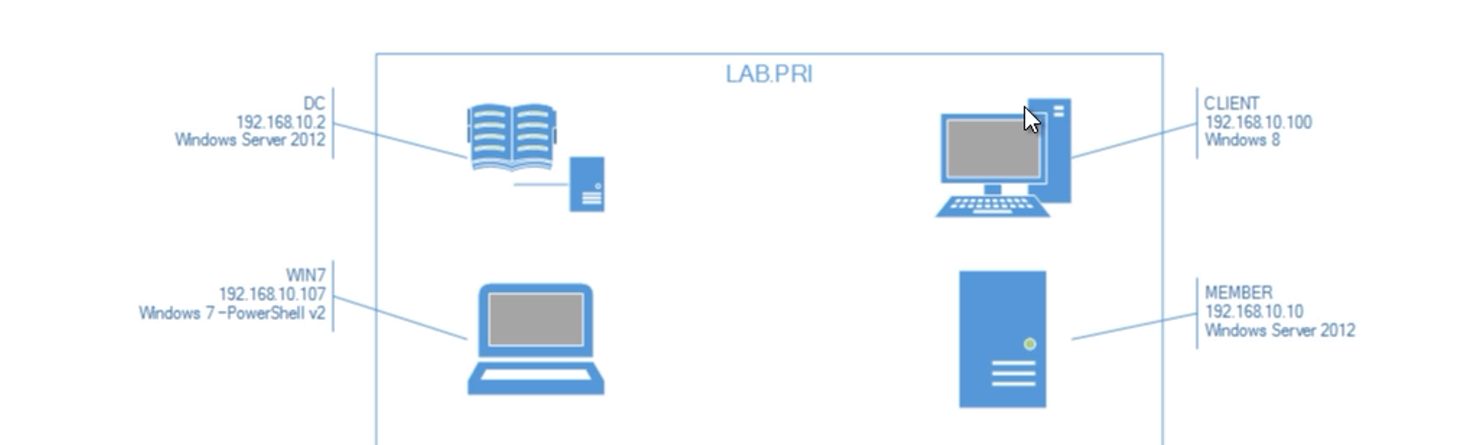
LAB



Find and discover commands

|  |  |
| --- | --- |
| Get-command | Get-command -name \*service\* -commandtype cmdlet,function |
|  | Get-command -verb get -noun \*serv\* |
|  | Get-command -verb stop |
| Module | Get-module -listavailable |
|  | Get-command -noun \*computer\*  Check module name |
|  | $env:PSModuePath |
|  | Get-command -name \*print\* |
|  | Get-command -verb get -noun net\* |
| Alias | Get-alias |
|  | Get-alias -name dir |
| Help | Get-help -name dir |
|  | Get-help get-childitem -example |
|  | Get-help get-childitem -online |
|  | Help \*service\* |
|  | Help about\*  These are powershell user manual |
|  | Help about\_escape\_characters |
|  | Help \*breaking\*  Nothing matches, it will check full string in all commands |
|  | get-help get-eventlog  explain how to read parameter and bracket meaning  get-eventlog -logname security -newest 10 |
|  | Update-help  This will update help section |
|  | Help dir -showwindow |
|  | Show-command -name get-eventlog |
| Run Command | Get-process |
|  | Get-process -name notepad |
|  | Get-process -name notepad -computername server2 |
|  | Get-process -id 5588 |
|  | Gt-help get-process -full  Position? 1 (positional parameter)  Position? Named (named parameter) |
|  | Dir -recurse  ls -recurse  Get-childItem -recurse |
|  | Get-process -name notepad  Ps notepad |
| Providers and Drives | Get-psdrive  Technologies like IIS and AD depends a lot on these providers and drives |
|  | C:  D:  Get-psdrive  Get-psprovider: it makes provider looks like drive |
|  | Get-psdrive  Get-command -noun psdrive  Get/new/remove |
|  | New-psdrive -name UTIL -PSProvier filesystem -Root C:\temp\UtilityFolder  I can map network drive as well  Only appear in currently PS session |
|  | Get-command -noun item |
|  | Get-itemproperty -path .\test.txt  Get-itemproperty -path .\test.txt | select -property \*  Set-itemproperty -path .\test.txt -name IsReadOnly -value $true |
|  | Cd hkcu:  Cd software\software\microsoft\windows\currentversion  Ls  Get-itemproperty .\explorer |
|  | Import-module ActiveDirectory  Get-psprovider  Cd AD: |
| Variable | Variable, String, Hashtables, Core operatoros  Double quotes and single quote difference  You can give \_ in variable name  $b = “`n`n`n`my new line”  $b = ‘`n`n`n`my new line’ |
|  | Arrary  $s = get-service  $s[0]  “$($s[0].name)” |
|  | “this  Is a  tring”  @“this  Is a  Tring  ”@ |
|  | [int]$repeat = Read-host “How many times loop should run?”  [string] / [float] / [bool] |
|  | Hashtable:  @{  ‘key’ = ‘value’;  ‘key’ = ‘value’;  } |
| Regula Expression | “abcd12312dfdsfds” -match “\d+” |
|  | “teaching.method@mydomain.com” -match “([a-z]+)\.[a-z][+@[a-z]+\.com](mailto:+@[a-z]+\.com)”  $matches |
|  | Help About\_regular\_expression |
| Replace | “this is a test line” -replace “test” “success” |
| Pattern | Get-childitem -recurse -include \*.txt | select-string -pattern “error” |
| External command | Start-process -file “ping.exe” -argument “127.0.0.1” |
|  | icacls c:\logs\\* /grant “Administrator:(D,WDAC)”  or  icacls --% c:\logs\\* /grant Administrator:(D,WDAC) |
| Pipeline | Get-process | stop-process |
|  | Ger-process | export-clixml proc.xml  Notpad  Calc  Mspaint  Compare-object | -referenceObject (inmport-clixml proc.xml) -diff $(get-process) -property Name |  Convertto-HTML | out-file report.html |
