środowiska PowerShell 7

Pierwsze wersje Windows PowerShell dla systemów Windows XP i Windows Server 2008 oferowane były w ramach dostępnych do pobrania plików, które instalowane były przez użytkownika. Dzisiaj systemy Windows Server i Windows 10 dostarczane są z domyślnie zainstalowanym środowiskiem Windows PowerShell w wersji 5.1, które w tej książce nazywam po prostu Windows PowerShell, aby odróżnić go od PowerShell 7 (i środowiska Windows PowerShell Integrated Scripting Environment). Windows PowerShell oferuje wiele różnych poleceń pozwalających na podstawową administrację systemem Windows.

W czasie pisania tej książki PowerShell 7 nie jest dostarczany z systemem Windows. W przyszłości zespół PowerShell może zacząć dostarczać PowerShell 7 w postaci składnika systemu Windows, ale dopóki tak się nie stanie, będziemy musieli sami pobierać i instalować to środowisko.

Środowisko graficzne Windows PowerShell Integrated Scripting Environment (ISE) nie obsługuje PowerShell 7. Profesjonaliści IT, którzy potrzebują dobrego interaktywnego środowiska programistycznego dla PowerShell mogą korzystać z programu Visual Studio Code (VS Code) – darmowego narzędzia, które również w prosty sposób możemy pobrać i zainstalować. VS Code dostarczany jest z szeregiem rozszerzeń, które oferują znacząco ulepszone możliwości programistyczne dla profesjonalistów IT (i nie tylko).

We wcześniejszych wersjach powłoki PowerShell znaczna większość poleceń dostarczana była wraz z systemem Windows, przy czym wiele z nich można było uzyskać po zainstalowaniu jakiejś aplikacji (takiej jak Exchange Server) lub włączeniu wybranej funkcji systemu Windows. W przypadku PowerShell 7 głównym źródłem modułów/poleceń, za pomocą których możemy wykonywać różne zadania administracyjne, jest galeria programu PowerShell. Aby móc w pełni korzystać ze wszystkich możliwości galerii PowerShell, należy zaktualizować moduł PowerShellGet.

Co nowego w PowerShell 7

PowerShell 7 jest najnowszą wersją powłoki PowerShell. Zespół rozwojowy PowerShell wydał PowerShell 7.0 w marcu 2020 roku. Zanim ktoś przeczyta tę książkę, zespół rozwojowy z pewnością zdąży już wydać nową pomniejszą aktualizację. PowerShell 7 ma wiele nowych kluczowych funkcji, które z pewnością będą przydatne dla profesjonalistów IT.

Jeśli jesteśmy już zaznajomieni i potrafimy wykorzystywać Windows PowerShell do zarządzania naszymi systemami Windows, będziemy mogli bezpośrednio przenieść całą naszą dotychczasową wiedzę do nowego środowiska. Potrzebujemy uzyskać pomoc dla jakiegoś polecenia? Wystarczy wpisać `Get-Help` w wierszu polecenia PowerShell. Podstawowa architektura PowerShell pozostaje taka sama i zawiera wiele wewnętrznych zmian i znaczących usprawnień, ale też kilka problemów z kompatybilnością.

Patrząc z perspektywy profesjonalisty IT z wiedzą na temat zarządzania systemem Windows z poziomu Windows PowerShell, kluczowe zmiany są następujące:

- **Opracowane na nowo polecenia cmdlet, oparte na technologii .NET Core i dostępne w wersji źródłowej w witrynie GitHub:** Możemy teraz czytać kod, a nawet pomagać rozszerzać dowolne polecenia w PowerShell 7. Oznacza to również, że polecenia zostały napisane w taki sposób, aby wykorzystywały .NET Core – co doprowadziło do powstania kilku niewielkich problemów ze zgodnością.
- **Solidna warstwa kompatybilności:** Za jej pomocą uzyskujemy dostęp do modułów Windows PowerShell, które nie działają bezpośrednio w PowerShell 7. Oznacza to, że niemal wszystkie (poza niewielką liczbą) moduły Windows PowerShell 5.1 są dostępne i działają w PowerShell 7. Rozdział 2., „Zgodność PowerShell 7 z Windows PowerShell", opisuje tę warstwę zgodności bardziej szczegółowo, wyjaśnia sposób jej działania i jej ograniczenia oraz dostarcza rozwiązania obejściowe.
- **Znaczące usprawnienia w zakresie wydajności:** Podczas przenoszenia modułów Windows PowerShell do PowerShell 7, zespół rozwojowy był w stanie przejrzeć ich kod i dostarczyć dla nich usprawnienia w zakresie wydajności. Przykładowo przetwarzanie olbrzymich kolekcji za pomocą `Foreach` jest teraz znacznie szybsze. Polecenie `Foreach-Object` ma teraz przełącznik `-Parallel`, który pozwala nam wykonywać bloki skryptu równolegle, co może znacząco skrócić czas wykonywania, zwłaszcza na potężniejszych wielordzeniowych i wieloprocesorowych serwerach.
- **Nowe operatory języka PowerShell: PowerShell oferuje teraz trzy nowe zestawy operatorów: operator trójargumentowy** (`a ? b : c`), operatory łańcucha potoku (`||` i `&&`), oraz nullowe operatory koalescencji (`??` i `??=`). Operatorów tych – zaimplementowanych również w innych powłokach, takich jak Bash czy Zsh – możemy teraz używać również w PowerShell 7.

