Windows PowerShell & SharePoint Better Together

Introduction and Hands on Guide

ABSTRACT

This document is written for developers who would like to get started with writing PowerShell script. It also include examples of using PowerShell 3.0 with SharePoint 2013, Office365 and Active Directory.

Jerry Yasir

SharePoint Server MVP, MCT

Table of Contents

About the Author

Jerry Yasir is a SharePoint Server MVP since 2010 and works as a Senior SharePoint Architect at US Tech Solutions based on Jersey City, New Jersey. He holds Master degree in Computer Science with over 12 years of experience working on Microsoft Technologies specially SharePoint, Microsoft .NET & other Information Worker Technologies. He is currently leading SharePoint teams in various projects and is responsible for architecting, designing & installing SharePoint farms. He specializes in providing advanced SharePoint administration and development training to teams and customers. He is Microsoft Certified Trainer (MCT), MCSE SharePoint 2013, and MCPD SharePoint 2013 (Still Waiting Results). He also holds MCPD, MCTIP, MCTS for SharePoint 2010, MCTS MOSS 2007 & WSS 3.0, Silverlight 4, MCPD, MCITP (EPM 2010 & 2007) and MCSD.NET. Jerry formed once of the biggest SharePoint User Group in middle with over 2000 members, he speaks in various user groups, conferences and online and offline events in US and middle-east.

Blog: http://jerryyasir.wordpress.com
Twitter: @jerryyasir
LinkedIn: http://www.linkedin.com/in/yasirbutt

This book has not been reviewed by anyone so please ignore or identify any spelling mistakes by sending an email to Yasir.attiq@gmail.com.

The lab type, step by step guide is written for SharePoint community and sharing purpose only and is not intended for sale. Please do not pay anyone if you get this book.

The document will be updated continuously as there are so many other cmdLets that can use in SharePoint.

US Tech Solutions

USTECH is a global firm providing a wide-range of talent on-demand and total workforce solutions.

Through the **USTECH** Talent Network of 100% company-owned and managed offices, we provide highly-skilled professionals whose education, skills and experience are vetted and matched to your unique hiring needs, work environment and company requirements.

Our 24x7 global service delivery drives time and cost out of any recruiting and staffing process (15-30% cost reduction in most cases) across all of our services and solutions, providing you with the talent you need on-demand – when, where and how you need it.

Learn more about **USTECH. Or Contact US**

Learning Programs

Microsoft

Microsoft Server
- Biztalk
- Exchange
- Lync
- SQL
- SharePoint
- Windows Server OS
- Powershell
- System Center/Private Cloud

Operating Systems
- Windows Client OS
- Windows Mobile

Microsoft Office Suite
- Word
- Excel
- PowerPoint
- Access
- Outlook

Developer Tools & Apps
- Visual Studio / .Net

Design
- Silverlight

Microsoft Dynamics
- AX, GP, CRM, Navision

Virtualization
- Hyper-V

Cloud Computing
- Azure

Adobe
- InDesign
- Acrobat
- Photoshop
- Flash
- Dreamweaver
- Illustrator
- Premiere Pro
- After Effects

PMI
- Project Mgmt Professional (PMP)

Autodesk
- AutoCAD
- Revit

Novell
- SUSE Linux

Citrix
- XenApp
- Xendesktop
- XenServer
- NetScaler

ITIL
- Foundation Certification
- Lifecycle Management
- Capability Management

CompTIA
- A+
- Network+
- Security+
- CTT+
- Linux+
- Healthcare IT

Check Point
- Check Point Certified Security Administrator (CCSA)
- Check Point Certified Security Expert (CCSE)

Oracle / Sun
- Oracle DBA
- Java
- Solaris
- My SQL

Soft Skills
- Computing Fundamentals
- Time Management
- Effective Communication
- Customer Service
- Leadership
- Management
- MBA Essentials

VMSources
- Virtualization Training

Cisco
- CCENT
- CCNA
- CCNA Security
- CCNA Voice
- CCDA
- CCNP
- CCNP Security
- CCNP Voice

EC-Council
- Certified Ethical Hacker (CEH)
- Computer Hacking Forensic Investigator (CHFI)

Apple/Google
- C++
- Website Development Pro
- Mobile Development Pro

CAP / CISSP
- Certified Information System Security Professional (CISSP)
- Certification & Accreditation Professional (CAP)

Red Hat
- Core System Administration
- Advanced System Administration
- Linux Development
- JBoss Enterprise Middleware

Key: Technical Disciplines

Help Desk	Development
Networking	Design
Security	Drafting
Virtualization	Business Prod.
Database	PM / Soft Skills

Learn more about training & certification:

Call Us: (888) 563-8266
http:// NetComLearning.com

Getting Started with Windows PowerShell

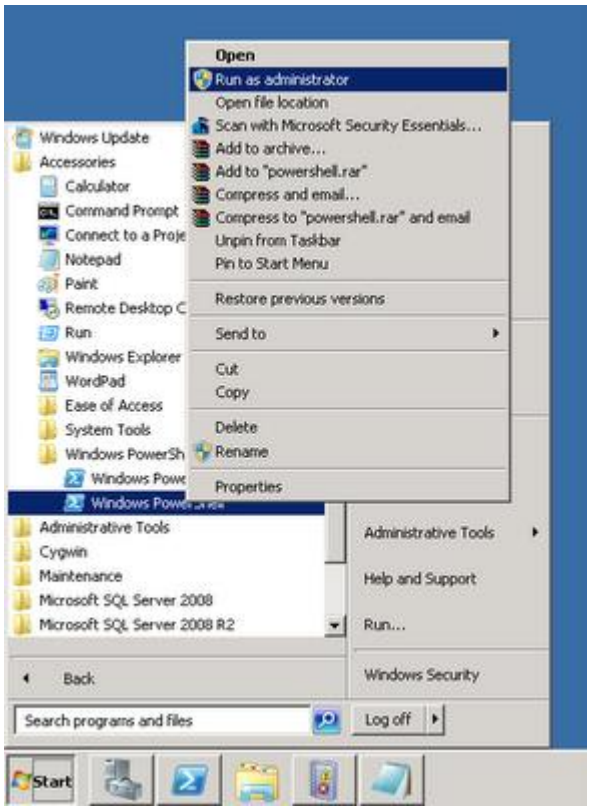
Windows PowerShell Basics

Starting Windows PowerShell Console

1. Log on to your SharePoint VM or Environment.

Windows Server 2008 R2

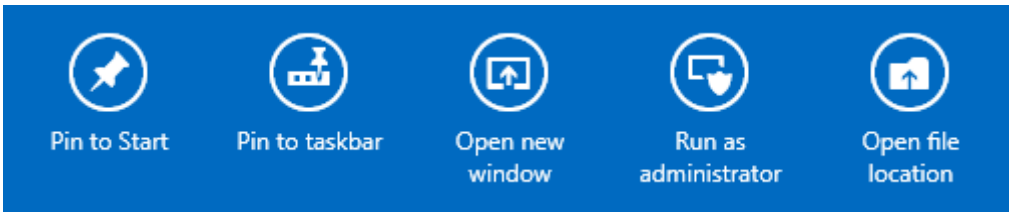
2. Click Start → All Programs → Accessories → Windows PowerShell → Right Click on **Windows PowerShell** and Choose **"Run as Administrator"**



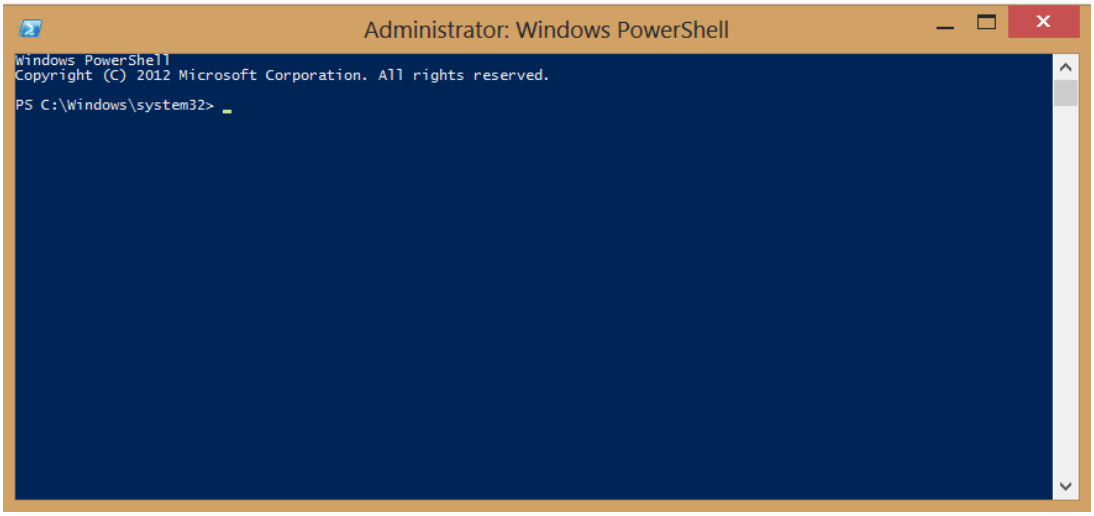
Windows Server 2012

3. Press Windows + C and click on Search Charm and type **PowerShell** and Right Click on **Windows PowerShell**

From Common Tasks Pane click on **Run as Administrator**

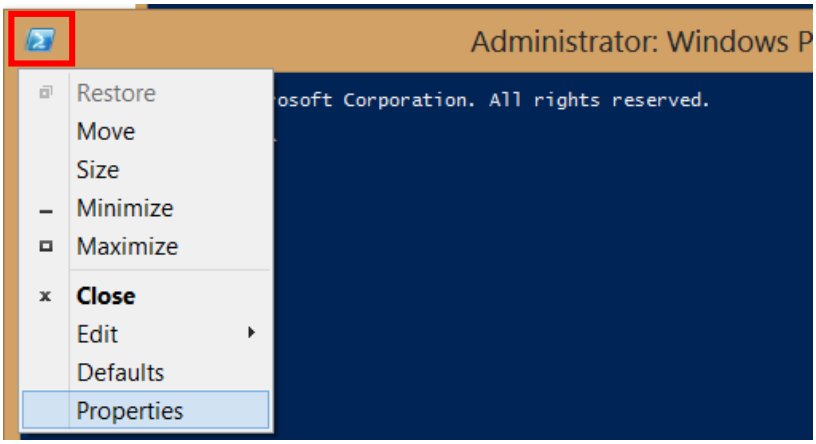


Windows PowerShell® will open and will look like this.

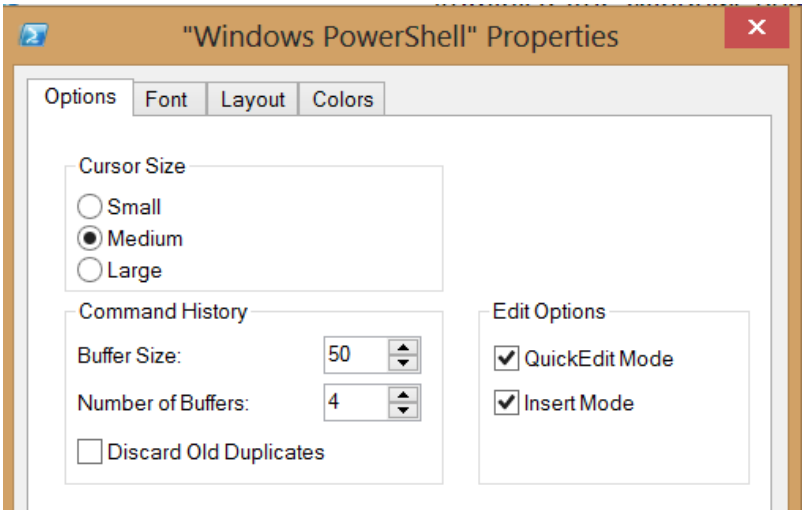


Setting Common Properties of PowerShell (Fonts, Cursor)

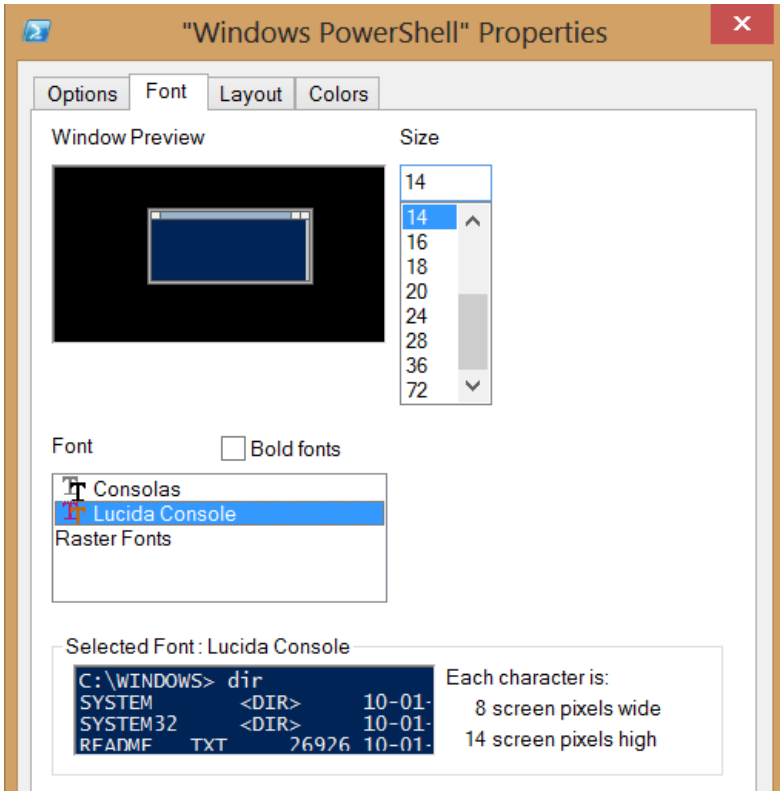
1. Click on the PowerShell **ICON** on the top left of Windows and Choose **Properties**.



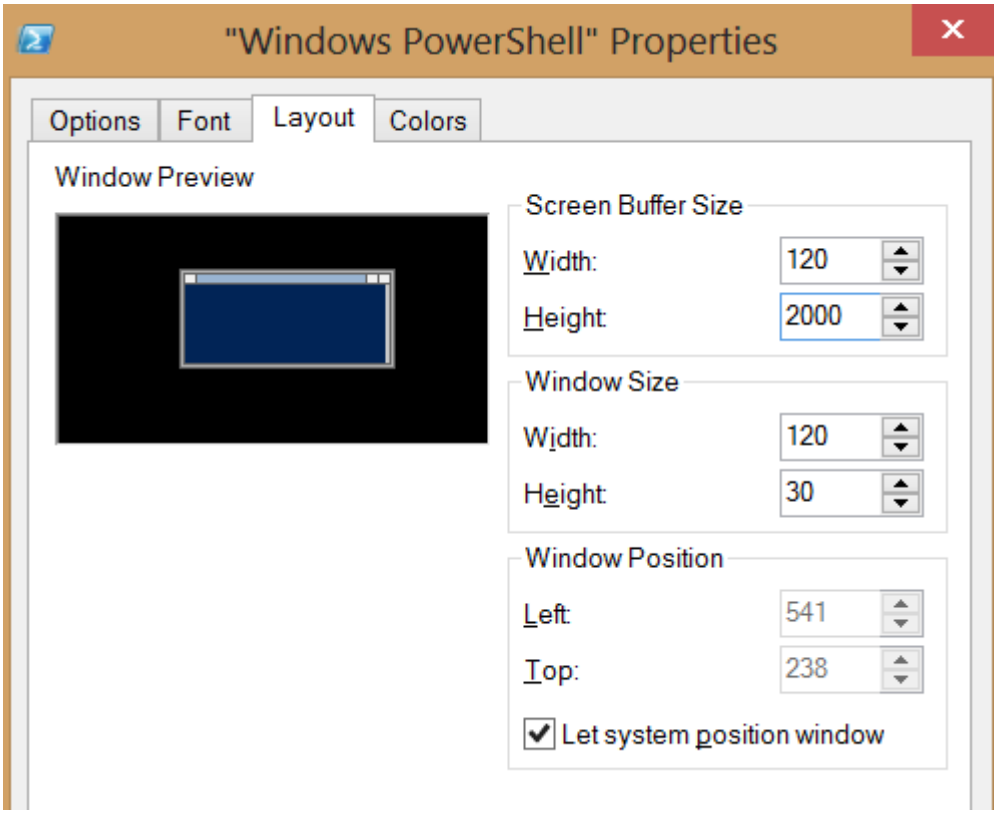
2. Set the Cursor Size to Medium.
3. Notice the Edit Options
   a. **Quick Edit**
   This mode enables you to use the mouse to copy and paste within the PowerShell window. You can use mouse to copy existing command and right click to Paste.
   Note: Quick Edit Mode is turned Off in SharePoint Management Shell.
   b. **Insert Mode**
   This mode enables you to insert text when typing by overwriting the exiting value.



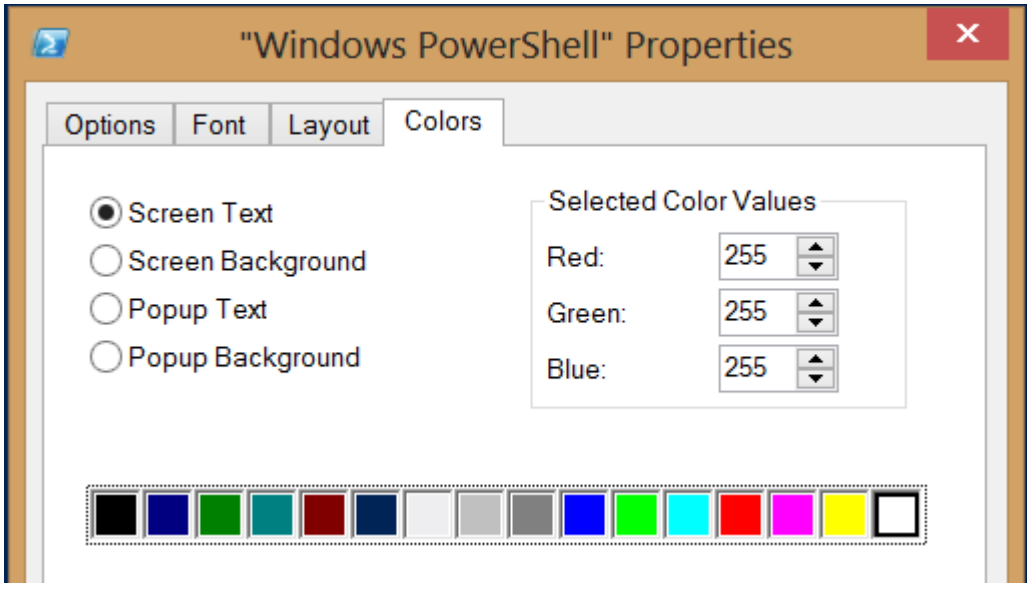
4. Click on **Font** Tab and Set **Size** as 14.



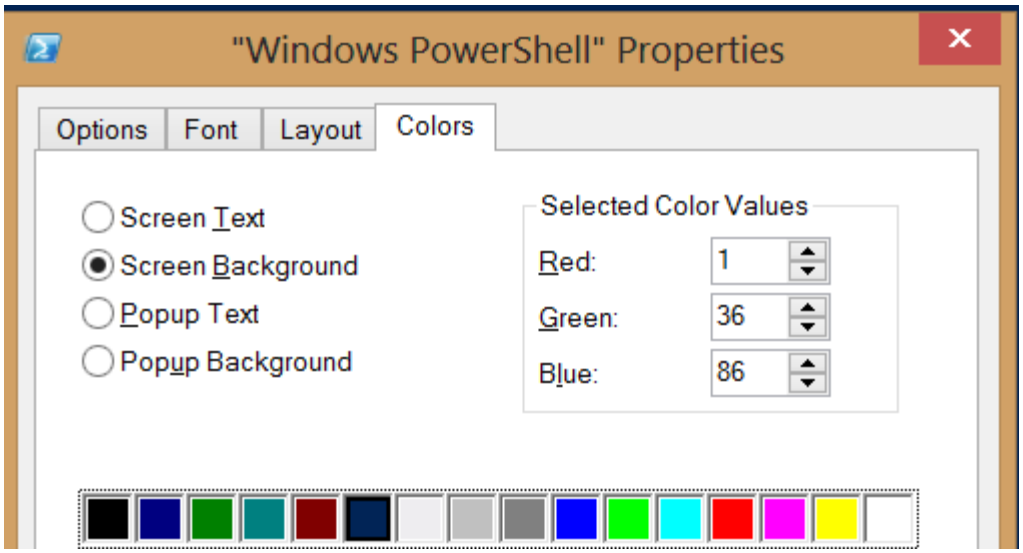
5. Click on **Layout** Tab and Screen Buffer Size Set Height to **3100** and Screen Buffer Size Width to **150**.
    a. Screen Butter Size Width will enable you to set number of characters for the buffer.
    b. Screen Buffer Size Height will allow you to see extended outputs without breaking.



6. Click **Colors** Tab.
   a. Select Screen Text and Choose White as Color.



   b. Select Screen Background and Choose Dark Blue



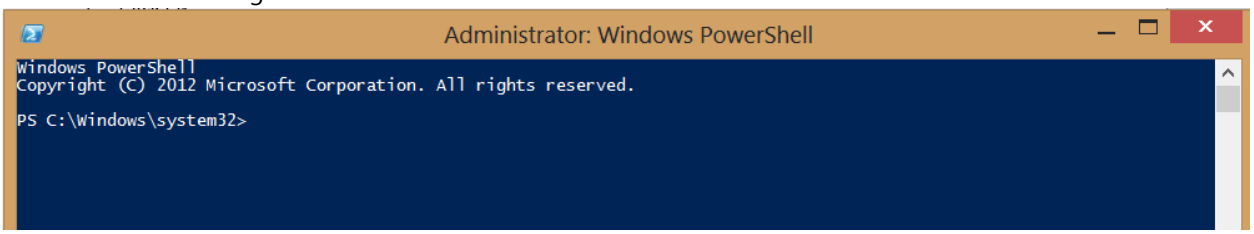
7. Click OK.

Windows Server 2008

When you're finished making all your changes click OK; that will bring up the following dialog box:

a. Select Modify shortcut that start this window and click OK.

8. Observe the Changes.



For more information please check out http://technet.microsoft.com/en-us/library/ee156814.aspx

PowerShell Commands Basics

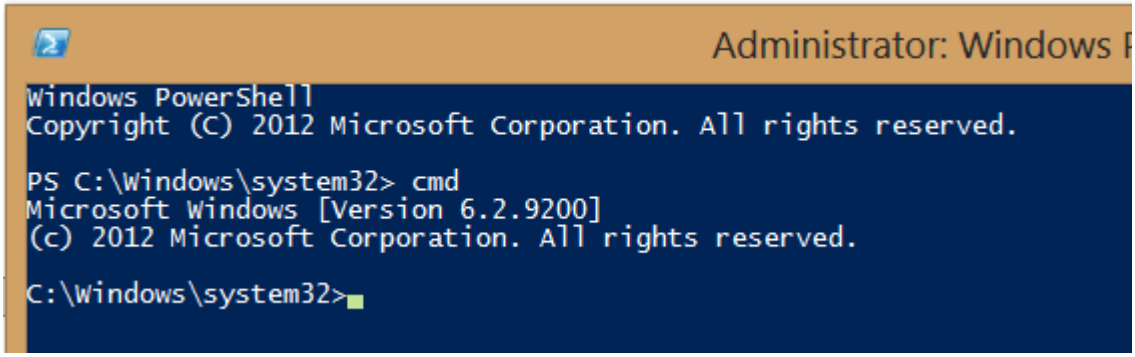
PowerShell Commands are commonly referred as "**cmdLets**" which are based on same concepts as Object, Properties and Methods. Cmdlets are mostly like any other commands that you can write on other command line tools. PowerShell command cannot be executed at any other location but some of the most common commands like cd, cmd, dir, ping, ipconfig are still available in PowerShell. Some of these commands have their PowerShell predecessor command. But some other command might not be available to you in PowerShell window that are normally available in command prompt.

Note: PowerShell commands are typically case-insensitive.

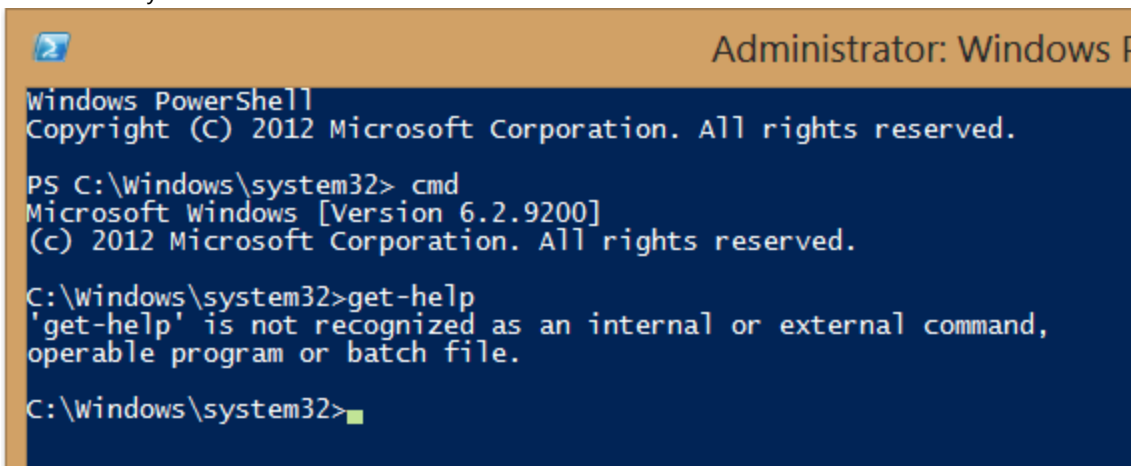
PowerShell and Command Prompt commands

In that case you can still use PowerShell Window to write normal Commands. But you have to Load the **cmd**. Once you are in command line mode you will not be able to write PowerShell cmdLets.

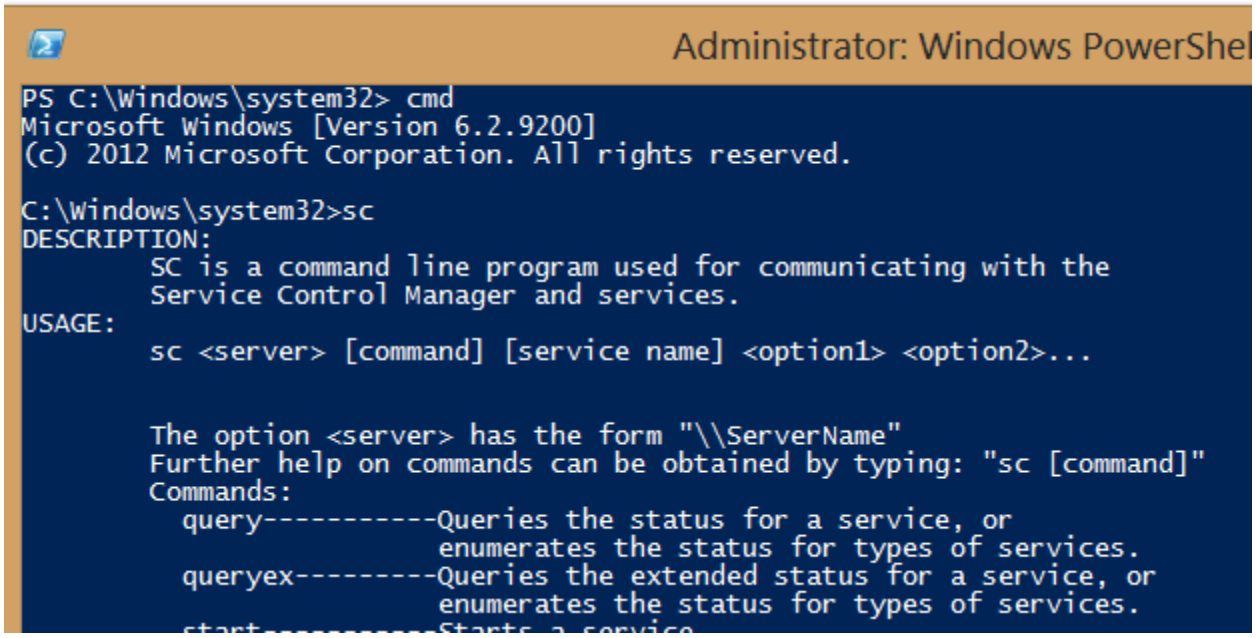
1. Go to your PowerShell Window that is already open and type "cmd" and press Enter



2. Now write Get-Help and press enter "You should get an error that this command is not recognized because you are now in command window."

