Reboot and Resume PowerShell **Script**

Lasse W, 8 Jul 2011

★★★★★ 4.71 (9 votes) Rate:

A first step in creating that magic "Developer PC Setup **Script**".

Download source - 1.43 KB

Introduction

Gone are the days where daily reboots of Windows were needed. However, when dealing with system maintenance or installing multiple applications from **script**, rebooting is sometimes unavoidable. This article will describe a PowerShell **script** capable of running through a number of steps, rebooting the computer, and continuing the **script** at another step. Simple, yet very useful.

Background

The ideas shown in this article stem from a real life **script** for setting up a developer PC with everything from the right version of the .NET Framework, enabling the right features in IIS, installing Visual Studio and MS SQL Server locally, plus a lot more. The advantages of setting up a developer PC in this way are many:

- It is a lot faster than letting every developer go through the steps themselves
- It is a lot more flexible than using a disc image
- It documents very clearly which technologies are used
- It can be used for updating existing developer PCs as well as installing from scratch

The main disadvantage of the above mentioned setup **script** is the upfront time investment in getting such a **script** in place. This article will however help you to get started and you should really take up the challenge and try creating your own **script** - you will be impressed with just how powerful Power**Script** really is.

Running the Test **Script**

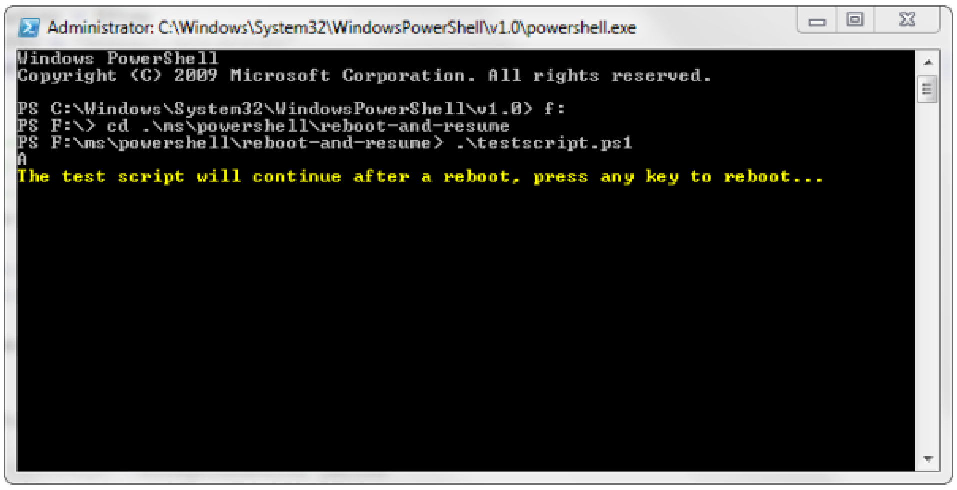
The simple test **script** covered in this article is shown below:

Hide Shrink ▲ Copy Code

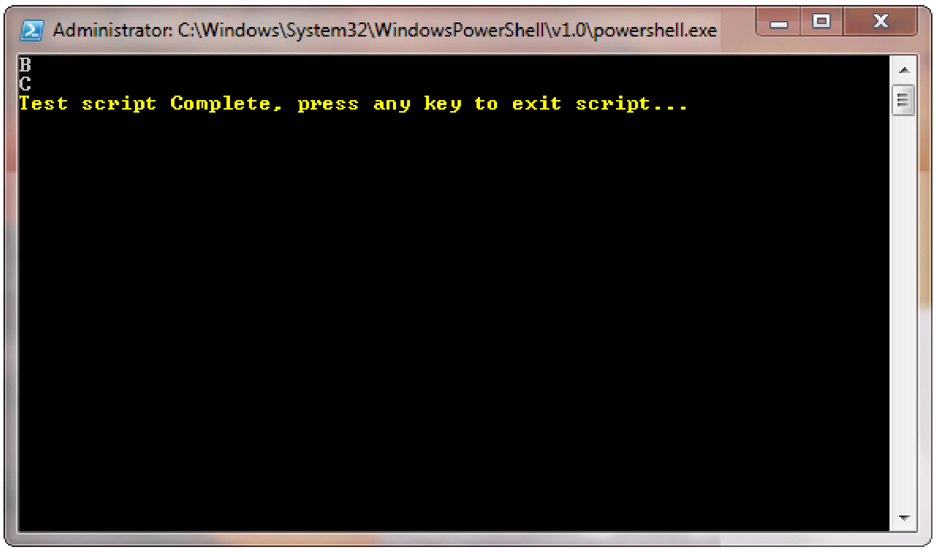
```
param($Step="A")
# -------------------------------------
# Imports
# -------------------------------------
$script = $myInvocation.MyCommand.Definition
$scriptPath = Split-Path -parent $script
. (Join-Path $scriptpath functions.ps1)


Clear-Any-Restart

if (Should-Run-Step "A")
{
    Write-Host "A"
    Wait-For-Keypress "The test script will continue after a reboot, press any key to reboot..."
    Restart-And-Resume $script "B"
}

if (Should-Run-Step "B")
{
    Write-Host "B"
}
```

```
if (Should-Run-Step "C")
{
    Write-Host "C"
}

Wait-For-Keypress "Test script Complete, press any key to exit script..."
```

