Tworzenie profesjonalnego skryptu przesyłania plików w PowerShell

Przygotowanie

```
# Utwórz folder C:\SourceFolder
New-Item -Path "C:\SourceFolder" -ItemType Directory
# Utwórz folder \\127.0.0.1\c$\DestinationFolder
New-Item -Path "\\127.0.0.1\c$\DestinationFolder" -ItemType Directory
# Utwórz plik z zawartością
Set-Content -Path "C:\SourceFolder\plik.txt" -Value "testowa informacja"
```

Transfery plików nie są sobie równe i mają różne znaczenie. Jednego dnia możesz być w domu i kopiować niektóre pliki MP3 na swój komputer osobisty, a następnego jesteś w pracy, wykonując aktualizację aplikacji, kopiując krytyczne pliki, aby zapewnić ciągłość działania firmy. Dzisiaj, interesuje nas to drugie podejście.

Jesteśmy tutaj, aby zbudować narzędzie w PowerShell, które wydawałoby się przesadą, gdyby wykonywać prostą kopię, ale jeśli chodzi o tworzenie kopii krytycznych danych, nie ma zbyt wielu zabezpieczeń.

1. Pierwsze kroki ze skryptem FTP

Zbudujmy funkcję PowerShell, która będzie działać jako nasze rusztowanie:

```
function Move-Stuff {
    param(
        [Parameter(Mandatory)]
        [string]$Path,
```

```
    [Parameter(Mandatory)]
    [string]$DestinationPath
)
}
```

Powyższy kod przedstawia definicję funkcji PowerShell o nazwie "Move-Stuff". Funkcja ta przyjmuje dwa parametry: "Path" i "DestinationPath", oba są obowiązkowe (oznaczone atrybutem [Parameter(Mandatory)]) i oba są typu string.

1. **param**: Jest to polecenie używane w funkcjach PowerShell do deklaracji parametrów funkcji. W tym przypadku zdefiniowano dwa parametry: "Path" i "DestinationPath".
2. **[Parameter(Mandatory)]**: Atrybut ten wskazuje, że dany parametr jest obowiązkowy, czyli musi zostać przekazany podczas wywoływania funkcji.
3. **[string]$Path**: Definiuje parametr "Path" jako string, co oznacza, że funkcja oczekuje, iż zostanie przekazany ciąg znaków reprezentujący ścieżkę do pliku lub katalogu.
4. **[string]$DestinationPath**: Definiuje parametr "DestinationPath" również jako string, co oznacza, że funkcja oczekuje, iż zostanie przekazany ciąg znaków reprezentujący docelową ścieżkę, do której ma zostać przeniesiony plik lub katalog.

Obecnie funkcja nie zawiera żadnej implementacji kodu, co oznacza, że nie wykonuje żadnych działań. Jeśli chcesz, aby funkcja realizowała przenoszenie plików lub katalogów z podanej ścieżki do docelowej ścieżki, musisz dodać odpowiednią logikę do funkcji. Na przykład, możesz użyć cmdletu "Move-Item" do przenoszenia plików i katalogów w środowisku PowerShell.

Funkcja Move-Stuff służy do przenoszenia folderu z jednego miejsca do drugiego, z zachowaniem integralności danych poprzez porównanie skrótów (hashy) przed i po transferze. Szczegółowy opis poszczególnych kroków w funkcji:

Parametry

Funkcja przyjmuje dwa parametry:

- $Path: Ścieżka źródłowa folderu, który ma być przeniesiony.
- $DestinationPath: Ścieżka docelowa, gdzie folder ma być przeniesiony.

Krok 1: Wyświetlanie parametrów i sprawdzanie ścieżek

1. **Wyświetlanie parametrów:**

```
Write-Host "Inside Function - Ścieżka źródłowa: $Path"
Write-Host "Inside Function - Ścieżka docelowa: $DestinationPath"
```

Funkcja wyświetla wartości parametrów wewnątrz funkcji dla celów debugowania.

2. **Sprawdzanie czy ścieżki nie są puste:**

```
if (-not $Path) {
    Write-Host "Path parameter is null or empty"
    return
}
if (-not $DestinationPath) {
    Write-Host "DestinationPath parameter is null or empty"
    return
}
```

Funkcja sprawdza, czy ścieżki nie są puste i, jeśli są, kończy działanie.

Krok 2: Kompresowanie folderu

3. **Kompresowanie folderu:**

```
$zipFilePath = "$Path\$([guid]::NewGuid()).zip"
Compress-Archive -Path $Path -DestinationPath $zipFilePath -CompressionLevel Optimal
Write-Host "Zip file created at: $zipFilePath"
```

Funkcja tworzy unikalną nazwę pliku ZIP, kompresuje folder źródłowy i zapisuje go w pliku ZIP.