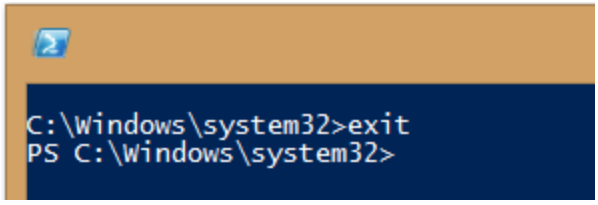


3. Now write sc and press Enter and examine the output.



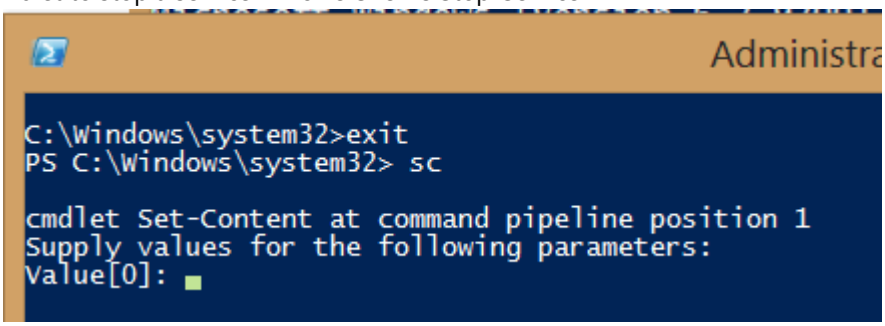
You can see that you are able to write a command in PowerShell. Let's Exit now and try to run this command from PowerShell.

4. Type **cls** and press enter to clear the window contents.
5. Now type **Exit** and press enter key.

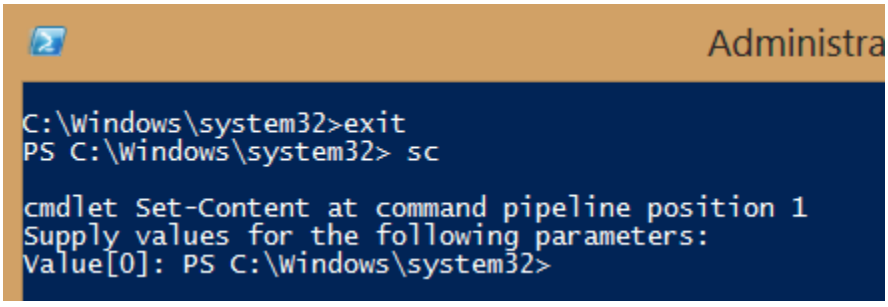


6. Now type **sc** and press enter key. This will execute set-content cmdlet which is not the right command.

Note: The cmdlet to stop a service in PowerShell is Stop-Service



7. Now Press **Ctrl + C** to cancel the command.



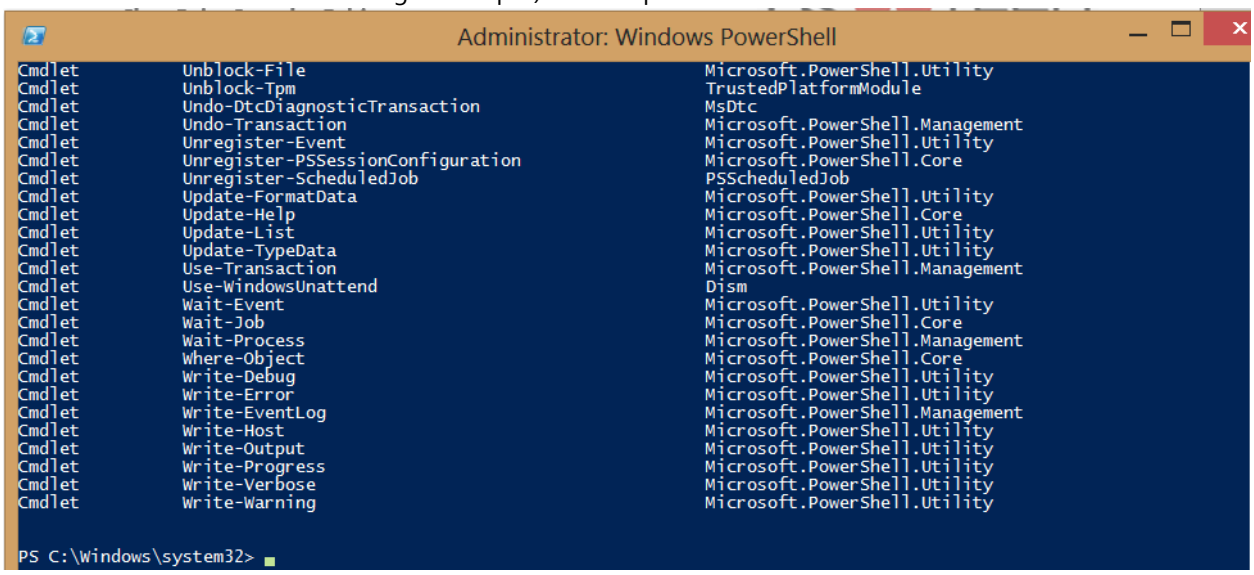
Writing your First cmdLets

Cmdlet support a normalized naming scheme with the VERB-NOUN syntax. These are compiled code that are available to the PowerShell environment. Some of the common Verbs are Get, Set, Delete, Create, New, Remove, Stop, Start, Write, Read. On the other than some verbs are Process, Computer, Event, Job, File, Object, Host etc.

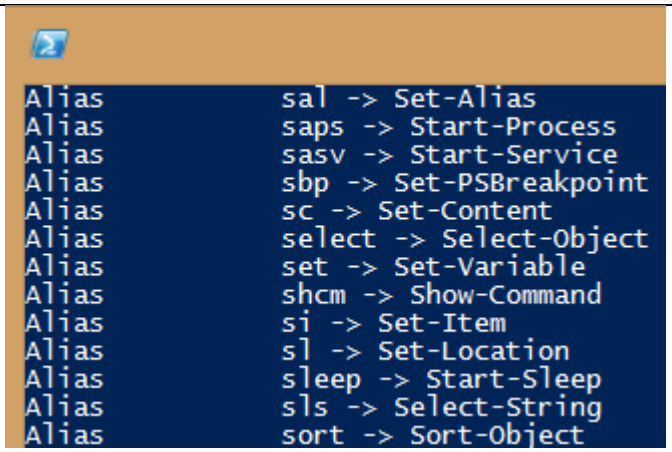
By looking concepts mentioned above it is not difficult to understand the basic concepts of cmdLets. Now if we want to **Get** all the **Commands** that are available in PowerShell so we need to run a cmdlet. The command will be made as **Get** then a **–** or **hyphen** or **dash** and then **Command.** The Actual command will look like **Get-Command.**

Note: cmdLets do not use **spaces** in their names. The parameters and value will be separated by space and of parameter value with space must be between double "C:\Program Files" or single 'C:\Program Files' quotes.

1. Now go to your Windows PowerShell window and type Get-Command and press enter (Wait for command to finish showing the output). Scroll up to view some of the commands.



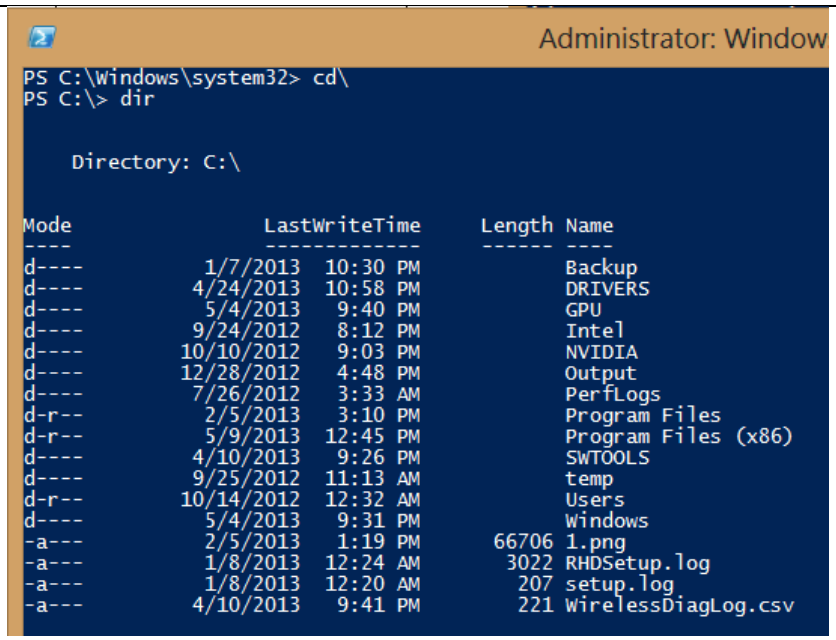
2. Now try some other commands.

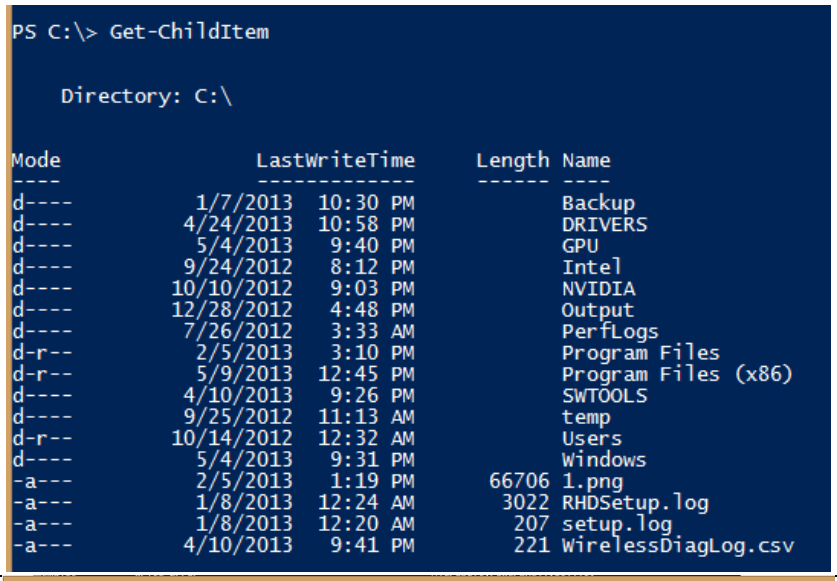
Get-Alias This command will show you the short name for cmdLets that you can use.	

Type **CD** and press Enter
Now type **dir** and press Enter
Now type **Get-ChildItem** and press Enter

Note: The **Alias** for Get-ChillItem is dir

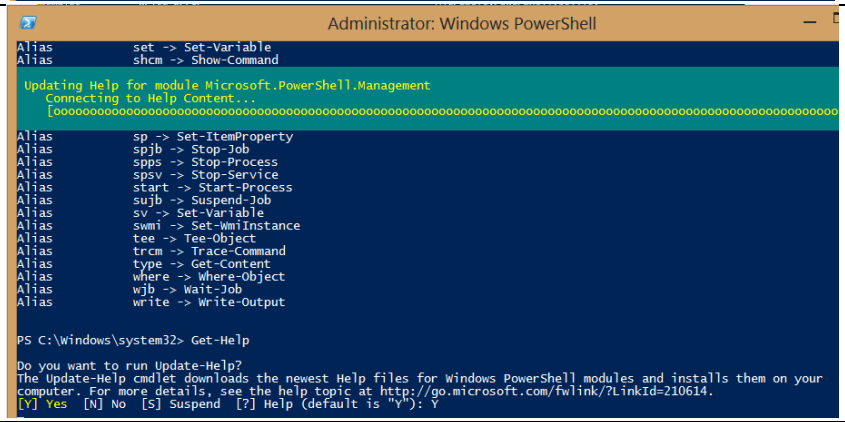
You can add your Own Alias.
Set-Alias -name list -value get-childitem

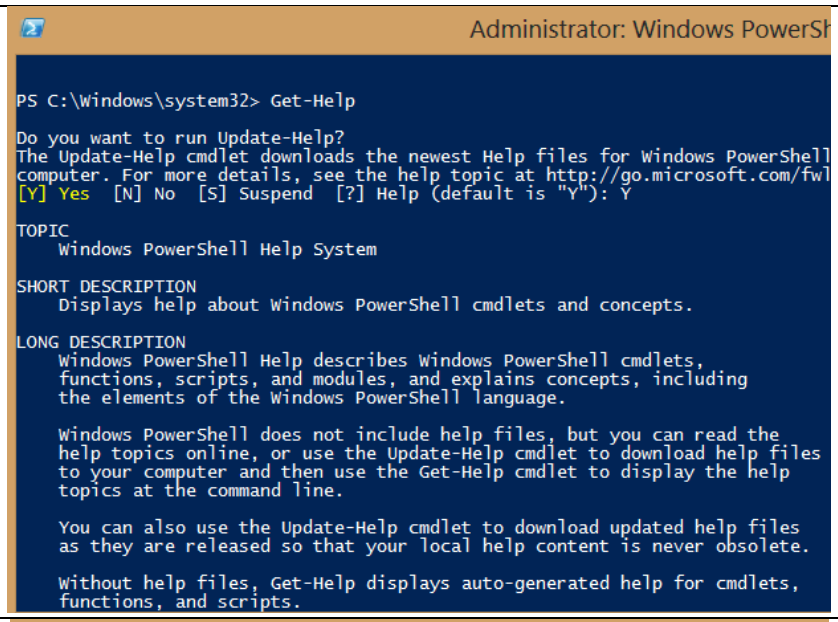
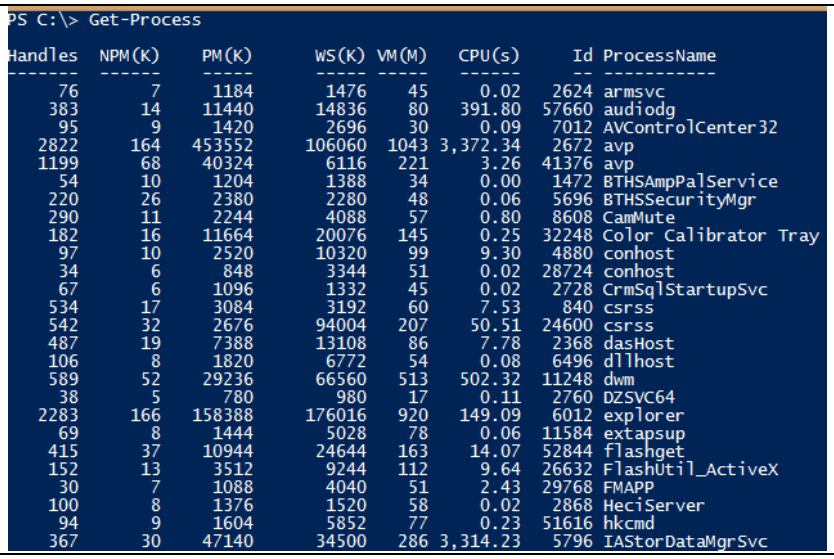




Get-Help
Note: If you are connected to Internet type **Y** for PowerShell to download the latest help files.

Download Latest Help
Latest Help Documents is a feature that is added in PowerShell 3.0



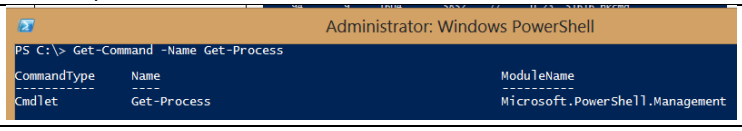
	
Get-Process This command will get all the running processes from the computer.	

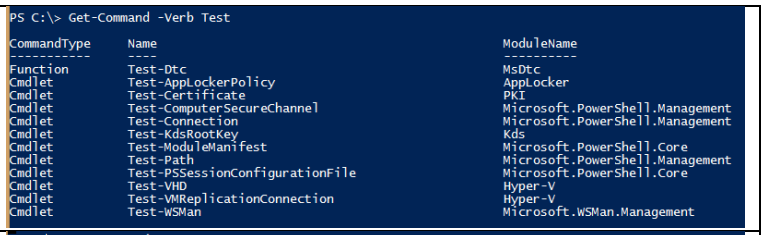
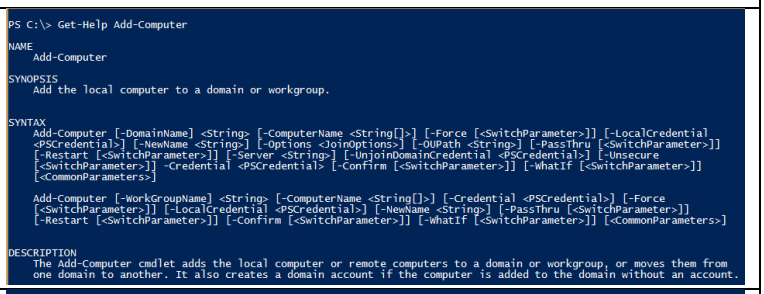
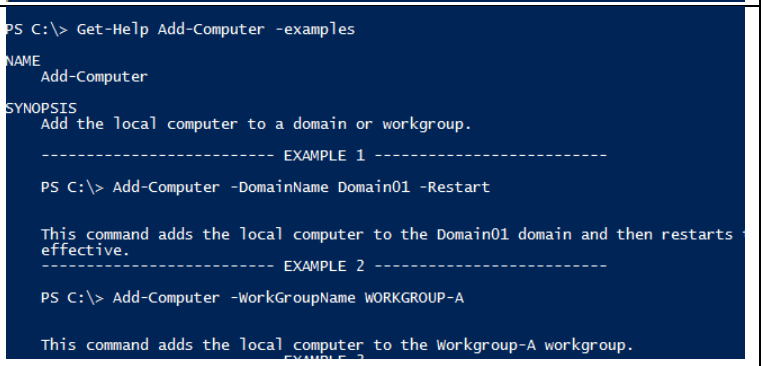
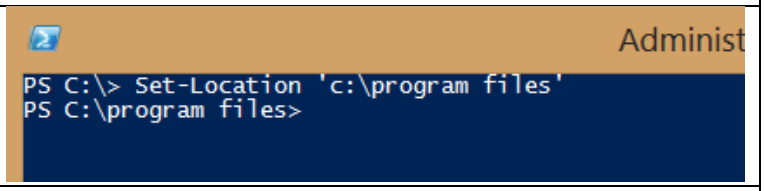
Passing Parameters and values to cmdLets

You can passing parameters and value to cmdLets. Parameters names must be specified with – (Dash or hyphen) and proceeded by value of the parameter. The value of the parameter can be any data type like string, number or Boolean ($true or $false).

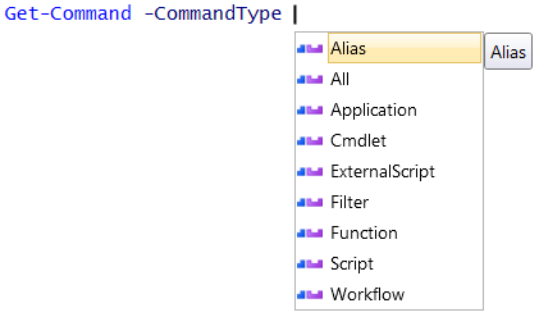
Let's try some commands.

Switch to your PowerShell Window and type cls and press enter.

Get-Command –Name Get-Process	

Get-Command –Verb Test **also try** Get-Command –Verb Add Get-Command –Verb Get Get-Command –Verb Set Get-Command –Verb Stop	
Get-Command –Noun Computer **Also try** Get-Command –Noun Path Get-Command –Noun Certificate Get-Command –Noun Cert*	
Get-Help Add-Computer	
Get-Help Add-Computer –examples Also Try the following command for help Get-Help Add-Computer –Detailed Get-Help Add-Computer –Full Observe the output of the command. To read about Get-Help Command you can say Get-Help Get-Help ☺ Or type help Get-Help	
Set-Location c:\Demo	

Note: PowerShell 3.0 provided lot more new cmdlets that might be difficult to find so new parameters are added to Get-Command to find cmdLets faster.
Get-Command –CommandType Cmdlet
Get-Command –CommandType Functions



Use Help Get-Command to find more information.

Tab Completion

If you have used Visual Studio to write any code in your life you must be aware of this feature that allows you to auto complete a class or object name without typing its full time. The feature is available in PowerShell. Tab completion also completes parameters for you.

All you have to do is to type of the first few characters that match your cmdlet and press tab key. E.g. if you are trying to run Get-Command then you can Get-Comm and press tab. If you type Get-com so first the buffer will show you first command that match this text and on next tab it will show you next one and it will loop back to first one. Tab completion can also show you all the parameters of cmdlet once the cmdlet name is typed completely.

Note: There are some long and matching cmdLets in PowerShell so tab completion might require more characters in typing e.g. Get-SPEnterpriseSearchServiceAppliationProxy
Let's try it in PowerShell.

Switch to Windows PowerShell and Clear your screen using cls (Please don't run any command just use Tab Completion and then press ESC key to clear the line)

1. Now type **Get-Comm** and press **TAB** key
2. Now Type Get-H and press TAB key
3. Now type Get-Co and press TAB Key and keep pressing tab to see all command matching this name

Let's try the parameters

1. Type Get-Comm and press TAB key then after space type –n and press TAB key
2. Type Get-Proc and press TAB then after space type –i and press TAB key.
3. Type Get-Proc and press TAB Key and then after space type –pro and then press TAB key.

Make sure you do some practice of Tab Completion and make it a habit as it is vital for fast PowerShell Script Writing.

Using Variables in PowerShell

Variables are used to save output of PowerShell. You can define variables to store different kind of data, including string, integer, datetime. Variables can be define using New-Variable cmdlet. But using the cmdlet is not required. You can define the variables directly without specifying the type of the variable. PowerShell will automatically set the type based on the assigned value.

Create a new Variable CustomID that will store an Integer	New-Variable -Name CustomerID -Value 10
Get the Variable by Name	Get-Variable -Name CustomerID or Get-Variable CustomerID
Get all variables including system variables	Get-Variable
Set Variable Value to new Value	Set-Variable -Name CustomerID -Value 20
Clear the value of the variable	Clear-Variable –Name CustomerID
Delete the variable	Remove-Variable -Name CustomerID

You can create common variables directly without using New-Variable cmdlet.

[System.Int16]$ID=1	[int]$EmployeeID=10
[System.String]$FirstNam="Tom"	[string]$LastName="Jerry"
[System.DateTime]$Today = Get-Date	[datetime]$Tomorrow = Get-Date

Common Shortcuts

Shortcut	Data Type
[datetime]	Date or time
[string]	String of characters
[char]	Single character
[double]	Double-precision floating number
[single]	Single-precision floating number
[int]	32-bit integer
[wmi]	Windows Management Instrumentation (WMI) instance or collection
[adsi]	Active Directory Services object
[wmiclass]	WMI class
[Boolean]	True or False value

To get the value of the variable you just need to type the name of the variable.
$ID

As every variable is a .NET type you can get the type of the variable by using GetType() method. This will show you the type and details of the object.
$ID.GetType()

```
PS C:\Users\YasirAttiq> $ID.GetType()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     Int16                                    System.ValueType
```

As you can see that type of ID is Int16. You cannot set a string value to integer.

```
PS C:\Users\YasirAttiq> $ID = "Trying to Set String to Integer"
Cannot convert value "Trying to Set String to Integer" to type "System.Int16". Error: "Input string was not in a
correct format."
At line:1 char:1
+ $ID = "Trying to Set String to Integer"
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : MetadataError: (:) [], ArgumentTransformationMetadataException
    + FullyQualifiedErrorId : RuntimeException
```

On the other hand you can set a string value to integer. PowerShell will allow you to change the value will not change the type.

```
PS C:\Users\YasirAttiq> $FirstNam
Tom
PS C:\Users\YasirAttiq> $FirstNam.GetType()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     String                                   System.Object


PS C:\Users\YasirAttiq> $FirstNam = 10
PS C:\Users\YasirAttiq> $FirstNam.GetType()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     String                                   System.Object
```

You can write string variables on PowerShell Window directly or using Write-Host command. To print simple text on the screen you can type the string text in double or single quote.

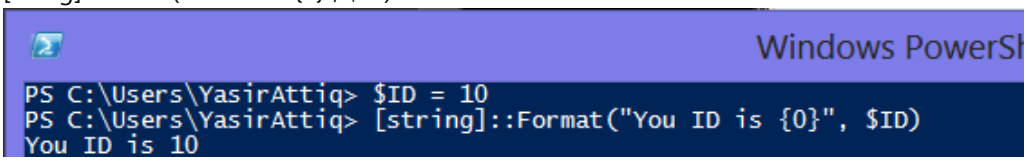
Type the following in PowerShell Window
"Please enter your name"

To type long string with spaces and carriage returns you must put @ sign around the string.

```
$LongString= @'
        This is a string
        This is another line'@
```

You can also using string.format for concatenation of strings.

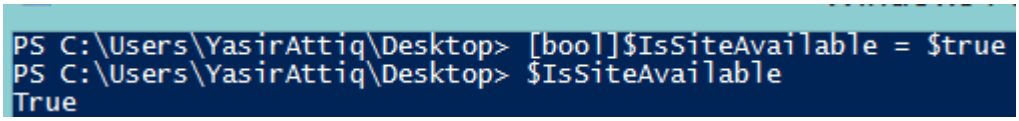
```
$ID = 10
[string]::Format("You ID is {0}", $ID)
```



Boolean Variables

Boolean values can be used in PowerShell but the value of Boolean can be represented by $true and $false

```
[bool]$IsSiteAvailable = $true
```



The output of Boolean will be in form of True and False.

Escape Sequences

If you want to format your string using escape sequences you can use the following symbols in your strings.