- **Uproszczone widoki błędów:** W programie Windows PowerShell komunikaty o błędach były bardzo przydatne i dostarczały wielu cennych informacji. Jednak w większości przypadków zawierały one więcej informacji niż to było potrzebne. Komunikaty o błędach w PowerShell 7 są teraz znacznie bardziej zwięzłe. A gdy *faktycznie* potrzebujemy tych dodatkowych informacji, za pomocą polecenia `Get--Error` możemy pozyskać pełne szczegóły na temat dowolnego błędu. Jeśli zmiennej `$ErrorView` nadamy wartość `NormalView`, będziemy mogli wyświetlać komunikaty o błędach w starszym stylu Windows PowerShell. Jeśli z kolei nadamy jej wartość `CategoryView`, wyświetlane będą wyłącznie kategorie błędów.
- **Funkcje eksperymentalne:** Zespół PowerShell zaimplementował zestaw nowych funkcji w fazie eksperymentalnej. Jeśli chcemy, możemy korzystać z tych funkcji. Daje nam to możliwość wypróbowania nowych rzeczy i przesłania naszej opinii o nich.
- **Automatyczne powiadamianie o nowej wersji:** W czasie pisania tej książki nie jest świadczone wsparcie dla PowerShell 7 za pośrednictwem sklepu Windows Store czy poprzez Windows Update. Oznacza to, że aktualizacjami musimy zarządzać sami. Komunikaty te powiadamiają nas o dostępności nowej wersji PowerShell do pobrania.
- **`Set-Location` obsługuje teraz - i + na ścieżce:** W przypadku korzystania z polecenia `Set-Location` do resetowania naszego bieżącego katalogu roboczego, możemy skorzystać z przełącznika `-Path "-"` w celu poinstruowania polecenia `Set-Location`, że chcemy przenieść się do ostatnio odwiedzonego folderu. Po cofnięciu się możemy ustawić lokalizację za pomocą `+`, aby przemieścić się ponownie do przodu.
- **Zdolność do bezpośredniego wywoływania zasobu DSC:** PowerShell 7 nie obsługuje konfiguracji żądanego stanu (Desired State Configuration, DSC), a tym samym serwerów ściągania/raportów, menedżera lokalnej konfiguracji i tak dalej. Możemy jednak ręcznie wywoływać zasoby DSC na danym hoście, co stanowi częściowe rozwiązanie.

Zaprezentowane w tej książce fragmenty kodu PowerShell 7 wykorzystują i demonstrują większość z tych nowych funkcji. Więcej informacji na temat dowolnej z nich – wliczając w to przypadki użycia oraz przykłady – można znaleźć wyszukując je po prostu w sieci, jako że społeczność PowerShell stworzyła już pokaźną ilość treści opisujących te funkcje. Dostępnych jest wiele takich wpisów ogólnych, jak choćby artykuł pod adresem *https://www.thomasmaurer.ch/2020/03/whats-new-in-powershell-7-check--it-out*. Istnieją również bardziej szczegółowe artykuły, które omawiają konkretne nowe funkcje, jak na przykład artykuł *tfl09.blogspot.com/2020/03/introduction-and-background-welcome-to.html*, który dostarcza szczegóły na temat nowych operatorów trójargumentowych i łańcucha potoku.

Systemy używane w tej książce i w tym rozdziale

Ta książka omawia sposób wykorzystywania programu PowerShell 7 do wykonywania szerokiego zakresu zadań, wliczając w to przypisywanie uprawnień do udziału sieciowego, zbieranie i raportowanie danych na temat wydajności czy instalowanie i konfigurowanie usługi Active Directory. W celu zademonstrowania tych oraz wielu innych zadań książka ta wykorzystuje zbiór hostów oraz dwie domeny: Reskit.Org i Kapoho .Com. Systemy te można zaimplementować na dwa różne sposoby.

Skrypty tworzące maszyny wirtualne serwerów

Skrypty w tej książce zakładają, że dysponujemy zestawem serwerów gotowych do skonfigurowania. Jeśli ktoś chce, może skonfigurować każdy z tych komputerów przy użyciu sprzętu fizycznego, jednak znacznie prostszą alternatywą jest zbudowanie niezbędnych serwerów z maszyn wirtualnych Hyper-V z użyciem skryptów dostępnych na stronie *github.com/doctordns/ReskitBuildScripts*. To repozytorium GitHub zawiera plik `README.MD` (*github.com/doctordns/ReskitBuildScripts/blob/master/README.md*), który wyjaśnia, jak za pomocą tych skryptów można zbudować własną farmę maszyn wirtualnych.

Skrypty te wykorzystywane są do tworzenia maszyn wirtualnych na potrzeby różnych szkoleń oraz dla innych książek. Nie ma potrzeby tworzenia *wszystkich* maszyn wirtualnych. We wprowadzeniu do każdego rozdziału podawane są konkretne maszyny wirtualne wykorzystywane w danym rozdziale.

Te skrypty tworzą maszyny wirtualne, ale należy zwrócić uwagę na kolejność, w jakiej są one tworzone, oraz na lokalizację, w której przechowywane są maszyny wirtualne, wirtualne dyski twarde i tak dalej.

Skrypty tworzą maszyny wirtualne z podstawową obsługą sieci (jedna karta sieciowa), przy czym możliwe jest ich rozszerzenie poprzez dodanie do nich kolejnych kart sieciowych. Skrypty tworzą maszyny wirtualne na potrzeby tej książki z użyciem konkretnego zestawu adresów sieciowych. Dokument *github.com/doctordns/ReskitBuildScripts/blob/master/ReskitNetwork.md* pokazuje szczegóły hostów sieciowych i adresy IP.

Dostęp maszyn wirtualnych do internetu

Maszyny wirtualne (lub hosty, jeśli wykorzystujemy fizyczne komputery) wymagają dostępu do internetu. Wszystkie maszyny wirtualne znajdują się w sieci IPv4 10.10.10.0/24 zaimplementowanej jako wewnętrzna sieć Hyper-V za pomocą wewnętrznego wirtualnego przełącznika Hyper-V. Istnieją dwa mechanizmy, za pomocą których możemy to osiągnąć.

Możemy wyposażyć każdą maszynę wirtualną w drugą wirtualną kartę sieciową. Kartę tę należy skonfigurować w taki sposób, aby wykorzystywała zewnętrzny

przełącznik powiązany z zewnętrzną kartą sieciową hosta naszej maszyny wirtualnej. Jest to proste rozwiązanie, które można szybko skonfigurować.

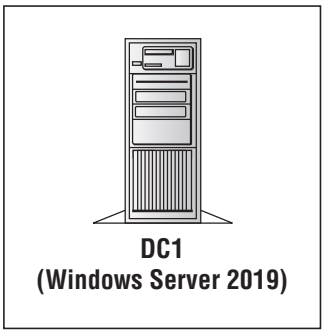
Alternatywą jest skonfigurowanie maszyny wirtualnej z systemem Windows Server z usługą Routing and Remote Access. W maszynie tej konfigurujemy dwie karty sieciowe (jedną wewnętrzną, drugą zewnętrzną) i konfigurujemy routing między podsiecią 10.10.10.0/24 używaną przez maszyny wirtualne wykorzystywane w tej książce a internetem.