Krok 3: Generowanie skrótu przed transferem

4. **Generowanie skrótu (hash) przed transferem:**

```
$beforeHash = (Get-FileHash -Path $zipFilePath).Hash
Write-Host "Before Hash: $beforeHash"
```

Funkcja oblicza skrót (hash) pliku ZIP przed transferem i wyświetla go.

Krok 4: Transfer pliku ZIP do folderu tymczasowego

5. **Przygotowanie ścieżki docelowej na komputerze zdalnym:**

```
$destComputer = $DestinationPath.Split('\')[2]
Write-Host "Destination Computer: $destComputer"
$remoteTempPath = "\\$destComputer\c$\Temp"
$remoteZipFilePath = "$remoteTempPath\$([System.IO.Path]::GetFileName($zipFilePath))"
Write-Host "Remote Zip File Path: $remoteZipFilePath"
```

Funkcja wyodrębnia nazwę komputera docelowego i przygotowuje ścieżki do folderu tymczasowego na tym komputerze.

6. **Tworzenie sesji PowerShell na komputerze zdalnym:**

```
$session = New-PSSession -ComputerName $destComputer
Write-Host "PowerShell Session Created"
```

7. **Tworzenie folderu tymczasowego na komputerze docelowym, jeśli nie istnieje:**

```
Invoke-Command -Session $session -ScriptBlock {
    if (-not (Test-Path -Path "C:\Temp")) {
        New-Item -Path "C:\Temp" -ItemType Directory
    }
}
Write-Host "Temp folder created on destination computer if it didn't exist"
```

Funkcja tworzy folder C:\Temp na komputerze docelowym, jeśli ten folder nie istnieje.

8. **Transfer pliku ZIP:**

```
Start-BitsTransfer -Source $zipFilePath -Destination $remoteZipFilePath
Write-Host "Bits Transfer Started"
```

Krok 5: Porównanie skrótów po transferze

9. **Obliczanie skrótu (hash) po transferze:**

```
$localFolderPath = $DestinationPath.Replace("\\$destComputer\c$\",'C:\')
$localZipFilePath = $remoteZipFilePath.Replace("\\$destComputer\c$\",'C:\')
Write-Host "Local Folder Path: $localFolderPath"
Write-Host "Local Zip File Path: $localZipFilePath"
$afterHash = Invoke-Command -Session $session -ScriptBlock { (Get-FileHash -Path "$using:localZipFilePath").Hash }
Write-Host "After Hash: $afterHash"
```

Funkcja oblicza skrót pliku ZIP po transferze i porównuje go z wartością przed transferem.

10. **Sprawdzenie integralności pliku:**

```
if ($beforeHash -ne $afterHash) {
    throw 'Plik został zmodyfikowany podczas transferu!'
}
```

Krok 6: Dekompresowanie pliku ZIP

11. **Dekompresowanie pliku ZIP na komputerze docelowym:**

```
Invoke-Command -Session $session -ScriptBlock { Expand-Archive -Path "$using:localZipFilePath" -DestinationPath $using:localFolderPath }
Write-Host "File Decompressed"
```

Krok 7: Czyszczenie

12. **Czyszczenie po zakończeniu operacji:**

```
if ($session) {
    Invoke-Command -Session $session -ScriptBlock { Remove-Item "$using:localZipFilePath" -ErrorAction Ignore }
    Remove-PSSession -Session $session -ErrorAction Ignore
}
if ($zipFilePath) {
    Remove-Item -Path $zipFilePath -ErrorAction Ignore
}
```

Zakończenie

Funkcja Move-Stuff dokładnie przenosi folder z jednego miejsca do drugiego, upewniając się, że plik ZIP nie został zmodyfikowany podczas transferu. Wykorzystuje sesje PowerShell do zdalnego wykonania komend na komputerze docelowym, co pozwala na bezpieczne i skuteczne przenoszenie danych.

