Basic PowerShell Cmdlets

Abhishek K P

These basic PowerShell commands are helpful for getting information in various formats, configuring security, and basic reporting.

1 Get-Command

Get-Command is an easy-to-use reference cmdlet that brings up all the commands available for use in your current session.

2 Get-Help

The Get-Help command is essential for anyone using PowerShell, providing quick access to the information you need to run and work with all of the available commands.

If you wanted some examples, for instance, you'd enter the following :

$Get-Help[[-Name] < String >][-Path < String >][-Category < String[] >][-Component < String[] >]$
$[-Functionality < String[] >][-Role < String[] >][-Examples][< CommonParameters >]$

3 Set-ExecutionPolicy

Microsoft disables scripting by default to prevent malicious scripts from executing in the PowerShell environment. Developers want to be able to write and execute scripts, however, so the Set-ExecutionPolicy command enables you to control the level of security surrounding PowerShell scripts. You can set one of four security levels:

1.Restricted: This is the default security level which blocks PowerShell scripts from running. In this security level, you can only enter commands interactively.
2.All Signed: This security level allows scripts to run only if they are signed by a trustworthy publisher.
3.Remote Signed: In this security level, any PowerShell scripts that were created locally are permitted to run. Scripts created remotely are permitted to run only if they've been signed by a reputable publisher.
4.Unrestricted: As its name suggests, the unrestricted security level permits all scripts to run by removing all restrictions from the execution policy.

Similarly, if you're working in an unfamiliar environment, you can easily find out what the current execution policy is using this command:

```
Get-ExecutionPolicy
```

4 Get-Service

It's also helpful to know what services are installed on the system. You can easily access this information with the following command:

```
Get-Service
```

The output will look something like the following :

```
Status Name DisplayName
− − − − − − − − − − − − − − − − − − − − − − −
Running AdobeActiveFile... Adobe Active File Monitor V4
Stopped Alerter Alerter
Running ALG Application Layer Gateway Service
Stopped AppMgmt Application Management
Running ASChannel Local Communication Channel
```

If you need to know if a specific service is installed, you can append the -Name switch and the name of the service, and Windows will show the state of the service.
Additionally, you can leverage filtering capabilities to return a specific subset of currently installed services.
The following example will result in an output of data from the Get-Service command that's been piped to the Where-Object cmdlet, which then filters out everything other than the services that have been stopped:

```
Get − Service|Where − Object{$_.status − eq”stopped”}
```

5 ConvertTo-HTML

If you need to extract data that you can use in a report or send to someone else, the ConvertTo-HTML is one simple way to do so.
To use it, pipe the output from another command to the ConvertTo-HTML command and use the -Property switch to specify which output properties you want in the HTML file. You'll also need to provide a file name.

For example, the following code creates an HTML page that lists the PowerShell aliases in the current console:

```
PS C : > get − alias|convertto − html > aliases.htm
PS C : > invoke − itemaliases.htm
```

Bonus:

The Export-CSV cmdlet functions in much the same way, but exports data to a .CSV file rather than HTML.
Use Select-Object to specify which properties you want to be included in the output.

6 Get-EventLog

You can actually use PowerShell to parse your machine's event logs using the Get-EventLog cmdlet.
There are several parameters available.
Use the -Log switch followed by the name of the log file to view a specific log.
You'd use the following command, for example, to view the Application log:

```
Get-EventLog -Log "Application"
```

1.-Verbose.
2.-Debug.
3.-ErrorAction.
4.-ErrorVariable.
5.-WarningAction.
6.-WarningVariable.
7.-OutBuffer.
8.-OutVariable.

7 Get-Process

Much like getting a list of available services, it's often useful to be able to get a quick list of all the currently running processes.
The Get-Process command puts this information at your fingertips.

Bonus: Use Stop-Process to stop processes that are frozen or is no longer responding.
If you're not sure what process is holding you up, use Get-Process to quickly identify the problematic process.
Once you have the name or process ID, use Stop-Process to terminate it.

Here's an example.
Run this command to terminate all currently running instances of Notepad :

Stop-Process -processname notepad You can use wildcard characters, too, such as the following example which terminates all instances of Notepad as well as any other processes beginning with note:

```
Stop-Process -processname note*
```

8 Clear-History

What if you want to clear the entries from your command history? Easy – use the Clear-History cmdlet. You can also use it to delete only specific commands. For example, the following command would delete commands that include "help" or end in "command":

$PS\ C :> Clear - History - Command * help*, *command$ If you want to add entries to a session, use:

Add-History

9 Where-Object

Where-Object is one of the most important cmdlets to know, as it enables you to take a dataset and pass it further down your pipeline for filtering :

$Get - Service|Where - Object\{\$\_.Status - eq'Running'\}$

```
Status Name DisplayName
− − − − − − − − − − − − − − − − − − − − − − −
Running AdobeARMservice Adobe Acrobat Update Service
Running AppHostSvc Application Host Helper Service
Running Appinfo Application Information
Running AudioEndpointBu... Windows Audio Endpoint Builder
Running Audiosrv Windows Audio
Running BFE Base Filtering Engine
Running BITS Background Intelligent Transfer Ser...
Running BrokerInfrastru... Background Tasks Infrastructure Ser...
Running Browser Computer Browser
Running CDPSvc Connected Devices Platform Service
```

10 Set-AuthenticodeSignature

If you want to keep your work secure in production and prevent modification, use Set-AuthenticodeSignature to add an Authenticode signature to a script or file.

$> Set-AuthenticodeSignaturesomescript.ps1@(Get-ChildItemcert : \backslash CurrentUser\backslash My-codesigning)[0]-IncludeChain"All"-TimestampServer"http : //timestamp.verisign.com/scripts/tims$