Systemy używane w tym rozdziale

W tym rozdziale użyjemy PowerShell 7 do zarządzania różnymi aspektami sieci. Skrypty z tego rozdziału wykorzystują jeden z nich.

DC1: Na potrzeby tego rozdziału, DC1 jest po prostu hostem z systemem Windows Server 2019.

Rysunek 1.1 przedstawia systemy wykorzystywane w tym rozdziale.



Rysunek 1.1 Systemy używane w tym rozdziale

Na potrzeby tego rozdziału, `DC1` jest maszyną wirtualną z zainstalowanym systemem Windows Server 2019 Datacenter. Serwer ten można utworzyć za pomocą wspomnianych wcześniej skryptów. W rozdziale 3., „Zarządzanie Active Directory”, wypromujemy ten serwer do roli kontrolera domeny.

Instalowanie PowerShell 7

Firma Microsoft nie oferuje programu PowerShell 7 z żadną wersją systemu Windows, taką jak Windows 10, Windows Server 2019 czy dowolnym innym wspieranym wydaniem klienckim lub serwerowym Windows. Aby zainstalować i korzystać z programu PowerShell 7, musimy sami zainstalować go w naszym systemie operacyjnym.

W chwili pisania tej książki zespół PowerShell zapewnia wsparcie dla PowerShell 7 na następujących systemach operacyjnych:

- Windows 7, 8.1 i 10

- Windows Server 2008 R2, 2012, 2012 R2, 2016 i 2019
- macOS 10.13+
- Red Hat Enterprise Linux (RHEL) / CentOS 7+
- Fedora 29+
- Debian 9+
- Ubuntu 16.04+
- openSUSE 15+
- Alpine Linux 3.8+

Lista ta jest stale aktualizowana. Zespół produktu PowerShell będzie rozszerzał ją o dystrybucje linuksowe i zapewniał wsparcie dla nowszych wersji wszystkich wymienionych platform.

Ta książka omawia instalowanie i korzystanie z programu PowerShell 7 na platformie Windows. Zamieszczone w niej rozdziały wykorzystują funkcje systemu Windows Server, takie jak Active Directory, które nie mają swoich odpowiedników na innych platformach. Niemniej jednak możliwe jest korzystanie z programu PowerShell 7 na platformach innych niż Windows w celu zarządzania klientami i serwerami Windows.

Książka ta wymaga, aby program PowerShell 7 został zainstalowany na każdym wykorzystywanym przez nas hoście, jako że wszystkie prezentowane w niej skrypty wymagają PowerShell 7.

Przed rozpoczęciem

Przedstawiony w tej części kod wykonywany jest na komputerze `DC1` z systemem Windows Server 2019. Możliwe jest również użycie fragmentów tego kodu na innych wykorzystywanych hostach w celu zainstalowania na nich programu VS Code.

Zaczynamy od wykonania kodu z użyciem programu Windows PowerShell 5.1 (lub ISE) w konsoli z podwyższonym poziomem uprawnień. Możemy pobrać odpowiednie pliki skryptów, otworzyć je lokalnie i wykonać je na każdej maszynie wirtualnej w środowisku ISE (początkowo). Gdy mamy już zainstalowany program PowerShell 7, możemy korzystać z konsoli PowerShell 7 (z programu VS Code skorzystamy w dalszej części tego rozdziału).

Włączanie zasad wykonywania

Domyślnie PowerShell nie pozwala nam wykonywać skryptów. Aby to zrobić, musimy najpierw skonfigurować zasady wykonywania.

```
# 1. Włącz wykonywanie skryptów
Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Force
```

W tym miejscu ustawiamy zasady wykonywania na wartość Unrestricted (Bez ograniczeń). W przypadku systemów produkcyjnych rozsądniejsze będzie skonfigurowanie bardziej restrykcyjnych zasad wykonywania.

Instalowanie najnowszej wersji NuGet i PowerShellGet

PowerShell 7 dostarczany jest z olbrzymią liczbą modułów zawierających przydatny zbiór poleceń, ale nie zawiera on wszystkich modułów i poleceń do wykorzystania w każdej możliwej sytuacji. Społeczność programu PowerShell stworzyła kilka wspaniałych modułów dodatkowych, które możemy pobrać i wykorzystywać. Niektóre z nich omawiane są w tej książce.

Do pobrania modułów PowerShell omawianych w tej książce używamy galerii programu PowerShell. Na przykład, w dalszych rozdziałach za pomocą modułu NTFS Security będziemy zarządzać zabezpieczeniami plików i folderów NTFS.

Moduł z galerii programu PowerShell lub innego repozytorium modułów pobieramy i instalujemy za pomocą polecenia `Import-Module`. Aby móc pracować z galerią programu PowerShell, musimy mieć pewność, że korzystamy z najnowszych wersji narzędzi, które polecenie `Install-Module` wykorzystuje do pracy z galerią PowerShell.

Aby upewnić się, że pobieramy najnowszą wersję kluczowych modułów z galerii programu PowerShell, używamy następujących poleceń:

```
# 2. Zainstaluj najnowsze wersje Nuget i PowerShellGet
Install-PackageProvider Nuget -MinimumVersion 2.8.5.201 -Force |
  Out-Null
Install-Module -Name PowerShellGet -Force -AllowClobber
```

Aby pobrać zawartość z galerii programu PowerShell, należy skorzystać z poleceń w module PowerShellGet. To z kolei wymaga narzędzia NuGet przynajmniej w wersji 2.8.5.201.

Więcej szczegółów na temat galerii programu PowerShell znajduje się na stronie *docs.microsoft.com/powershell/scripting/gallery/overview*. Aby zobaczyć polecenia dostępne w module PowerShellGet, przejdź na stronę *docs.microsoft.com/powershell/module/powershellget/*.