Symbol	Output
'b	Backspace
'n	New Line
'r	Carriage Return

""""	The value will have double quote in string
't	Tab

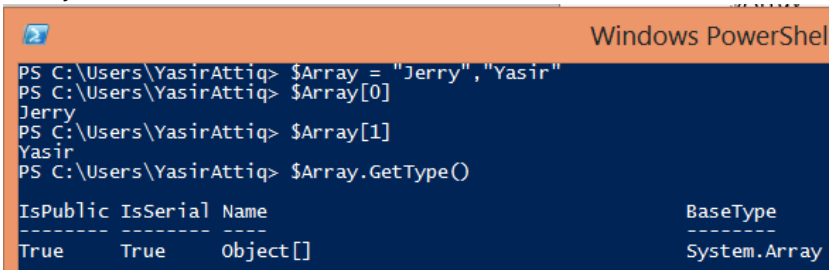
String Comparison

Script	Output
"UnitedStates" -like "Uni*"	True (Include Uni and everything after that)
"UnitedStates" -like "?ni*"	True (Does not know first character then uni and everything

Using Arrays

You can also use Arrays in PowerShell. There are defined in same ways as you define them in C#.
Note: Arrays Start with 0 (Zero)

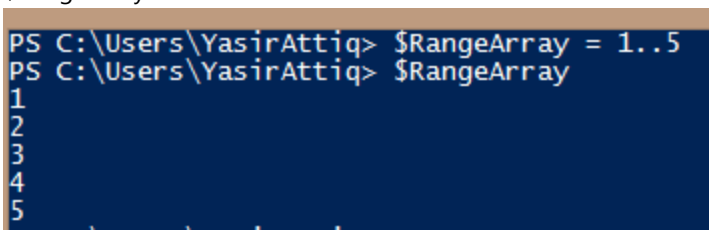
```
$Array = "Jerry", "Yasir"
$Array[0]
$Array[1]
```



You can also Set Array values using Index e.g. $Array[0] = "Jersey"

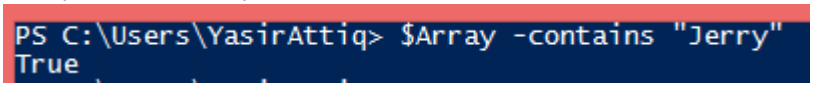
You can also create Arrays with numeric ranges (using double dots).

```
$RangeArray = 1..5
```



You can also search from Arrays which will return true and false if the item exist in array.

```
$Array -contains "Jerry"
```



Note: PowerShell is support Hashtables which are Unordered by default. In PowerShell 3.0 you can also define a Hashtables as Ordered.

```
#Un-ordered Hashtable
cls
$hashtable = @{x=1; y=2;z=3}
$hashtable
```

```
Name                           Value
----                           -----
y                              2
z                              3
x                              1
```

```
#Ordered Hashtable
cls
$hashtable = [ordered]@{x=1;
y=2;z=3}
$hashtable
```

```
Name                           Value
----                           -----
x                              1
y                              2
z                              3
```

Comparing Variables

PowerShell allow you to use comparison operators to compare two variables. Most programming langues using =, <>, > or < symbols but PowerShell uses different format. You can use the following symbols.

-eq	-ne	-gt
-ge	-lt	-le
-Like	-NotLike	-Match
-NotMatch	-Contains	-NotContains
-In	-NotIn	-Replace

The comparison return true or false or even objects.
$A = 20
$B = 40

$A -gt $B	False $A is not greater than $B
$A -eq $B	False Both are not equal
$A -lt $B	True A is less then B

String Comparisons

$Name = "Tom Franklin"

$Name -like "Tom*"	True Tom is part of string

You can try some other operators by yourself. Practice is key.

Using Objects as Variable

As we have spent good amount of time with basic variables lets learn about Variables that will hold objects instead of values.
Run the following cmdlet.

Clear-Host
Get-Process

Get-Process will give you list of all the processes running on the system.

```
PS C:\Users\YasirAttiq> Get-Process

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
     75       7     1152       4056    45               2532 armsvc
    382      13     9892      13288    75    14.38      7404 audiodg
   2323     155   296096      74080   719               2588 avp
   1194      67    40956       5740   215     2.92      6676 avp
     54      10     1172       4076    34               3564 BTHSAmpPalService
    221      25     2340       7092    48               1764 BTHSSecurityMgr
    184      16    11628      20056   145     0.34      6300 Color Calibrator Tray
     48       7     1668       5792    59     2.01      7480 conhost
     33       6      852       3368    51     0.02      7740 conhost
     69       6     1036       3724    45               2652 CrmSqlStartupSvc
    506      16     2036       4380    54                844 csrss
    515      32     2760      90996   280                964 csrss
    492      20     6380      14816    84               4076 dasHost
    108       8     1800       6668    54               4432 dllhost
    297      33    27288      37812   348               1300 dwm
```

Let's use a variable to hold the output

$Processes = Get-Process

```
PS C:\Users\YasirAttiq> $Processes = Get-Process
```

Now type the variable to see the output.

```
PS C:\Users\YasirAttiq> $Processes

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
     75       7     1152       4056    45               2532 armsvc
    382      13     9892      13288    75    14.38      7404 audiodg
   2294     151   296588      76120   720               2588 avp
   1192      67    40956       6192   215     2.92      6676 avp
     54      10     1172       4076    34               3564 BTHSAmpPalService
    221      25     2340       7092    48               1764 BTHSSecurityMgr
    184      16    11628      20056   145     0.34      6300 Color Calibrator Tray
     48       7     1648       5808    59     2.11      7480 conhost
```

Let's get a single process from these processes.

Open Notepad from your computer.

$NotePad = Get-Process notepad

```
PS C:\Users\YasirAttiq> $NotePad = Get-Process notepad
PS C:\Users\YasirAttiq> $NotePad

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
     77       9     1660       6264    93     0.08   9788 notepad
```

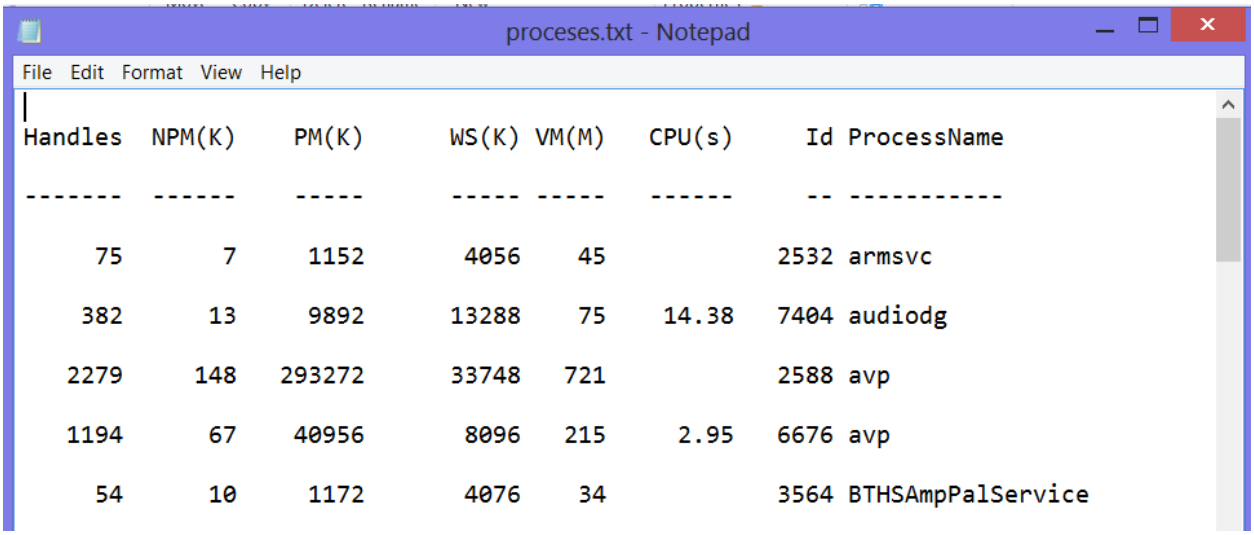
Combining Multiple cmdlets using Pipe | Sign

Pipe sign is used to pass output of first command as an input to next command. Here are some examples

Get-Process | Out-File d:\proceses.txt #this command will write the information of processes in text file

```
PS C:\Users\YasirAttiq> Get-Process | Out-File d:\proceses.txt
PS C:\Users\YasirAttiq>
```

Because the output is written to a text file nothing will show up on powershell.



Now you can see that Get-Process command returns all the properties of the object. If you want to get only limited columns you have to use **Select**.

Passing objects in Pipe

You can pass the object to next cmdlet using Pipe sign. E.g. in the example below I will exit or kill the notepad process.

#Start Notepad from your computer.

```
$NotePad = Get-Process notepad
$NotePad | Stop-Process
```

```
PS C:\Users\YasirAttiq> $NotePad | Stop-Process
```

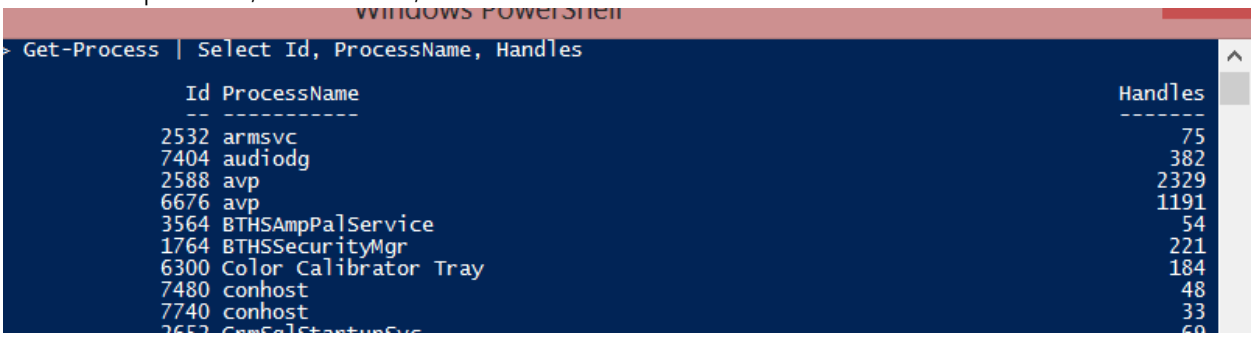
PowerShell will not show an output.

Using Select

To use select cmdlet you must your first cmdlet or variable on the left and after pipe sign you will add Select and the columns you need separated by comma.

#Type the following cmdlet on PowerShell Window

```
cls
Get-Process | Select Id, ProcessName, Handles
```



Using Sort

To Sort the output you have to use Sort cmdlet with another pipe sign. You can any properties of the object.

#Type the following cmdlet on PowerShell Window

cls

Get-Process | Select Id, ProcessName, Handles | sort ProcessName

```
ss | Select Id, ProcessName, Handles | sort ProcessName

    Id ProcessName                                                      Handles
    -- -----------                                                      -------
  2532 armsvc                                                             75
  7404 audiodg                                                           382
  6676 avp                                                              1191
  2588 avp                                                              2359
  3564 BTHSAmpPalService                                                  54
  1764 BTHSSecurityMgr                                                   221
  6300 Color Calibrator Tray                                             184
  7740 conhost                                                            33
  7480 conhost                                                            48
  2652 CrmSqlStartupSvc                                                   69
   964 csrss                                                             526
   844 csrss                                                             510
  4076 dasHost                                                           492
  4432 dllhost                                                           108
```

You can also use Sort-Object property

Formatting the output

You can format the output of the command using the following way.

Format-List

cls

Get-Process | Select Id, ProcessName, Handles | sort ProcessName | Format-List

```
PS C:\Users\YasirAttiq> Get-Process | Select Id, ProcessName, Handles | sort ProcessName | Format-List


Id          : 2532
ProcessName : armsvc
Handles     : 75

Id          : 7404
ProcessName : audiodg
Handles     : 382

Id          : 6676
ProcessName : avp
Handles     : 1194
```

Format-Table (Default)

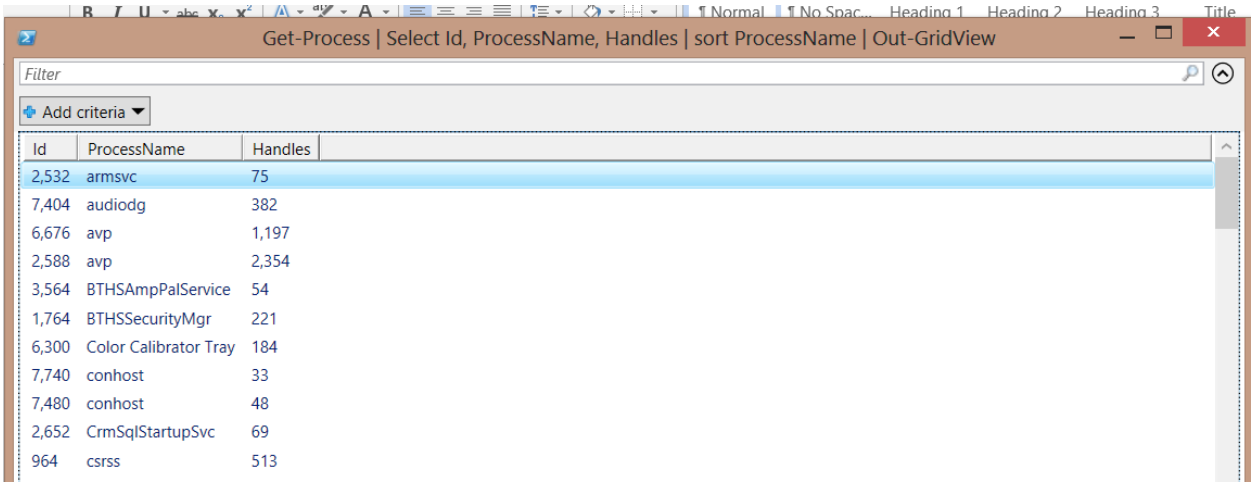
cls

Get-Process | Select Id, ProcessName, Handles | sort ProcessName | Format-Table

```
iq> Get-Process | Select Id, ProcessName, Handles | sort ProcessName | Format-Table

                Id ProcessName                                              Handles
                -- -----------                                              -------
              2532 armsvc                                                     75
              7404 audiodg                                                   382
              6676 avp                                                      1194
              2588 avp                                                      2337
              3564 BTHSAmpPalService                                          54
```

Out-GridView (Provides you a Windows Dialog to sort and filter cmdlets)

Get-Process | Select Id, ProcessName, Handles | sort ProcessName | Out-GridView



Filtering Objects

You can also Filter objects in PowerShell using multiple ways. Here is a simple example to start with

1,2,3,4 | Where-Object {$_ -lt 3} #This will return you 1 and 2

```
PS C:\Users\YasirAttiq> 1,2,3,4 | Where-Object {$_ -lt 3}
1
2
```

Get All the process where name has svc in start

Get-Process | Where-Object {$_.ProcessName -like "svc*"}

```
PS C:\Users\YasirAttiq> Get-Process | Where-Object {$_.ProcessName -like "svc*"}

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    451      17     5236      11312    47             1044 svchost
    603      17     6420      10476    51             1184 svchost
    874      34    25052      34340   136             1244 svchost
   2012      91    33896      47892   373             1284 svchost
    810      44    13264      26688   111             1348 svchost
   1015      46    14784      31172   121             1436 svchost
    913      51    17616      21240  1402             1956 svchost
    524      31     6032      13644    62             2308 svchost
    598      42    25208      31180   146             2416 svchost
     95       8     1676       4440    22             3028 svchost
    143      11     2340       7492    47             3128 svchost
    111       8    19972       9496    39             3148 svchost
    398      30     8880      13296   831             4408 svchost
```

You can also use Where instead of Where-object.

Get-Process | Where {$_.ProcessName -like "svc*"}

```
2
PS C:\Users\YasirAttiq> Get-Process | Where {$_.ProcessName -like "svc*"}

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    446      16     5124      11256    46              1044 svchost
    607      17     6420      10476    51              1184 svchost
    867      32    24948      34304   135              1244 svchost
   2007      91    33844      47860   373              1284 svchost
    810      44    13264      26672   111              1348 svchost
   1015      46    14784      31172   121              1436 svchost
    910      51    17620      21416  1401              1956 svchost
    526      32     6188      13692    64              2308 svchost
    599      42    25208      31196   146              2416 svchost
     95       8     1676       4440    22              3028 svchost
    143      11     2340       7492    47              3128 svchost
    111       8    19972       9496    39              3148 svchost
    393      30     8816      13260   831              4408 svchost
```

Get-ChildItem is same as dir in DOS.

Get-ChildItem | Where-Object {$_.length -gt 10000}

```
PS C:\Users\YasirAttiq\Desktop> Get-ChildItem | Where-Object {$_.length -gt 10000}


    Directory: C:\Users\YasirAttiq\Desktop


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         3/27/2013  10:00 PM     772672 adfs2-how-to-setup-lab-environment-for-federated-collabora
-a---         4/18/2013   8:23 PM      88073 AutoSPInstaller.zip
-a---         3/28/2013  10:28 PM    5482524 Drive License Manual.pdf
-a---          5/6/2013   8:51 PM      43929 I'mSpeakingDC2013.png
-a---         4/12/2013   3:23 PM      59492 microsoft_mvp_logo2-191x300.png
-a---          4/2/2013   9:05 PM    1487871 NetExtender6.zip
```

You can also use –and –or –not inside the FilterExpression

Get-ChildItem | Where-Object {$_.length -gt 10000 -and $_.name -like "*SharePoint*"}

```
Windows PowerShell
PS C:\Users\YasirAttiq\Desktop> Get-ChildItem | Where-Object {$_.length -gt 10000 -and $_.name -like "*SharePoint*"}


    Directory: C:\Users\YasirAttiq\Desktop


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         4/12/2013   5:40 PM     207076 Why Upgrading to SharePoint 2013.docx
-a---          5/8/2013  11:59 AM     101217 Yasir_Attiq - SharePoint MVP_US Tech Solutions.docx
```

You can pass KB, or MB or GB in length.

Note: Not all of the properties will be exposed when you print the object. To get information about the object's properties and method you can use Get-Member. Some properties might not be simple to present as they represent a collection.

Get-Member

object | Get-Member

object | Get-Member –MemberType Methods

object | Get-Member –MemberType Properties

#Example (Please start Notepad)

$Notepad = Get-Process notepad

$NotePad | Get-Member -MemberType Properties

```
PS C:\Users\YasirAttiq\Desktop> $NotePad | Get-Member -MemberType Properties


   TypeName: System.Diagnostics.Process

Name                       MemberType     Definition
----                       ----------     ----------
Handles                    AliasProperty  Handles = Handlecount
Name                       AliasProperty  Name = ProcessName
NPM                        AliasProperty  NPM = NonpagedSystemMemorySize
PM                         AliasProperty  PM = PagedMemorySize
VM                         AliasProperty  VM = VirtualMemorySize
WS                         AliasProperty  WS = WorkingSet
__NounName                 NoteProperty   System.String __NounName=Process
BasePriority               Property       int BasePriority {get;}
Container                  Property       System.ComponentModel.IContainer Container {get;}
EnableRaisingEvents        Property       bool EnableRaisingEvents {get;set;}
ExitCode                   Property       int ExitCode {get;}
ExitTime                   Property       datetime ExitTime {get;}
Handle                     Property       System.IntPtr Handle {get;}
HandleCount                Property       int HandleCount {get;}
```

$NotePad | Get-Member –MemberType Method

```
PS C:\Users\YasirAttiq\Desktop> $NotePad | Get-Member -MemberType Method


   TypeName: System.Diagnostics.Process

Name                      MemberType Definition
----                      ---------- ----------
BeginErrorReadLine        Method     void BeginErrorReadLine()
BeginOutputReadLine       Method     void BeginOutputReadLine()
CancelErrorRead           Method     void CancelErrorRead()
CancelOutputRead          Method     void CancelOutputRead()
Close                     Method     void Close()
CloseMainWindow           Method     bool CloseMainWindow()
CreateObjRef              Method     System.Runtime.Remoting.ObjRef CreateObjRef(type requestedType)
Dispose                   Method     void Dispose(), void IDisposable.Dispose()
Equals                    Method     bool Equals(System.Object obj)
GetHashCode               Method     int GetHashCode()
GetLifetimeService        Method     System.Object GetLifetimeService()
GetType                   Method     type GetType()
InitializeLifetimeService Method     System.Object InitializeLifetimeService()
Kill                      Method     void Kill()
Refresh                   Method     void Refresh()
Start                     Method     bool Start()
ToString                  Method     string ToString()
WaitForExit               Method     bool WaitForExit(int milliseconds), void WaitForExit()
WaitForInputIdle          Method     bool WaitForInputIdle(int milliseconds), bool WaitForInputIdle()
```

Interacting with PowerShell

To Read and write on PowerShell we have the following important cmdlets.

Write-Host is used to write output to the powershell window. You can also show variables.

#Example

$Name = "I am PowerShell"

Write-Host "This is some Text " $Name

```
PS C:\Users\YasirAttiq\Desktop> $Name = "I am PowerShell"
PS C:\Users\YasirAttiq\Desktop> Write-Host "This is some Text " $Name
This is some Text  I am PowerShell
```

Read-Host is used to get some value from the window.

$Name = Read-Host "Please Enter Your Name"

$Name

```
PS C:\Users\YasirAttiq\Desktop> $Name = Read-Host "Please Enter Your Name"
Please Enter Your Name: Jerry Yasir
PS C:\Users\YasirAttiq\Desktop> $Name
Jerry Yasir
```

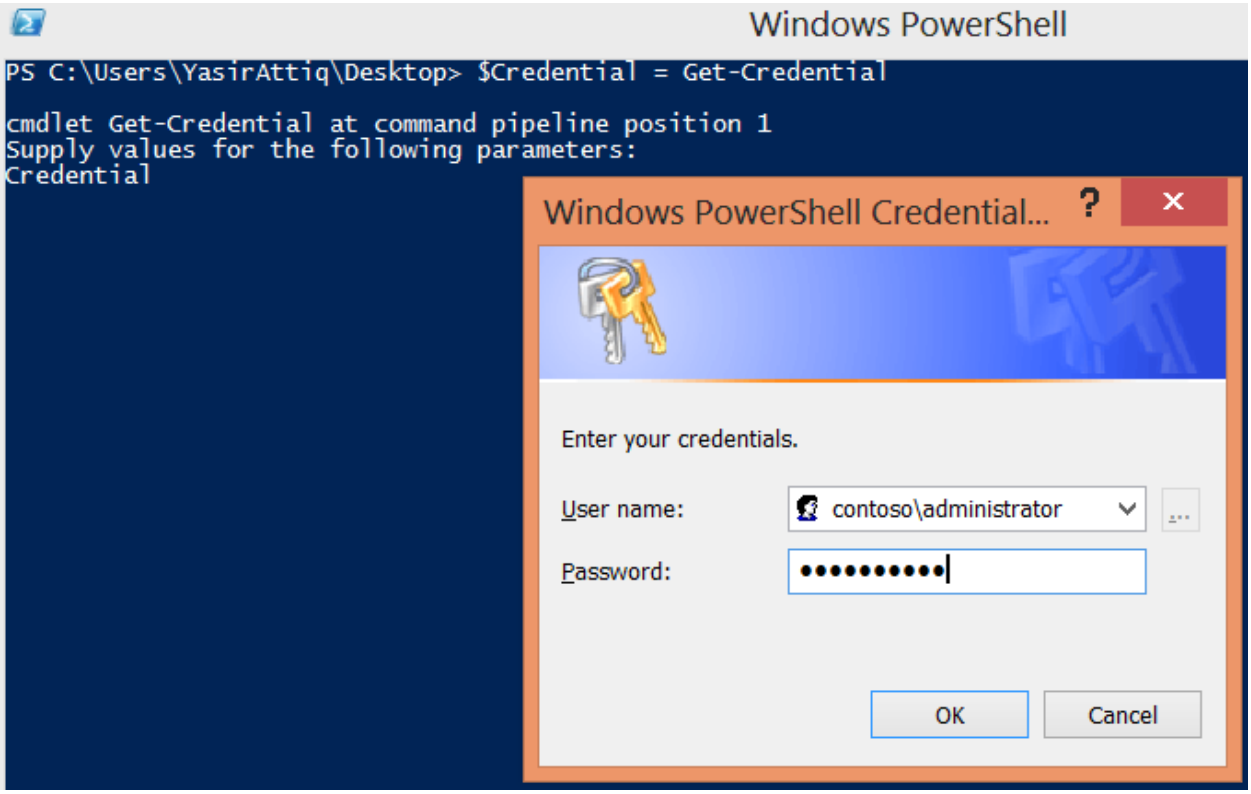
Getting Secure Information like Passwords AsSecureString (You cannot print these values). The value you will type will be shown as *****.

$Password = Read-Host "Please Enter your password." –AsSecureString

```
PS C:\Users\YasirAttiq\Desktop> $Password = Read-Host "Please Enter your password. " -AsSecureString
Please Enter your password. : **********
PS C:\Users\YasirAttiq\Desktop> $Password
System.Security.SecureString
```

Get-Credential is used to get credentials using Windows User Name and password Dialog.

$Credential = Get-Credential



$Credential

```
PS C:\Users\YasirAttiq\Desktop> $Credential = Get-Credential

cmdlet Get-Credential at command pipeline position 1
Supply values for the following parameters:
Credential
PS C:\Users\YasirAttiq\Desktop> $Credential

UserName                                                      Password
--------                                                      --------
contoso\administrator                                         System.Security.SecureString
```

Loops

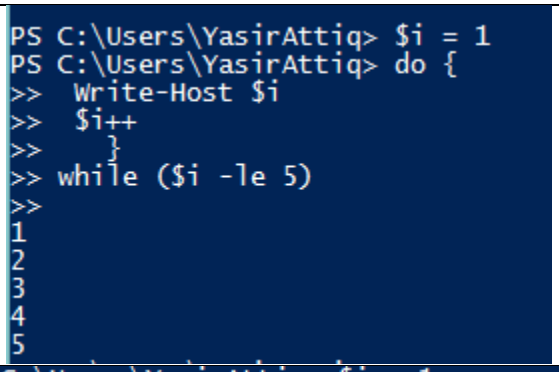
PowerShell support 5 type of Loops.

- **do while** - Script block executes as long as condition value = True.
- **while** - Same as "do while"
- **do until** – Script block executes until the condition value = True.
- **for** – Script block executes a specified number of times.
- **foreach** - Executes script block for each item in a collection or array.

Note: The code between { (Curly Brackets) is called a Script Block.

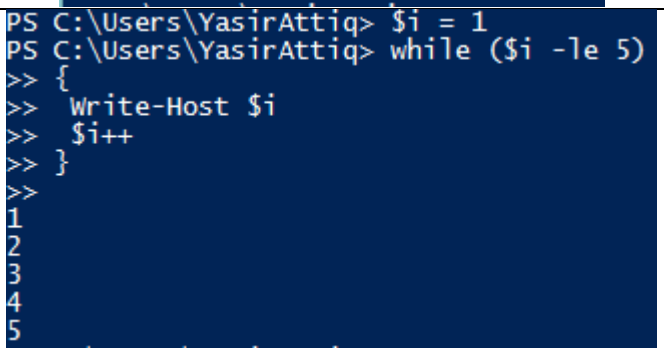
#Examples - **Do While**

```
$i = 1
do {
        Write-Host $i
        $i++
  }
while ($i -le 5)
```



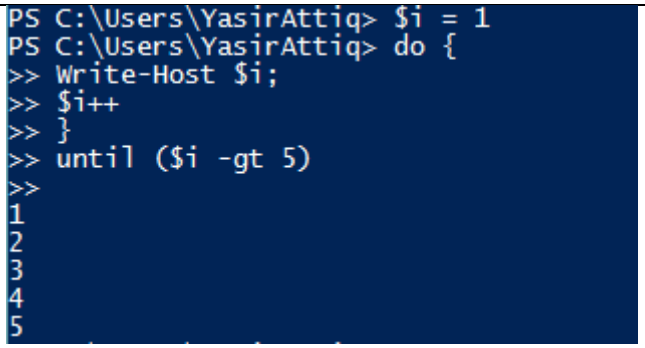
#Examples – **While**

```
$i = 1
while ($i -le 5)
{
 Write-Host $i
 $i++
}
```



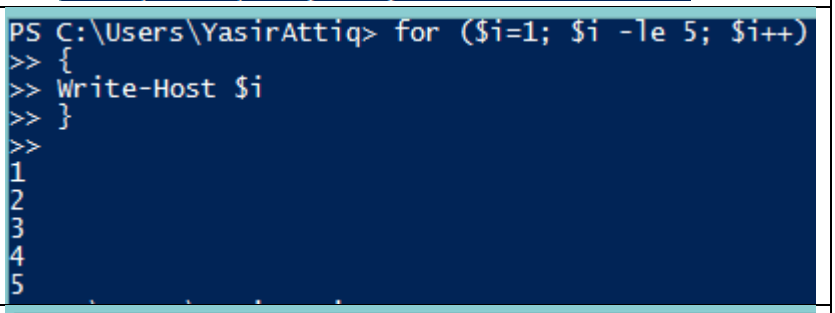
#Examples – **Do Until**

```
$i = 1
do {
        Write-Host $i;
        $i++
}
until ($i -gt 5)
```



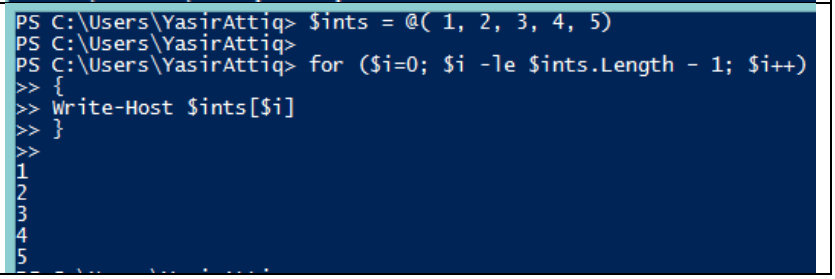
#Example **Loop**

```
for ($i=1; $i -le 5; $i++)
{
Write-Host $i
}
```



#Example **Loop with Array**

```
$ints = @( 1, 2, 3, 4, 5)

for ($i=0; $i -le $ints.Length
– 1; $i++)
{
        Write-Host $ints[$i]
}
```



#Example **For Loop**

```
$ints = @(1, 2, 3, 4, 5)

foreach ($i in $ints)
{
        Write-Host $i
}
```

```
PS C:\Users\YasirAttiq> $ints = @(1, 2, 3, 4, 5)
PS C:\Users\YasirAttiq>
PS C:\Users\YasirAttiq> foreach ($i in $ints)
>> {
>> Write-Host $i
>> }
>>
1
2
3
4
5
```

>> will show you line break. You need to press one more enter to execute after last line.

