Explain what is the significance of brackets in PowerShell? Parenthesis Brackets (): Curved parenthesis style brackets are used for compulsory arguments. Braces Brackets {} : Curly brackets are employed in blocked statements Square Brackets []: They define optional items, and they are not frequently used
What does it mean cmdlet's? Cmdlet's are simple build in commands written in .net language like C# or VB introduced by Windows PowerShell
Explain what is PowerShell Loop? Automating repetitive task with the help of PowerShell loop is known as PowerShell Loop. Through PowerShell, you can execute For each loop, While loop and Do While loop.
Explain what is PowerShell pipeline is used for? PowerShell pipeline is used for joining two statements such that the output of one statement becomes the input of the second.
Explain what is PowerShell get-command? Get-Command -> For all Cmdlet's Get-Command S* -> All Cmdlet's starting with 'S' Get-Command [A-C]* -> All cmdlet's between 'A' and 'C'
What is the use of hash table in PowerShell? It is an array that allows you to store data in a "key-value" pair association. To declare a hash table you have to use @ followed by curly braces.
Explain what is the use of Array in PowerShell? An array is a data structure that is designed to store a collection of items. The items can be the same type or different types. The use of Array in PowerShell is to run a script against remote computers. In order to create an array, you have to create a variable and assign the array. Arrays are represented by "@"symbol, they are represented as hashtable but not followed by curly braces. For example, $arrmachine = @ ("machine1" , "machine2" , "machine3")
Define the key features of PowerShell? ***The prime characteristics of Microsoft PowerShell can be summed up as follows -*** PowerShell is a scripting environment PowerShell commands are customizable The programming language isn't text-based. It's object-based.
What is the best way to find all the sql services on one server? There are two ways to do this. 1. get-wmiobject win32_service | where-object {$_.name -like "*sql*"} 2. get-service sql*

Name		Class	Output	
System Info	Get-WmiObject -Class	(win32_computersystem)	.Domain	ushustech.com
			.Manufacturer	Dell Inc.
			.Model	Vostro 460
			.Name	KWS1A045
			.TotalPhysicalMemory (KB)	8431009792
System Info	Get-WmiObject -Class	(win32_bios)	.SMBIOSBIOSVersion	A06
			.Manufacturer	Dell Inc.
			.SerialNumber (System Sl.No.)	J81HVQ1
OS Info	Get-WmiObject -Class	(win32_operatingsystem)	.BuildNumber	16299
			.SerialNumber (Product ID)	00329-10438-15609-AA631
			.Version	10.0.16299
			.OSArchitecture	64-bit
			.Caption	Microsoft Windows 10 Enterprise
			.TotalVisibleMemorySize (KB)	8431009792
			.FreePhysicalMemory (KB)	1431009792
Processor Info	Get-WmiObject -Class	(win32_processor)	.Name	Intel(R) Core(TM) i3-2100 CPU @ 3.10GHz
			.LoadPercentage	23
Hard Disk	Get-WmiObject -Class	(win32_diskdrive)	.Model	WDC WD5000AZLX-22JKKA0
			.Size (KB)	500105249280
	(Get-WmiObject -Class win32_diskdrive).Size		Output : **500105249280**	
Disk Partitions	Get-WmiObject -Class	(win32_logicaldisk)	**.DeviceID**	**C:**
			.DriveType	**3**
	Get-WmiObject -Class win32_logicaldisk |Where-Object DeviceID -Match "C:"		**.FreeSpace (KB)**	**86900658176**
			.Size (KB)	**156709679104**
	Get-WmiObject -Class win32_logicaldisk |Where-Object DeviceID -Match "C:" | Select-Object @{ Name = "FreeSpace"; Expression = {"{0:N2}" -f ((($_.FreeSpace) / ($_.Size))*100) + "%"}}			**FreeSpace**
				86900658176

Network	Get-WmiObject -Class	(Get-NetIPAddress)	.IPAddress	192.168.150.30
	Get-NetIPAddress | Where-Object InterfaceAlias -eq "Ethernet" | Where-Object AddressFamily -eq "IPv4" | Select-Object IPAddress			**192.168.150.30**
	Test-Connection -ComputerName $ComputerName -Count 1 | Select-Object IPV4Address			**192.168.150.30**
	Get-NetAdapter		**.InterfaceDescription**	**Broadcom NetLink (TM) Gigabit Ethernet**
	Get-NetAdapter | Where-Object Name -Eq "Ethernet" | Select InterfaceDescription,MacAddress		**.MacAddress**	**78-2B-CB-A5-51-EA**
Software	Get-WmiObject -Class	(win32_product)	**.Name**	**Microsoft Office Professional Plus 2016**
	Get-WmiObject -Class win32_product | Where-Object Name -eq "Microsoft Office Professional Plus 2016"		**.Vendor**	**Microsoft Corporation**
			.Version	**16.0.4266.1001**
			.Caption	**Microsoft Office Professional Plus 2016**