Tworzenie folderu Foo

Zawarte w tej książce przykładowe fragmenty kodu wykorzystują folder `C:\Foo` do przechowywania różnych plików wykorzystywanych w tym kodzie. Możemy go utworzyć za pomocą polecenia `New-Item`.

```
# 3. Utwórz lokalny folder C:\Foo
$LFHT = @{
  ItemType    = 'Directory'
```

```
  ErrorAction = 'SilentlyContinue' # jeśli już istnieje
}
New-Item -Path C:\Foo @LFHT | Out-Null
```

Polecenia te wykorzystują technikę PowerShell o nazwie *splatting* (tworzenie pakietów parametrów). Najpierw tworzymy tabelę skrótów zawierającą nazwy i wartości parametrów. Następnie przekazujemy tę tabelę do polecenia, w tym wypadku `New-Item`. Przekazujemy tabelę skrótów zamiast parametrów i ich wartości. Ta książka wykorzystuje tę funkcję na szeroką skalę z kilku powodów. Przede wszystkim dzięki niej wszystkie polecenia mieszczą się na stronach, co pozwala uniknąć poleceń rozciągających się na wiele linii. Z kolei w przypadku kodu produkcyjnego podejście to upraszcza czytanie oraz aktualizowanie skryptów zawierających polecenia z duża liczbą parametrów. Aby uzyskać więcej informacji na ten temat, możemy wpisać **`Get-Help about_splatting`** w programie PowerShell lub wyświetlić plik pomocy dostępny na stronie *docs.microsoft.com/powershell/module/microsoft.powershell.core/about/about_splatting*. Zwróćmy uwagę, że zanim będziemy mogli podejrzeć w programie PowerShell 7 plik pomocy dotyczący techniki splattingu, będziemy musieli wykonać polecenie `Update-Help`.

Pobieranie skryptu instalacyjnego PowerShell 7

Program PowerShell 7 możemy zainstalować na wiele różnych sposobów. Jednym z najprostszych jest pobranie i uruchomienie skryptu instalacyjnego utworzonego przez zespół PowerShell. Skrypt ten dostępny jest w internecie (na stronie GitHub) i możemy go pobrać w następujący sposób:

```
# 4. Pobierz skrypt instalacyjny PowerShell 7
Set-Location C:\Foo
$URI = "https://aka.ms/install-powershell.ps1"
Invoke-RestMethod -Uri $URI |
  Out-File -FilePath C:\Foo\Install-PowerShell.ps1
```

Polecenia te pobierają skrypt instalacyjny z witryny GitHub i zachowują go w folderze `C:\Foo`.

Wyświetlanie informacji pomocniczych z pliku instalacyjnego

Zawsze dobrze jest wyświetlić dowolny pobrany przez nas skrypt i zobaczyć, jak jego twórca chciał, aby był wykorzystywany. Z tego względu skrypt instalacyjny zawiera pewne podstawowe informacje pomocnicze, które możemy podejrzeć za pomocą polecenia `Get-Help`.

```
# 5. Wyświetl pomoc skryptu instalacyjnego
Get-Help -Name C:\Foo\Install-PowerShell.ps1
```

Wynik tego polecenia widoczny jest na rysunku 1.2.

```
PS C:\Foo> # 5. View Installation Script Help
Get-Help -Name C:\Foo\Install-PowerShell.ps1
Install-PowerShell.ps1 [-Destination <string>] [-Daily] [-DoNotOverwrite] [-AddToPath] [-Preview] [<CommonParameters>]
Install-PowerShell.ps1 [-UseMSI] [-Quiet] [-AddExplorerContextMenu] [-EnablePSRemoting] [-Preview] [<CommonParameters>]
```

Rysunek 1.2 Przeglądanie informacji pomocniczych

Skrypt instalacyjny pozwala nam zainstalować program PowerShell 7 za pomocą pliku MSI lub poprzez pobranie pliku ZIP i jego wyodrębnienie do wskazanego przez nas folderu. Możemy zainstalować bieżącą wersję programu PowerShell 7, najnowsze wydanie poglądowe nadchodzącej wersji, lub też zainstalować kompilację dzienną programu PowerShell. Wszystkie te wersje możemy zainstalować jednocześnie obok najnowszej pełnej wersji PowerShell 7.

Dla większości profesjonalistów IT najszybsze i najprostsze będzie skorzystanie z pliku MSI i dyskretne zainstalowanie programu. Ewaluacja przyszłych wersji i poprawek wydawanych w ramach dziennych kompilacji przeznaczona jest raczej dla najbardziej odważnych. Oczywiście zawsze powinniśmy postępować ostrożnie podczas korzystania z jakiejkolwiek niewspieranej wersji dowolnego produktu. Mając to na uwadze, te zaawansowane wersje są bardzo stabilne i pozwalają nam na wczesny dostęp do nowych funkcji i poprawek.

Instalowanie PowerShell 7

Aby zainstalować program PowerShell 7 na komputerze `DC1`, należy uruchomić skrypt instalacyjny z użyciem poniższego fragmentu kodu:

```
# 6. Zainstaluj PowerShell 7
$EXTHT = @{
UseMSI = $true
Quiet = $true
AddExplorerContextMenu = $true
EnablePSRemoting = $true
}
C:\Foo\Install-PowerShell.ps1 @EXTHT
```

Skrypt ten pobierze pakiet instalacyjny MSI programu PowerShell 7, a następnie dyskretnie wykona ten kod. Nie powinien zostać wyświetlony żaden znaczący rezultat. Ponieważ skorzystaliśmy z pliku MSI, program PowerShell został pobrany do dobrze znanej lokalizacji, a rejestr systemu Windows został zaktualizowany, aby wskazać, które wersje programu PowerShell zostały na nim zainstalowane.

Bez użycia tego pliku, na przykład w przypadku instalacji kompilacji dziennej, skrypt instalacyjny pobierze odpowiednią wersję w formie pliku ZIP i wyodrębni jego zawartość do wskazanego przez nas folderu.

Po zakończeniu wykonywania skryptu `Install-PowerShell`, w naszym systemie zainstalowany będzie program PowerShell 7.

Sprawdzanie folderu instalacji

Gdy mamy już zainstalowany PowerShell 7, możemy przyjrzeć się zawartości folderu instalacji za pomocą poniższej składni:

```
# 7. Przejrzyj folder instalacji
Get-Childitem -Path $env:ProgramFiles\PowerShell\7 -Recurse |
  Measure-Object -Property Length -Sum
```

Wynik działania tych poleceń widoczny jest na rysunku 1.3.

```
PS C:\Foo> # 7. Examine the installation folder
PS C:\Foo> Get-Childitem -Path $env:ProgramFiles\PowerShell\7 -Recurse |
             Measure-Object -Property Length -Sum

Count    : 986
Average  :
Sum      : 234332610
Maximum  :
Minimum  :
Property : Length
```

Rysunek 1.3 Sprawdzanie folderu instalacji

Jak widzimy na powyższym rysunku, folder instalacji PowerShell 7 różni się od folderu instalacji programu Windows PowerShell (który będąc komponentem systemu Windows znajduje się w folderze `C:\Windows\System32\WindowsPowerShell`).