Variable Scope

You can define variables inside and outside the script block just like any other programming language.

#Example

```
$Var =10
& {$Var =15; Write-Host "Local Variable $Var"}
Write-Host "Global Var $Var"
```

```
PS C:\Users\YasirAttiq> $Var = 10
PS C:\Users\YasirAttiq> & { $Var =15; Write-Host "Local Variable $Var"} Write-Host "Global Var $Var"
Local Variable 15
PS C:\Users\YasirAttiq> Write-Host "Global Var $Var"
Global Var 10
```

Writing Functions

You can also write functions in PowerShell if you have some code to execute multiple time. You can take parameters.

```
#Example Function
function HelloWorld()
{
    clear-host
    Write-Host "Hello World is being Called"
}

#Calling the Function
HelloWorld
```

```
Hello World is being Called
PS C:\Users\YasirAttiq>
```

```
#Example Function with Parameter
function SayYourName($YouName)
{
    clear-host
    Write-Host "Your Name is " $YouName
}

SayYourName "Tommy Jonaa"
```

```
Your Name is Tommy Jonaa
PS C:\Users\YasirAttiq>
```

Handling Errors in PowerShell

PowerShell provides powerful way for handling errors in scripts. If you have used .NET to catch errors in try catch block. PowerShell has "Trap" which allows you to catch errors. You can trap errors just like you can trap exceptions. You can use .NET Exception inside Trap to catch a specific error like System.DivideByZeroException etc.

```powershell
#Example of Basic Method with Paramter
function Divide($FirstValue, $SecondValue)
{
    clear-host
    $ThirdValue = $FirstValue / $SecondValue
    Write-Host "Result is" $ThirdValue
}

# Calling the Method
Divide 20 10
```

```
Result is 2
PS C:\Users\YasirAttiq>
```

```powershell
#Example with Continue on error of Script Result
function Divide($FirstValue, $SecondValue)
{
    clear-host
    $ThirdValue = $FirstValue / $SecondValue
    Write-Host "Result is" $ThirdValue
    trap
    {
        Write-Host "Can not divide by Zero"
        Write-Host $_.ErrorID
        Write-Host $_.Exception.Message
        continue;
    }
}

# Calling the Method
Divide 20 0
```

```
Can not divide by Zero

Attempted to divide by zero.
Result is
PS C:\Users\YasirAttiq>
```

```powershell
#Example with Breaking of Script Result
function Divide($FirstValue, $SecondValue)
{
    clear-host
    $ThirdValue = $FirstValue / $SecondValue
    Write-Host "Result is" $ThirdValue
    trap
    {
        Write-Host "Can not divide by Zero"
        Write-Host $_.ErrorID
        Write-Host $_.Exception.Message
        break
    }
}
# Calling the Method
Divide 20 0
```

```
Can not divide by Zero

Attempted to divide by zero.
Attempted to divide by zero.
At line:11 char:5
+     $ThirdValue = $FirstValue / $SecondValue
+     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [], Par
    + FullyQualifiedErrorId : RuntimeException
```

PowerShell 3.0 Integrated Scripting Environment (PowerShell ISE)

Before we jump into some advanced topic like writing loops and script blocks lets learn about the script writing environment of PowerShell which is called PowerShell Integrated Scripting Environment.

PowerShell ISE is the script editing environment for PowerShell. According to TechNet article "The Windows PowerShell Integrated Scripting Environment (ISE) is a host application for Windows PowerShell. In Windows PowerShell ISE, you can run commands and write, test, and debug scripts in a single Windows-based graphic user interface with multiline editing, tab completion, syntax coloring, selective execution, context-sensitive help, and support for right-to-left languages. You can use menu items and keyboard shortcuts to perform many of the same tasks that you would perform in the Windows PowerShell console."

Try these features in Windows PowerShell ISE.

- Multiline editing: To insert a blank line under the current line in the Command pane, press SHIFT+ENTER.

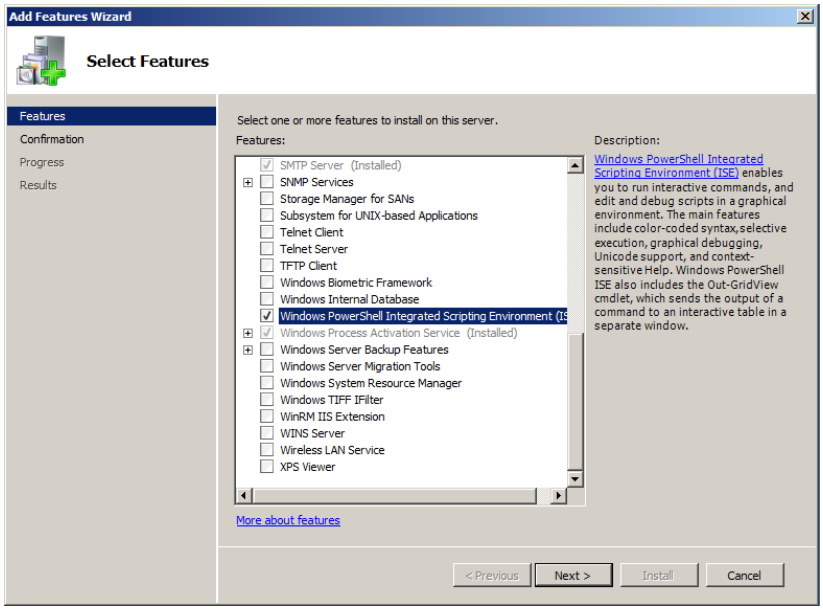
- Selective execution: To run part of a script, select the text you want to run, and then click the Run Script button. Or, press F5.

- Context-sensitive help: Type Invoke-Item, and then press F1. The Help file opens to the Help topic for the Invoke-Item cmdlet.

Source: http://technet.microsoft.com/en-us/library/dd315244.aspx

PowerShell ISE is not enabled by default on Window Server 2008/2008 R2. So you have to enable it. It is part of a feature. So you must enable that feature.

Go to Server Manager → Feature → Add Feature and Check the box next to PowerShell Integrated Scripting Environment.

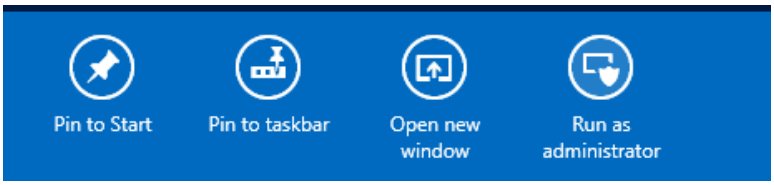


Click Next and Finish when done.

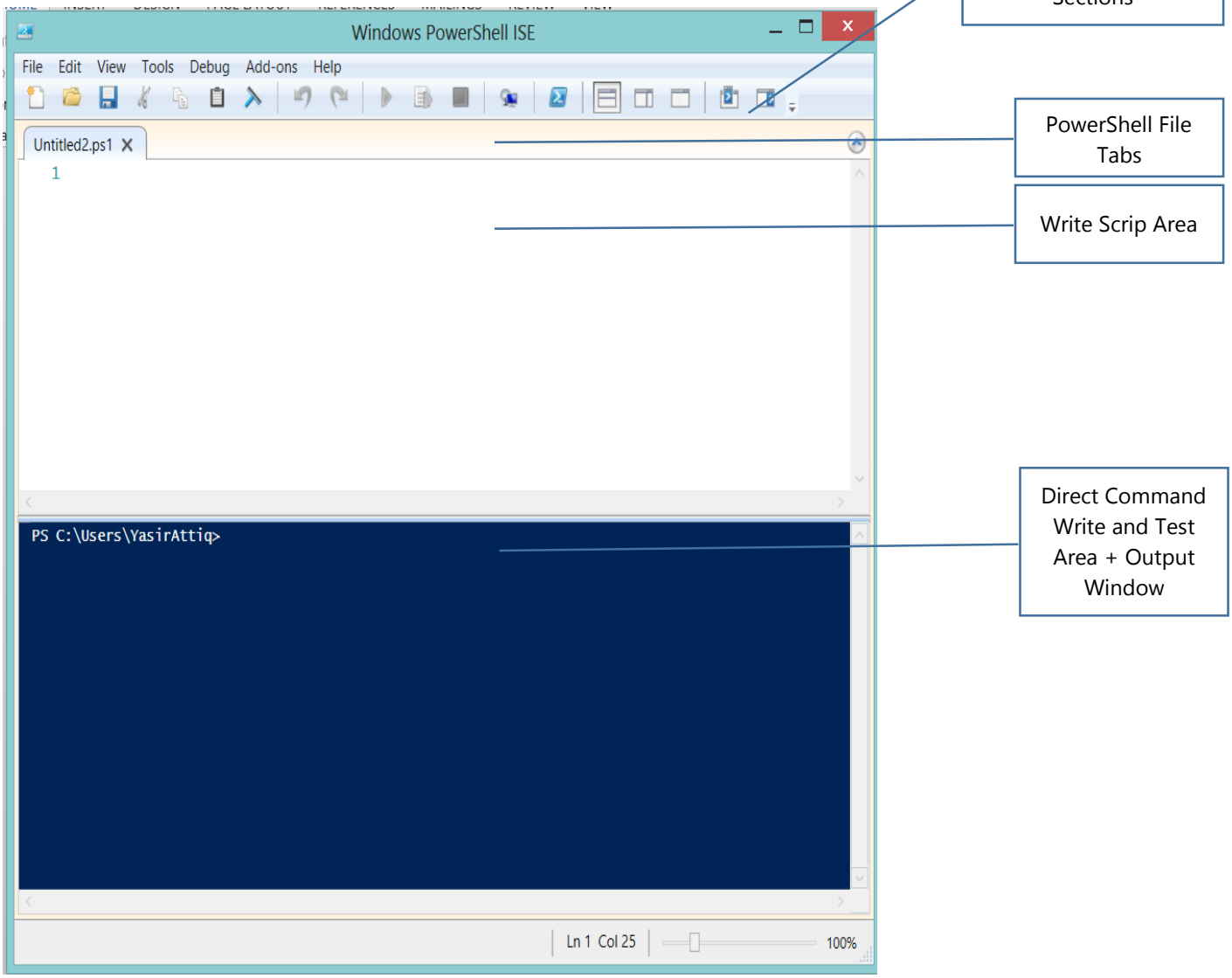
Windows Server 2012 and Windows 8

PowerShell 3.0 ISE is enabled on Windows Server 2012 and Windows 8 by default.

Make sure you right click on it and choose Run as Administrator



PowerShell ISE 3.0 User Interface



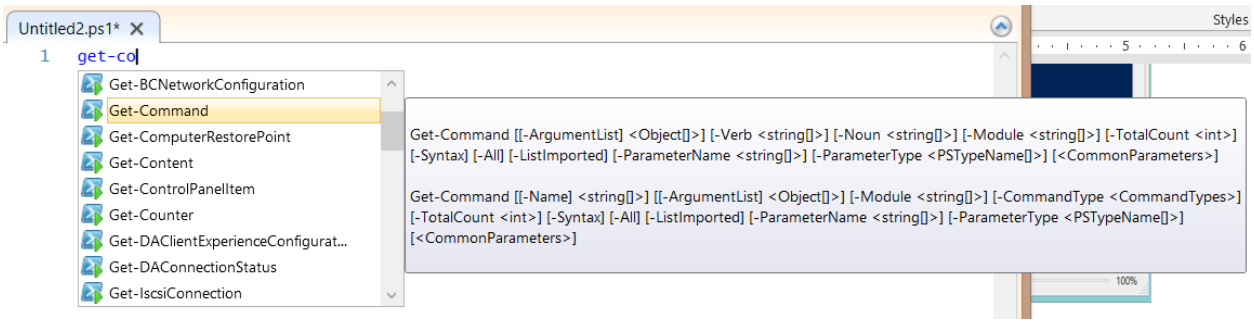
Note: PowerShell ISE 2.0 has 3 windows on home page the middle windows that show the results of script is not gone. You can use the output window to type and test script and same windows will show you output**.**

Some Feature of PowerShell 3.0 Integrated Scripting Environment

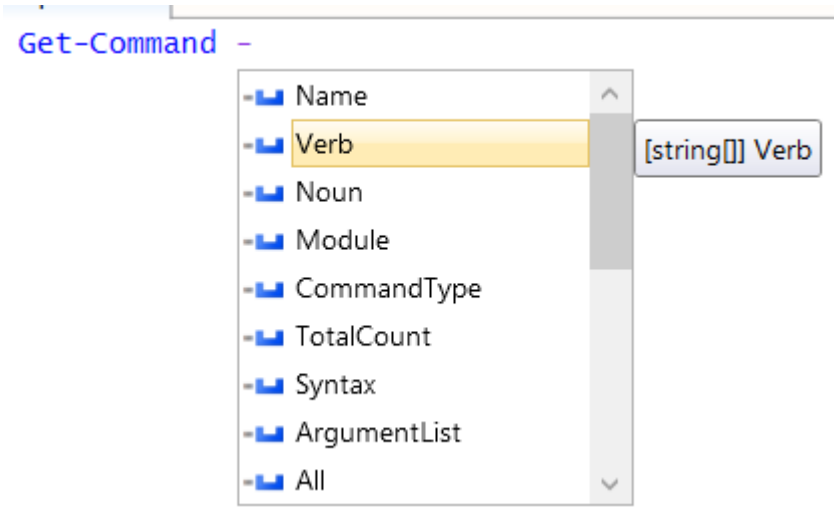
IntelliSense Support

PowerShell 3.0 Introduce the concepts of IntelliSense which an essential part of programming using Visual Studio. With IntelliSense you can find the commands faster and as you type the help will be provided for commands as well as parameter. The IntelliSense is available for both Script window and command text window.

IntelliSense about Get-Command

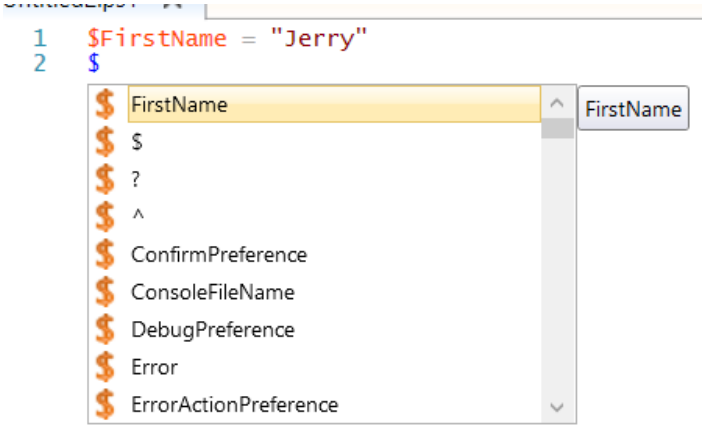


IntelliSense with Parameters

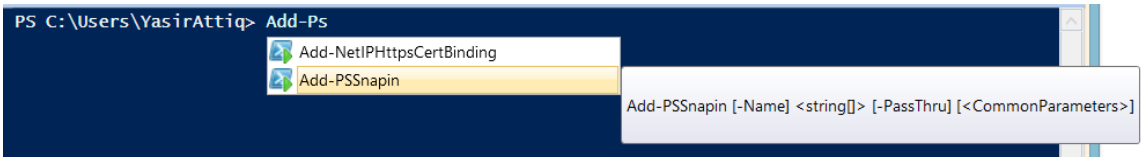


IntelliSense with Variables

As you might have many in your scripts you also have the ability to find them using the same feature. All the variables defined by you will show up on top automatically.



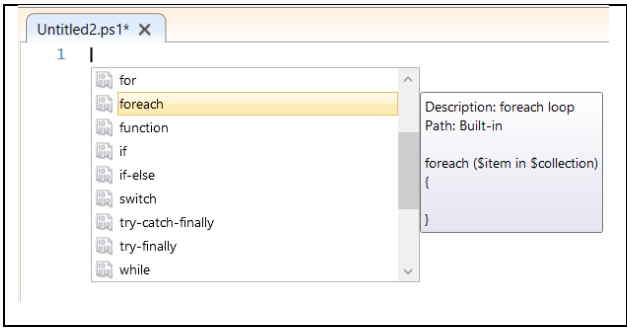
Output Window IntelliSense

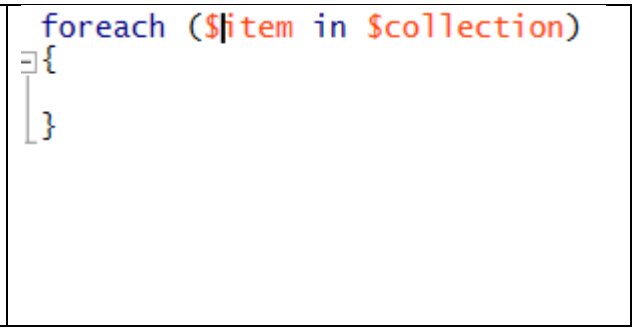


Using Script Snippets

If you have developed applications using Visual Studio, you may have used Code Snippets. Code snippets allows you to reuse scripts. PowerShell 3.0 provides snippets.

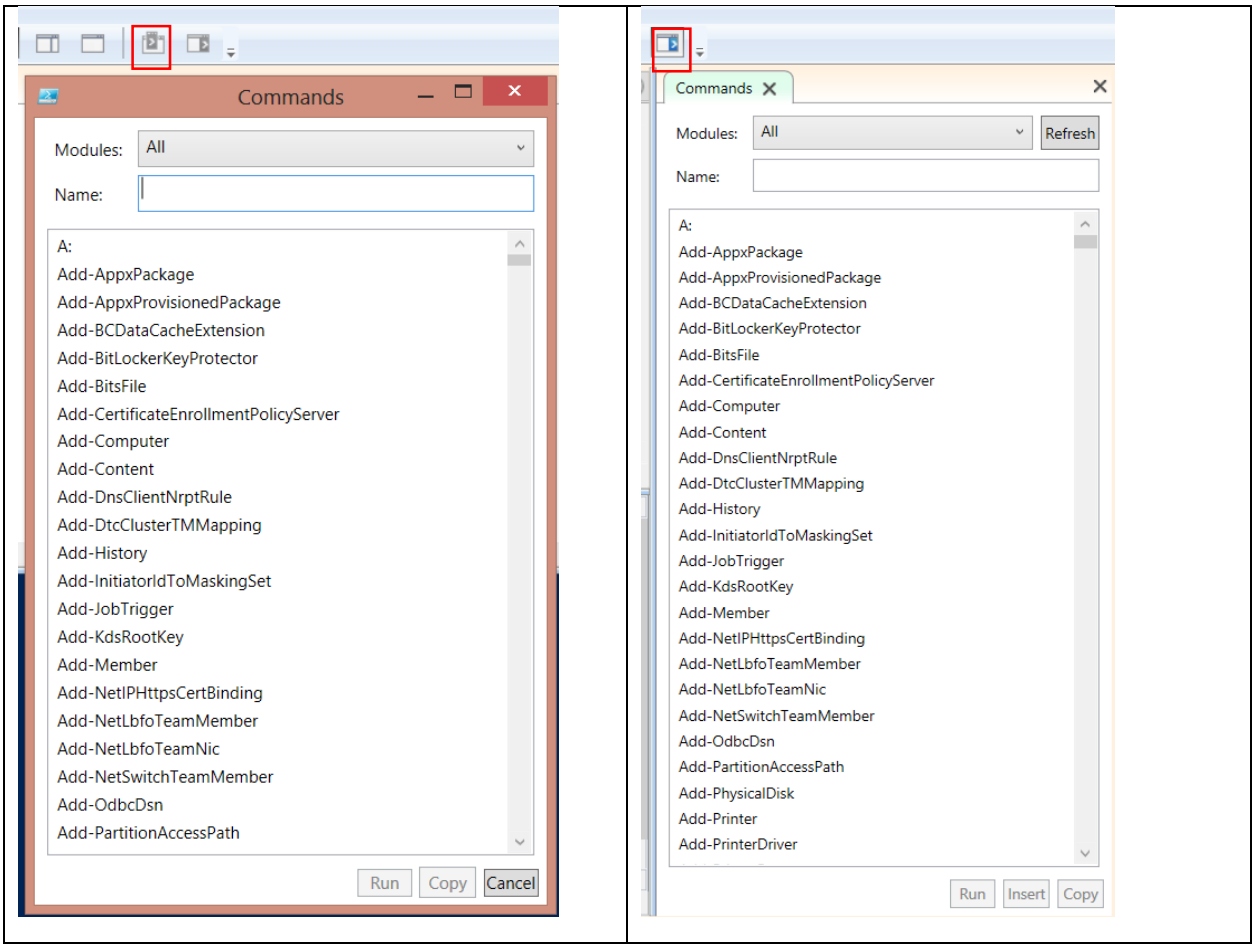
Simply right click on Script Window and Select **Start Snippets**.





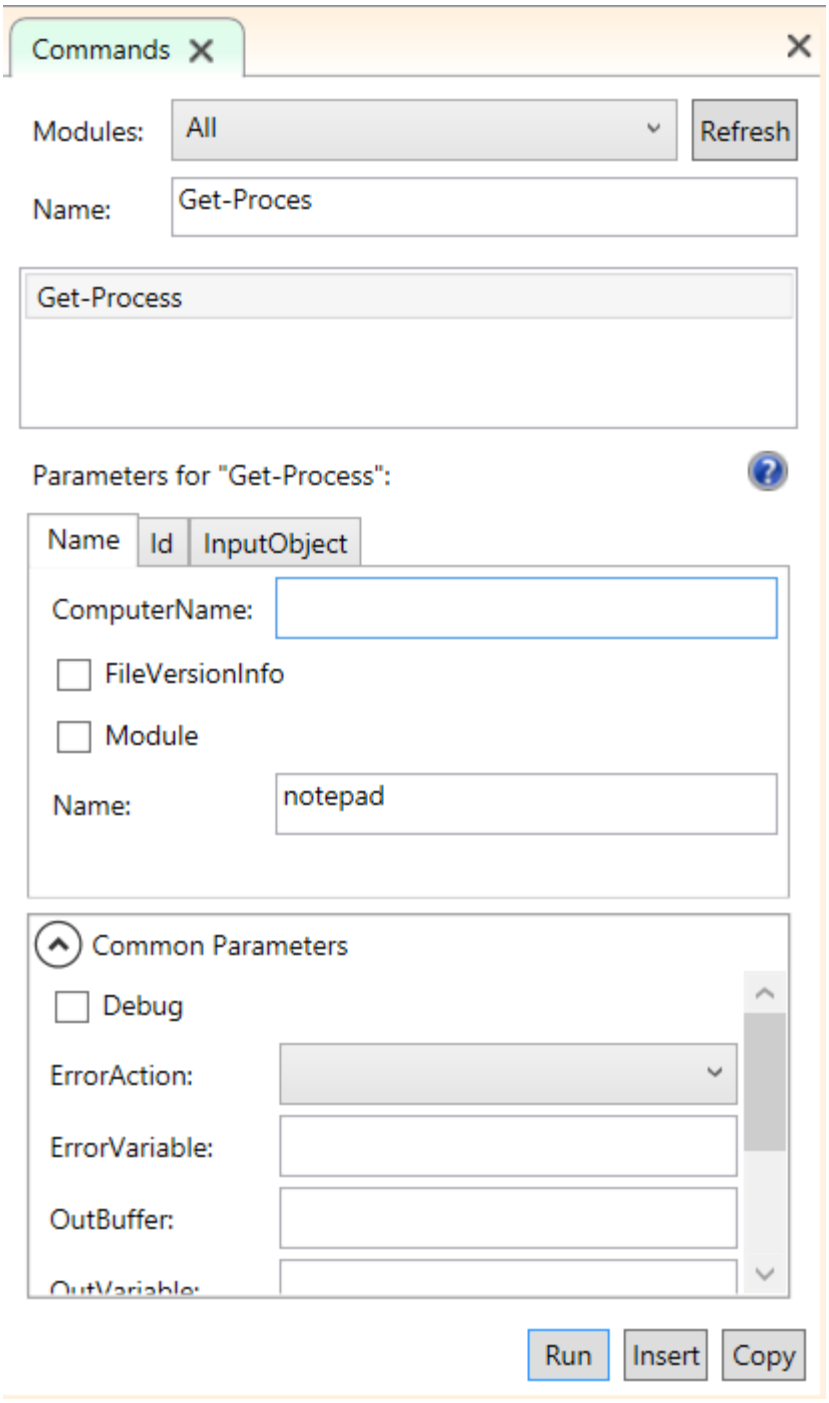
Command and Command Float Window

Clicking on this ICON will open a floatable Command window which can be useful if you have multiple monitors. You can filter and select commands for help based on loaded modules and snap ins.	Click on this ICON will open a Docked command window where users can find and select the command they type. Once a command is selected the command parameters will be shown.



Using the Command Window

In the command "Name" Text box type Get-Process and click on the command from the List.



You can now have the option to provide possible values to the cmdlet and either Run it inside the Output window. Insert it to Output window or simply copy to clipboard. Common Parameters will be collapsed while optional parameters will be in separate tab for easy access.

You can use **Alt + Shift** Keys to select and rename common Variables directly.

```
$FirstName = "Jerry"
$FirstName
$FirstName = "Tom"
$FirstName
```

You can also use Regions to hide long section of scripts or may be functions.

```
#region "Region 1"
    $VariableInRegsion = "Something"
#endregion
```

```
#region Region1
    $VariableInRegsion = "Something"
#endregion Region1
```

In and Not In

Is and Not In Today will allow you to find whether an object is part of a collection or not.