2. Funkcja przesyłania plików PowerShell

Ostateczna funkcja będzie wyglądać tak:

```
function Move-Stuff {
    param(
        [Parameter(Mandatory=$true)]
        [string]$Path,


        [Parameter(Mandatory=$true)]
```

```
    [string]$DestinationPath
)
try {
    # Debug: Wyświetl parametry wewnątrz funkcji
    Write-Host "Inside Function - Ścieżka źródłowa: $Path"
    Write-Host "Inside Function - Ścieżka docelowa: $DestinationPath"

    # Sprawdź, czy ścieżki nie są puste
    if (-not $Path) {
        Write-Host "Path parameter is null or empty"
        return
    }
    if (-not $DestinationPath) {
        Write-Host "DestinationPath parameter is null or empty"
        return
    }

    ## Kompresuj folder
    $zipFilePath = "$Path\$([guid]::NewGuid()).zip"
    Compress-Archive -Path $Path -DestinationPath $zipFilePath -CompressionLevel Optimal
    Write-Host "Zip file created at: $zipFilePath"
```

```
    ## Utwórz hasz przed transferem
    $beforeHash = (Get-FileHash -Path $zipFilePath).Hash
    Write-Host "Before Hash: $beforeHash"

    ## Przenieś do folderu tymczasowego
    $destComputer = $DestinationPath.Split('\')[2]
    Write-Host "Destination Computer: $destComputer"
    $remoteTempPath = "\\$destComputer\c$\Temp"
    $remoteZipFilePath = "$remoteTempPath\$([System.IO.Path]::GetFileName($zipFilePath))"
    Write-Host "Remote Zip File Path: $remoteZipFilePath"

    ## Utwórz sesję PowerShell
    $session = New-PSSession -ComputerName $destComputer
    Write-Host "PowerShell Session Created"

    ## Utwórz folder tymczasowy na komputerze docelowym, jeśli nie istnieje
    Invoke-Command -Session $session -ScriptBlock {
        if (-not (Test-Path -Path "C:\Temp")) {
            New-Item -Path "C:\Temp" -ItemType Directory
        }
```

```
    }
    Write-Host "Temp folder created on destination computer if it didn't exist"

    Start-BitsTransfer -Source $zipFilePath -Destination $remoteZipFilePath
    Write-Host "Bits Transfer Started"

    ## Porównaj hasze przed i po transferze
    ## Zakładając, że używamy udziału administracyjnego c$
    $localFolderPath = $DestinationPath.Replace("\\$destComputer\c$\",'C:\')
    $localZipFilePath = $remoteZipFilePath.Replace("\\$destComputer\c$\",'C:\')
    Write-Host "Local Folder Path: $localFolderPath"
    Write-Host "Local Zip File Path: $localZipFilePath"
    $afterHash = Invoke-Command -Session $session -ScriptBlock { (Get-FileHash -Path "$using:localZipFilePath").Hash }
    Write-Host "After Hash: $afterHash"
    if ($beforeHash -ne $afterHash) {
        throw 'Plik został zmodyfikowany podczas transferu!'
    }

    ## Dekompresuj plik zip
    Invoke-Command -Session $session -ScriptBlock { Expand-Archive -Path "$using:localZipFilePath" -DestinationPath $using:localFolderPath }
    Write-Host "File Decompressed"
```

```
    } catch {
        $PSCmdlet.ThrowTerminatingError($_)
    } finally {
        ## Czyszczenie
        if ($session) {
            Invoke-Command -Session $session -ScriptBlock { Remove-Item "$using:localZipFilePath" -ErrorAction Ignore }
            Remove-PSSession -Session $session -ErrorAction Ignore
        }
        if ($zipFilePath) {
            Remove-Item -Path $zipFilePath -ErrorAction Ignore
        }
    }
}
```

3. Przygotuj polencie które użyje powyższą funkcję

Upewnij się, że zamienisz "C:\SourceFolder" i \\RemoteComputer\c$\DestinationFolder na rzeczywiste ścieżki źródłowe i docelowe.

To polecenie spakuje folder źródłowy, przeniesie go do miejsca docelowego, zweryfikuje integralność transferu, a następnie rozpakowuje go w miejscu docelowym.

```
# Zdefiniuj wartości dwóch zmiennych ścieżek i wyświetlenia ich wartości
Write-Host "Definiowanie zmiennych:"
$sourcePath = "C:\SourceFolder"
```

```
$destinationPath = "\\127.0.0.1\c$\DestinationFolder"

Write-Host "Ścieżka źródłowa: $sourcePath"

Write-Host "Ścieżka docelowa: $destinationPath"


# Debug: Sprawdź typy zmiennych przed wywołaniem funkcji

Write-Host "Typ zmiennej sourcePath: $($sourcePath.GetType().Name)"
```

$sourcePath: Jest to zmienna, która została wcześniej zdefiniowana jako ścieżka źródłowa do folderu, który ma być przeniesiony.

$sourcePath.GetType().Name: Ta część kodu wykorzystuje metodę GetType() obiektu $sourcePath, aby uzyskać informacje o typie tego obiektu. Metoda GetType() zwraca obiekt typu Type, który zawiera informacje o typie runtime. Następnie używamy właściwości Name tego obiektu, aby uzyskać nazwę typu jako ciąg znaków (string).