Jeśli uważnie przyjrzymy się plikom w tym folderze, ujrzymy pewne różnice względem programu PowerShell 7. W przypadku PowerShell 7 w folderze `$PSHome` znajduje się znacznie więcej plików (co może być mylące), a nazwą programu wykonywalnego, za pomocą którego uruchamiamy konsolę PowerShell 7, jest `pwsh.exe`. Kolejną zauważalną zmianą jest to, że w przypadku programu PowerShell 7 nie ma żadnych plików `.PS1XML`. W programie Windows PowerShell pliki XML definiowały domyślne formatowanie dla szerokiego zakresu obiektów i pozwalały administratorom na korzystanie z rozszerzeń dla obiektów środowiska .NET Framework. W przypadku PowerShell 7 formatowanie XML zostało zaimplementowane w samym programie w celu podniesienia wydajności. Pliki rozszerzenia typu pozwalały zapewnić spójność między nazwami właściwości w klasach .NET. Funkcja ta również jest teraz częścią PowerShell. Wciąż możemy tworzyć nasze własne typy lub formatować XML i rozszerzać PowerShell przy użyciu zaktualizowanych typów lub domyślnego formatowania, które lepiej pasuje do naszych potrzeb.

Wyświetlanie lokalizacji folderów modułów

W programie Windows PowerShell wykorzystywane przez nas polecenia zawarte są w modułach. Możemy dyskretnie zaimportować dany moduł przed jego użyciem, aby upewnić się, że jest dostępny. Możemy również skorzystać z funkcji automatycznego wczytywania modułów programu Windows PowerShell. Dzięki tej funkcji, jeśli użyjemy polecenia, które nie jest zawarte w żadnym z wczytanych już modułów, PowerShell przeszuka dostępne moduły i sprawdzi, czy polecenie to istnieje w jakimś innym module. Jeśli tak, PowerShell wczyta ten moduł i wykona to polecenie. PowerShell wykorzystuje wbudowaną zmienną środowiskową Windows do przechowywania listy ścieżek rozdzielanych średnikami. PowerShell przeszukuje każdą ścieżkę po kolei w celu odkrycia wszystkich wymaganych modułów. Za pomocą poniższego kodu w programie Windows PowerShell możemy podejrzeć zbiór folderów modułów:

```
# 8. Wyświetl foldery modułów
# Wyświetl foldery modułów do automatycznego wczytania
$I = 0
$env:PSModulePath -split ';' |
  Foreach-Object {
    "[{0:N0}]   {1}" -f $I++, $_}
```

Wynik tych poleceń widoczny jest na rysunku 1.4.

```
PS C:\Foo> # 8. View Module folders
PS C:\Foo> #  View module folders for autoload
PS C:\Foo> $I = 0
PS C:\Foo> $env:PSModulePath -split ';' |
             Foreach-Object {
               "[{0:N0}]   {1}" -f $I++, $_}

[0]   C:\Users\tfl\Documents\WindowsPowerShell\Modules
[1]   C:\Program Files\WindowsPowerShell\Modules
[2]   C:\Windows\system32\WindowsPowerShell\v1.0\Modules
```

Rysunek 1.4 Wyświetlanie ścieżek modułów

Jak widzimy, w przypadku programu Windows PowerShell domyślnie istnieją jedynie trzy ścieżki modułów. Po dodaniu do naszego systemu funkcji lub innych aplikacji/narzędzi, programy instalacyjne mogą dodać do zmiennej zawierającej ścieżki do plików modułów dodatkowe ścieżki.

Aby zoptymalizować wydajność, przy każdym uruchomieniu program Windows PowerShell tworzy wątek o niskim priorytecie, który przegląda dostępne moduły i przechowuje szczegóły na ich temat w lokalnej pamięci podręcznej. Funkcja automatycznego wczytywania modułów wykorzystuje tę pamięć podręczną do odkrywania modułów, które muszą zostać zaimportowane przed użyciem danego polecenia. Jednym

ze skutków ubocznych tej funkcji jest to, że gdy różne moduły implementują polecenie o tej samej nazwie, zaimportowany zostanie moduł znajdujący się na wyżej położonej ścieżce. Oznacza to, że możemy tworzyć własne wersje polecenia `Get-Command`.

Wyświetlanie lokalizacji plików profilów

PowerShell definiuje cztery pliki profilów.

- AllUsersAllHosts
- AllUsersCurrentHost
- CurrentUserAllHosts
- CurrentUserCurrentHost

Każdy plik profilu powiązany jest z dobrze znaną nazwą pliku profilu. PowerShell ma wbudowaną zmienną o nazwie `$Profile`. Podczas uruchamiania PowerShell dodaje cztery właściwości do tej zmiennej, która przechowuje dobrze znane ścieżki do każdego z plików profilu. Aby zobaczyć te właściwości, należy jawnie skorzystać z parametru `-Force`.

Te pliki profilów zapewniają nam znaczącą elastyczność w odniesieniu do profilów startowych. Każdy plik profilu (który PowerShell wczytuje podczas swojego uruchamiania) może tworzyć obiekty/zmienne, ustawiać opcje środowiskowe, wysyłać wiadomości e-mail, tworzyć transkrypt, tworzyć dyski PowerShell i tak dalej. W efekcie pliki profilów oferują możliwość utrwalania niestandardowego środowiska. Jeśli chcemy, aby przed rozpoczęciem naszej pracy w ramach sesji PowerShell wykonane zostały jakieś polecenia, wystarczy dodać je do profilu.

Plików profilów PowerShell możemy używać dla wszystkich użytkowników, chociażby do definiowania pewnych aliasów dla korporacji lub wydziału, tworzenia pewnych „dobrze znanych" lokalizacji plików dla wszystkich użytkowników lub wykonywania bardziej niestandardowych akcji wyłącznie dla bieżącego użytkownika. Do dyspozycji mamy profile dla wszystkich hostów PowerShell (programów, które hostują i wykorzystują środowisko uruchomieniowe PowerShell), jak również oddzielne pliki profilów dla różnych hostów. Przykładowo środowisko ISE zawiera zmienną `$PSISE`, która pozwala nam kontrolować to środowisko. Zmienna ta nie istnieje ani w konsoli PowerShell, ani w programie VS Code. Posiadanie różnych profilów dla różnych hostów PowerShell pozwala nam dostosować każdy host za pomocą różnych technik.

