Powershell :

- Notepad
- Calc
- Ipconfig /all
- Get-childitem
- Set-location C:\
- Clear-host
- cd\ > step back
- cd dir_name\dir_name
- cd ..
- cls
- dir
- ls
- clear
- get-alias
- get-alias cls > returns full command name
- get-alias dir > return full command name
- update-help –force > go out to the internet and download most recent update
- help > gives you option to scroll through by pressing space bar
- help *firewall*
- get-help
- get-help * process* list bunch of process
- get-help Get-Process
- get-help *ipaddress*
- get-help new-NetIPAddress
- get-help get-*service*
- get-help *verb*
- get-verb
- get-help *dns*
- get-help *array*
- get-help about_*
- get-process
- get-help get-process -Detailed
- get-help get-service -Detailed
- get-help get-service -example
- get-help get-service -full
- get-help get-process -ShowWindow
- get-help get-service –online
- get-help get-help
- hostname

Syntax Structure:

- Get-Fake –param Arg –param -param arg,arg
- Get-help Get-Service –Detailed > then page through till syntax

Command	parameter name	parameter value
Get-service	–name	bits
Get-service		bits
Get-service	–name	bits, bfe

- Get-service –name bits,bfe –ComputerName dc
- Get-sevice –DisplayName "app*" > required param

Alias:

- Gsv bits
- Get-alias –Definition get-service
- Ps –C dc > process from computer dc
- Get-help ps
- Get-process –ComputerName
- Get-help *snapin*
- mmc
- get-PSSnapin –Registered
- Add-PSSnapin -name *exch*

MODULE:

- Get-help *module*
- get-module
- get-module –ListAvailable
- import-Module -name ActiveDirectory
- get-command -Module activeDirectory
- get-help get-AdComputer
- get-AdComputer
- get-help get-ad*
- get-adcomputer –filter *
- get-module
- get-help *AdComputer*
- get-help *module*

Find-Module

- Install-Module PSREadline
- Or Import-module PSREadline
- Set-PSREadlineOption -EditMode Emacs

Find-Package

- Stop-service –name bits;   start-service –name bits
- Get-service -name bits | stop-service
- Get-service | stop-service –WhatIf
- Get-service | stop-service –Confirm
- Test-Connection –ComputerName Khadijah
- Get-help *file*
- get-help *out-*

Text File

- get-service | out-file –FilePath c:\service.txt
- get-childitem -Path C:\ -Filter *.txt*
- get-help *content*
- get-content –Path c:\service.txt
- notepad c:\service.txt
- get-service | out-printer
- get-eventlog -list
- get-eventlog
- get-eventlog –LogName security –EntryType error –Newest 5
- get-eventlog –LogName system –EntryType error –Newest 5
- get-eventlog –LogName system –EntryType error –Newest 5 | out-file c:\error.txt

- notepad .\error.txt

CSV

- get-help *csv*
- get-service | export-csv -path c:\service.csv
- import-csv -path c:\service.csv
- notepad c:\service.csv
- get-service | export-csv -path c:\service.csv –NoTypeInformation
- get-service | Convertto-csv –NoTypeInformation | out-file c:\test.csv

XML

- get-process | Export-Clixml   -Path c:\process.xml
- notepad process.xml
- import-Clixml c:\process.xml

- get-process | Export-Clixml c:\gold.xml
- notepad;calc;mspaint
- Compare-Object –ReferenceObject Import-Clixml c:\gold.xml
- Ctrl + c to fix that means it wont run
- Compare-Object –ReferenceObject (Import-Clixml c:\gold.xml)

- Compare-Object –ReferenceObject (Import-Clixml c:\gold.xml) -DifferenceObject (get-process) -Property Processname

WebPage:

- Get-eventlog   -LogName system –Newest 5 –EntryType error
- Get-eventlog   -LogName system –Newest 3 –EntryType error   –ComputerName
- Get-eventlog   -LogName system –Newest 3 –EntryType error | ConvertTo-html | out-file c:\error.html
- .\error.html

