How-to: Run a PowerShell script

There are several ways to run a PowerShell script.

Before running any scripts on a new PowerShell installation, you must first set an appropriate Execution Policy,
e.g. `Set-ExecutionPolicy RemoteSigned`

If the script has been downloaded from the internet and saved as a file then you may also need to right click on the script, select properties, then `unblock`. If you just copy and paste the text of the script, this is not needed.

A PowerShell script is the equivalent of a Windows CMD or MS-DOS batch file, the file should be saved as plain ASCII text with a .ps1 extension, e.g. `MyScript.ps1`

Call or Invoke a script to run it

The most common (default) way to run a script is by *calling* it:

```
PS C:\> & "C:\Batch\My first Script.ps1"

PS C:\> & cscript /nologo
"C:\Batch\another.vbs"
```

If the path does not contain any spaces, then you can omit the quotes and the '`&`' operator

```
PS C:\> C:\Batch\Myscript.ps1
```

If the script is in the current directory, you can omit the path but must instead explicitly indicate the current directory using `.\` (or `./` will also work)

```
PS C:\> .\Myscript.ps1
```

An important caveat to the above is that the currently running script might not be located in the current directory.

Call one PowerShell script from another script saved in the *same* directory:

```
#Requires -Version 3.0
& "$PSScriptRoot\set-consolesize.ps1" -
height 25 -width 90
```

When you *invoke* a script using the syntax above, variables and functions defined in the script will disappear when the script ends.[1]

An alternative which allows running a script (or command) on local or remote computers is Invoke-Command

```
PS C:\> invoke-command -filepath
c:\scripts\test.ps1 -computerName Server64
```

[1]unless they are explicitly defined as globals: `Function SCOPE:GLOBAL` or `Filter SCOPE:GLOBAL` or `Set-Variable -scope "Global"`

Run a PowerShell Script from the GUI or with a shortcut

This can be done by running PowerShell.exe with parameters to launch the desired script.

Run As Administrator (Elevated)

See the PowerShell elevation page for ways of running a script or a PowerShell session "As admin"

Dot Sourcing

When you *dot source* a script, all variables and functions defined in the script will persist even when the script ends.

Run a script by *dot-sourcing* it:

```
PS C:\> . "C:\Batch\My first Script.ps1"
```

Dot-sourcing a script in the current directory:

```
PS C:\> . .\Myscript.ps1"
```

Run a CMD batch file

Run a batch script from PowerShell:

```
PS C:\> ./demo.cmd
```

Early versions of PowerShell would only run *internal* CMD commands if the batch file was run by explicitly calling the CMD.exe shell and passing the batch file name.

Run a single CMD internal command

This will run the CMD.exe version of DIR rather than the powershell DIR alias for Get-ChildItem:

```
PS C:\> CMD.exe /C dir
```

Run a VBScript file

Run a vb script from PowerShell:

```
PS C:\> cscript c:\batch\demo.vbs
```

The System Path

If you run a script (or even just enter a command) without specifying the fully qualified path name, PowerShell will search for it as follows:

1. Currently defined aliases
2. Currently defined functions
3. Commands located in the system path.

#Yeah, I'm gonna run to you, cause when the feelin's right I'm gonna stay all night, I'm gonna run to you# ~ Bryan Adams

Related PowerShell Cmdlets:

#requires - Prevent a script from running without a required element.
Basic PowerShell script Template - HowTo.
Invoke-Command - Run commands on local and remote computers.
Invoke-Expression - Run a PowerShell expression.
Invoke-Item - Invoke an executable or open a file (START).
The call operator (&) - Execute a command, script or function.
Set-Variable - Set a variable and its value.
Functions - Write a named block of code.
CMD Shell: Run a PowerShell script from the CMD shell.
VBScript: Run a script from VBScript