W miarę naszych potrzeb za pomocą kombinacji tych czterech plików profilów możemy realizować różne scenariusze wdrożeniowe. Większość profesjonalistów IT wykorzystuje jedynie profil Current User Current Host, który jest wartością zmiennej `$Profile`.

Cztery profile oraz powiązane z nimi dobrze znane nazwy plików możemy wyświetlić w następujący sposób:

```
# 9. Wyświetl lokalizacje plików profilów
# Wewnątrz ISE
$PROFILE |
  Format-List -Property *Host* -Force
# Z konsoli Windows PowerShell
powershell -Command '$Profile|   Format-List -Property *Host*' -Force
```

Wynik powyższego fragmentu kodu, widoczny na rysunku 1.5, pokazuje lokalizacje plików profilów zarówno dla środowiska ISE, jak i dla konsoli Windows PowerShell.

```
PS C:\Foo> # 9. View Profile File locations
PS C:\Foo> # Inside the ISE
PS C:\Foo> $PROFILE |
             Format-List -Property *Host* -Force

AllUsersAllHosts       : C:\Windows\System32\WindowsPowerShell\v1.0\profile.ps1
AllUsersCurrentHost    : C:\Windows\System32\WindowsPowerShell\v1.0\Microsoft.PowerShellISE_profile.ps1
CurrentUserAllHosts    : C:\Users\tfl\Documents\WindowsPowerShell\profile.ps1
CurrentUserCurrentHost : C:\Users\tfl\Documents\WindowsPowerShell\Microsoft.PowerShellISE_profile.ps1

PS C:\Foo> powershell -Command '$PROFILE |
             Format-List -Property *Host*' -Force

AllUsersAllHosts       : C:\Windows\System32\WindowsPowerShell\v1.0\profile.ps1
AllUsersCurrentHost    : C:\Windows\System32\WindowsPowerShell\v1.0\Microsoft.PowerShell_profile.ps1
CurrentUserAllHosts    : C:\Users\tfl\Documents\WindowsPowerShell\profile.ps1
CurrentUserCurrentHost : C:\Users\tfl\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1
```

Rysunek 1.5 Wyświetlanie lokalizacji plików profilów

Uruchamianie PowerShell 7

Po zainstalowaniu programu PowerShell 7 możemy otworzyć jego konsolę klikając przycisk Start w systemie Windows, wpisując **pwsh** i naciskając Enter. Gdy otworzy się konsola PowerShell 7, możemy zweryfikować wersję posiadanego programu poprzez wyświetlenie wartości zmiennej `$PSVersionTable`.

```
# 10. Uruchom konsolę PowerShell 7, a następnie…
$PSVersionTable
```

Jak widzimy na rysunku 1.6, rezultat przechwycony został w programie PowerShell 7. Gdy ktoś czyta tę książkę, z pewnością istnieje już jakaś nowsza wersja programu (np. 7.0.1 lub 7.0.2), tak więc wyświetlana wersja programu PowerShell może być nieznacznie wyższa.

```
PS C:\Users\tfl> # 10. Run PowerShell 7 console and then...
PS C:\Users\tfl> $PSVersionTable

Name                           Value
----                           -----
PSVersion                      7.0.0
PSEdition                      Core
GitCommitId                    7.0.0
OS                             Microsoft Windows 10.0.18363
Platform                       Win32NT
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0}
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1
WSManStackVersion              3.0
```

Rysunek 1.6 Przeglądanie informacji pomocniczych

Wyświetlanie nowych lokalizacji dla folderów modułów

W poprzedniej części, „Wyświetlanie lokalizacji folderów modułów", wyświetliliśmy foldery, w których Windows PowerShell poszukuje modułów. W programie PowerShell 7 możemy osiągnąć to samo w następujący sposób:

```
# 11. Wyświetl foldery modułów
$ModFolders = $Env:PSModulePath -split ';'
$I = 0
$ModFolders |
  ForEach-Object {"[{0:N0}]   {1}" -f $I++, $_}
```

Wynik tych poleceń widoczny jest na rysunku 1.7.

```
PS C:\Foo> # 11. View Modules folders
PS C:\Foo> $ModFolders = $Env:PSModulePath -split ';'
PS C:\Foo> $I = 0
PS C:\Foo> $ModFolders |
             ForEach-Object {"[{0:N0}]   {1}" -f $I++, $_}

[0]   C:\Users\tfl\Documents\PowerShell\Modules
[1]   C:\Program Files\PowerShell\Modules
[2]   c:\program files\powershell\7\Modules
[3]   C:\Program Files\WindowsPowerShell\Modules
[4]   C:\Windows\system32\WindowsPowerShell\v1.0\Modules
```

Rysunek 1.7 Wyświetlanie ścieżek do modułów PowerShell 7

Zwróćmy uwagę, że w uzyskanym wyniku mamy pięć ścieżek do plików modułów.

Wyświetlanie nowych lokalizacji dla plików profilów

Podobnie jak w programie Windows PowerShell, w PowerShell 7 mamy wiele plików profilów, przy czym znajdują się one teraz w nieco innym miejscu na dysku. Lokalizacje dla czterech plików profilów PowerShell 7 możemy wyświetlić w następujący sposób:

```
# 12. Wyświetl lokalizacje profilów
$PROFILE | Format-List -Property *Host* -Force
```

Jak widać na rysunku 1.8, istnieją cztery pliki profilów, każdy w innej lokalizacji niż w przypadku programu Windows PowerShell.

```
PS C:\Foo> # 12. View Profile Locations
PS C:\Foo> $PROFILE | Format-List -Property *Host* -Force

AllUsersAllHosts       : C:\Program Files\PowerShell\7\profile.ps1
AllUsersCurrentHost    : C:\Program Files\PowerShell\7\Microsoft.PowerShell_profile.ps1
CurrentUserAllHosts    : C:\Users\tfl\Documents\PowerShell\profile.ps1
CurrentUserCurrentHost : C:\Users\tfl\Documents\PowerShell\Microsoft.PowerShell_profile.ps1
```

