

Set-Alias

Das Cmdlet `Set-Alias` erstellt oder verändert einen Alias für ein Cmdlet, eine Funktion oder einen Befehl.

Ein Alias ist dabei einfach ein alternativer Kurzname für einen bestehenden Command.

Es wird häufig verwendet, um:

- häufig genutzte Befehle schneller aufzurufen
- eigene Kurzbefehle zu definieren
- bestehende Aliase umzubiegen oder anzupassen
- Shell-Umgebungen individueller zu gestalten

□ Syntax

```
Set-Alias [-Name] <String> [-Value] <String>  
        [-Description <String>]  
        [-Option <ScopedItemOptions>]  
        [-PassThru]  
        [-Scope <String>]  
        [-Force]  
        [-WhatIf]  
        [-Confirm]  
        [<CommonParameters>]
```

□ Parameter

-Name

- **Typ:** `String`
- **Pflicht:** Ja
- Name des Alias

```
Set-Alias -Name ll -Value Get-ChildItem
```

-Value

- **Typ:** String
- **Pflicht:** Ja
- Zielbefehl, auf den der Alias zeigen soll

```
Set-Alias -Name edit -Value notepad
```

-Description

- **Typ:** String
- **Pflicht:** Nein
- Fügt dem Alias eine Beschreibung hinzu

```
Set-Alias -Name gs -Value Get-Service -Description "Listet Dienste auf"
```

-Option

- **Typ:** ScopedItemOptions
- **Pflicht:** Nein
- Steuert das Verhalten des Alias

Mögliche Optionen:

Option	Bedeutung
None	Keine besondere Einschränkung
ReadOnly	Alias kann nur mit <code>-Force</code> geändert werden
Constant	Alias kann gar nicht mehr geändert werden
Private	Nur im aktuellen Scope sichtbar

```
Set-Alias -Name test -Value Get-Date -Option ReadOnly
```

-PassThru

- Gibt das erstellte Alias-Objekt zurück

```
Set-Alias -Name now -Value Get-Date -PassThru
```

-Scope

- Legt fest, in welchem Scope der Alias existiert

Beispiele:

Scope	Bedeutung
Local	Nur aktueller Scope
Global	Überall verfügbar
Script	Nur innerhalb des Skripts

```
Set-Alias -Name ll -Value Get-ChildItem -Scope Global
```

-Force

- Erzwingt das Überschreiben von `ReadOnly`-Aliases

```
Set-Alias -Name ls -Value Get-Process -Force
```

⚠ Hinweise zur Verwendung

- Aliase speichern **keine Parameter**
- Ein Alias ersetzt nur den Befehlsnamen
- Aliase gelten standardmäßig nur für die aktuelle Sitzung
- Für dauerhafte Aliase muss man sie ins Profil (`$PROFILE`) schreiben
- `Constant`-Aliases können nicht mehr entfernt werden

☐ Verhalten

Eigenschaft	Beschreibung
Rückgabewert	Standardmäßig keiner
Überschreibbar	Ja, außer <code>Constant</code>
Persistenz	Nur aktuelle Sitzung
Unterstützt Funktionen	Ja
Unterstützt EXE-Dateien	Ja

☐ Beispiele

Einfachen Alias erstellen

```
Set-Alias -Name ll -Value Get-ChildItem
```

Jetzt funktioniert:

```
ll
```

Alias für Programme

```
Set-Alias -Name np -Value notepad
```

Bestehenden Alias überschreiben

```
Set-Alias -Name ls -Value Get-Process -Force
```

Jetzt startet `ls` plötzlich Prozesse statt Dateien aufzulisten.

Ein hervorragender Weg, sich selbst drei Stunden später maximal zu verwirren.

Alias dauerhaft speichern

```
Add-Content -Path $PROFILE -Value 'Set-Alias -Name ll -Value Get-ChildItem'
```

Alias anzeigen

```
Get-Alias ll
```

⚙ Typische Anwendungsfälle

- Eigene Kurzbefehle erstellen
- Linux-ähnliche Befehle nachbauen
- Lange Befehlsnamen abkürzen
- Eigene Shell-Umgebungen konfigurieren
- Schnellzugriffe für Skripte

☐ Alternativen / Ergänzungen

New-Alias

```
New-Alias -Name ll -Value Get-ChildItem
```

Unterschied zu `Set-Alias`:

Cmdlet	Verhalten
<code>New-Alias</code>	Erstellt nur neue Aliase
<code>Set-Alias</code>	Erstellt oder überschreibt

Funktionen statt Alias

```
function ll {  
    Get-ChildItem -Force  
}
```

Vorteil:

- Kann Parameter enthalten
 - Deutlich flexibler
 - Besser für komplexe Logik
-

☐ Typische Fehler

1. Denken, dass Aliase Parameter speichern

```
# ☐ Funktioniert NICHT wie erwartet  
Set-Alias -Name ll -Value "Get-ChildItem -Force"
```

Ein Alias verweist nur auf einen Commandnamen.
Nicht auf eine komplette Befehlszeile.

→ Dafür nutzt man Funktionen.

2. Alias nach Neustart weg

```
Set-Alias -Name test -Value Get-Date
```

Nach neuer PowerShell-Sitzung verschwunden.

→ Alias ins `$PROFILE` schreiben.

3. Wichtige Standard-Aliase überschreiben

```
Set-Alias -Name cd -Value Get-Date
```

Technisch möglich.
Psychologisch fragwürdig.

☐☐ Best Practices

- Nur sinnvolle Kurzformen verwenden
- Standard-Aliase nicht leichtfertig überschreiben
- Für komplexe Logik lieber Funktionen nutzen
- Dauerhafte Aliase ins `$PROFILE` auslagern
- In Skripten sparsam mit Aliases umgehen, damit der Code lesbar bleibt

Revision #1

Created 2026-05-15 04:47:33 UTC by John-Andreas Borinas

Updated 2026-08-13 11:28:07 UTC by John-Andreas Borinas