Execution Policy

Set-ExecutionPolicy -ExecutionPolicy	AllSigned RemoteSigned Restricted Unrestricted
Get-ExecutionPolicy	Gets the execution policies for the current session

Event Log

-LogName

- Application
- HardwareEvents
- Security
- System

-EntryType

- Error
- Warning
- Information

Get-EventLog -LogName System -EntryType Error -Newest 5 -ComputerName KWS1A045

Index	Time	EntryType	Source	InstanceID	Message
64308	Dec 12 14:32	Error	DCOM	10010	The description for Event ID '10010' in Source 'DCOM' cannot be found. The local computer ...
64303	Dec 12 14:00	Error	DCOM	10010	The description for Event ID '10010' in Source 'DCOM' cannot be found. The local computer ...
64298	Dec 12 12:25	Error	DCOM	10010	The description for Event ID '10010' in Source 'DCOM' cannot be found. The local computer ...
64294	Dec 12 12:19	Error	DCOM	10028	The description for Event ID '10010' in Source 'DCOM' cannot be found. The local computer ...
64293	Dec 12 12:19	Error	DCOM	10028	The description for Event ID '10010' in Source 'DCOM' cannot be found. The local computer ...

Get-EventLog -LogName System -EntryType Error -Newest 5 -ComputerName KWS1A045 | Sort-Object InstanceID -Unique

Index	Time	EntryType	Source	InstanceID	Message
64308	Dec 12 14:32	Error	DCOM	10010	The description for Event ID '10010' in Source 'DCOM' cannot be found. The local computer ...
64293	Dec 12 12:19	Error	DCOM	10028	The description for Event ID '10010' in Source 'DCOM' cannot be found. The local computer ...

E Mail

```
$EmailTo = "To mail id"
$EmailFrom = "From Mail id"
$Subject = "System Report" + " - " + $computername
$Body = $Report
$SMTPServer = "smtp.gmail.com"
$SMTPMessage = New-Object System.Net.Mail.MailMessage($EmailFrom,$EmailTo,$Subject,$Body)
$SMTPClient = New-Object Net.Mail.SmtpClient($SmtpServer, 587)
$SMTPClient.EnableSsl = $true
$SMTPClient.Credentials = New-Object System.Net.NetworkCredential("from mail id", "password");
$SMTPClient.Send($SMTPMessage)
```

or

```
Send-MailMessage -to <To Email Id> -Subject "<Subject>" -body "<Mail Body>" -smtpserver <SMTP Server> -from <From Mail Id>
```

1. Create a local user account on remote machine

```
Clear-Host
$computer = Read-Host "Enter Machine Name"
$username = Read-Host "Enter User Name"
$password = Read-Host "Enter Password"
$fullname = Read-Host "Enter Full Name"
$local_group = "Administrators"
$description = "LocalUser"
$comp = [ADSI]"WinNT://$computer"

Try {
$users = $comp.psbase.children | select -expand name

   if ($users -like $username) {
      Write-Host "$username already exists on $computer"
      }
      else {
#Create the account
      $user = $comp.Create("User","$username")
      $user.SetPassword("$password")
      $user.Put("Description","$description")
      $user.Put("Fullname","$fullname")
      $user.SetInfo()

#Set password to never expire and Set user cannot change password
      $DONT_EXPIRE_PASSWD = 0x10000
      $PASSWD_CANT_CHANGE = 0x40
      $user.userflags = $DONT_EXPIRE_PASSWD + $PASSWD_CANT_CHANGE
      $user.SetInfo()

#Add the account to the local admins group
      $group = [ADSI]"WinNT://$computer/$local_group,group"
      $group.add("WinNT://$computer/$username")

#Validate whether user account has been created or not
      $users = $comp.psbase.children | select -expand name

         if ($users -like $username) {
            Write-Host "$username has been created on $computer"
           }
           else {
           Write-Host "$username has not been created on $computer"
           }
        }
    }
    Catch {
        Write-Host "Error creating $username on $($computer.path):  $($Error[0].Exception.Message)"
        }
```

1. Remove a local user account on remote machine

```
$computer = Read-Host "Enter Machine Name"
$username = Read-Host "Enter User Name"

$comp = [ADSI]"WinNT://$computer"
$comp.Delete("User","$username")
```

2. Enable and Disable Local User

```
$computer = Read-Host "Enter Machine Name"
$username = Read-Host "Enter User Name"

#Enable
$User = [ADSI]"WinNT://$computer/$username,user"
$User.AccountDisabled = $False
$User.SetInfo()

#Disable
$User = [ADSI]"WinNT://$computer/$username,user"
$User.AccountDisabled = $True
$User.SetInfo()
```

3. Reset Password

```
$computername = Read-Host "Enter Host Name"
$username = Read-Host "Enter User Name"
$adminPassword = Read-Host "Reset password"
$adminUser = [ADSI] "WinNT://$computername/$username"
$adminUser.SetPassword($adminPassword)
```