| Objects | Get-service  Earch row is object  Each col is property |
|  | Create a variable and do get-member  Get-service | get-member |
|  | $d = get-date  $d | gm  $d.hour  $d.minute  $d.toshortdatestring() |
|  | $proc = Get-process  $proc | gm  $proc[0].kill() |
| Core Commands | Sorting, selecting, deasuring, grouping  Sort-object  Select -expandproperty / -property  Get-process | sort-object -property ws | select-object -first 5 |
|  | Get-service | group-object -property status |
|  | Get-process | measure-object -property ws -max -min -ave -sum |
|  | Get-process | Select-object -property name, id, @{  Name=’VM(MB)’;expression={**$\_**.vm / 1GB -as [int]};  }  Get-process | Select-object -property name, id, @{  Name=’VM(MB)’;expression={**$PSITEM**.vm / 1GB -as [int]};  } |
|  | Get-adcomputer -filter \* |  select @{  Label=’computername’;  Expression={$\_.name}  } |  Get-service -name \* |
|  | Get-eventlog -logname security -newest 10  -computername (get-adcomputer -filter \* | select -expandproperty name) |
| Formatting | Get-process | format-wide  Format-list  Format-table -autosize -wrap  Get-service | Format-table -groupby status | select -property name, id, status  Get-process |  Format-table -property Name, ID,  @{  Name=’vm(mb)’;  Expression={$\_.vm / 1MB};  Formatstring=’N2’;  Align=’right’  },  pm |
|  | Help Format-table -examples |
| Comparison operator | eq, lt, gt, le, ge,  ne, ceq, like, not like, clike |
|  | Get-service | where -filterscript {  ($\_.status -eq “Running”) -or  ($\_.name -like \*win\*)  } |
|  | -as [string]  - is [string]  “powershell” -like “\*shell\*”  “powershell” -contains “\*shell\*”  $x = 1,2,3,5,”seven”,”eight”  $x += “nine”,”ten”  $x -contains “seven”  -contains = -in  5 -in $x |
|  | $arr = 1,2,3,4,5  $arr -join ‘,’  $arr -split “,”  $arr[-1] |
| Format | "{0} {1:N2} {3} {2:n4}" -f "hello",4.5626444,6.4564564,"three"  “{0:N2}” -f “5.6525” |
| Foreach | Notepad;notepad;notepad  Get-process -name notepad  Get-process -name notepad | foreach-object -process {$\_.kill()}  Get-process -name notepad | % {$\_.kill()}  ps notepad | % {$\_.kill()} |
| Measure | Measure-command -expression { notepad; notepad; get-process -name notepad | foreach { $\_.kill() } }  Measure-command -expression { notepad; notepad; stop-process -name notepad } |
|  | notepad;notepad  get-process -name notepad | foreach-object id  get-process -name notepad | select-object id  get-process -name notepad | foreach-object { $PSITEM.name.toupper()} |
|  | Get-windowsfeature  Get-windowsfeature | where-object -filterscript {$\_.installed -eq $true} |
|  | Install-windowsfeature -name telnet-client  UnInstall-windowsfeature -name telnet-client |
|  |  |
| **Best practice**  Baseline using compare-object | Get-windowsfeature | export-clixml baseline.xml  compare-object -ref (import-clixm baseline.xml)  -diff (get-windowsfeature | where {$\_.installed}  )  -property name | select -expandproperty name  Uninstall-windowsfeature -computername dc -name *(*  *compare-object -ref (import-clixm baseline.xml)*  *-diff (get-windowsfeature | where {$\_.installed}*  *)*  *-property name | select -expandproperty name*  *)* |
|  | Get-hotfix -computername dc  Get-hotfix -computername (get-adcomputer -filter \* | select -expand name) |
|  | Get-history |
| Powershell Remoting | WS-MAN protocol  WinRM service  HTTP(s)  Ports: 5985 & 5986  Trust machines in network:  Trustedhost \* |