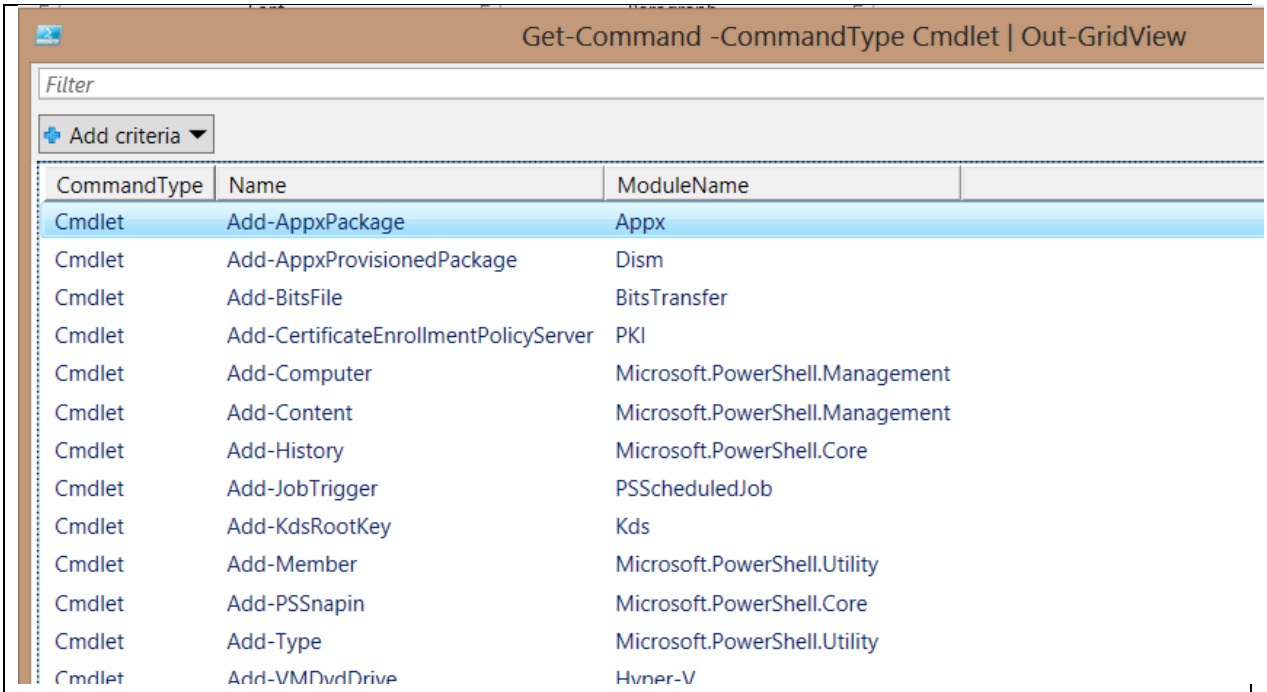
```
$array = (1,2,3,4)
if(3 -in $array)
{
    "3 is Part of Array"
}
if(5 -notin $array)
{
    "5 is not in Array"
}
```

```
PS C:\Users\YasirAttiq> $array = (1,2,3,4)
if(3 -in $array)
{
    "3 is Part of Array"
}
if(5 -notin $array)
{
    "5 is not in Array"
}
3 is Part of Array
5 is not in Array
```

Out-GridView

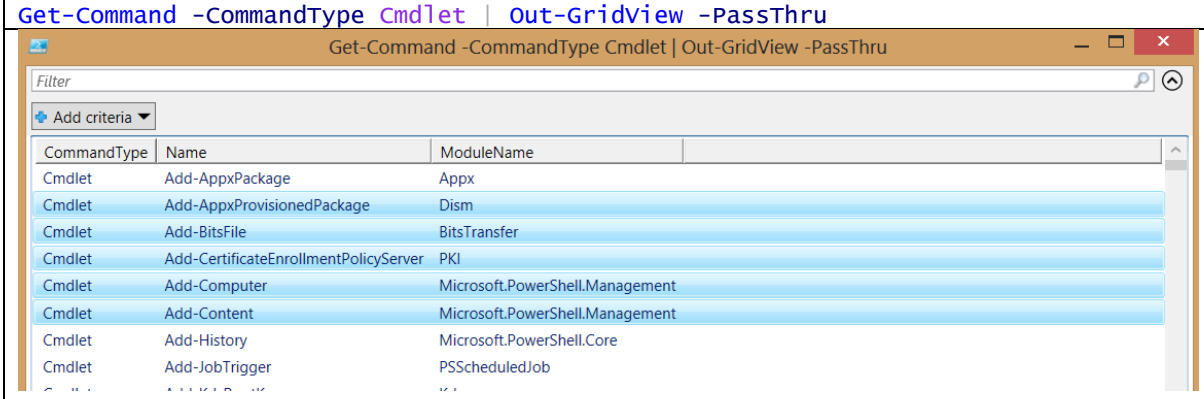
If you have used PowerShell you might have noticed the cmdlet Out-GridView which allows you to put your output inside a Windows Forms dialog. This dialog allow you to filter search commands. In PowerShell 3.0 you Out-GridView has many useful features that will help you develop powerful scripts.

```
#Basic out-GridView cmdlet
Clear-Host
Get-Command -CommandType Cmdlet | Out-GridView
```



#Pass Through will allow you to select from the items and get them as object or print them to output.

```
Clear-Host
Get-Command -CommandType Cmdlet | Out-GridView -PassThru
```

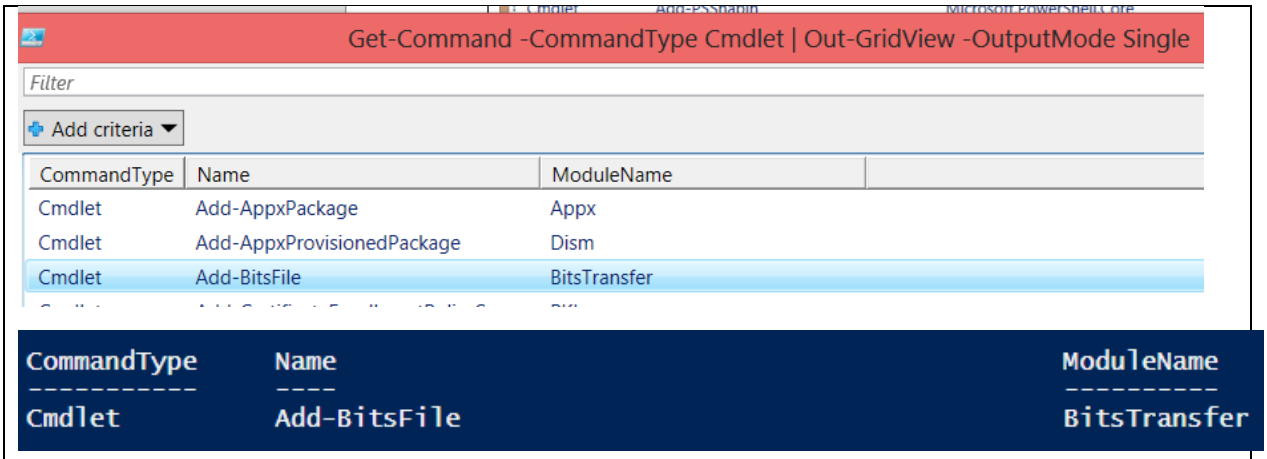


Selected cmdlets

```
CommandType     Name                                ModuleName
-----------     ----                                ----------
Cmdlet          Add-AppxProvisionedPackage          Dism
Cmdlet          Add-BitsFile                        BitsTransfer
Cmdlet          Add-CertificateEnrollmentPolicyServer PKI
Cmdlet          Add-Computer                        Microsoft.Po...
Cmdlet          Add-Content                         Microsoft.Po...
```

#Output Mode will allow you to select single items and get it as object or print it to output.

```
Clear-Host
Get-Command -CommandType Cmdlet | Out-GridView -OutputMode Single
```




```
#Adding Selected item from grid view to variable
$CommandInVariable
Get-Command -CommandType Cmdlet |
    Out-GridView -OutputMode Single -OutVariable $CommandInVariable
```

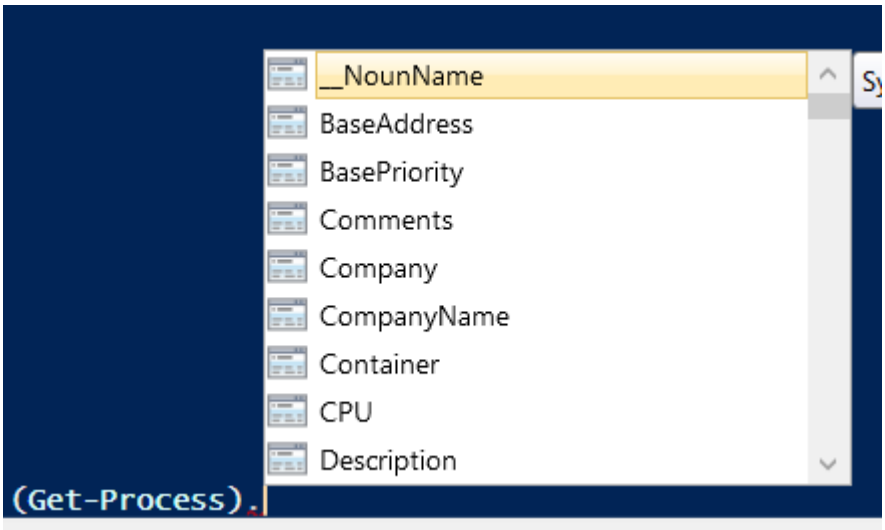
```
CommandType     Name                                               ModuleName
-----------     ----                                               ----------
Cmdlet          Copy-Item                                          Microsoft.PowerShell.Management
```

Another nice feature which is added in PowerShell 3.0 is access to properties of an Object collection directly.

In PowerShell 2.9 to access an object collection property you must use ForEach.

Get-Process | ForEach($_.Id) which allows you to read Ids of all the process

Now you can get the property values directly using parentheses () around your object.
(Get-Process).Id



Full Support for CMD scripts.
You can get Count or Length of Property even if object does not have Count Property
You can Index any object even if it does not support it.
You don't need to use script blocks in Where-Object command to filter objects.

```
Get-Process | where ProcessName -Match "avp"
```

You don't need to import any module prior to running the command.
You can download latest help files using PowerShell using Update-Help cmdlet. If your server does not have internet you can use Save-Help cmdlet from an updated server to save the file on the disk and then use Update-Help cmdlet to update from this file.

Workflows in PowerShell 3.0

PowerShell 3.0 introduced the concepts of Workflows. Workflows in PowerShell are built on top of Windows workflow Foundation. Any code that you write will be converted to WWF objects. The main focus of Workflows in is to enable robust and pause and resume able long term processes. Every line of code you write in a workflow is considered as an activity in WWF base workflows. Every activity is a standalone unit and executes as if it is being run for the first time.

There are some restriction in the workflows for more information on workflows please read.

http://technet.microsoft.com/en-us/library/hh857339.aspx

To Begin writing workflows in PowerShell you can start by importing the workflow module in PowerShell Window. This will load all the cmdlets for workflows.

```
Import-Module PSWorkflow
```

#Syntax is workflow WorkflowName { *script* }. You can also use Invoke-WorkflowName

```
workflow Run-MyFirstWorkflow
{
    "Hi Everyone I am running in a workflow"
}
```

#To run the Workflow call it as function.
```
Run-MyFirstWorkflow
```

#You can also run the function as Job. Running a workflow as job will hide the output of the workflow.
```
$WorkflowJob = Run-MyFirstWorkflow -AsJob
```
#You can check the status of the Job if it is long running and pause and continue it.
```
$Get-Job $WorkflowJob.Name
```

```
PS C:\Users\YasirAttiq> Get-Job $WorkflowJob.Name

Id     Name            PSJobTypeName   State         HasMoreData     Location             Command
--     ----            -------------   -----         -----------     --------             -------
18     Job18           PSWorkflowJob   Completed     True            localhost            Run-MyFirstW...
```

```
$WorkflowJob.State
```

Parallel Operations in Workflows

To execute discrete operations in workflow you can use Parallel operation in Workflow. The cmdlets in parallel workflow will execute in random order or parallel way.

#Example of Workflow with Parallel Tasks

```
Import-Module PSWorkflow
workflow Run-ParallelWorkflow
{
    parallel
    {
        Get-Command -Verb
"Start*"
        "Value is  1"
        "Value is  2"
        "Value is  3"
        "Value is  4"
        "Value is  5"
        "Value is  6"
        "Value is  7"
        "Value is  8"
        "Value is  9"
        "Value is  10"
        Get-Command -Noun
"Computer*"
    }
}

#Run the Workflow
Run-ParallelWorkflow
```

```
Value is  1
Value is  2
Value is  3
Value is  4
Value is  5
Value is  6
Value is  7

CommandType     Name                                               ModuleName                     PSComp
-----------     ----                                               ----------                     ------
Function        Start-Dtc                                          MsDtc                          local
Function        Start-DtcTransactionsTraceSession                  MsDtc                          local
Function        Start-ScheduledTask                                ScheduledTasks                 local
Function        Start-Trace                                        PSDiagnostics                  local
Cmdlet          Start-BitsTransfer                                 BitsTransfer                   local
Cmdlet          Start-DtcDiagnosticResourceManager                 MsDtc                          local
Cmdlet          Start-Job                                          Microsoft.PowerShell.Core      local
Cmdlet          Start-Process                                      Microsoft.PowerShell.Management local
Cmdlet          Start-Service                                      Microsoft.PowerShell.Management local
Cmdlet          Start-Sleep                                        Microsoft.PowerShell.Utility   local
Cmdlet          Start-Transaction                                  Microsoft.PowerShell.Management local
Cmdlet          Start-Transcript                                   Microsoft.PowerShell.Host      local
Cmdlet          Start-VM                                           Hyper-V                        local
Cmdlet          Start-VMFailover                                   Hyper-V                        local
Cmdlet          Start-VMInitialReplication                         Hyper-V                        local
Value is  8
Value is  9
Value is  10
Cmdlet          Add-Computer                                       Microsoft.PowerShell.Management local
Cmdlet          Checkpoint-Computer                                Microsoft.PowerShell.Management local
Cmdlet          Disable-ComputerRestore                            Microsoft.PowerShell.Management local
Cmdlet          Enable-ComputerRestore                             Microsoft.PowerShell.Management local
Cmdlet          Get-ComputerRestorePoint                           Microsoft.PowerShell.Management local
Cmdlet          Remove-Computer                                    Microsoft.PowerShell.Management local
Cmdlet          Rename-Computer                                    Microsoft.PowerShell.Management local
Cmdlet          Reset-ComputerMachinePassword                      Microsoft.PowerShell.Management local
Cmdlet          Restart-Computer                                   Microsoft.PowerShell.Management local
Cmdlet          Restore-Computer                                   Microsoft.PowerShell.Management local
Cmdlet          Stop-Computer                                      Microsoft.PowerShell.Management local
Cmdlet          Test-ComputerSecureChannel                         Microsoft.PowerShell.Management local
```

Parallel Operations with Sequential Tasks in Workflows

If you have parallel workflow with some tasks that must be completed in a sequence you can use sequence script block to execute those tasks as sequence.

```
Import-Module PSWorkflow
workflow Run-
ParallelWorkflowWithSequenceTasks
{
    parallel
    {
        Get-Command -Verb
"Start*"
        sequence
        {
        "Task with Sequence  1"
        "Task with Sequence  2"
        "Task with Sequence  3"
        "Task with Sequence  4"
        "Task with Sequence  5"
        "Task with Sequence  6"
        "Task with Sequence  7"
        "Task with Sequence  8"
        "Task with Sequence  9"
        "Task with Sequence  10"
        }
        Get-Command -Noun
"Computer*"
    }
}

Run-
ParallelWorkflowWithSequenceTasks
```

```
Task with Sequence  1
Task with Sequence  2
Task with Sequence  3
Task with Sequence  4
Task with Sequence  5
Task with Sequence  6
Task with Sequence  7

CommandType     Name                                               Module
-----------     ----                                               ------
Function        Start-Dtc                                          MsDtc
Function        Start-DtcTransactionsTraceSession                  MsDtc
Function        Start-ScheduledTask                                Schedu
Function        Start-Trace                                        PSDiag
Cmdlet          Start-BitsTransfer                                 BitsTr
Cmdlet          Start-DtcDiagnosticResourceManager                 MsDtc
Cmdlet          Start-Job                                          Micros
Cmdlet          Start-Process                                      Micros
Cmdlet          Start-Service                                      Micros
Cmdlet          Start-Sleep                                        Micros
Cmdlet          Add-Computer                                       Micros
Cmdlet          Start-Transaction                                  Micros
Cmdlet          Checkpoint-Computer                                Micros
Cmdlet          Disable-ComputerRestore                            Micros
Cmdlet          Enable-ComputerRestore                             Micros
Cmdlet          Get-ComputerRestorePoint                           Micros
Task with Sequence  8
Cmdlet          Remove-Computer                                    Micros
Cmdlet          Start-Transcript                                   Micros
Cmdlet          Start-VM                                           Hyper-
Cmdlet          Start-VMFailover                                   Hyper-
Cmdlet          Start-VMInitialReplication                         Hyper-
Cmdlet          Rename-Computer                                    Micros
Cmdlet          Reset-ComputerMachinePassword                      Micros
Cmdlet          Restart-Computer                                   Micros
Cmdlet          Restore-Computer                                   Micros
Cmdlet          Stop-Computer                                      Micros
Cmdlet          Test-ComputerSecureChannel                         Micros
Task with Sequence  9
Task with Sequence  10
```

Parallel For-Each

You can also run for-Each in parallel mode.

```powershell
Import-Module PSWorkflow
workflow Run-ForEachParallel
{
    param([string[]]$Products)
    foreach -parallel($Product in $Products)
    {
        "Reading $Product"
    }
}

#Create Array and run the workflow

$Products = (1..20)
Run-ForEachParallel $Products;
```

```
PS C:\Users\YasirAttiq> $Products = (1..20)
Run-ForEachParallel $Products;
Reading 20
Reading 19
Reading 18
Reading 17
Reading 16
Reading 15
Reading 14
Reading 13
Reading 12
Reading 11
Reading 10
Reading 9
Reading 7
Reading 5
Reading 4
Reading 8
Reading 6
Reading 3
Reading 2
Reading 1
```

Part II – SharePoint and PowerShell

In this part we will look at how we can use PowerShell in SharePoint. Most of the cmdlets will work same way in SharePoint 2010 and 2013. If a cmdlet is only available in SharePoint 2013 we will point it out.

Add the SharePoint snap-in using the Add-PSSnapin cmdlet.

Although PowerShell automatically loads the cmdlets for Windows and computer management but you can check what is loaded and what is not. Test the below mentioned section by yourself.

1. Close All Existing instances of PowerShell and start a new instance of Windows PowerShell as Administrator
2. Type the following command and then press ENTER:

   Get-PSSnapin

   #The output lists the snap-ins that have been added to the current session. The SharePoint snap-in is not listed.
3. Type the following command and then press ENTER:

   Get-PSSnapin –Registered

   #The output lists the snap-ins that are registered on the system, except for those that are installed with Windows PowerShell.
4. Type the following command and then press ENTER:

   Add-PSSnapin Microsoft.SharePoint.PowerShell

   #Now you will be able to write SharePoint cmdLets in Windows PowerShell.
5. Type the following cmdlet and press ENTER:

   Get-SPSite

   Get all the Site Collections
6. Remote SharePoint Snapin from PowerShell.

   Remove-PSSnapin Microsoft.SharePoint.PowerShell

   #If you try to write a SharePoint cmdlet, PowerShell will not recognize it.
7. Close Windows PowerShell.

Delegating the Ability to Use Windows PowerShell to Manage SharePoint

Using PowerShell for SharePoint require certain permissions. If you have installed SharePoint with Least Privileges then you have given required permissions (Shell Admin role in SQL) to the Human Installation account (e.g. SPAdmin) or any other account which needs to run PowerShell for a specific web application. The task below will allow you to do that using PowerShell.

If you try to start SharePoint Management Shell and try to run a SharePoint cmdlet you will get the following error.

```
PS C:\Users\adamb> Get-SPContentDatabase
Get-SPContentDatabase : The farm is unavailable.
At line:1 char:1
+ Get-SPContentDatabase
+ ~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidData: (Microsoft.Share...ContentDatabase:
   SPCmdletGetContentDatabase) [Get-SPContentDatabase], SPException
    + FullyQualifiedErrorId : Microsoft.SharePoint.PowerShell.SPCmdletGetConte
   ntDatabase
```

Configure least privilege rights to manage SharePoint with Windows PowerShell.

1. Start SharePoint Management Shell based on your operating System (Run as Different User)
    i. W2k8R2: Click **Start**, click **All Programs**, click **Microsoft SharePoint 2010 Products**,
    ii. Win2012: Start Screen → SharePoint Management Shell or use Search Charm

    The Windows Security dialog box appears.
2. In the **User name** box, type **Domain\User Name** who as access to run PowerShell.
3. In the **Password** box, type the password of the account.
4. Click **OK**.
5. Type the following cmdlets each followed by ENTER:

```
$Contentdb = Get-SPContentDatabase WSS_Content_DB
Add-SPShellAdmin -UserName CONTOSO\SPAdmin -Database $Contentdb
```

```
PS C:\Users\Administrator> $ContentDB = Get-SPContentDatabase -Identity WSS_Content_80
PS C:\Users\Administrator> Add-SPShellAdmin -UserName adventure\adamb -database $ContentDB
PS C:\Users\Administrator> Get-SPShellAdmin

UserName
--------
ADVENTURE\adamb
```

Note: the example above I have added Adam Barr as SPShellAdmin.

6. Close the SharePoint Management Shell Window and try to login with SPAdmin user.

```
PS C:\Users\adamb> Get-SPShellAdmin

UserName
--------
ADVENTURE\adamb
```

Use Windows PowerShell to Access SharePoint Objects

1. Start New SharePoint Management Shell Window as administrator.
2. Type the following command and then press ENTER:

Get-SPFarm

This will show us basic information of Farm e.g. Configuration Database Name and Farm Version.
Use Get-Members cmdlet to find properties of Farm.

```
PS C:\Users\adamb> Get-SPFarm | Get-Member -MemberType Properties


   TypeName: Microsoft.SharePoint.Administration.SPFarm

Name                                      MemberType Definition
----                                      ---------- ----------
AlternateUrlCollections                   Property   Microsoft.SharePoint.Adm
BuildVersion                              Property   version BuildVersion {ge
CanBackupRestoreAsConfiguration           Property   bool CanBackupRestoreAsC
CanMigrate                                Property   bool CanMigrate {get;}
CanRenameOnRestore                        Property   bool CanRenameOnRestore
CanSelectForBackup                        Property   bool CanSelectForBackup
CanSelectForRestore                       Property   bool CanSelectForRestore
CanUpgrade                                Property   bool CanUpgrade {get;}
CEIPEnabled                               Property   bool CEIPEnabled {get;se
DaysBeforePasswordExpirationToSendEmail Property   int DaysBeforePasswordEx
```

#Let's try to read more information from Farm and format the results.
Get-SPFarm | Select Servers, Products, Solutions | Format-List

```
PS C:\Users\adamb> Get-SPFarm | Select Servers, Products, Solutions | Format-Lis
t


Servers   : {smtp.gmail.com, SP2013}
Products  : {9ff54ebc-8c12-47d7-854f-3865d4be8118,
            b7d84c2b-0754-49e4-b7be-7ee321dce0a9}
Solutions : {cavalieritlists.wsp, cavalieritwebparts.wsp,
            spcustomclaimsprovider.wsp, testwp.wsp}
```

Searching for Object Properties

As SharePoint Object has many properties it will be difficult to find by scrolling. So you can use Pipe sign and Where to filter and fine some properties that match a criteria. You can also filter methods etc.

Object | Get-Member -MemberType Properties | Where{$_.Name -like "*User*"}
#Example shows you finding property matching user in property name.
$Portal = Get-SPSite http://intranet.adventure.com
$Portal | Get-Member -MemberType Properties | Where{$_.Name -like "*User*"}

```
   TypeName: Microsoft.SharePoint.SPSite

Name                        MemberType Definition
----                        ---------- ----------
UserAccountDirectoryPath    Property   string UserAccountDirectoryPath {get;set;}
UserCodeEnabled             Property   bool UserCodeEnabled {get;}
UserCustomActions           Property   Microsoft.SharePoint.SPUserCustomActionCollection UserCustomActions {get;}
UserDefinedWorkflowsEnabled Property   bool UserDefinedWorkflowsEnabled {get;set;}
UserIsSiteAdminInSystem     Property   bool UserIsSiteAdminInSystem {get;}
UserToken                   Property   Microsoft.SharePoint.SPUserToken UserToken {get;}
```

Working with Web Applications

To Get all the Web Applications you must use the following cmdlet.

#Get All Web Apps
Get-SPWebAppliction

```
PS C:\Users\Administrator> Get-SPWebApplication

DisplayName                     Url
-----------                     ---
Sales Web App                   http://sales.adventure.com/
SharePoint - 80                 http://sp2013/
SharePoint - 82                 http://sp2013:82/
SharePoint - my.adventure.c...  http://my.adventure.com/
```

#Get One Web App

```
PS C:\Users\Administrator> Get-SPWebApplication -Identity http://sales.adventure.com

DisplayName                 Url
-----------                 ---
Sales Web App               http://sales.adventure.com/
```

#Get All Web Application including Site Collection using Nested Piping
Get-SPWebAppliction | Get-SPSite

```
PS C:\Users\Administrator> Get-SPWebApplication | Get-SPSite

Url                                                CompatibilityLevel
---                                                ------------------
http://sales.adventure.com                         15
http://sales.adventure.com/sites/Products          15
http://sp2013                                      15
http://sp2013/my                                   15
http://sp2013/personal/adamb                       15
http://sp2013/personal/administrator               15
http://sp2013/sites/Apps                           15
http://sp2013/sites/Community                      15
http://dev.adventure.com                           15
http://intranet.adventure.com                      15
http://search.adventure.com                        15
http://my.adventure.com                            15
http://my.adventure.com/personal/sp_search         15
```

Note: The Default Limit of Site Collection return by Get-SPSite cmdLet is 20. Use –Limit All to Get all. Not a good practice to run this cmdlet on production environment.