Rysunek 1.8 Wyświetlanie lokalizacji plików profilów

Tworzenie profilu bieżącego użytkownika/bieżącego hosta

Aby zademonstrować sposób działania plików profilów w programie PowerShell 7, możemy pobrać przykładowy profil z internetu i utworzyć profil bieżącego użytkownika/bieżącego hosta (CurrentUserCurrentHost) za pomocą następujących poleceń:

```
# 13. Utwórz profil bieżącego użytkownika/bieżącego hosta
$URI = 'https://raw.githubusercontent.com/doctordns/Wiley20/master/' +
       'Goodies/Microsoft.PowerShell_Profile.ps1'
$ProfileFile = $Profile.CurrentUserCurrentHost
New-Item $ProfileFile -Force -WarningAction SilentlyContinue |
   Out-Null
(Invoke-WebRequest -Uri $uri -UseBasicParsing).Content |
  Out-File -FilePath  $ProfileFile
```

Skrypty dla tej książki można pobrać ze strony wydawnictwa Wiley oraz z repozytorium GitHub autora. Repozytorium autora zawiera folder Goodies, w którym znajduje się przykładowy plik profilu ilustrujący sposób korzystania z profilów w konsoli PowerShell 7.

Ten przykładowy plik ustawia pewne przydatne wartości domyślne i aliasy, tworzy pewne zmienne, konfiguruje nagłówek konsoli i ustawia bieżący folder roboczy na tytułowy folder `C:\Foo`. Profil ten możemy dowolnie zmodyfikować, aby dostosować go do naszego stylu pracy. Po uruchomieniu tych poleceń należy uruchomić ponownie PowerShell, aby nowe profile zaczęły być wykorzystywane.

Instalowanie i konfigurowanie VS Code

PowerShell ISE jest interaktywnym środowiskiem programistycznym dla Windows PowerShell. ISE pozwala nam edytować, zarządzać i uruchamiać skrypty z poziomu pojedynczego programu. Choć dla PowerShell dostępnych jest wiele takich innych środowisk, środowisko ISE jest wbudowane i darmowe, a przy tym oferuje dobrą funkcjonalność, która będzie przydatna dla profesjonalisty IT. Wykonane dla tej książki zrzuty ekranu ukazujące kod PowerShell 7 ilustrują działanie programu VS Code.

Firma Microsoft poinformowała, że nie ma w planach aktualizować środowiska ISE pod kątem wsparcia dla PowerShell 7. Rekomendowaną alternatywą jest VS Code. VS Code firmy Microsoft jest darmowym, wieloplatformowym edytorem kodu źródłowego na licencji open source. VS Code działa w formie aplikacji tradycyjnej w systemach Windows, macOS i Linux, zapewniając świetne wsparcie dla PowerShell. VS Code jest doskonałym narzędziem do zarządzania nie tylko kodem źródłowym PowerShell, ale również dokumentami z kodem w językach Perl, Python, Markdown i wielu innych.

Choć VS Code jest narzędziem wieloplatformowym, ten rozdział opisuje korzystanie z VS Code w systemie Windows. Więcej informacji na temat programu VS Code można znaleźć na stronie *code.visualstudio.com/*.

VS Code obsługuje bogaty zestaw rozszerzeń, które dodatkowo poszerzają doświadczenie programistyczne. Możemy przykładowo zainstalować narzędzie do sprawdzania pisowni. Jeśli pracujemy z kodem Markdown (*en.wikipedia.org/wiki/Markdown*), również możemy skorzystać z utworzonych przez społeczność wbudowanych rozszerzeń VS Code, które nam tę pracę ułatwią. Za pomocą programu VS Code możemy dodać nowe rozszerzenia do naszego środowiska, w tym również zainstalować je podczas instalacji VS Code.

Aby zainstalować VS Code na naszym komputerze, musimy pobrać i uruchomić skrypt instalacyjny ze strony Visual Studio, która umożliwia nam zainstalowanie programu VS Code wraz z dowolnymi wymaganymi rozszerzeniami w ramach pojedynczej operacji.

Jeśli nie pracujemy wyłącznie z kodem PowerShell, warto odwiedzić witrynę sklepu Visual Studio pod adresem *marketplace.visualstudio.com*, gdzie znajdziemy bogatą kolekcję rozszerzeń. Niektóre rozszerzenia są darmowe, a niektóre komercyjne (zwykle oferują one darmowe wersje próbne).

Po zainstalowaniu programu VS Code możemy w prosty sposób zmienić wykorzystywaną przez niego czcionkę. VS Code może wykorzystywać dowolną czcionkę zainstalowaną w naszym systemie. Czcionkę możemy zmienić z poziomu interfejsu graficznego programu VS Code lub za pośrednictwem pliku ustawień użytkownika w formacie JSON.

Microsoft opublikował niedawno nową czcionkę o nazwie Cascadia Code. W programie VS Code czcionka ta wygląda bardzo dobrze. Wszystkie zaprezentowane w tej książce zrzuty ekranu z kodem PowerShell korzystają z tej czcionki.

Jeśli planujemy korzystać z programu VS Code i PowerShell 7, możemy zechcieć dodać do nich skróty do naszego paska zadań.

Uwaga Zrzuty ekranu dla tej książki zostały wykonane z użyciem programu VS Code. Z programu VS Code możesz korzystać we wszystkich swoich maszynach wirtualnych w celu przetestowania skryptów zawartych w tej książce, ale w pewnych środowiskach może to być zabronione lub uznane za złą praktykę.

Przed rozpoczęciem

Przedstawiony w tej części kod wykonywany jest na maszynie wirtualnej `DC1` z systemem Windows Server 2019 Datacenter. Możemy również użyć tych fragmentów kodu na innych hostach, aby za ich pomocą zainstalować VS Code i przetestować skrypty zawarte w tej książce.

Skrypty przedstawione w tej części należy uruchamiać za pomocą programu PowerShell 7, który zainstalowaliśmy w poprzedniej części „Instalowanie PowerShell 7".

Pobieranie skryptu instalacyjnego VS Code

Zespół VS Code stworzył skrypt instalacyjny, za pomocą którego możemy zainstalować VS Code. Jest on dostępny w galerii programu PowerShell. Aby go pobrać, należy wykonać poniższy kod:

```
# 1. Pobierz skrypt instalacyjny VS Code
$VSCPATH = 'C:\Foo'
Save-Script -Name Install-VSCode -Path $VSCPATH
Set-Location -Path $VSCPATH
```

Ten fragment kodu pozyskuje skrypt `Install-VSCode` i zapisuje go w folderze `C:\Foo`.