|  | Enable-psremoting  Enable-PSRemoting -SkipNetworkProfileCheck |
|  | Group policy |
|  | Start WinRM service via policy |
|  | Allow port |
|  | Get-command -noun \*wsman\*  Cd wsman:  Connect-wsman -computername dc  Cd dc  Wsman:dc\>cd service  Ls  Cd defaultports  Ls  Cd dc\shell  Cd .\listener #check host and port  Cd .\client #check trustedhost |
|  | Enter-pssession -computername member.lab.pri |
|  | Invoke-command -scriptblock {  Get-eventlog -logname security -newest 10  }  -computername dc, member |
|  | Get-help invoke-command  throttleLimit  UseSSL |
| Second hop | Enable-WSManCredSSP  Help enable-wsmancredssp  Computer Configuration  Administrative Templates  System  Credential delegation    Windows components  Windows Remote Management (winRM)  WinRM client & WinRM Service    Now while using enter-pssession cmdlet use below options  -authentication  -useSSL |
|  | Default security  Get-PSSessionConcifiguration | ft -wrap  Get-PSSessionConcifiguration | select permission  Set-PSSEssionConfiguration -SHowSecurityDescriptionUI -name Microsoft.powershell |
| Possession | Enter-pssession -comptername dc  Wsmprovhost process would be running – web services management for provider host |
|  | Ad-hoc session  Connect and disconnection after use  Invoke-command -computername dc -scriptblock {get-process}  Get-process -computername dc |
|  |  |
|  | $dc = new-pssession -computername dc  $servers = get-pssession -computername dc, member  Get-pssession  Enter-psesssion -session $dc  Invoke-command -session $servers -scriptblock (dir c:\users)  Get-pssession | remove-pssession  *No need to do it manually. Close ps windows, it will close all sessions remotely.* |
|  | **Implicit remoting** which continues to run over a long period of time  $dc = new-pssession -computername dc  $rdc = new-pssession -computername rdc -credential administrator  Get-pssession  Invoke-command -session $dc, $rdc -scriptblock {import-module ActiveDirectory}  Import-pssession -session $dc -prefix lab -module ActiveDirectory  *It is not installing module. Just creating a shortcut of module on local computer with a prefix “lab”*  Get-command -noun lab\*  Get-labADUser -filter \*  **Is equal to**  Invoke-command -computername dc -scriptblock { get-aduser -filter \* }  Get-pssession  Get-pssession | remove-pssession  In many cases it will not support pipeline command.  **This will fail**  get-labaduser -identity Pradeep | set-labaduser -department developer -city “India”  **this will work on dc computer**  enter-pssession -session $dc  get-aduser -identity Pradeep | set-aduser -department developer -city “India”  get-module -session $dc -Listavailable  import-module -name netadapter -pssession $dc -prefix DC  set-execution policy -executionpolicy RemoteSigned  try again if it failed in previous run  get-module  import-module |
|  | Remoting happens on HTTP  All native commands doesn’t support -computername parameter  **It won’t work**  Get-process -computername dc -name dns | stop-process  **It will work**  Invoke-command -computername dc -scriptblock{  Get-process -name dns | stop-process  } |
| Invoke command with variable | $logname = “security”  $newest = 10  Invoke-command -computername dc, member -scriptblock{  Param($var1, $var2)  Get-eventlog -logname $var1 -newest $var2  } -argumentlist $logname, $newest  **We can also use**  Get-eventlog -logname $using:logname -newest $using:newest |