- Get-eventlog   -LogName system –Newest 3 –EntryType error | ConvertTo-html   -Title "Windows Errors"   -Body (Get-Date) -PreContent   "<p> Generated by IT <p>"
- -PostContent   "For more details check the full server log " | out-file c:\error.html

Working with OBJECT:

- Get-service | Get-member
- Get-process | Get-member
- Get-service –name bits
- Get-service –name bits | Select-Object –Property Name, Status, MachineName
- Get-help *format*
- Get-service –name bits | Select-Object –Property Name, Status, MachineName
  | format-table –AutoSize
- Get-service –name bits –Computername Khadija, s1, client | Select-Object –Property Name, Status, MachineName | format-table –AutoSize
- Get-service –name bits –Computername Khadija, s1, client | Select-Object –Property Name, Status, MachineName | format-table –AutoSize | out-file c:\services.tx
- Get-process | select-object -Property name, cpu

Sorting:

- Get-service | select-object -Property name,status | Sort-Object –Property status -Descending
- Get-Childitem -Path c:\ | sort-object –Property length –Descending
- Get-Childitem -Path c:\ | sort-object –Property length –Descending | get-member
- Get-Childitem -Path c:\ | sort-object –Property length –Descending | select name, length
- Get-service –name bits | select-object -Property name | sort-object -Property Status
- Get-service | select-object -Property name | Get-member
- Get-service | sort-object -property status | select-object –Property name

Customization:

- Get-service | select-object -Property naem, stitus
- Get-service | select-object -Property naem, stitus | get-member
- Get-service -name bits | select-object -Property name, @{name="ServiceName" ; expression={"hello"}}
- Get-service -name bits | select-object -Property name, @{name="ServiceName" ; expression={$_.name}}
- Get-service -name bits | select-object -Property @{name="ServiceName" ; expression={$_.name}}, status
- Get-service -name bits | select-object -Property @{n="ServiceName" ; e={$_.name}}, status
- Get-service -name bits | select-object -Property @{label="ServiceName" ; e={$_.name}}, status
- Get-WmiOjbect -Class win32_logicaldisk -filter "DeviceID= 'C:' "
- Get-WmiOjbect -Class win32_logicaldisk -filter "DeviceID= 'C:' " | select-object -property DeviceID, Freespace
- Get-WmiOjbect -Class win32_logicaldisk -filter "DeviceID= 'C:' " | select-object -property DeviceID, @{n="FreeGB"; e={$_.Freespace / 1gb}}
- Get-WmiOjbect -Class win32_logicaldisk -filter "DeviceID= 'C:' " | select-object -property DeviceID, @{n="FreeGB"; e={$_.Freespace / 1gb –as [int] }}

Filtering:

- Get-service | Where-Ojbect -FilterScript { } ctlr + c to break out
- Get-help *Operator*
- 4 –gt 3
- 4 –lt 3
- 4 –ne 3
- 4 –le 3
- "hello" –eq "HELLO" true
- "hello" –ceq "HELLO" false [case sensitive]
- Get-service | Where-Object {$_.status -eq "Running" }
- Get-service | Where-Object {$_.status -eq "Running" -and $_.name –like "b*"}
- Get-Wmiobject -Filter
- Get-Service –name b*
- Get-Service –name b* -ComputerName dc,s1,a2,f2| Where_Object {$_.name="Stopped"}

Method:

- Get-service –name bits | ForEach-Object {$_.start()}
- Get-service –name bits | ForEach-Object {$_.stop()}
- Get-service –name bits | ForEach-Object {$_.status}

Automation Security:

- Get-ExecutionPolicy
- Set-ExecutionPolicy remotesigned
- Set-ExecutionPolicy unrestricted

Get-eventlog -Logname system -Newest 5 -EntryType error -ComputerName dc1,dc2,dc4 | select-object -propert index, source , message | convertTo-HTML | out-file c:\error.html

How to create script:

Open Notepad
Copy below command paste it on notepad
Get-eventlog -Logname system -Newest 5 -EntryType error -ComputerName dc1,dc2,dc4 | select-object -propert index, source , message | convertTo-HTML | out-file c:\error.html
and save it as error.ps1

How to run script:

C:\error.ps1
Or to run from inside folder
.\error.ps1
Or open with: notepad .\error.ps1

Variable:

- Get-help *variable*
- $var = "Hello"
- Write-output $var or type $var hit enter
- $var=get-service –name bits
- $var
- $var | get-member
- $var.status > stopped
- $var.start()
- $var.status > stopped
- $var.refresh()
- $var.status > Running
- $var= 1,2,3,4,5
- $var = 1234
- $var[3] > 4
- $var[0] > first index
- $var[-1] > last index

Open Integrated Script Environment

- Ise hit enter. Hit "CTRL + R" to toggle between screen.

- Get-CinInstance -ClassName Win32-logicaldisk -filter " DeviceID='C:' " -ComputerName DC | Select-Object PSComputerName, FreeSpace
- Get-CinInstance -ComputerName DC -ClassName Win32_logicaldisk -filter " DeviceID='C:' " | Select-Object –Property @{n="ComputerName"; e={$_.PSComputername}}, @{name="FreeGB";e=$_.Freespace / 1gb -as [int]}}
- Copy command from step 3. Paste it on ISE editor and save it as Diskinfor.ps1
- $computername = 'DC'
- Get-CinInstance -ComputerName $computername -ClassName Win32_logicaldisk -filter " DeviceID='C:' " | Select-Object –Property @{n="ComputerName"; e={$_.PSComputername}}, @{name="FreeGB";e=$_.Freespace / 1gb -as [int]}}
-

Parameterized Script :

-
  ```
  Param(
        $computername='DC'
  )
  ```
- Get-CinInstance -ComputerName $computername -ClassName Win32_logicaldisk -filter " DeviceID='C:' " | Select-Object –Property @{n="ComputerName"; e={$_.PSComputername}}, @{name="FreeGB";e=$_.Freespace / 1gb -as [int]}}
- ./Diskinfo.ps1 –computername dc
- ./Diskinfo.ps1 -computername s1

-
  ```
  Param(
        $computername='DC'
        $NotForUse
  )

  Get-help .\Diskinfo.ps1
  ```

Mulitple Parameter:

-
  ```
  Param(
        [string [] ]$computername='DC'
        $NotForUse
  )
  ```
- .\Diskinfo.ps1 –computername dc, s1

-
  ```
  Param(
        [Parameter (Mandatory=$true)]
        [string [] ] $computername = 'DC',
        $NotForUse
  )
  ```

Block Comment:

NB: Build a help file make sure leave no space:

```
<#      .Synopsis
        This is brief comments
        .Description
        This is the long comments
        .Parameter ComputerName
        This is the name of a remote computer
        .Example
```

```
                Connecting to remote computer
                Diskinfo  –computername  DC
                .Example
                Connecting to local computer
                Diskinfor  –computername  localhost
#>
Param(
    [Parameter (Mandatory=$true)]
    [string [] ] $computername = 'DC',
    $NotForUse
)
```

Single Comment:

```
# Main code here
        Get-CinInstance  -ComputerName  $computername    -ClassName  Win32_logicaldisk  -filter  "
DeviceID='C:' "   | Select-Object –Property  @{n="ComputerName"; e={$_.PSComputername}},
@{name="FreeGB";e=$_.Freespace / 1gb  -as [int]}}
```

Remoting:

- Servermanager
- Get-service -Computername dc, s2, s3 -name bits

Enable PowerShell Remoting:

- Enable-PSRemoting
- Enable-PSRemoting –force

Establish session single computer:

- Enter-PSSession -ComputerName dc
- Hostname > dc
- Ipconfig > dc ip info
- Start-service -name bits

Establish session multiple computer

- Invoke-Command -ComputerName dc, s3, s4 {Get-servcie –name bits }
- Invoke-Command -ComputerName dc, s3, s4 {Get-servcie –name bits } | out-file c:\info.txt
- Invoke-Command -ComputerName dc, s3, s4 {format : c } very dangerous
- Invoke-Command -ComputerName dc, s3, s4 {$var=2 }
- Invoke-Command -ComputerName dc, s3, s4 {write-output $var }
- $sessions = New-PSSession -Computername s3, s4
- Invoke-Command –Session $sessions {$var=2}
- Invoke-Command –Session $sessions {write-output $var } > 2
- Get-PSSession
- Disconnect session if   close out PowerShell window.