$($sourcePath.GetType().Name): Ta część kodu jest wstawiana do ciągu znaków, aby wynik był wyświetlany jako część komunikatu. Użycie $() jest potrzebne do wykonania zagnieżdżonego wyrażenia w kontekście ciągu znaków.

Write-Host: Wyświetla komunikat zawierający typ zmiennej sourcePath. Przykładowy wynik to: Typ zmiennej sourcePath: String.

```
Write-Host "Typ zmiennej destinationPath: $($destinationPath.GetType().Name)"
```

$destinationPath: Jest to zmienna, która została wcześniej zdefiniowana jako ścieżka docelowa, gdzie folder ma być przeniesiony.

$destinationPath.GetType().Name: Podobnie jak w przypadku zmiennej sourcePath, ta część kodu używa metody GetType() obiektu $destinationPath, aby uzyskać informacje o typie tego obiektu. Właściwość Name zwraca nazwę typu jako ciąg znaków.

$($destinationPath.GetType().Name): Ta część kodu jest wstawiana do ciągu znaków, aby wynik był wyświetlany jako część komunikatu.

Write-Host: Wyświetla komunikat zawierający typ zmiennej destinationPath. Przykładowy wynik to: Typ zmiennej destinationPath: String.

Ten kod jest używany do sprawdzenia i wyświetlenia typów zmiennych sourcePath i destinationPath. Wyświetlanie typu zmiennej jest szczególnie przydatne podczas debugowania, ponieważ pozwala upewnić się, że zmienne mają oczekiwane typy danych. W tym przypadku obie zmienne powinny mieć typ String, co jest potwierdzane przez wyniki wyświetlane przez Write-Host.

Skonfiguruj środowisko do zdalnego zarządzania

Enable-PSRemoting -Force -SkipNetworkProfileCheck

Wywołaj funkcję

Move-Stuff -Path $sourcePath -DestinationPath $destinationPath

Efekt:

```
Zip file created at: C:\SourceFolder\6c8a2353-891a-471b-a502-300718449c67.zip
Before Hash: B523C0B2A84C917333FE7B70FE6BD07775CA9FED8A4F6C41930A91413F0A97E1
Destination Computer: 127.0.0.1
Remote Zip File Path: \\127.0.0.1\c$\Temp\6c8a2353-891a-471b-a502-300718449c67.zip
PowerShell Session Created


    Directory: C:\


Mode                 LastWriteTime         Length Name                                                                  PSComputerName
----                 -------------         ------ ----                                                                  --------------
d-----        30.07.2024     12:34                Temp                                                                  127.0.0.1
Temp folder created on destination computer if it didn't exist
Bits Transfer Started
Local Folder Path: C:\DestinationFolder
Local Zip File Path: C:\Temp\6c8a2353-891a-471b-a502-300718449c67.zip
After Hash: B523C0B2A84C917333FE7B70FE6BD07775CA9FED8A4F6C41930A91413F0A97E1
File Decompressed
```

Co się dzieje po wywołaniu funkcji

1. **Wykonanie funkcji**: Po wywołaniu Move-Stuff, funkcja przetwarza przekazane argumenty zgodnie z jej definicją. W szczególności:
   - **Kompresuje folder**: Funkcja tworzy plik ZIP z folderu wskazanego przez -Path.
   - **Tworzy hasz**: Oblicza sumę kontrolną (hash) dla pliku ZIP przed przeniesieniem.
   - **Przenosi plik**: Przenosi plik ZIP do lokalizacji docelowej wskazanej przez -DestinationPath.

- **Weryfikuje integralność**: Porównuje sumę kontrolną pliku ZIP przed i po przeniesieniu, aby upewnić się, że plik nie został zmodyfikowany.
- **Dekompresuje plik**: Rozpakowuje plik ZIP w docelowej lokalizacji.

2. **Sprawdzenie wyników**: Po wykonaniu funkcji, można sprawdzić, czy operacje zostały przeprowadzone poprawnie, w tym:
   - Czy plik ZIP został poprawnie przeniesiony.
   - Czy integralność pliku została zachowana.
   - Czy plik ZIP został poprawnie rozpakowany w lokalizacji docelowej.

Wywołana funkcja wykona następujące operacje:

1. Skompresuje C:\SourceFolder do pliku ZIP.
2. Przeniesie ten plik ZIP do folderu \\127.0.0.1\c$\Temp na zdalnym komputerze.
3. Weryfikuje, czy plik ZIP nie został zmodyfikowany w trakcie transferu.
4. Rozpakowuje plik ZIP w \\127.0.0.1\c$\DestinationFolder.

To zapewnia, że zawartość folderu źródłowego została poprawnie przeniesiona i że integralność pliku została zachowana.