| Trustedhost setting | Trustedhost contains machins current machine should trust in network  Start (\*) means trust all computers  Trust specific d omain means: \*.lab.local  To do so  Set-item -path .\TrstedHosts -value “\*.lab.local” |
|  | **New-adgroup** -name helpdesk -scope global -samaccountname helpdesk  **New-aduser** -name user1 -samaccountname user1 -passwordneverexpires $true -accountpassword (  Convertto-securestring -asplaintext “P@ssword” -force  ) -department help -title technician -city Pune  **Enable-adacount** user1  **Add-adgroupmember** -identity helpdesk -members user1  Get-adgroup helpdesk |
| Ignore this topic | **New-PSSessionConfiguration**  -path .pssc file extension  - modulesToImport  - visibleCmdLets  -visibleFunctions  AD module to visible  Enable-ADAccount  Disable-ADAccount  New-PSSessionConfigurationFile -Pass c:\helpdesk.pssc -modulesToImport Activedirectory  -VisibleCmdlets ‘Enable-ADAccont’,’Disable-ADAccount’  Get-content helpdesk.pssc  **Register endpoint**  Register-PSSessionConfigration -name test -showsecuritydescriptorUI  Y  Y  Y  Add helpdesk group, give read, write execute permission  Get-PSSessionConfiguration -Name test | fl \*  *Copy security descriptor of test user*  UnRegister-pssessionconfiguration -name test  Register-PSSessionConfigration -name test |
| Powershell web remoting | **Add-WindowsFeature** WIndowsPowerShellWebAccess  Get-module -listavailable  Get-command -module powershellwebaccess |
|  | **Install-PSwebApplication -useTestCertificate**  [http**s**://Member.lab.pri/pswa](https://Member.lab.pri/pswa)  username: pradeep  password: lab@123  computername: dc.lab.pri  use ssl: no  This will not allow to login by default |
|  | **Enter-pssession -computer member.lab.pri**  **Add-pswaAuthorizationRule**  -**ruleName** ‘Default configuration for admins to computer’  -**ConfigurationName** ‘microsoft.powershell’  -**usergroupname** ‘lab\domain admins’  -**computername** ‘dc.lab.pri’  You are on win7, you connecting to member.lab.pri, authenticated by dc.lab.pri.  It is 2nd hop authentication. For this we need CredSSP Delegation |
|  | On win7  Enable-WSmanCredSSP -role client -delegateComputer \*.lab.pri  Cd wsman:\localhost\client\auth  Ls |
|  | Enter-Pssession -computername member.lab.pri  Enable-WSManCredSSP -role server  exit |
|  | On win7  Enter-pssession computername member.lab.pri -authentication credssp -credential lab\administrator |
|  | Now run this on member.lab.pri  **Add-pswaAuthorizationRule**  -**ruleName** ‘Default configuration for admins to computer’  -**ConfigurationName** ‘microsoft.powershell’  -**usergroupname** ‘lab\domain admins’  -**computername** ‘dc.lab.pri’ |
|  | Open pswa portal again   |  |  | | --- | --- | |  |  |   Enter-pssession -computername member.lab.pri  Get-webbinding -protocol htts | select \*  Dir cert: -recurse |
| WMI and CIM | Windows management instrumentation  Common Information Model |
|  | Open mmc and connect **WMI Control** on local computer |
|  | Namespace is a container |
|  | [WMI Explorer from The PowerShell Guy • The Lonely Administrator (jdhitsolutions.com)](https://jdhitsolutions.com/blog/powershell/2848/wmi-explorer-from-the-powershell-guy/) |
|  | Get-wmiobject -listavailabe |
|  | **WMI**  It uses RPC protocol to connect remote server which is part of DCOM model (distributed component object model). It is very old model.  This solution was created before DMTF (Distributed management task force) finalized standards around Common Information Model.  RPC is Microsoft proprietary protocol  It uses port 139. |