#Get All Web Application including Site Collection and Sites (Webs) using Nested Piping

```
PS C:\Users\Administrator> Get-SPWebApplication | Get-SPSite | Get-SPWeb

Url
---
http://sales.adventure.com
http://sales.adventure.com/sites/Products
http://sp2013
http://sp2013/my
http://sp2013/personal/adamb
http://sp2013/personal/administrator
http://sp2013/sites/Apps
http://sp2013/sites/Community
http://dev.adventure.com
http://app-02fdfdf4f9f3b8.apps.adventure.com/HelloSh...
http://app-02fdfdf4f9f383.apps.adventure.com/RemoteE...
http://app-02fdfdf4f9f3a4.apps.adventure.com/RESTSPJ...
http://app-02fdfdf4f9f399.apps.adventure.com/UserPro...
http://intranet.adventure.com
http://search.adventure.com
http://my.adventure.com
http://my.adventure.com/personal/sp_search
```

#Get One Web Application and its Site collections using Variable
$SalesWebApp = Get-SPWebApplication -Identity http://sales.adventure.com

$SalesWebApp | Get-SPSite

```
PS C:\Users\Administrator> $SalesWebApp = Get-SPWebApplication -Identity http://sales.adventure.com
PS C:\Users\Administrator> $SalesWebApp | Get-SPSite

Url                                                CompatibilityLevel
---                                                ------------------
http://sales.adventure.com                         15
http://sales.adventure.com/sites/Products          15
```

#Get Web Application Properites
$SalesWebApp | Select MaximumFileSize, Id, MaxItemsPerThrottledOperation, UseClaimsAuthentication, ContentDatabases | Format-List

```
PS C:\Users\Administrator> $SalesWebApp | Select MaximumFileSize, Id, MaxItemsPerThrottledOperation, UseClaimsAuthentic
tion, ContentDatabases | Format-List


MaximumFileSize              : 250
Id                           : 2f5075a3-04c0-42ed-8662-fd14a3e33622
MaxItemsPerThrottledOperation : 5000
UseClaimsAuthentication      : True
ContentDatabases             : {WSS_Content_Sales}
```

#Change Maximum Upload File Size
$SalesWebApp.MaximumFileSize = 500
$SalesWebApp.Update()
$SalesWebApp | Select MaximumFileSize, Id, MaxItemsPerThrottledOperation, UseClaimsAuthentication, ContentDatabases | Format-List

```
PS C:\Users\Administrator> $SalesWebApp | Select MaximumFileSize, Id, MaxItemsPerThrottledOperation, UseClaimsAuthentica
tion, ContentDatabases | Format-List


MaximumFileSize              : 500
Id                           : 2f5075a3-04c0-42ed-8662-fd14a3e33622
MaxItemsPerThrottledOperation : 5000
UseClaimsAuthentication      : True
ContentDatabases             : {WSS_Content_Sales}
```

#**Creating a New Web Application**
In SharePoint 2010 you have the option to choose the Authentication Method to Classic Claims but on ther other than 2013 only have Claims as default. So If you don't not specify the authtication provider in SharePoint 2013. The Web application will be created as Classic. See the Example Below.

#**SharePoint 2010 – Class Mode Authentication in SharePoint 2013 VM**
New-SPWebApplication -Port 89 -Name "SharePoint Web Application Classic" -DatabaseName "WSS_Content_Classic_89" -ApplicationPool "WSS_Content_Classic_AppPool" -ApplicationPoolAccount adventure\administrator

```
PS C:\Users\Administrator> New-SPWebApplication -Port 89 -Name "SharePoint Web Application Classic" -DatabaseName "WSS_C
ontent_Class_89" -ApplicationPool "WSS_Content_Class_AppPool" -ApplicationPoolAccount adventure\administrator
WARNING: The Windows Classic authentication method is deprecated in this release and the default behavior of this
cmdlet, which creates Windows Classic based web application, is obsolete. It is recommended to use Claims
authentication methods. You can create a web application that uses Claims authentication method by specifying the
AuthenticationProvider parameter set in this cmdlet. Refer to the http://go.microsoft.com/fwlink/?LinkId=234549 site
for more information. Please note that the default behavior of this cmdlet is expected to change in the future release
to create a Claims authentication based web application instead of a Windows Classic based web application.

DisplayName                    Url
-----------                    ---
SharePoint Web Application ... http://sp2013:89/
```

#SharePoint 2013 Default Web Application
$AuthenticationProvider = New-SPAuthenticationProvider
New-SPWebApplication -Name "SharePoint 2013 Claims Web App" -Port 90 -DatabaseName WSS_Content_Claims_90 -ApplicationPool "SharePoint 2013 Default App Pool" -ApplicationPoolAccount adventure\administrator -AuthenticationProvider $AuthenticationProvider

```
PS C:\Users\Administrator> New-SPWebApplication -Name "SharePoint 2013 Claims Web App" -Port 90 -DatabaseName WSS_Conten
t_Claims_90 -ApplicationPool "SharePoint 2013 Default App Pool" -ApplicationPoolAccount adventure\administrator -Authent
icationProvider $AuthenticationProvider

DisplayName                    Url
-----------                    ---
SharePoint 2013 Claims Web App http://sp2013:90/
```

#Check Authentication Provider

Get-SPWebapplication | Select Url, UseClaimsAuthentication

```
PS C:\Users\Administrator> Get-SPWebapplication | Select Url, UseClaimsAuthentication

Url                                                                    UseClaimsAuther
---                                                                    ---------------
http://sales.adventure.com/
http://sp2013/
http://sp2013:82/
http://my.adventure.com/
http://sp2013:90/
http://sp2013:89/
```

#Delete a Web Application

Remove-SPWebApplication -Identity http://sp2013:89/ -DeleteIISSite -RemoveContentDatabases –Confirm

```
PS C:\Users\Administrator> Remove-SPWebApplication -Identity http://sp2013:89/ -DeleteIISSite -RemoveContentDatabases -C
onfirm

Confirm
Are you sure you want to perform this action?
Performing operation "Remove-SPWebApplication" on Target "SharePoint Web Application Classic".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"): Y
PS C:\Users\Administrator>
```

Note: It is recommended to use –Confirm before doing a delete operation.

Some Other important cmdLets for Web Application

New-SPContentDatabase

Dismount-SPContentDatabase

Mount-SPContentDatabase

Upgrade-SPContentDatabase

Convert-SPContentDatabase

Creating Site Collection

You can create site collection using PowerShell. SharePoint 2010 we used to have URL based site collection e.g. http://sharepoint/sites/sales but in sharePoint 2013 Microsoft Suggest to use Host name site collections.

#Traditional Site Collections

New-SPSite -Url http://sp2013:90 -OwnerAlias adventure\administrator -Name "My Team Site" –Template "STS#0"

```
PS C:\Users\Administrator> New-SPSite -Url http://sp2013:90 -OwnerAlias adventure\administrator -Name "My Team Site" -Te
mplate "STS#0"

Url                                    CompatibilityLevel
---                                    ------------------
http://sp2013:90                       15
```

#Creating Host Name Site Collections

$hnscWebApp = Get-SPWebApplication http://sp2013

$rootSite = New-SPSite -Name "Adventure Works Web Site" -Url "http://portal.adventure.com" -HostHeaderWebApplication $hnscWebApp -OwnerAlias "adventure\Administrator" -Template "STS#0" -SecondaryOwnerAlias "adventure\adamb"

Get-SPSite -Identity http://portal.adventure.com

```
PS C:\Users\Administrator> $hnscWebApp = Get-SPWebApplication http://sp2013
PS C:\Users\Administrator> $rootSite = New-SPSite -Name "Adventure Works Web Site" -Url "http://portal.adventure.com" -H
ostHeaderWebApplication $hnscWebApp -OwnerAlias "adventure\Administrator" -Template "STS#0" -SecondaryOwnerAlias "advent
ure\adamb"
PS C:\Users\Administrator> Get-SPSite -Identity http://portal.adventure.com

Url                                    CompatibilityLevel
---                                    ------------------
http://portal.adventure.com            15
```

#Change Site Collection Administrator

$Portal = Get-SPSite -Identity http://portal.adventure.com

Set-SPSite -Identity $Portal -SecondaryOwnerAlias "adventure\alans"

```
$Portal = Get-SPSite -Identity http://portal.adventure.com
$Portal.SecondaryContact
```

```
PS C:\Users\Administrator> $Portal = Get-SPSite -Identity http://portal.adventure.com
PS C:\Users\Administrator> $Portal.SecondaryContact

UserLogin           DisplayName
---------           -----------
i:0#.w|adventure\... Adam Barr


PS C:\Users\Administrator> Set-SPSite -Identity $Portal -SecondaryOwnerAlias "adventure\alans"
```

```
PS C:\Users\Administrator> $Portal = Get-SPSite -Identity http://portal.adventure.com
PS C:\Users\Administrator> $Portal.SecondaryContact

UserLogin            DisplayName
---------            -----------
i:0#.w|adventure\... Alan Steiner
```

```
#Deleting Site Collections
Remove-SPSite -Identity $Portal
```

```
PS C:\Users\Administrator> Remove-SPSite -Identity $Portal

Confirm
Are you sure you want to perform this action?
Performing operation "Remove-SPSite" on Target "http://portal.adventure.com".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"): Y
PS C:\Users\Administrator>
```

```
#Some Important things to do with SPSite
-ContentDatabase (Creating Site Collection in a Separate Content Database)
Copy-SPSite
Move-SPSite
```

Reading and Creating Web Sites

You can also create website in site collections using PowerShell using New-SPWeb. To get a web object you must provide the site collection as input for the web application. Get-SPWeb cmdlet does not work without the reference of Get-SPSite object

```
#Reading Web Sites
Get-SPSite http://sales.adventure.com | Get-SPWeb
```

```
PS C:\Users\Administrator> Get-SPSite http://sales.adventure.com | Get-SP

Url
---
http://sales.adventure.com
```

```
#Create Web Sites in Collection
New-SPWeb -Url http://intranet.adventure.com/Jobs -Template "STS#0" -Language 1033 -Name
"Jobs"
```

```
PS C:\Users\Administrator> New-SPWeb -Url http://intranet.adventure.com/Jobs -Template "STS#0" -Language 1033 -Name "Job
s"

Url
---
http://intranet.adventure.com/Jobs
```

```
#You can also read information from web site e.g. Lists, Users, Groups etc. Use Filter with Get-
Member
$WebSite.Users | Select DisplayName, LoginName,Id
```

```
PS C:\Users\Administrator> $WebSite.Users | Select DisplayName, LoginName,Id

DisplayName                    LoginName                                ID
-----------                    ---------                                --
adventure Administrator        i:0#.w|adventure\administrator            1
System Account                 SHAREPOINT\system                 1073741823
```

```
#Get Lists Information
```

```
$WebSite.Lists | Select Id, Title, ItemCount
```

```
PS C:\Users\Administrator> $WebSite.Lists | Select Id, Title, ItemCount

ID                                   Title                                                ItemCount
--                                   -----                                                ---------
94b8291e-6d03-45a9-901a-08aafe765183 Composed Looks                                              18
dde59de9-0b24-4236-8f4e-f68bd145d802 Documents                                                    0
a36cb2f1-6786-4b55-ad07-121bf02b52d2 Master Page Gallery                                          6
21a33048-8d19-4ce0-ac7f-ed10a46b5c6a MicroFeed                                                    2
e6244a19-c877-4c89-b070-f5af1327f7d8 Site Assets                                                  0
2a979f24-29a1-4f29-9f4c-ba2f19a155c9 Site Pages                                                   2
```

```
#You can also Add Items to Lists using PowerShell
```

Using PowerShell with Activity Directory

While creating test environment we need to create test accounts in Active Directory. User Active Directory cmdLets we can create users as well. Find the script below to create user in active directory.

```
New-ADUser -Name "SharePoint Administrator" -SamAccountName "SPAdmin" -UserPrincipalName "SPAdmin@adventure.com" -Type "User"

$user = "SPAdmin"
$password = ConvertTo-SecureString -String "pass@word1" -AsPlainText -Force
Set-ADAccountPassword -Identity $user -NewPassword $password -Reset

Enable-ADAccount -Identity $user

Set-ADAccountControl -Identity $user -CannotChangePassword $true -PasswordNeverExpires $true
Set-ADUser -Identity $user -ChangePasswordAtLogon $false
```

There are many other cmdlets for Active Directory to find them use the following cmdlet.
Get-Command –Noun "AD*" | Out-GridView

Provisioning Service Application using PowerShell

You can provision Service application using PowerShell. Some of the Service Applications are easy to create and some other require complex PowerShell. Search Service Application would be the most complex due to different steps involved and on the other than Translation Management Service is the easiest one through PowerShell.

Note: Creation of Service Application does not provision the Service Instance. It has to be started manually or using PowerShell.

Creating Work Management Service Application
New-SPWorkManagementServiceApplication –Name "Work Management Service Application" -ApplicationPool "SharePoint Web Services Default" New-SPWorkManagementServiceApplicationProxy -name "Work Management Service Proxy" -ServiceApplication "WM Service
Creating Subscription Settings Service Application

```
$account = Get-SPManagedAccount "<farm account>"

$appPoolSubSvc = New-SPServiceApplicationPool -Name SettingsServiceAppPool -Account
$account

$appSubSvc = New-SPSubscriptionSettingsServiceApplication –ApplicationPool $appPoolSubSvc –
Name SettingsServiceApp –DatabaseName SettingsServiceDB

$proxySubSvc = New-SPSubscriptionSettingsServiceApplicationProxy –ServiceApplication
$appSubSvc
```

Creating App Management Service Application

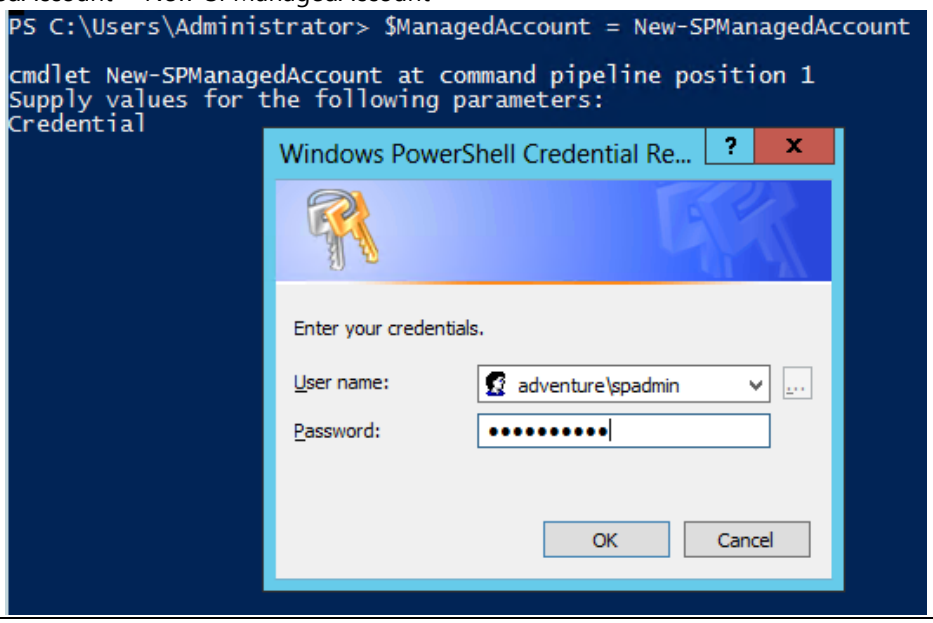
```
New-SPAppManagementServiceApplication -Name "App Management Service Application" -
DatabaseServer MyDatabaseServer -DatabaseName AppManagementDB –ApplicationPool
MyServiceAppPool

New-SPAppManagementServiceApplicationProxy -Name "App Management Proxy" -
UseDefaultProxyGroup -ServiceApplication $appManagementServiceApplication

Set-SPAppDomain –Domain apps.domain.com
Set-SPAppSiteSubscriptionName -Name "app" -Confirm:$false
```

Creating Translation Management Service Application

```
New-SPTranslationServiceApplication –Name "Translation Service Application" -ApplicationPool
“SharePoint Web Services Default” –DatabaseServer MyDatabaseServer –DatabaseName
TranslationServiceDatabase

New-SPTranslationServiceApplicationProxy –Name "Translation Service Application" –
ServiceApplication TranslationService –DefaultProxyGroup
```

Creating Creating Managed Metadata Service Application using PowerShell with Separate application Pool

```
$ManagedAccount = New-SPManagedAccount
```



```
PS C:\Users\Administrator> $ManagedAccount

UserName              PasswordExpiration     Automatic ChangeSchedule
                                             Change
--------              ------------------     --------- --------------
ADVENTURE\SPAdmin     6/26/2013 11:11:02 PM  False
```

$AppPool = New-SPServiceApplicationPool -Name ContosoSalesMMSAppPool -Account $ManagedAccount

```
PS C:\Users\Administrator> $AppPool = New-SPServiceApplicationPool -Name ContosoSalesMMSAppPool -Account $ManagedAccount

PS C:\Users\Administrator> $AppPool

Name                                ProcessAccountName
----                                ------------------
ContosoSalesMMSAppPool              ADVENTURE\SPAdmin
```

$MMS = New-SPMetadataServiceApplication -Name "Adventure Managed Metadata SA" -ApplicationPool $AppPool -DatabaseName AdventureMMSSADB

```
PS C:\Users\Administrator> $MMS = New-SPMetadataServiceApplication -Name "Adventure Managed Metadata SA" -ApplicationPoo
l $AppPool -DatabaseName AdventureMMSSADB
```

New-SPMetadataServiceApplicationProxy -Name "Adventure Managed Metadata SA Proxy" –ServiceApplication $MMS –DefaultProxyGroup

```
PS C:\Users\Administrator> New-SPMetadataServiceApplicationProxy -Name "Adventure Managed Metadata SA Proxy" -ServiceApp
lication $MMS -DefaultProxyGroup

DisplayName          TypeName            Id
-----------          --------            --
Adventure Managed... Managed Metadata ... 07577491-94d1-4c2c-9aaa-d2b8f461d1fd
```

Get-SPServiceApplication |?{$_.name -eq "Adventure Managed Metadata SA"}

```
PS C:\Users\Administrator> Get-SPServiceApplication |?{$_.name -eq "Adventure Managed Metadata SA"}

DisplayName          TypeName            Id
-----------          --------            --
Adventure Managed... Managed Metadata ... fbd7f6ff-3c1a-44fb-a9d0-bc5b5ebbb78d
```

Update Content Type Hub URI

#If you want to change it the content type Hub Uri which is not possible directly once you set it from user interface. To Set the HubUri you have to first get the Metadata Service application and then use Set-SPMetaDataSerivceApplication.

$AMMS = Get-SPServiceApplication |?{$_.name -eq "Adventure Managed Metadata SA"}
Set-SPMetadataServiceApplication -Identity $AMMS -HubUri http://sales.adventure.com

```
PS C:\Users\Administrator> Set-SPMetadataServiceApplication -Identity $AMMS -HubUri http://sales.adventure.com

Confirm
Are you sure you want to perform this action?
Performing operation "Hub uri will be changed to http://sales.adventure.com/. By default, all published packages will
become unpublished, and all subscriber sites will redo syndication. This is a very costly operation.
    And packages have to be republished on the new hub site." on Target "Adventure Managed Metadata SA".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"): Y
```

Creating Excel Services Service App and Starting the Service

```
New-SPExcelServiceApplication –Name "Excel Services Service Application" -
ApplicationPool “SharePoint Web Services System” -Default

#You can do the same for provisioning Services for other Service Applications
$ExcelService = Get-SPServiceInstance | Where TypeName -like "Excel*"
$ExcelService.Provision()
```

Creating Search Service Application using PowerShell.

#The Most Complex Service application script would be SharePoint Server Search. You need to configure the following

- Must provision individual components
- Use **Get-SearchServiceInstance** to get reference to search on a server
- Use **Start-SPSearchServiceInstance** to start search on a server
- Create new topology with **New-SPEnterpriseSearchTopology**

• Add components to the topology • Set the topology with **Set-SPEnterpriseSearchTopology**
Create 4 Search Users • SPSearch. • SPSearchAdmin. • SPSearchQuery. • SPContentAccess To create these accounts quickly with same cmdlets from section **Creating Users in Activity Directory**
start search on individual servers $server1 = Get-SPEnterpriseSearchServiceInstance -Identity "server1" Start-SPEnterpriseSearchServiceInstance –Identity $server1
create search service application $appPool = New-SPServiceApplicationPool -name "SsaAppPool" -account domain\SPSearch $ssa = New-SPEnterpriseSearchServiceApplication -Name "NewSSA" -ApplicationPool $appPool
create topology $newTopology = New-SPEnterpriseSearchTopology -SearchApplication $ssa
add components New-SPEnterpriseSearchAdminComponent -SearchTopology $newTopology -SearchServiceInstance $server1 New-SPEnterpriseSearchCrawlComponent -SearchTopology $newTopology –SearchServiceInstance $server1
New-SPEnterpriseSearchContentProcessingComponent -SearchTopology $newTopology - SearchServiceInstance $server1 New-SPEnterpriseSearchAnalyticsProcessingComponent -SearchTopology $newTopology - SearchServiceInstance $server1 New-SPEnterpriseSearchQueryProcessingComponent -SearchTopology $newTopology - SearchServiceInstance $server1 New-SPEnterpriseSearchIndexComponent -SearchTopology $newTopology –SearchServiceInstance $server1 –IndexPartition 0
set the topology Set-SPEnterpriseSearchTopology -Identity $newTopology # verify the topology Get-SPEnterpriseSearchTopology -SearchApplication $ssa # verify all components Get-SPEnterpriseSearchStatus -SearchApplication $ssa -Text

Creating Warm-Up Script for SharePoint 2013

To Improve the Load time for SharePoint site collections you have use the following cmdlets.

```
$warmupscript = New-Object System.Net.WebClient
$warmupscript.Credentials = [System.Net.CredentialCache]::DefaultCredentials
$warmupscript.DownloadString("http://intranet.adventure.com")
```

Working with Service Instances in SharePoint

If you are facing problem with any of the services in Central Admin site when the service is stuck at either Starting or Stopping and you want to force stop then you can use the following to fix this issue.

Get-SPServiceInstance | Select Id, TypeName, Status | sort typeName, Status

```
Id                                   TypeName                                                 Status
--                                   --------                                                 ------
d289db95-f8c5-4d87-9200-6d03253e70a7 Access Database Service 2010                          Disabled
321b9f0a-4f77-4263-9a7b-239583c65a2c Access Services                                       Disabled
126707ef-fb87-40a4-95da-d7606a91e423 App Management Service                                  Online
6cfebc0f-bcd3-49e8-81cc-9a21742b8f85 Business Data Connectivity Service                      Online
c359d416-fa46-495f-b409-5d5135677d30 Central Administration                                  Online
60019a63-d7e0-42b5-8309-2238f56d5856 Claims to Windows Token Service                       Disabled
3a22c713-6edd-4bf3-8dc6-38091cab6c18 Distributed Cache                                       Online
be3eb909-3e82-44f0-8b34-b38fce2e4680 Document Conversions Launcher Service                 Disabled
315ead7a-9e96-4adf-a741-e848eccfd54b Document Conversions Load Balancer S...               Disabled
bfb9f121-b47b-4dc7-92da-024b14b9d43a Excel Calculation Services                            Disabled
53a5fa19-9c4a-48a2-ab84-4875e499361d Lotus Notes Connector                                 Disabled
badd873b-9aed-48eb-abf0-6da85dba6156 Machine Translation Service                           Disabled
17c99ebf-5b0c-42ed-8e19-a133c8b7f4fe Managed Metadata Web Service                            Online
4d91dac8-cf01-4150-bbc7-06092021f53d Microsoft SharePoint Foundation Inco...                 Online
3ee2e540-fd8e-4a0f-8961-658f8a669c0e Microsoft SharePoint Foundation Sand...               Disabled
05f19501-7fa3-4a29-b986-4f3fe368e97b Microsoft SharePoint Foundation Subs...                 Online
e06cf485-c9b1-4373-a8c7-80a184e203b2 Microsoft SharePoint Foundation Web ...                 Online
84e09c6f-60cc-43bf-8d11-ebc00be95542 Microsoft SharePoint Foundation Work...                 Online
41d82cea-5ab3-4a2e-8181-df7ba2f1c082 PerformancePoint Service                              Disabled
ac6ba573-4aaa-46ee-a218-7e0c8e9b5e34 PowerPoint Conversion Service                         Disabled
0a696a94-8c6e-428d-9f5e-b42942774f94 Request Management                                    Disabled
c5e01d12-544d-46e6-b9eb-7020ba91661e Search Host Controller Service                          Online
f85f9b05-e55d-46bd-91a3-de6751f63318 Search Query and Site Settings Service                  Online
a9e4151a-d4e7-4129-b0d5-c227bf0d8de7 Secure Store Service                                  Disabled
4b280d80-24f1-4e21-9358-4d84ecd852c4 SharePoint Server Search                                Online
8e8ce005-7d48-4390-8090-f2f8fab49fb5 User Profile Service                                    Online
a0054458-4447-4bd1-9e20-c5b918dbe564 User Profile Synchronization Service                    Online
a365f4d1-ef61-4544-80c1-524f7f94991c Visio Graphics Service                                Disabled
5f37d5ba-1b29-4afc-8a15-ab1b836856c7 Word Automation Services                              Disabled
3b8a5811-e29d-4ae4-989b-c1fd25d8689d Work Management Service                               Disabled
```

$Get One Service by TypeName

$BCS = Get-SPServiceInstance | Select Id, TypeName, Status | Where{$_.TypeName -like "Busines
s*"}

```
PS C:\Users\Administrator> $BCS = Get-SPServiceInstance | Select Id, TypeName, Status | Where{$_.TypeName -like "Busines
s*"}
PS C:\Users\Administrator> $BCS

Id                                   TypeName                                                 Status
--                                   --------                                                 ------
6cfebc0f-bcd3-49e8-81cc-9a21742b8f85 Business Data Connectivity Service                      Online


PS C:\Users\Administrator> $BCS.Status
Online
```

Note: You can use $Var.UnProvision() or $Var.Stop() to stop the service.

Upgrading Content Databases and Site Collections using PowerShell

As most of you will be performing upgrades from SharePoint 2010 to 2013. Learning and understanding PowerShell is vital because you can perform Upgrade only using PowerShell. You must be aware that SharePoint 2013 Content DB upgrade is different than upgrade from 2007 to 2010. In SharePoint 2010 the upgrade process upgrade everything in the content database when we mount it to 2010 farm. After that we have to run some PowerShell script to upgrade UI Version to 4. This has changed in 2013 content database upgrade. The mount database will only upgrade the schema of the database but will not do anything to site collections. We will have to options to upgrade every site collection from UI. We can also do the upgrade from PowerShell. We also have an evaluation site which we get as a copy of our site collection for review. The upgrade process is managed by a Timer Job which runs overnight. You can control how many site collection can be upgraded at one time to tweak the process using PowerShell. Below are some examples of how to upgrade your content databases and site collection.

1. Install and configure your SharePoint 2013 environment.
2. Go to your SharePoint 2010 Farm and Set the Read-only property to true using SQL Server Management Studio.

3. Take Backup of your content Database from SQL Server Management Studio of 2010 and restore to SharePoint 2013 SQL Server Management Studio in 2013 environment
4. Set the Read-Only property to false.
5. In SharePoint 2013 Central Administration site → Create a new Web Application but do not create a site collection →
6. Delete the empty content database created by the web application.
7. Install Solutions

   If your web application has any custom solutions installed then you have to get those solutions and install them to the farm. You can use the script mentioned below to download all the solutions from SharePoint 2010 farm.

   Go to SharePoint 2010 Farm → Start SharePoint Management Shell (as Administrator)

```
$SolutionsDirectory = "c:\CustomSolutions"
foreach ($solution in Get-SPSolution)
{
   $title = $Solution.Name
   $filename = $Solution.SolutionFile.Name
   $solution.SolutionFile.SaveAs("$SolutionsDirectory\$filename")
}
```

8. Copy the Solutions from SharePoint 2010 to 2013 environment.
9. You can add and deploy the solutions using PowerShell.

   Add-SPSolution "c:\CustomSolutions\MySharePointSolution.wsp"
   Install-SPSolution –Identity MySharePointSolution.wsp –WebApplication http://sp2013webapp –GACDeployment -CompatibilityLevel (14 or 15)

   #You can write bit fancy step to automate the installation of solutions using #Get-ChildItems cmlet with foreach loop.
   #Why writing if something is already there
   http://gallery.technet.microsoft.com/projectserver/Deploy-multiple-wsp-file-1114692f
10. Now open SharePoint 2013 Management Shell and test your content database
    Test-SPContentDatabase –Name WSS_ContentDBName –WebApplication http://sp2013webapp
11. Review and Fix any errors specially if there is any "Upgrade Blocking" set to true.
12. Now upgrade your content database schema to 2013 using the following cmdlet.
    Mount-SPContentDatabase –Name WSS_ContentDBName –WebApplication http://sp2013webapp
13. You must identity the Site collection that needs to be upgrade. Step one is to check for any issues with the upgrade.
    Test-SPSite –Identity http://yoursitecollection
    #If you get any errors in the report then you have to Repair the site collection. You can ignore the warnings.
    Repair-SPSite –Identity http://yoursitecollection
14. If you want to request an evaluation site you can use the following cmdlet.
    Request-SPUpgradeEvaluationSite http://yoursitecollection
15. Now it's time to run the upgrade
    Upgrade-SPSite http://yoursitecollection –VersionUpgrade
    #The default number of site collection upgrade thottole is 10. If you want to upgrade this site collection immediately use –Unthrottled with the above command.