Instalowanie VS Code i rozszerzeń

Instalacji programu VS Code dokonujemy przy użyciu pobranego pliku `Install-VSCode.ps1`. Za pomocą poniższego kodu możemy wskazać konkretne rozszerzenia VS Code, które instalator powinien zainstalować razem z programem:

```
# 2. Uruchom go i dodaj pewne popularne rozszerzenia VS Code
$Extensions =  "Streetsidesoftware.code-spell-checker",
               "yzhang.markdown-all-in-one"
$InstallHT = @{
  BuildEdition         = 'Stable-System'
  AdditionalExtensions = $Extensions
  LaunchWhenDone       = $true
}
```

```
.\Install-VSCode.ps1 @InstallHT
```

Wynik uzyskany po uruchomieniu skryptu instalacyjnego VS Code widoczny jest na rysunku 1.9.

```
PS C:\Foo> # 2. Now run it and add in some popular VSCode Extensions
PS C:\Foo> $Extensions =  "Streetsidesoftware.code-spell-checker",
                          "yzhang.markdown-all-in-one"
PS C:\Foo> $InstallHT = @{
             BuildEdition         = 'Stable-System'
             AdditionalExtensions = $Extensions
             LaunchWhenDone       = $true
             }
PS C:\Foo> .\Install-VSCode.ps1 @InstallHT

Installing extension ms-vscode.PowerShell...
Installing extensions...
Installing extension 'ms-vscode.powershell' v2020.4.0...
Extension 'ms-vscode.powershell' v2020.4.0 was successfully installed.

Installing extension Streetsidesoftware.code-spell-checker...
Installing extensions...
Installing extension 'streetsidesoftware.code-spell-checker' v1.8.0...
Extension 'streetsidesoftware.code-spell-checker' v1.8.0 was successfully installed.

Installing extension yzhang.markdown-all-in-one...
Installing extensions...
Installing extension 'yzhang.markdown-all-in-one' v2.8.0...
Extension 'yzhang.markdown-all-in-one' v2.8.0 was successfully installed.

Installation complete, starting Visual Studio Code (64-bit)...
```

Rysunek 1.9 Instalowanie VS Code na maszynie wirtualnej DC1

Ten fragment kodu najpierw pobiera z internetu, a następnie instaluje najnowszą stabilną wersję programu VS Code. W ramach procesu instalacji dodawane są dwa rozszerzenia. Pierwszym jest narzędzie do sprawdzania pisowni, a drugie przydaje się w przypadku, gdy edytujemy pliki z kodem Markdown.

Istnieje wiele innych rozszerzeń, które mogą być dla nas przydatne, w zależności od tego, z czego składają się nasze obciążenia robocze. Społeczność stale rozwija, rozszerza i utrzymuje szeroką gamę rozszerzeń.

Polecenia te uruchamiają program VS Code, tak więc w ramach uzyskiwanego wyniku pojawia się również okno programu VS Code. Możemy wykorzystać to okno do uruchamiania fragmentów kodu zaprezentowanych w pozostałej części tego rozdziału.

Tworzenie przykładowego pliku profilu osobistego

Pliki profilu PowerShell pozwalają nam utrwalać modyfikacje dokonywane w ramach sesji PowerShell. Możemy tworzyć nowe dyski PowerShell, tworzyć i wypełniać wartościami niestandardowe zmienne, definiować przydatne funkcje i wiele więcej. Domyślnie

PowerShell 7 dostarczany jest bez żadnych plików profilów. W przypadku programu Windows PowerShell możemy utworzyć (pusty) plik profilu poprzez uruchomienie poniższych poleceń w oknie VS Code:

```
# 3. Utwórz przykładowy plik profilu
$SAMPLE = 'https://raw.githubusercontent.com/doctordns/Wiley20/master/' +
          'Goodies/Microsoft.VSCode_profile.ps1'
(Invoke-WebRequest -Uri $Sample).Content |
  Out-File $Profile
```

Powyższy fragment kodu pobiera przykładowy profil VS Code i zapisuje go jako profil bieżącego hosta/bieżącego użytkownika. W zależności od naszej organizacji, przykład ten może być dla nas w pełni wystarczający. Jeśli korzystamy zarówno z konsoli PowerShell 7, jak i z programu VS Code, musimy utrzymywać dwie oddzielne wersje dla tego profilu – po jednym dla każdego hosta. Warto przejrzeć te dwa przykładowe profile i rozszerzyć je zgodnie z zapotrzebowaniem.

Pobieranie czcionki Cascadia Code

Oprócz programu VS Code, firma Microsoft opracowała również nową czcionkę Cascadia Code o stałej szerokości do wykorzystywania w VS Code (i dowolnej innej aplikacji Windows, która używa czcionek o stałej szerokości, wliczając w to przykładowo PowerShell, konsolę Windows PowerShell czy Microsoft Office Word). Twórcy czcionki Cascadia Code dostarczają końcową wersję tej czcionki poprzez witrynę GitHub (jak również poprzez sklep Microsoft Store). Istnieje kilka wersji tej czcionki, wliczając w to czcionkę podstawową (`Cascadia.ttf`). Możemy również pobrać wariant, który obsługuje symbole Powerline (`CascadiaPL.ttf`). Aby pobrać najnowszą wersję czcionki podstawowej, należy wykonać poniższy kod:

```
# 4. Pobierz czcionkę Cascadia Code z witryny GitHub
# Pozyskaj lokalizacje plików
$CascadiaFont    = 'Cascadia.ttf'    # nazwa czcionki
$CascadiaRelURL  = 'https://github.com/microsoft/cascadia-code/releases'
$CascadiaRelease = Invoke-WebRequest -Uri $CascadiaRelURL # Pobierz
wszystkie
$CascadiaPath    = "https://github.com" + ($CascadiaRelease.Links.href |
                     Where-Object { $_ -match "($CascadiaFont)" } |
                       Select-Object -First 1)
$CascadiaFile    = "C:\Foo\$CascadiaFont"
# Pobierz plik czcionki Cascadia Code
Invoke-WebRequest -Uri $CascadiaPath -OutFile $CascadiaFile
```