Start iexplore http://www.bing.com
Enter-PSSession
Get-WindowsFeature

Installing webser to the below computer:

- **$servers = 's3', 's4'**
- $sessions = New-PSSession -ComputerName $servers
- Invloke-Command -Session $sessions {install-windowsfeature web-servers}

Test:

- $servers | ForEach-Object {start iexplore http://$_}

Deploy: Webserver and Website

- $servers | ForEach-Object { Copy-Item .\default.html -Destination \\$_\C$\inetpub\wwwroot}
- $servers | ForEach-Object {start iexplore http://$_}

Running PowerShell form Win Server 2012 R2 without GUI

- Powershell
- Get-WindowsFeature *gui*
- Get-WindowsFeature *gui* | Install-WindowsFeature

Remoting Management Tools:

- **Get-ADComputer –filter ***
- **Get-module -ListAvaiable**
- **$adsession = New-PSSession -ComputerName dc**
- **Import-PSSession –Session $adSession -Module ActiveDirectory**
- **Get-help *ad***
- **Get-help get-ADComputer**
- **Get-ADComputer –filter ***

Profile:

- **$profile <load everytime powershell start>**

Create profile:

- **New-item $profile –ItemType file –force**

Open profile with ISE

- **Isc $profile**
- **$adSession=New-PSSession -ComputerName dc**
- **Import-PSSession -Session $adSession –Module Activedirectory**
- **Then save it**

Example 1: Create an SMB share

S C:\>New-SmbShare -Name "VMSFiles" -Path "C:\ClusterStorage\Volume1\VMFiles" -FullAccess "Contoso\Administrator", "Contoso\Contoso-HV1$"

Example 2: Create an encrypted SMB share

PS C:\>New-SmbShare -Name "Data" -Path "J:\Data" -EncryptData $True

Create local user:

New-LocalUser -Name "abu" -Password $Password -FullName "ST-W10P\abu" -Description "Administrator for ST-W10P Machine"

Network share enable:
Network Discovery:
netsh advfirewall firewall set rule group=”network discovery” new enable=yes
File and Printer Sharing:
netsh firewall set service type=fileandprint mode=enable profile=all
Service start/stop:
Start-Service <service name>
To get a list of the service names, run
Get-Service

Net SHARE share=d:\share /GRANT:EVERYONE`,FULL /REMARK:"

Get-Verb
(Get-Verb).count
help push
help pushd
powershell support standard window shell command
ipconfig cd \ ipconfig
hostname > direct.txt
dir >> direct.txt
type direct.txt
cat direct.txt
open notepad type hostname save it as test1.ps1
then run on powershell .\test1.ps1

```
$name = 'Abu'
$number = 42
$nlist = 1,2,3,5,4,8,10
($nlist).count
echo "hello world"
write-host "hellow world"
echo "The ist is  $nlist "
```

If else condition:

```
if($nlist[1] -gt 0)
{
  echo "Positive"
}
```

do while loop:

```
$i = 1
do {
        $val = $nlist | Select-Object -Index $i
        echo"Value is $val"
        $i = $i + 1
```

```
  }while($i -le ($nlist).count)



ForEach($val in $nlist){
  echo "Value is $val "
}
```

Get-NetConnectionProfile

See the network name you want to change its type and run the following command:

Set-NetConnectionProfile -Name <N/W name> -NetworkCategory Public

Online Help Document:

https://docs.microsoft.com/en-us/powershell/module/smbshare/new-smbshare?view=win10-ps

Free eBook PowerShell.org