|  | CIM  It communicates of WS-MAN (Win RM Service)  Both (CIM & WMI) access same repository.  CIM is also capable of communicating with old DCOM model just for backward combability.  Microsoft is no longer investing in old stack, that is WMI.  Command still exist but no upgrade will be there.  CIM is the way forward. |
| WMI | Get-wmiObject -class Win32\_service -first 1 -compuername localhost| select -property \*  It gives more information  Focus  \_\_Path  \_\_Server |
|  | Gwmi win32\_bios  Gwmi win32\_bios | fl \*  Gwmi win32\_bios -computername dc  Gwmi win32\_bios -computername dc, member  “dc”,”abc”,”member” | out-file c:\temp\computer.txt  Get-wmiobject -class win32\_bios -computername (get-content .\computer.txt)  Computer names are picked sequentially  Default output doesnot display computername  Get-wmiobject -class win32\_bios -computername (get-content .\computer.txt) |  Format-table -property \_\_server, serialNumber, version -autosize  \_\_Server = PSComputername |
|  | Get-wmiobject -computername (get-content .\computer.txt) -class win32\_OperatingSystem -credential lab\administrator  *Credential can’t be used to connect local repository* |
|  | Help Get-wmiobject -examples |
|  | Get-wmiobject -class win32\_logicaldisk  Search win32\_logicaldisk on google and check disk info |
|  | $os = get-wmiobject -class win32\_operatingsystem  $os.lastbootuptime  Wmi by default comes with some useful script methods  $os.ConvertToDateTIme($os.lastbootuptime)  $os.ConvertToDateTIme($os.lastbootuptime) | gm  $os.ConvertToDateTIme($os.lastbootuptime).toShortDateString() |
| CIM  Common Information Model | Get-command -noun cim\*  Get-cimclass  Get-ciminstance |
|  | Get-cimclass: it contains all namespace. It doesn’t necessarily mean we have physical component  Get-CimClass -class Win32\_Bios -Namespace root\cimv2 | fl \*  It is exptracting info from CIM\_BIOSElement super class |
|  | Get-cimInstance -className win32\_service |
|  | Compare in 2 different windows  Get-cimInstance -className win32\_service | gm  Get-WmiObject -class win32\_service | gm |
|  | If ciminstace fails on remote server then enable psremoting on that server  Enable-PSRemoting  And enable wmi on firewall |
| Filter | Get-wmiobject -class win32\_service | where {$\_.state -ne ‘Running’ -and $\_.startmode -eq ‘auto’}  Better way using filter  There is a trick  Help gwmi -full  It doesn’t support powershell comparison operator  Get-wmiobject -class win32\_service -filter “DeviceID = ‘c:’”  Gt >  Lt <  Equal =  Not equal <>  LIKE  In WMI string always goes in single quotes ‘ ‘  %% is wildcard not \*\* |
|  | Get-wmiobject -class win32\_service -coputername dc **-filter** “ state <> ’running’ and startmode = ‘auto’ and name like ‘Remote**%**’ ” |
| WML query | $wml = “select \* from win32\_logicaldisk where drivetype = 3”  Get-wmiobject -query $wml -computername dc |
|  | Get-ciminstance -classname win32\_service -filter "startmode = 'auto' and name like 'wlan%' and state = 'running'"  Get-ciminstance -classname win32\_service -filter "startmode = 'auto' and name like 'wlan%' and state <> 'running'" |
| Depended services | $s = get-ciminstance -classname win32\_service -filter "name='bits'"  Get-CimAssociatedInstance -Association win32\_dependentservice -InputObject $s |