16. If you want to look at current site collection upgrades that are in progress or have been completed you can use the following cmdlets.

    #Sit Collection Upgrade Session Status
    Get-SPSiteUpgradeSession –Site http://server/sitecolection

    #All Site Collection in a Content Database upgrade Status
    $database = Get-SPContentDatabase MyContentDb
    Get-SPSiteUpgradeSession –ContentDatabase $database –ShowInProgress
17. If you want to disable an upgrade for a site collection you can use the following cmdlet.
    $site.AllowSelfServiceUpgrade = $false;
18. While upgrading your site collection the end users will see a maintenance message which you can cusotmize using the following command.

    $wa=Get-SPWebApplication http://sp2013webapp
    $wa.UpgradeReminderDelay "We are Upgrade your site collection from SP 2010 to 2013. Why?"
    $wa.UpgradeMaintenanceLink = http://www.info.adventure.com/pages/whyupgrade.aspx

    #You can also increase or decrease the reminder for upgrade. Default is 30 days.
    $webApp.UpgradeReminderDelay = 90

    #You can also set the readonline status while an upgrade is in progress or for some maintenance
    $webApp.ReadOnlyMaintenanceLink = "http://link.adventure.com;

    #You Must update your web application object to confirm these changes.
    $webApp.Update()

Managing Licensing using PowerShell

If you ware installing and configuring SharePoint from previous versions you must be aware of Standard of Enterprise versions. You had to make sure that you have to install SharePoint with the right key. So if you by mistake installed the standard key you have to update the key and run the process of activating enterprise features. We do have seen folks having issues with this thing. But overall there was not control after that point. You can deactivate features but it will happen for everyone.

So to solve this problem Microsoft Introduced License management using PowerShell. License management works with Active Directory Groups. You must define your Groups in active directory and add people to the groups based on their licensing requirement. There are 4 Licensing Groups and by name understandable,

1. Standard
2. Enterprise
3. Project
4. WebApps

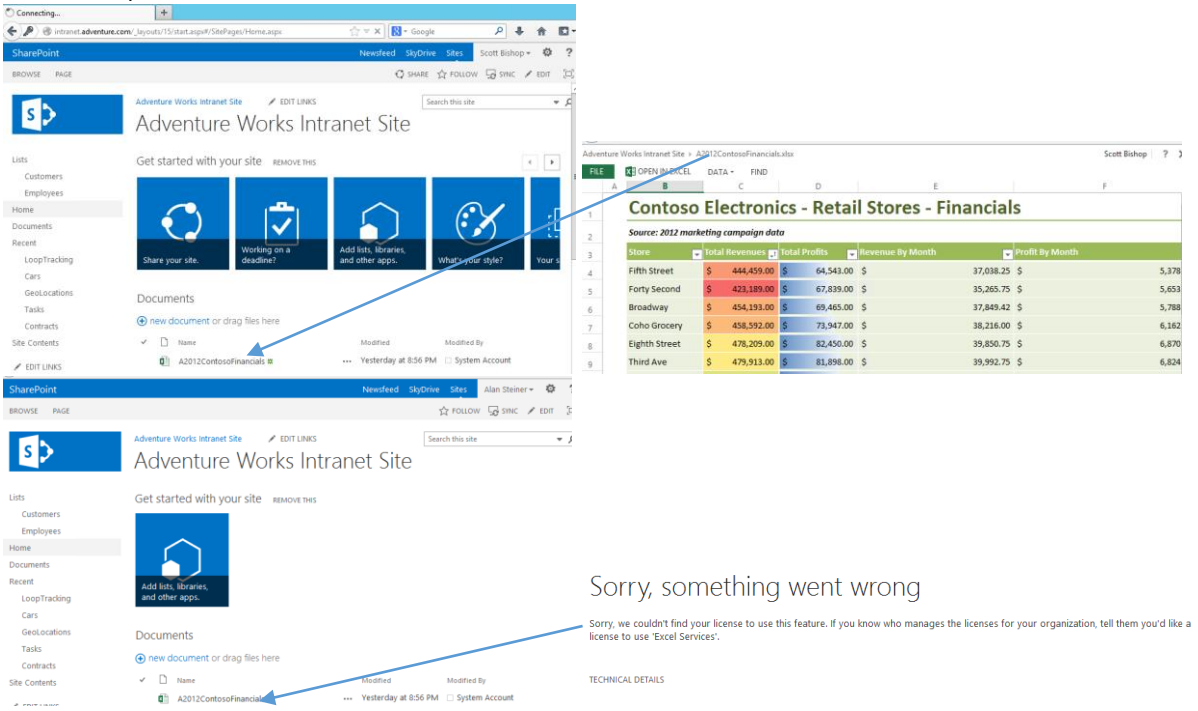
Let's say I have 2 Groups in my Active directory "TestStandard" and "TestEnterprise". Each have one user

```
Add-PSSnapin Microsoft.SharePoint.PowerShell
#License Management
```

```
Get-ADGroup -Filter * | Where Name -Like "Test*" | Select Id, Name

#Set Standard Group as Standard License
$standardusers = New-SPUserLicenseMapping –SecurityGroup “TestStandard” –License
Standard
#Set Enterprise License for Enterprise Groups
$enterpriseusers = New-SPUserLicenseMapping –SecurityGroup “TestEnterprise” –License
Enterprise

$standardusers | Add-SPUserLicenseMapping
$enterpriseusers | Add-SPUserLicenseMapping

Enable-SPUserLicensing
```

The PowerShell Script is easy to understand. Now what will happened when the user logs in to SharePoint? The enterprise group users will continue to use SharePoint without any issue. If the Standard users try to access an enterprise feature they will get an error that they don't have this features. In the below mentioned screen shot you can see that Scott Bishop has access to enterprise features so he is accessing the Excel Sheet using Excel Services. On the other hand Alan Steiner is getting the error.

Scott Bishop



Using PowerShell with SharePoint Online

You can also use PowerShell for Office365 which means PowerShell online support, Exchange, SharePoint, and Lync support.

First you have to download the SharePoint Online Management Shell from Microsoft Download Center installed it to your machine and fire it from your desktop.

First we need to connect the PowerShell Window to SharePoint Online.

Connect-SPOService -Url https://yourtenant-admin.sharepoin t.com -Credential admin@ yourtenant.onmicrosoft.com

To get all the cmdlets which are available in PowerShell for SharePoint online you can use the following cmdlet.
Get-Command –Module Microsoft.Online.SharePoint.PowerShell

```
PS C:\Windows\system32> Get-Command -Module Microsoft.Online.SharePoint.PowerShe
ll

CommandType     Name                                               ModuleName
-----------     ----                                               ----------
Cmdlet          Add-SPOUser                                        Microsoft...
Cmdlet          Connect-SPOService                                 Microsoft...
Cmdlet          Disconnect-SPOService                              Microsoft...
Cmdlet          Get-SPOAppErrors                                   Microsoft...
Cmdlet          Get-SPOAppInfo                                     Microsoft...
Cmdlet          Get-SPODeletedSite                                 Microsoft...
Cmdlet          Get-SPOExternalUser                                Microsoft...
Cmdlet          Get-SPOSite                                        Microsoft...
Cmdlet          Get-SPOSiteGroup                                   Microsoft...
Cmdlet          Get-SPOTenant                                      Microsoft...
Cmdlet          Get-SPOTenantLogEntry                              Microsoft...
Cmdlet          Get-SPOTenantLogLastAvailableTimeInUtc             Microsoft...
Cmdlet          Get-SPOUser                                        Microsoft...
Cmdlet          Get-SPOWebTemplate                                 Microsoft...
Cmdlet          New-SPOSite                                        Microsoft...
Cmdlet          New-SPOSiteGroup                                   Microsoft...
Cmdlet          Remove-SPODeletedSite                              Microsoft...
Cmdlet          Remove-SPOExternalUser                             Microsoft...
Cmdlet          Remove-SPOSite                                     Microsoft...
Cmdlet          Remove-SPOSiteGroup                                Microsoft...
Cmdlet          Remove-SPOUser                                     Microsoft...
Cmdlet          Repair-SPOSite                                     Microsoft...
Cmdlet          Request-SPOUpgradeEvaluationSite                   Microsoft...
Cmdlet          Restore-SPODeletedSite                             Microsoft...
Cmdlet          Set-SPOSite                                        Microsoft...
```

There are not many cmdlets for SharePoint as compared to SharePoint on premises.
Some commonly used cmdLets are given below with example

Get-SPOSite will show you the site collection which exist on your tenant.

```
Url                        Owner                                  Storage Quota
---                        -----                                  -------------
http://spsbmore-publi...                                                  1000
https://spsbmore.shar...                                                  1000
https://spsbmore.shar...                                                  1000
https://spsbmore-my.s...                                                    50
```

Get-SPOSite –Detailed will show you detailed information about your sites.

```
PS C:\Windows\system32> Get-SPOSite -Detailed

Url                        Owner                                  Storage Quota
---                        -----                                  -------------
http://spsbmore-publi... s-1-5-21-4198170677-951206642-2013710...         1000
https://spsbmore.shar... s-1-5-21-4198170677-951206642-2013710...         1000
https://spsbmore.shar... s-1-5-21-4198170677-951206642-2013710...         1000
https://spsbmore-my.s... s-1-5-21-4198170677-951206642-2013710...           50
```

To Get all the properties you can use Get-Members –membertype properties.

```
PS C:\Windows\system32> Get-SPOSite -Detailed | Get-Member -MemberType Properties


   TypeName: Microsoft.Online.SharePoint.PowerShell.SPOSite

Name                          MemberType Definition
----                          ---------- ----------
AllowSelfServiceUpgrade       Property   bool AllowSelfServiceUpgrade {get;set;}
CompatibilityLevel            Property   int CompatibilityLevel {get;}
DenyAddAndCustomizePages      Property   Microsoft.Online.SharePoint.TenantAdmin...
LastContentModifiedDate       Property   datetime LastContentModifiedDate {get;}
LocaleId                      Property   uint32 LocaleId {get;set;}
LockIssue                     Property   string LockIssue {get;}
LockState                     Property   string LockState {get;set;}
Owner                         Property   string Owner {get;set;}
ResourceQuota                 Property   double ResourceQuota {get;set;}
ResourceQuotaWarningLevel     Property   double ResourceQuotaWarningLevel {get;s...
ResourceUsageAverage          Property   double ResourceUsageAverage {get;}
ResourceUsageCurrent          Property   double ResourceUsageCurrent {get;}
Status                        Property   string Status {get;}
```

Creating a Site Collection

New-SPOSite -Url https://spsbmore.sharepoint.com/sites/sitecollection -Owner jerry@spsbmore.onmicrosoft.com -StorageQuota 100 -CompatibilityLevel 15 -LocaleId 1033 -ResourceQuota 100 -Template "STS#0" -Title "My Site Collection"

```
PS C:\Windows\system32> Get-SPOSite | Where Url -Like "*collection"

Url                           Owner                                Storage Quota
---                           -----                                -------------
https://spsbmore.shar...      jerry@spsbmore.onmicrosoft.com                 100
```

Using PowerShell to Management Office 365

Just like SharePoint Online you can also manage Office365 with PowerShell. First you must install the PowerShell Module for Active Directory which is available at link below.

You have to install the following

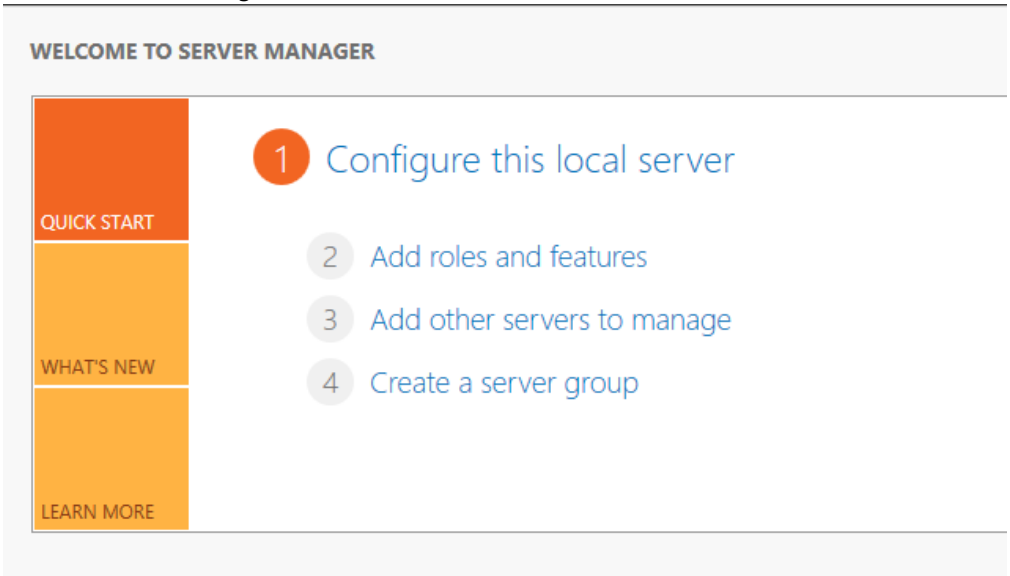
Microsoft Online Services Sign-In Assistant for IT Professionals RTW
http://www.microsoft.com/en-us/download/details.aspx?id=28177

Windows Azure Active Directory Module for Windows PowerShell (64-bit version)
http://go.microsoft.com/fwlink/p/?linkid=236297

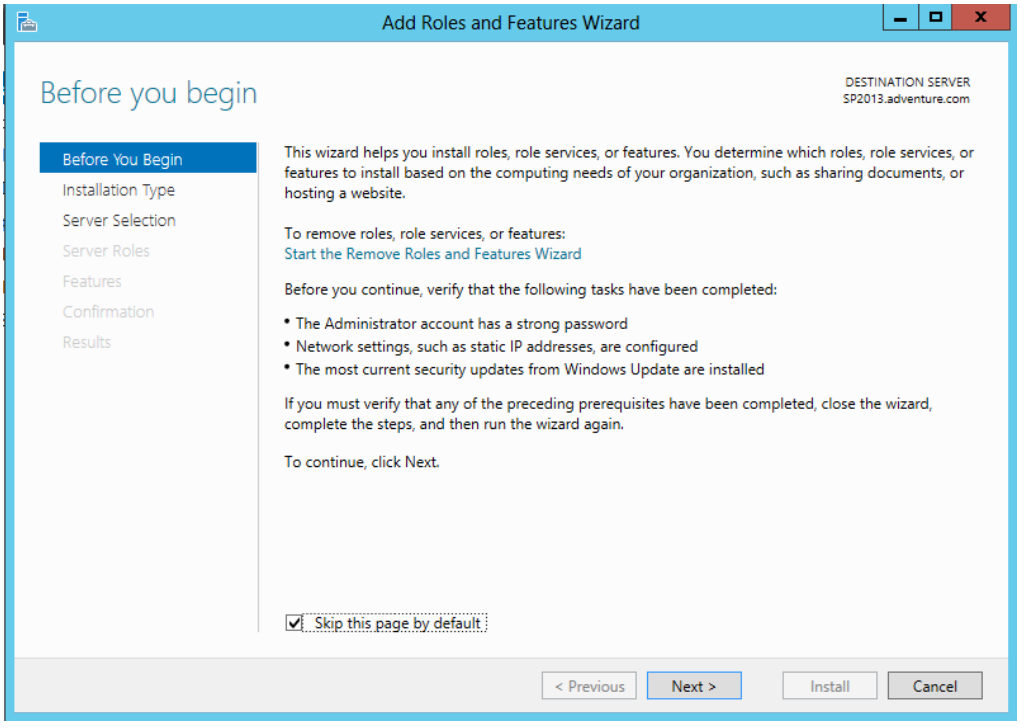
Enabling Windows PowerShell Web Access on Windows Server 2012

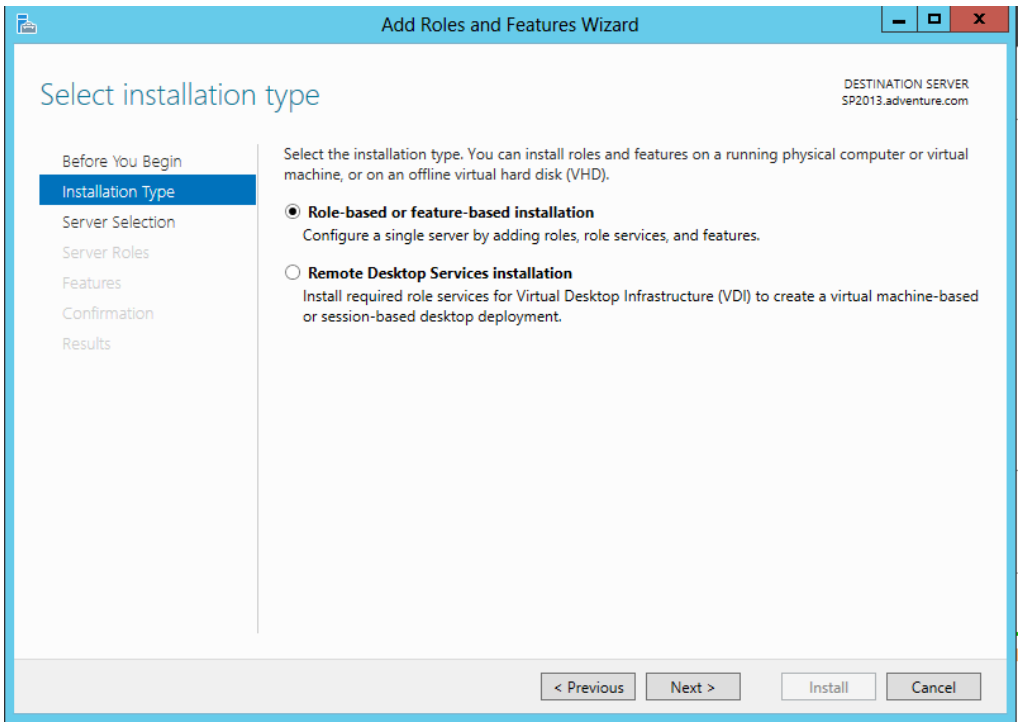
PowerShell allow Administrator to manage servers using web based management shell window.

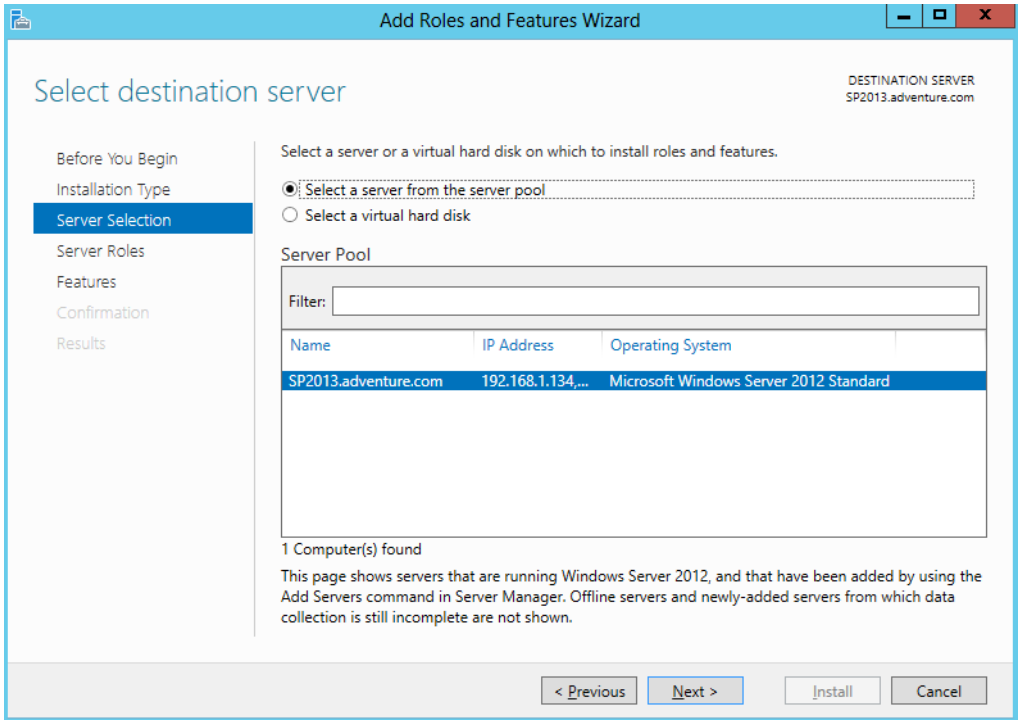
Click on Server Manager ICON from Task Bar.

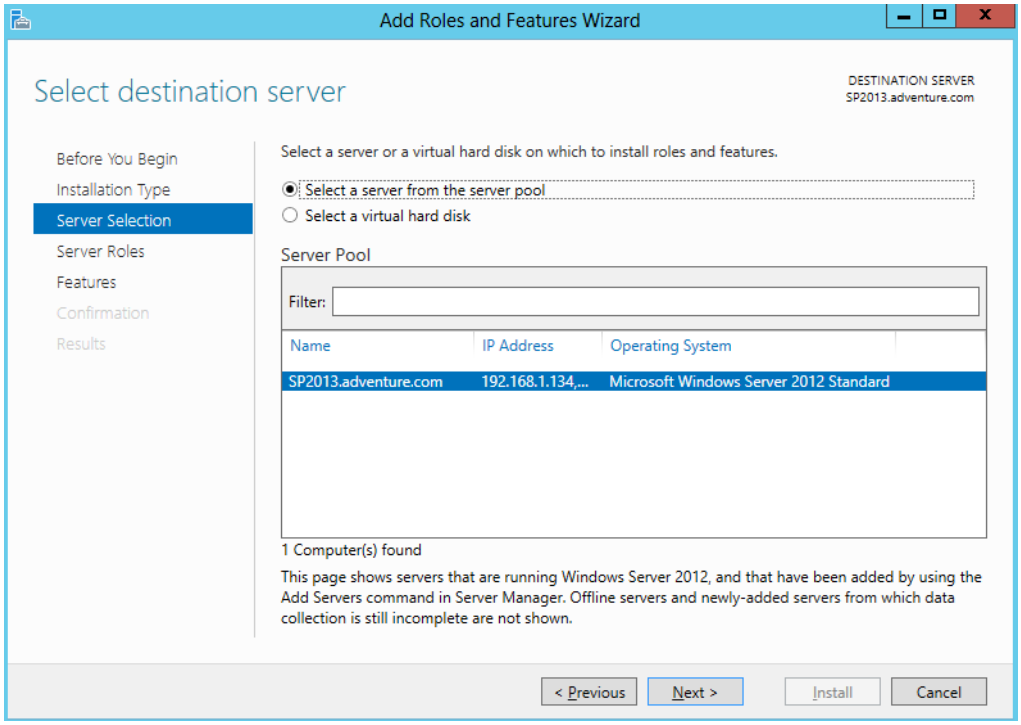


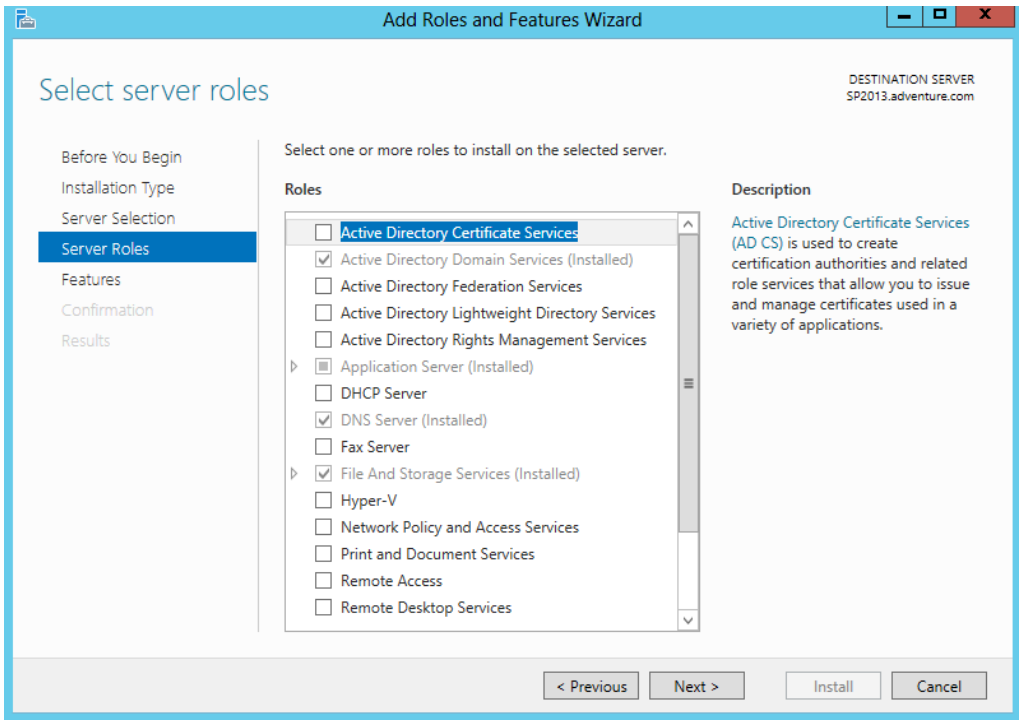
Click Add Role and Feature

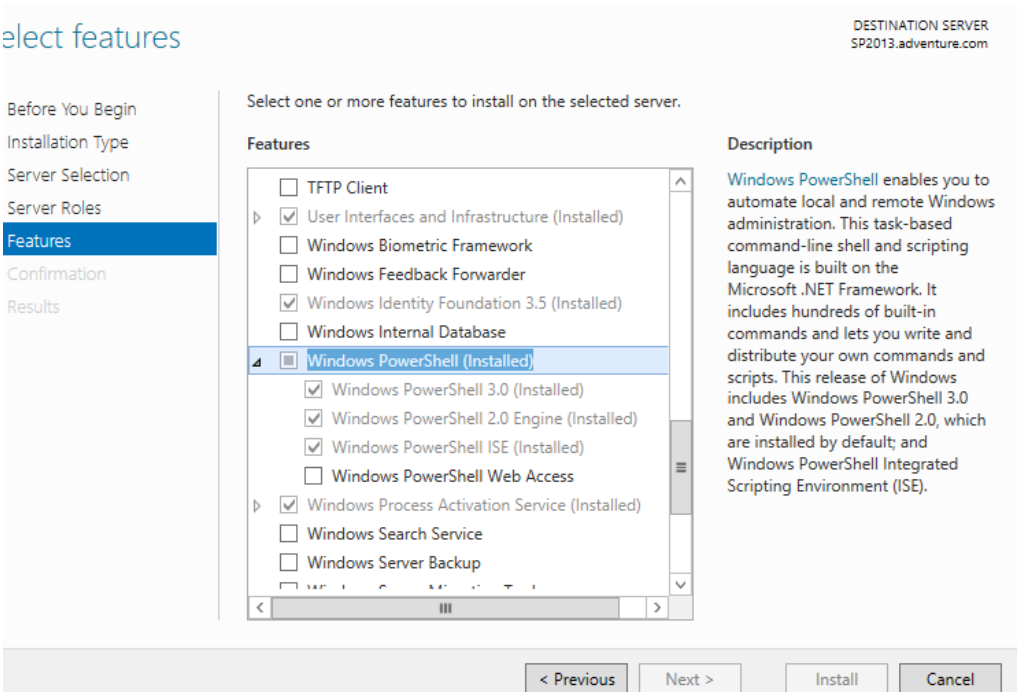


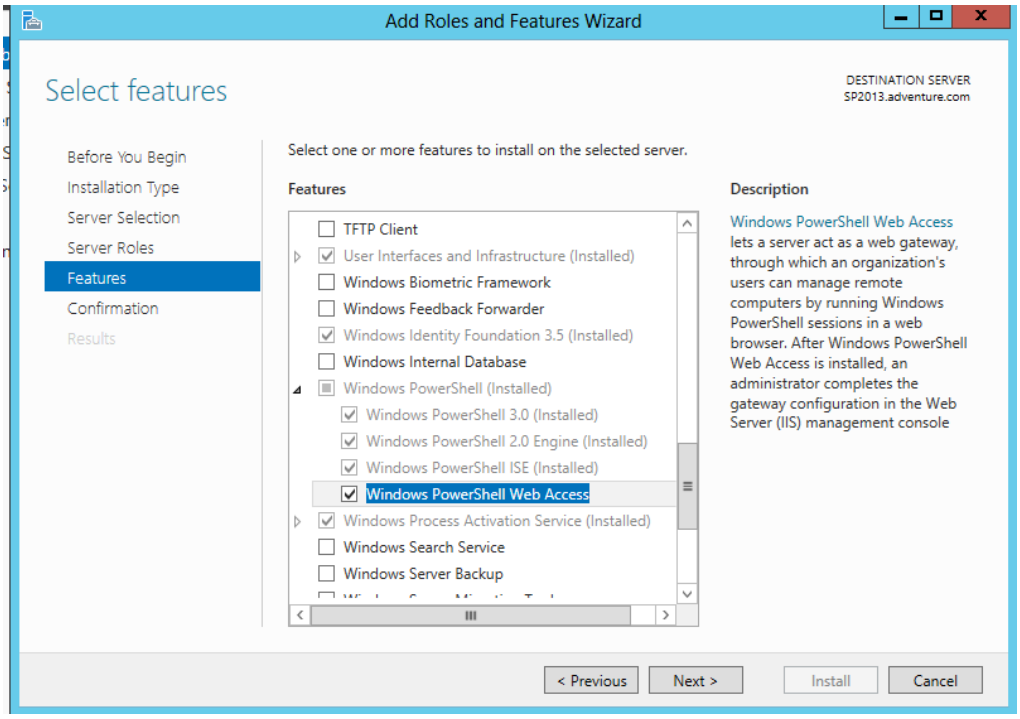

Add Roles and Features Wizard
Select installation type
DESTINATION SERVER
SP2013.adventure.com
Before You Begin
Installation Type
Server Selection
Server Roles
Features
Confirmation
Results
Select the installation type. You can install roles and features on a running physical computer or virtual machine, or on an offline virtual hard disk (VHD).
Role-based or feature-based installation
Configure a single server by adding roles, role services, and features.
Remote Desktop Services installation
Install required role services for Virtual Desktop Infrastructure (VDI) to create a virtual machine-based or session-based desktop deployment.
< Previous
Next >
Install
Cancel