4. Admin group members

```
$ComputerName = Read-Host "Computer Name"
$GroupName = "Administrators"

Get-WmiObject -ComputerName $ComputerName -Query "SELECT * FROM Win32_GroupUser WHERE
GroupComponent=`"Win32_Group.Domain='$ComputerName',Name='$GroupName'`"" | Select-
Object PSComputerName,PartComponent
```

5. Patch Management

Install PowerShell Modules (Install PSWindowsUpdate Module)

```
Install-Module -Name PSWindowsUpdate
```

List of available updates

```
Get-WindowsUpdate
```

Install all available updates and automatically reboot and afterward

```
Get-WindowsUpdate -install -acceptall -autoreboot
```

Install a specific KB

```
Get-WindowsUpdate -KBArticleID KB890830 -install
```

Remove an update

```
Remove-WindowsUpdate -KBArticleID KB890830
```

List of installed updates

```
Get-HotFix
Get-HotFix | Where-Object HotFixID -Match "KB4477136" [or Get-HotFix | Where-Object HotFixID -eq "KB4477136"]
Get-HotFix | Where-Object Description -Match "Security Update"
Get-HotFix | Sort-Object InstalledOn –Descending
```

7. Install application

```
Clear-Host
$computer = "KWS1A064"
$source = "\\ksrst004\Software-Lib\Share\Notepad ++\npp.7.6.2.Installer_x64.exe"
$destination = "\\$computer\D$\SCCMClient"
Get-Service remoteregistry -ComputerName $computer | start-service

Copy-Item -Path $source -Destination $destination -Recurse

$installString = "\\$computer\D$\Apps\npp.7.6.2.Installer_x64.exe /S"
$installation = ([WMICLASS]"\\$computer\ROOT\CIMV2:Win32_Process").Create($installString)

if ($installation.ReturnValue -eq 0) {
"Installation successfully completed. The process Id -"+$installation.ProcessId
}
else { "Installation failed"}
```

8. Uninstall application

```
$app = Get-WmiObject -ComputerName KWS1A064 -Class Win32_Product | Where-Object Name -Match "Notepad*"
$app.Uninstall()
```

9. Execute a Script file remotely

```
Invoke-Command -ComputerName KWS1A064 -ScriptBlock { D:\MailTest.bat }
```

Script for install Windows updates

```
Clear-Host
$version = ($PSVersionTable.PSVersion).Major

if ($version -le 4) {
Write-Host The PowerShell version 5.0 or newer is required to run this script. The current version is $version, please upgrade. -ForegroundColor Yellow
Break }

if ($version -ge 4) {
Write-Host The PowerShell version is $version -ForegroundColor Green
}

$PSModule = Get-Module -ListAvailable | Where-Object Name -Match "PSWindowsUpdate"

if ($PSModule.Name -ne "PSWindowsUpdate") {
Write-Host PSWindowsUpdate module is not installed. -ForegroundColor Yellow

$Install_PSWindowsUpdate_Module = Read-Host "Do you want to install PSWindowsUpdate module? Yes/No"

If ($Install_PSWindowsUpdate_Module -eq "No") {
Break
}

If ($Install_PSWindowsUpdate_Module -eq "Yes") {
Write-Host Installing PSWindowsUpdate module... -ForegroundColor Green
Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force
Install-Module pswindowsupdate -force
Set-ExecutionPolicy -ExecutionPolicy Unrestricted
Import-Module PSWindowsUpdate -force
Write-Host PSWindowsUpdate module successfully installed. -ForegroundColor Green
}

}

$PSModule = Get-Module -ListAvailable | Where-Object Name -Match "PSWindowsUpdate"

if ($PSModule.Name -eq "PSWindowsUpdate") {
Write-Host Checking for new updates available... -ForegroundColor Green
$updates = Get-wulist -verbose
$updatenumber = ($updates.kb).count

Write-Host Number of updates available - $updatenumber -ForegroundColor Green
Write-Host Retrieves a list of available updates..... -ForegroundColor Green

#$updates
$updates.kb | Out-File C:\KB_Files.txt
$updates.kb + " - " + $updates.size

if ($updatenumber -eq "0") {
Write-Host Windows is now up to date -ForegroundColor Green
Break
}

}

$Start_Installation = Read-Host "Install Windows Updates ? Yes/No"

if ($Start_Installation -eq "No") {
Break
}

$KB_List = "\\$computer\c$\KB_Files.txt"
$KB_Files = Get-Content $KB_List

ForEach($KB_File in $KB_Files) {
```

if ($KB_File -ne "") { *Write-Host Installing $KB_File...* *Get-WindowsUpdate -KBArticleID "$KB_File" -Install -AcceptAll -ErrorAction SilentlyContinue* *}* *else {}* *}* *Write-Host Please restart computer*	