| Cim session | $dc = new-cimsession -comutername dc  Get-ciminstance -classname win32\_NTEventlogFile -Cimsession $dc  Backup  Get-wmiobject -class win32\_NTEventLogFile -filter “LogFileName = ‘application’ ” | gm  Look for method backupEventLog  **This will not clear event. Just take backup**  Get-wmiobject -class win32\_NTEventLogFile -filter “LogFileName = ‘application’ ” | invoke-wmiMethod -name BackupEventLog -ArgumentList c:\temp\backup.evt  **This will clear event and take backup**  Get-wmiobject -class win32\_NTEventLogFile -EnableAllPrivileges -filter “LogFileName = ‘application’ ” | invoke-wmiMethod -name ClearEventLog -ArgumentList c:\temp\backup2.evt |
| Background job | **Local**  Start-job -scriptblock {dir c:\ -recurse}  Start-job -scriptblock {dir c:\ -recurse} -name myjobname  Get-job  Get-job -id 1  Stop-job -id 1  Once you receive data from job, PS doesnot keeps data in cache  Receive-job -id 1  Receive-job -id 1 -keep  Get-job | stop-job |
|  | **Remote Job**  Invoke-command -comuername dc -scriptblock {get-eventlog -logname security -newest 10} -asJob -jobname myEventLog |
|  | **WMI-Jobs**  Get-wmiobject -class win32\_logicaldisk -computername dc, member -asjob |
|  | Get-job -id 5 | select -expandProperty childjobs |
|  | Get-command -noun job |
|  |  |
| Scheduled Job | Get-command -module psscheduledjob |
|  | $trigger = new-jobTrigger -atlogon |
|  | $option = new-scheduledJobOption -requieNetwork -waketorun |
|  | Register-scheudledjob -name “logon action” -scriptblock {get-process} -maxresultcount 2 -trigger $trigger -scheduledJobOption $option  Get-scheduledjob  Get-scheduledjob | fl \*  Open task scheduler  Check active tasks, job name should be listed there  Output is saved here |
| Powershell scripting | Run a sample powershell script.  Trust a script?  Trusted root certificate contains list of certs which local computer trusts.  Use this CmdLet  Set-AuthenticodeSignature  Show one test signature specimen,  C:\windows\system32\_WIndoesPowerShell\v1.0\dotnetTypes.format.ps1xml  Get-AuthenticodeSignature .\DotNetTyypes.format.ps1xml |
|  | **Execution policy**  Get-ExecutionPolicy  Set-ExecutionPolicy unrestricted  Remotesigned: script coming remotes must come with signature  Unrestrcted: script will run irresppecive of signature  Bypass: used by developer who has implemented some other way of security |
|  | **New session will open with specific execution policy**  Powershell.exe -executionPolicy remotesigned |
|  | **Shows output**  Get-service  **No output**  Get-service | out-file 1.txt |
|  | **it prompts for input value**  Read-host “enter your name”  $name = Read-host “enter your name” |
|  | **get-command -Verb write**  Write-Debug  Write-Error  Write-EventLog  Write-Host  Write-Information  Write-Output  Write-Progress  Write-SqlTableData  Write-Verbose  Write-Warning |
|  | **Write a small script to get disk info**  Get-Diskinfo.ps1  Ge-wmiobject -class win32\_logicaldisk -filter ‘DriveType = 3’ -computername dc |  Select-object -property @{name=’Comutername’;expression{$\_.\_\_SERVER} },  @{name=’Drive’;expression{$\_.DeviceID} },  @{name=’FreeSpace(GB)’;expression{$\_.FreeSpace / 1GB -as [int] } },  @{name=Size(GB)’;expression{$\_.Size / 1GB as [int] } }, |
|  | **Add PARAM to above script**  PARAM(  [String]$Computername,  [int]$DriveType = 3  )  .\get-diskInfo.ps1 -Computername dc -DriveType 3 |