The **script** will execute step A through C, only when done with step A, it will reboot, and then continue at step B. Below are two screenshots showing what to expect:


Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Windows\System32\WindowsPowerShell\v1.0> f:
PS F:\> cd .\ms\powershell\reboot-and-resume
PS F:\ms\powershell\reboot-and-resume> .\testscript.ps1
A
The test script will continue after a reboot, press any key to reboot...
```


Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

```
B
C
Test script Complete, press any key to exit script...
```

Making the **Script** Step Aware

Making a PowerShell **script** to execute in sequential steps can be done in a number of ways. The only thing required in this example is that the starting step must be specified on the command line, as in:

Hide Copy Code

```
>.\testscript.ps1 -Step B
```

As seen from the test **script** above, we declare a **script** parameter `$Step` for taking in the starting step from the command line. We then create a simple guard method `Should-Run-Step` which will return false until the starting step specified on the command line is encountered, and from there on, it will return true:

Hide Copy Code

```
$global:started = $FALSE
$global:startingStep = $Step

function Should-Run-Step([string] $prospectStep)
{
    if ($global:startingStep -eq $prospectStep -or $global:started) {
        $global:started = $TRUE
    }
    return $global:started
}
```

This stepping mechanism could of course be way more sophisticated, including endpoints and step ranges, but that is not really the focus of the article and hence left as an exercise for the reader.

A Few Registry Helper Functions

As much as I like the super general `???-ItemProperty` functions of PowerShell, I tend to think these methods can clutter the **script**s and make them less readable. Therefore I will introduce four simple helper methods for dealing with the Registry:

Hide Copy Code

```
function Test-Key([string] $path, [string] $key)
{
    return ((Test-Path $path) -and ((Get-Key $path $key) -ne $null))
}

function Remove-Key([string] $path, [string] $key)
{
    Remove-ItemProperty -path $path -name $key
}

function Set-Key([string] $path, [string] $key, [string] $value)
{
    Set-ItemProperty -path $path -name $key -value $value
}

function Get-Key([string] $path, [string] $key)
{
    return (Get-ItemProperty $path).$key
}
```

Rebooting from PowerShell

Restarting a computer from PowerShell is done simply with the command `Restart-Computer`. Making Windows execute a command on startup is handled via the Registry key "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run". Putting this information together leads us to define a `Restart-And-Run` function which will reboot and then launch whatever is passed in the `$run` parameter.

Hide Copy Code

```
$global:RegRunKey ="HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
function Restart-And-Run([string] $key, [string] $run)
{
    Set-Key $global:RegRunKey $key $run
    Restart-Computer
    exit
}
```

If you do not want to launch the command specified every time you start Windows, you better remember to clear that Registry key. As we plan to resume our **script**, we can easily do that in a dedicated function which is to be placed at the top of our **script**.

Hide Copy Code

```
$global:restartKey = "Restart-And-Resume"
function Clear-Any-Restart([string] $key=$global:restartKey)
{
    if (Test-Key $global:RegRunKey $key) {
        Remove-Key $global:RegRunKey $key
    }
}
```

As you may have noticed, the `Clear-Any-Restart` function takes a default key of "Restart-And-Resume", and resuming the **script** was what this article set out to accomplish. Guess what, all we need to do is restart PowerShell with the wanted step parameter:

Hide Copy Code

