PiowerShell – Wstęp , Wersje

Z perspektywy dzisiejszego Windowsa warto myśleć o **dwóch „rodzinach" PowerShella**:

- **Windows PowerShell 5.1** – „stary", klasyczny, tylko Windows, domyślnie zainstalowany.
- **PowerShell (Core) 7.x** – nowy, multiplatformowy, rozwijany, doinstalowywany osobno.

1. Windows PowerShell 5.1 – co to jest i jaki jest status

Charakterystyka:

- Wersja: **5.1** (ostatnia, dalej rozwijana tylko w ramach poprawek bezpieczeństwa Windows).
- System: **tylko Windows** (wbudowany w system).
- Środowisko: oparte na **.NET Framework** (klasyczny, nie .NET / .NET Core).
- Ścisła integracja z Windows:
    - pełne wsparcie dla starych modułów (np. ActiveDirectory, DNSServer, FSRM, stare moduły Exchange/Skype On-Prem).
    - Dużo skryptów/produktów vendorów wciąż zakłada 5.1.

Status:
5.1 jest w trybie „maintenance mode" – **nie rozwijany funkcjonalnie**, ale wciąż masowo używany w środowiskach on-prem i w skryptach legacy.

2. PowerShell (Core) 7.x – co to jest

Charakterystyka:

- Aktualna „główna linia" – wersje **7.x** (7.5.x itd).
- Multiplatformowy:
    - **Windows, Linux, macOS**.
- Oparty na **nowym .NET (Core/.NET)** – wydajniejszy, lepsze wsparcie nowości.
- Rozwijany: nowe funkcje, poprawki, usprawnienia (pipeline parallelization, ForEach-Object -Parallel, lepsze error handling, tryb „predictive intellisense" itd).
- Instalacja obok 5.1:
    - domyślnie **nie zastępuje** 5.1.
    - uruchamiasz jako pwsh (a nie powershell).

3. Kluczowe różnice praktyczne

3.1. Kompatybilność modułów

- **5.1:**Najlepsza kompatybilność z **„starymi" modułami Windows**:
    - ActiveDirectory
    - GroupPolicy
    - DNSServer
    - Hyper-V
    - FSRM, DHCP, stare Exchange Management Shell on-prem, itp.
    - Część vendorów w dokumentacji wprost pisze: „uruchom w Windows PowerShell".
- **7.x:**W nowoczesnych systemach (Windows Server 2022+, aktualny Windows 11) **większość modułów działa** – dzięki tzw. **Windows Compatibility Layer** (automatyczne spinanie z 5.1 w tle).
    - Niektóre bardzo stare/niszowe moduły mogą dalej wymagać 5.1.
    - Nowe moduły (szczególnie cross-platform) są tworzone z myślą o 7.x:
    - Azure (Az),
    - Microsoft Graph,
    - moduły DevOps, CI/CD, narzędzia chmurowe.

Praktyczny wniosek:
Jeżeli coś jest typowo „systemowe, stare, mocno on-prem" → najpierw testuj w 5.1.
Jeżeli to **API, chmura, nowy kod** → preferuj 7.x.

3.2. Obsługiwane platformy

- **5.1:** tylko Windows (serwery, stacje).
- **7.x:** Windows, Linux, macOS.
  Ten sam skrypt możesz użyć w kontenerze, na serwerze Linux w Azure, na macbooku itd.

Wniosek:
Do automatyzacji „świata mieszanych systemów" -> **tylko 7.x ma sens**.

3.3. Wydajność i nowości w języku

PowerShell 7.x ma:

- Lepszą wydajność (start, parsy, pipeline).
- **ForEach-Object -Parallel** – prosty parallelizm w pipeline.
- Tryb **„Predictive IntelliSense"** (podpowiadanie komend na podstawie historii/PSReadLine).
- Ulepszony handling błędów (np. „concise view", filtrowanie).
- Rozszerzenia typu **cross-platform remoting** (SSH).

5.1 tego **nie dostanie** – tam czas się zatrzymał.

4. Co zaleca Microsoft (wysoki poziom)

Ogólny kierunek:

- **Do nowego rozwoju skryptów/automatyzacji – PowerShell 7.x.**
- **Windows PowerShell 5.1 zachować dla kompatybilności** z istniejącą infrastrukturą i modułami.

W dokumentacji Microsoftu bardzo często jest wzór:

„Windows PowerShell 5.1 is in maintenance mode. For new scripts and tools, use PowerShell 7+ unless you depend on specific modules that require Windows PowerShell."

5. Jakie scenariusze – co stosować w praktyce

5.1. Scenariusze typowo „Windows on-prem"

Przykłady:

- Administracja **AD DS, GPO, FSRM, DHCP, DNS** na Windows Server 2016/2019/2022.
- Skrypty instalacyjne starych aplikacji, które odwołują się wprost do 5.1 lub .NET Framework.
- Narzędzia producentów, które w dokumentacji podają „Windows PowerShell".

Rekomendacja:

- **Produkcja/konserwacja istniejącej infrastruktury:**
    - nadal używaj **Windows PowerShell 5.1** jako „baseline".
- Jednocześnie:
    - nowe elementy (np. integracja z chmurą, API) możesz wynosić do **PS 7.x**, jeśli moduły wspierają.

5.2. Azure, chmura, automatyzacja CI/CD

Przykłady:

- Moduł **Az** (Azure PowerShell).
- Skrypty w **GitHub Actions**, **Azure DevOps**, pipeline'y CI/CD.
- Integracje z REST API, Graph API, automatyzacja SaaS.

Rekomendacja:

- **Docelowo standard: PowerShell 7.x.**
- W pipeline'ach i kontenerach: używaj **oficjalnych obrazów Dockera** z PS 7.x.
- Lokalnie (na laptopie) – też 7.x jako „główne" środowisko do kodowania, 5.1 tylko gdy musisz użyć starego modułu.

5.3. Multiplatformowe skrypty (Windows + Linux + macOS)

Jeżeli:

- piszesz materiały szkoleniowe dla studentów z różnymi systemami,
- automatyzujesz serwery Linux (np. w Azure),
- chcesz ten sam kod uruchamiać na Windows i w kontenerach,

to **Windows PowerShell 5.1 odpada z definicji**.

Rekomendacja:

- Wyłącznie **PowerShell 7.x**.
- Zwracaj uwagę na:
    - ścieżki (/ vs \),
    - różnice w dostępności cmdletów typowo windowsowych (rejestr, WMI klasyczne).

6. Jak rozróżniać i współistnieć na jednej maszynie

Na typowym Windows 10/11/Server:

- **Windows PowerShell 5.1**
    - komenda: powershell.exe
    - lokalizacja: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
- **PowerShell 7.x**
    - komenda: pwsh.exe
    - domyślna lokalizacja (MSI): C:\Program Files\PowerShell\7\pwsh.exe
    - można też instalować z **winget**, **Microsoft Store**, **ZIP** itd.

7. Podsumowanie – krótka rekomendacja „co stosować”

1. **Nie rezygnuj z Windows PowerShell 5.1** – jest i długo będzie potrzebny do:
    - starej administracji Windows/AD/GPO/FSRM/DHCP/DNS,
    - istniejących skryptów i narzędzi vendorów.
2. **Do każdego nowego projektu / nowej automatyzacji**:
    - startuj w **PowerShell 7.x**,
    - wyjątek: gdy masz **konkretny moduł, który nie działa w 7.x** – wtedy zostajesz przy 5.1.