|  | PARAM(  [String]$Computername = read-host “Computer Name”,  [int]$DriveType = 3  ) |
| If construct | $disk = Get-wmiobject -class win32\_logicaldisk  If($disk.driveType -eq 2){  Write-host “floppy”  }else If($disk.driveType -eq 3){  Write-host “fixed”  }else If($disk.driveType -eq 5){  Write-host “optical”  }else{  Write-host “invalid drive”  } |
| Switch construct | *Switch will evaluate each option*  Switch ($disk.drivetype){  2 {write-host “”}  3 {write-host “”}  5 {write-host “”}  } |
|  | $name = read-host “Enter server name”  Switch -wildcard ($name) {  “\*DC\*” {write-host “is a domain controller”}  “\*FS\*” {write-host “is a file server”}  “\*NYC\*” {write-host “is in new york”}  “\*LON\*” {write-host “is in London”}  }    You can use break to brak after condition is met:  “\*LON\*” {  write-host “is in London”  break  } |
|  | Help about\_switch |
| Looping | Foreach  For  While / until / do |
|  | **Foreach**  1..100 | foreach {notepad} |
|  | Get-process -name notepad | foreach {$\_.kill()} |
|  | $process = get-process -name notepad  Foreach($proc in $process){  $proc.kill()  } |
|  | **For**  $arr1 = “one”,”two”,”three”  For([int]$x; $x -le 3; $x++){  Write-host $x  }  $x -le 3 can be  $x -le $arr.count |
|  | **Do-while**  $existing = “server1”,”server5”,”server6”  $candidate = 0  Do{  $candidate++  $possiblename = “SERVER$candidate”  } while($existing.contains($possiblename))  $possiblename  Until (-not $existing.contains($possiblename))  Until just reverse the logic |
|  | **While**  $a = 5  $b = 7  $  While($a -eq $b){  Write-host “$b”  $b++  }  This will never execute. Condition is not met |
| Function  Pipeline |  |
|  | **Using backtick `** |
|  | **Using param** |
| Command to script to module |  |
|  | Save in module location  Save as MyTools.psm1  Import-module .\mytools.psm1 will also work |
| Scripting scope |  |
|  |  |
| **Combining data from multiple sources** |  |
| Advanced function adding h elp |  |
| Adavanced help  Parameter attribute | Mandatory / help message / positional / validation / alias |
|  | Validae set |
|  | Pipeline |
|  |  |
|  | `` |
|  |  |
|  |  |
| Parameterset |  |
|  |  |
| Custom utility |  |
|  | Export specific funtion in module |
| Error Handling | $? |
| Debug | CmdLetBindng is enabled    Add debug and verbose code    Call the funciton |
| Custom formatting views |  |
| Enable whatif and confirm parameter |  |
|  |  |
|  | That if condition producded below output |
|  | Coded added in CmdLetBinding brings below popup |
| Hierarchy |  |
|  |  |

AD Installation

Domain: Lab.pri

NetBIOS: LAB

|  |  |
| --- | --- |
| #  # Windows PowerShell script for AD DS Deployment  #  Import-Module ADDSDeployment  Install-ADDSForest `  -CreateDnsDelegation:$false `  -DatabasePath "C:\Windows\NTDS" `  -DomainMode "WinThreshold" `  -DomainName "lab.pri" `  -DomainNetbiosName "LAB" `  -ForestMode "WinThreshold" `  -InstallDns:$true `  -LogPath "C:\Windows\NTDS" `  -NoRebootOnCompletion:$false `  -SysvolPath "C:\Windows\SYSVOL" `  -Force:$true `  -SafeModeAdministratorPassword ("Lab@123" | ConvertTo-SecureString -AsPlainText -Force) |  |

AD DS Snap-Ins and Command-Line Tools RSAT-ADDS-Tools

Active Directory Administrative Center RSAT-AD-AdminCenter

Active Directory module for Windows PowerShell RSAT-AD-PowerShell

Group Policy Management GPMC

RSAT-ADDS-Tools, RSAT-AD-AdminCenter, RSAT-AD-PowerShell, GPMC, AD-Domain-Services

172.31.39.156



IIS