```
$global:powershell = (Join-Path $env:windir "system32\WindowsPowerShell\v1.0\powershell.exe")
function Restart-And-Resume([string] $script, [string] $step)
{
    Restart-And-Run $global:restartKey "$global:powershell $script -Step $step"
}
```

That is it, we are all done, you are now ready to make your own setup **script**s. Enjoy, and please do let me know what you think and what crazy things you use this for.

Troubleshooting

If you are having trouble executing the **script**, make sure you have not specified the `restricted` execution policy:

Hide Copy Code

```
>Set-ExecutionPolicy RemoteSigned
```

History

- July 08, 2011: Initial version.

License

Share

TWITTER FACEBOOK

About the Author

Lasse W
Denmark 🇩🇰

No Biography provided

You may also be interested in...

Choosing a Hosting Environment – Linux vs Windows

Intel® Cyclone® 10 LP FPGA Board - How to Program Your First FPGA

Generating JSON Web Services from an Existing Database with CodeFluent Entities

SAPrefs - Netscape-like Preferences Dialog

Get Started: Intel® Cyclone® 10 LP FPGA kit

Window Tabs (WndTabs) Add-In for DevStudio

Comments and Discussions

Add a Comment or Question Search Comments

First Prev Next

Really nice script. Helped me a lot!
Member 12917391 22-Dec-16 4:58

Thanks for the script.
I've a short question. Is there an option to split those scripts? I mean to hold one file, not 2 separated (testscript+functions) ?

Reply · Email · View Thread

Doesn't work on Client computer - Need admin rights without UAC.
Guillaume T. 23-Feb-16 8:39

Hi,

Your script work very well, but only on a server environment with the predefined Administrator account.. Because UAC protects your script from being executed as Admin, this workaround just doesn't work with Windows 8.1 & other Client OS ..

Is there a way to execute a script with elevated permissions at startup ?

Reply · Email · View Thread

The Reboot and Resume Powershell Script runs the next step after logging in to the server. I want the script to keep running without actually logging in.

Member 12042865 14-Oct-15 3:44

I am following the "Reboot and Resume PowerShell Script" for AD and Exchange Installation. The script works fine but it runs the next step after logging in to the server. I want the script to continue running even if I do not actually login to the server after reboots. Is this possible? Any help would be appreciated.

Article: Reboot and Resume PowerShell Script[^]

Reply · Email · View Thread

Resume Script Under Different Login?

Member 10742736 11-Apr-14 13:41

I am just confirming that after I join a domain (which is included in my Step A here with a rename computer step) that I can login as a different user (now that AD users will be available) and resume the script under that login?

Please advise.

Thank you,

Reply · Email · View Thread

Re: Resume Script Under Different Login?

Member 10742736 11-Apr-14 13:54

Updating this, I do see the script resumes when I login under a different user. That is great!

I have one last piece of completing the autologin on reboot under that user and that should do it - except of course reviewing loose creds and moving to a databag of some soft.

Great post!

Reply · Email · View Thread

Any way to jump to specific steps?

Member 9587060 19-Feb-14 10:55

Thanks for publishing this! I was wondering if there was a way to jump to a specific step in the script ex. 'Should-run-Step D', without the need for a reboot? Similar to the GOTO function in batch.

Reply · Email · View Thread

UAC

rome7777 20-Nov-12 16:43

After reboot is there a way for the Administrator PowerShell instance to run?

Reply · Email · View Thread

My vote of 5

hermiodinator 30-Apr-12 16:51

I've been trying to figure out how to do exactly this, this has so many uses. Thanks a lot!

Reply · Email · View Thread

Just wanted to say thanks

Marin Marušić 26-Nov-11 16:58

I used your script in a larger project intended for replacing ADMT for workstations, providing 3 step disjoin, rename, network params update, and join a new domain.

Many thanks for a good idea to make a step aware PS script, it saved me a lot of time.

Reply · Email · View Thread

Re: Just wanted to say thanks

Lasse W 28-Nov-11 14:36

Glad you liked it.

Reply · Email · View Thread

My vote of 5
Member 4320844 15-Jul-11 18:27

power Shell scripts for administrators .... thanks.

Reply · Email · View Thread

Refresh 1

General News Suggestion Question Bug Answer Joke Praise Rant Admin