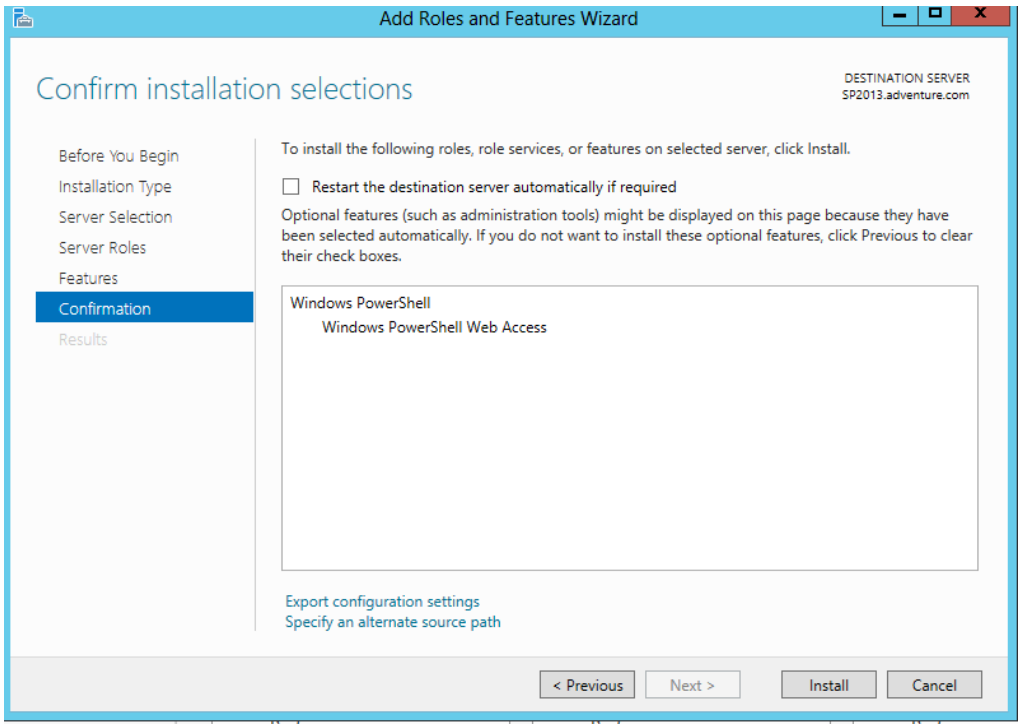

Add Roles and Features Wizard
Select destination server
DESTINATION SERVER
SP2013.adventure.com
Before You Begin
Installation Type
Server Selection
Server Roles
Features
Confirmation
Results
Select a server or a virtual hard disk on which to install roles and features.
Select a server from the server pool
Select a virtual hard disk
Server Pool
Filter:
Name
IP Address
Operating System
SP2013.adventure.com
192.168.1.134,...
Microsoft Windows Server 2012 Standard
1 Computer(s) found
This page shows servers that are running Windows Server 2012, and that have been added by using the Add Servers command in Server Manager. Offline servers and newly-added servers from which data collection is still incomplete are not shown.
< Previous
Next >
Install
Cancel

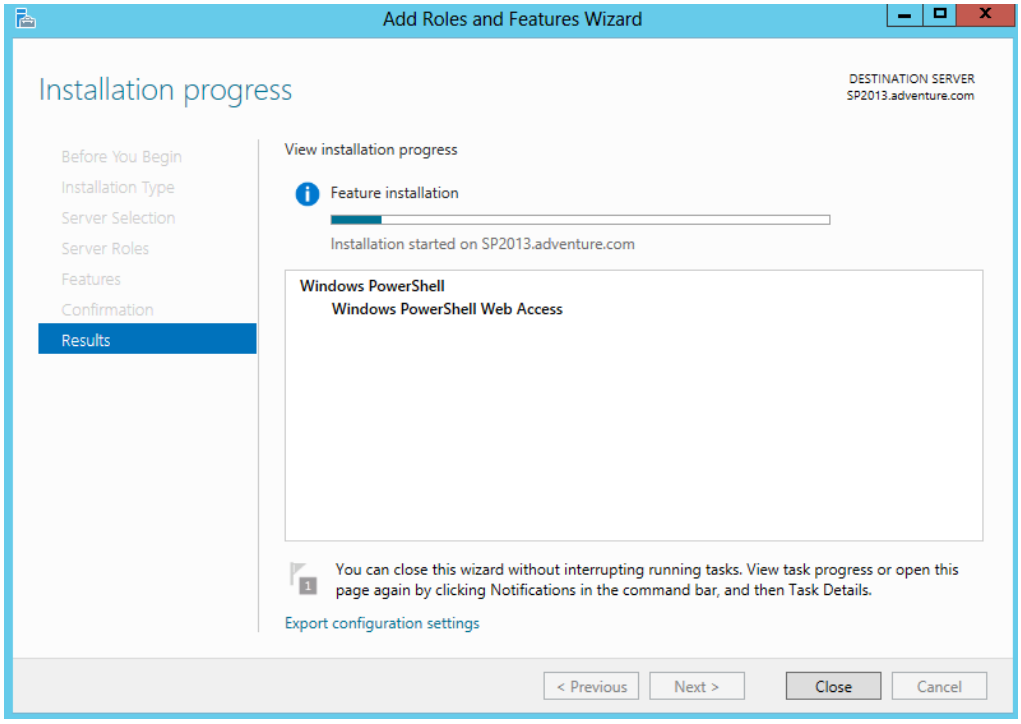

Add Roles and Features Wizard
Select destination server
DESTINATION SERVER
SP2013.adventure.com
Before You Begin
Installation Type
Server Selection
Server Roles
Features
Confirmation
Results
Select a server or a virtual hard disk on which to install roles and features.
Select a server from the server pool
Select a virtual hard disk
Server Pool
Filter:
Name
IP Address
Operating System
SP2013.adventure.com
192.168.1.134,...
Microsoft Windows Server 2012 Standard
1 Computer(s) found
This page shows servers that are running Windows Server 2012, and that have been added by using the Add Servers command in Server Manager. Offline servers and newly-added servers from which data collection is still incomplete are not shown.
< Previous
Next >
Install
Cancel

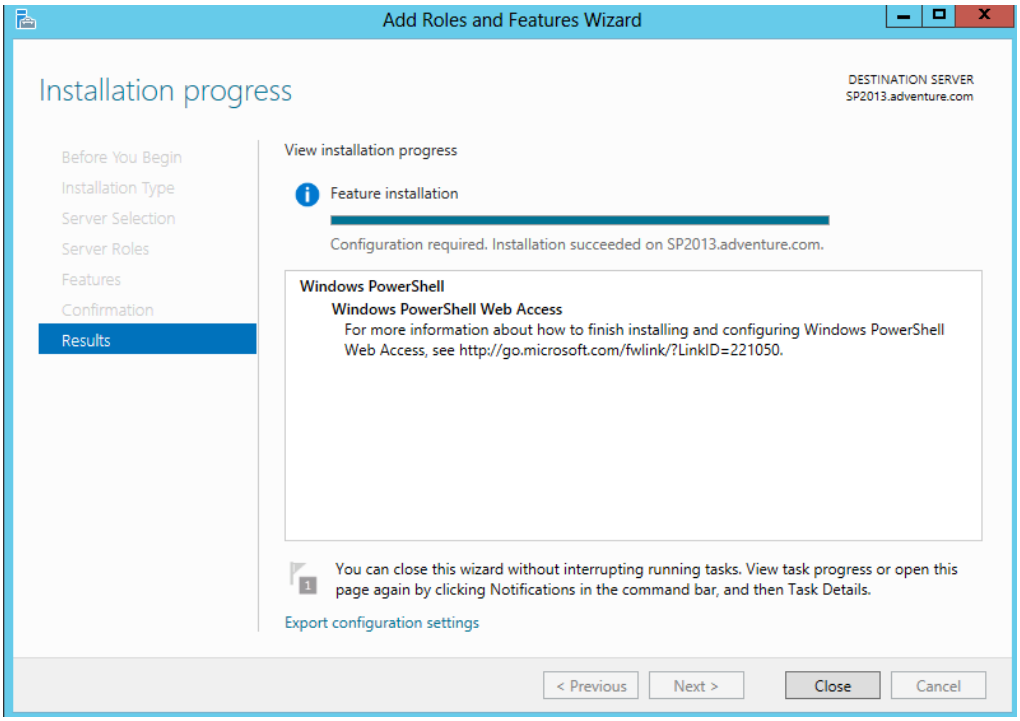

Add Roles and Features Wizard
Select server roles
DESTINATION SERVER
SP2013.adventure.com
Before You Begin
Installation Type
Server Selection
Server Roles
Features
Confirmation
Results
Select one or more roles to install on the selected server.
Roles
Active Directory Certificate Services
Active Directory Domain Services (Installed)
Active Directory Federation Services
Active Directory Lightweight Directory Services
Active Directory Rights Management Services
Application Server (Installed)
DHCP Server
DNS Server (Installed)
Fax Server
File And Storage Services (Installed)
Hyper-V
Network Policy and Access Services
Print and Document Services
Remote Access
Remote Desktop Services
Description
Active Directory Certificate Services (AD CS) is used to create certification authorities and related role services that allow you to issue and manage certificates used in a variety of applications.
< Previous
Next >
Install
Cancel


elect features
DESTINATION SERVER
SP2013.adventure.com
Before You Begin
Installation Type
Server Selection
Server Roles
Features
Confirmation
Results
Select one or more features to install on the selected server.
Features
TFTP Client
User Interfaces and Infrastructure (Installed)
Windows Biometric Framework
Windows Feedback Forwarder
Windows Identity Foundation 3.5 (Installed)
Windows Internal Database
Windows PowerShell (Installed)
Windows PowerShell 3.0 (Installed)
Windows PowerShell 2.0 Engine (Installed)
Windows PowerShell ISE (Installed)
Windows PowerShell Web Access
Windows Process Activation Service (Installed)
Windows Search Service
Windows Server Backup
Description
Windows PowerShell enables you to automate local and remote Windows administration. This task-based command-line shell and scripting language is built on the Microsoft .NET Framework. It includes hundreds of built-in commands and lets you write and distribute your own commands and scripts. This release of Windows includes Windows PowerShell 3.0 and Windows PowerShell 2.0, which are installed by default; and Windows PowerShell Integrated Scripting Environment (ISE).
< Previous
Next >
Install
Cancel


Add Roles and Features Wizard
Select features
DESTINATION SERVER
SP2013.adventure.com
Before You Begin
Installation Type
Server Selection
Server Roles
Features
Confirmation
Results
Select one or more features to install on the selected server.
Features
TFTP Client
User Interfaces and Infrastructure (Installed)
Windows Biometric Framework
Windows Feedback Forwarder
Windows Identity Foundation 3.5 (Installed)
Windows Internal Database
Windows PowerShell (Installed)
Windows PowerShell 3.0 (Installed)
Windows PowerShell 2.0 Engine (Installed)
Windows PowerShell ISE (Installed)
Windows PowerShell Web Access
Windows Process Activation Service (Installed)
Windows Search Service
Windows Server Backup
Description
Windows PowerShell Web Access lets a server act as a web gateway, through which an organization's users can manage remote computers by running Windows PowerShell sessions in a web browser. After Windows PowerShell Web Access is installed, an administrator completes the gateway configuration in the Web Server (IIS) management console
< Previous
Next >
Install
Cancel


Add Roles and Features Wizard
Confirm installation selections
DESTINATION SERVER
SP2013.adventure.com
Before You Begin
Installation Type
Server Selection
Server Roles
Features
Confirmation
Results
To install the following roles, role services, or features on selected server, click Install.
Restart the destination server automatically if required
Optional features (such as administration tools) might be displayed on this page because they have been selected automatically. If you do not want to install these optional features, click Previous to clear their check boxes.
Windows PowerShell
Windows PowerShell Web Access
Export configuration settings
Specify an alternate source path
< Previous
Next >
Install
Cancel


Add Roles and Features Wizard
Installation progress
DESTINATION SERVER
SP2013.adventure.com
Before You Begin
Installation Type
Server Selection
Server Roles
Features
Confirmation
Results
View installation progress
Feature installation
Installation started on SP2013.adventure.com
Windows PowerShell
Windows PowerShell Web Access
You can close this wizard without interrupting running tasks. View task progress or open this page again by clicking Notifications in the command bar, and then Task Details.
Export configuration settings
< Previous
Next >
Close
Cancel



Start Windows PowerShell as Administrator and run the following script
Install-PswaWebApplication –UseTestCertificate

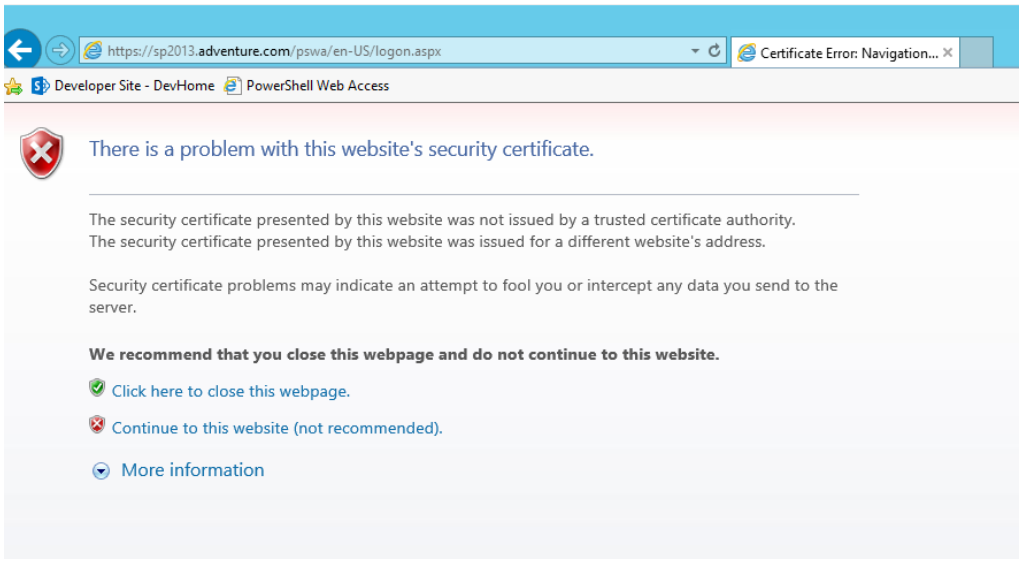
```
Windows PowerShell
Copyright (C) 2012 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> Install-PswaWebApplication -UseTestCertificate
WARNING: Using a test certificate in a production environment is not recommended for security reasons. This certificate
 should be used only for internal testing of Windows PowerShell Web Access. The test certificate expires in 90 days.
Creating application pool pswa_pool...

Name                    State        Applications
----                    -----        ------------
pswa_pool               Started


Creating web application pswa...

Path            : /pswa
ApplicationPool : pswa_pool
EnabledProtocols : http
PhysicalPath    : C:\Windows\Web\PowerShellWebAccess\wwwroot


Creating self-signed certificate...


Creating HTTPS binding...
```

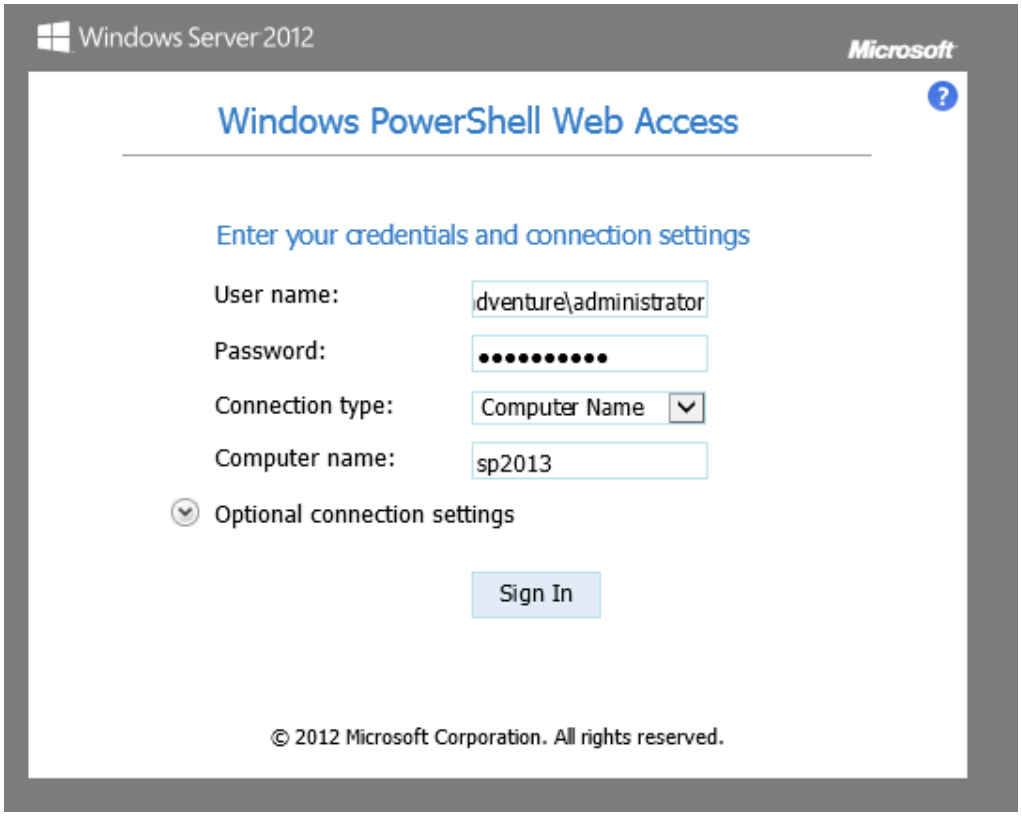
Add-PswaAuthorizationRule -UserName adventure\administrator -ComputerName sp2013 -ConfigurationName adminsonly

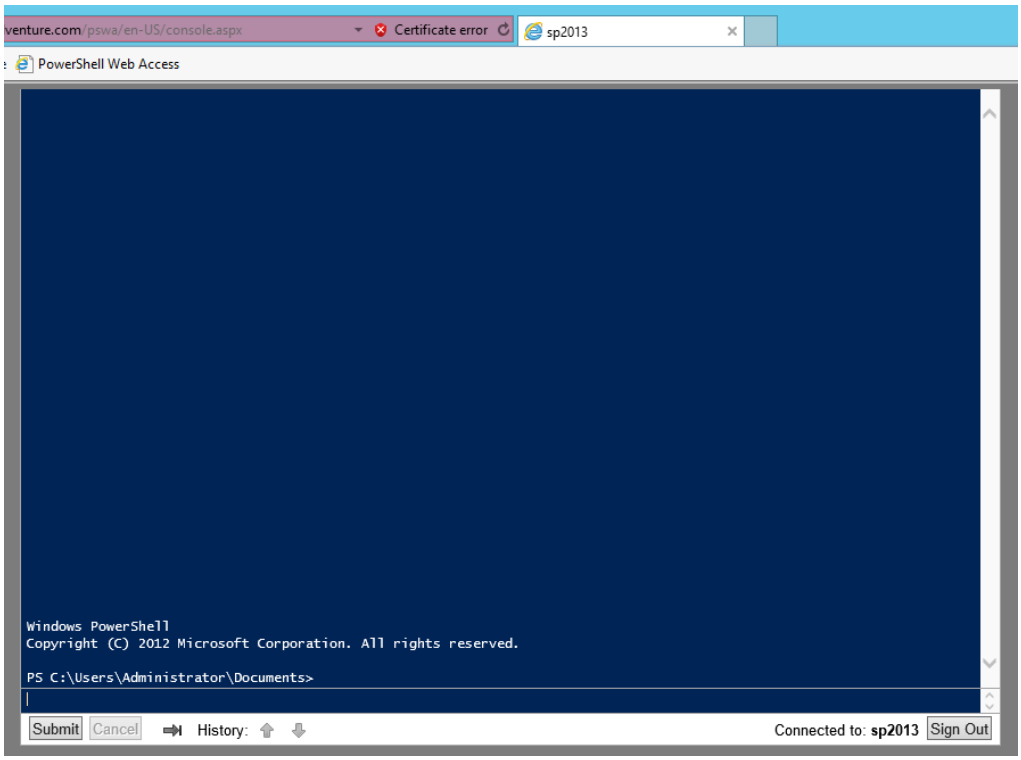
To Allow Everyone (Test Lab)

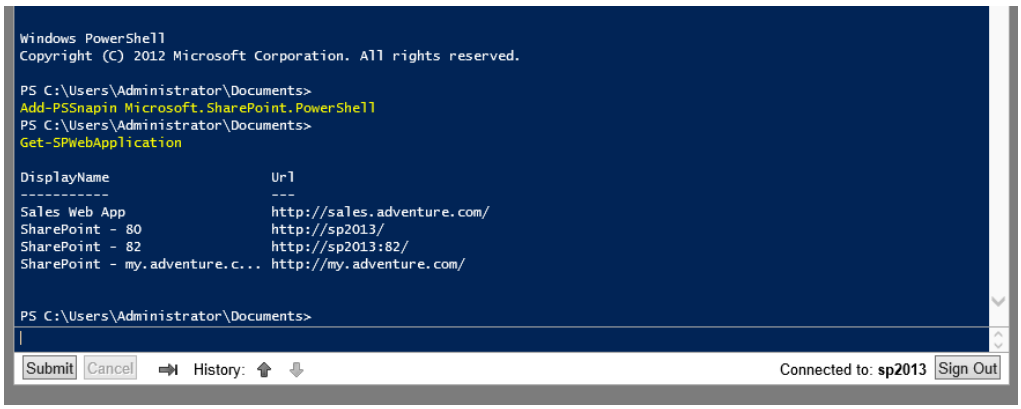
Add-PswaAuthorizationRule –UserName * -ComputerName * -ConfigurationName *

```
PS C:\Windows\system32> Add-PswaAuthorizationRule -UserName adventure\administrator -ComputerName sp2013 -ConfigurationN
ame adminsonly

Confirm
You are creating a rule for computer "SP2013.adventure.com". If this is not the computer for which you want to create a
 rule, then try again with a fully-qualified domain name.

Do you want to continue?
[Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"): y

Id     RuleName         User                        Destination                 ConfigurationName
--     --------         ----                        -----------                 -----------------
0      Rule 0           adventure\administrator     SP2013.adventure.com        adminsonly
```


https://sp2013.adventure.com/pswa/en-US/logon.aspx
Certificate Error: Navigation...
Developer Site - DevHome
PowerShell Web Access
There is a problem with this website's security certificate.
The security certificate presented by this website was not issued by a trusted certificate authority.
The security certificate presented by this website was issued for a different website's address.
Security certificate problems may indicate an attempt to fool you or intercept any data you send to the server.
We recommend that you close this webpage and do not continue to this website.
Click here to close this webpage.
Continue to this website (not recommended).
More information


Windows Server 2012
Microsoft
Windows PowerShell Web Access
Enter your credentials and connection settings
User name:
dventure\administrator
Password:
Connection type:
Computer Name
Computer name:
sp2013
Optional connection settings
Sign In
© 2012 Microsoft Corporation. All rights reserved.


venture.com/pswa/en-US/console.aspx
Certificate error
sp2013
PowerShell Web Access
Windows PowerShell
Copyright (C) 2012 Microsoft Corporation. All rights reserved.
PS C:\Users\Administrator\Documents>
Submit
Cancel
History:
Connected to: sp2013
Sign Out


Windows PowerShell
Copyright (C) 2012 Microsoft Corporation. All rights reserved.
PS C:\Users\Administrator\Documents>
Add-PSSnapin Microsoft.SharePoint.PowerShell
PS C:\Users\Administrator\Documents>
Get-SPWebApplication
DisplayName Url
Sales Web App http://sales.adventure.com/
SharePoint - 80 http://sp2013/
SharePoint - 82 http://sp2013:82/
SharePoint - my.adventure.c... http://my.adventure.com/
PS C:\Users\Administrator\Documents>
Submit
Cancel
History:
Connected to: sp2013
Sign Out

Windows PowerShell the Power of Variables
http://technet.microsoft.com/en-us/magazine/2007.03.powershell.aspx

Introducing the Windows PowerShell ISE
http://technet.microsoft.com/en-us/library/dd315244.aspx
Index of Windows PowerShell cmdlets for SharePoint 2013

Managing Office 365 using PowerShell

http://technet.microsoft.com/en-us/library/jj151815.aspx

Overview of the upgrade process to SharePoint 2013

http://technet.microsoft.com/en-us/library/cc262483.aspx