Post Config - PowerShell

Tuesday, September 6, 2022 9:48 PM

```
#################################################################################################

# Step 1: General Setup

function Set-PageFile {

   # RAM in System (MB)
      ## Size of D:\ (MB)
      $ram = (Get-ComputerInfo).CsPhyicallyInstalledMemory / 1KB ## Calculates the total server memory in MB
      $SAPPageFile = $ram * 1.5 ## Requirements for SAP is 1.5 of total server memory
      $diskSize = (Get-Volume -DriveLetter D).Size / 1MB

      if ($SAPPageFile -gt $diskSize) {
         Write-Host "Temporary Disk is less than calculated Page File size" -ForegroundColor Yellow
      }

      $Pagefile = Get-WmiObject Win32_PagefileSetting | Where-Object {$_.name -eq "D:\pagefile.sys"}
      $Pagefile.InitialSize = $SAPPageFile / 2
      $Pagefile.MaximumSize = $SAPPageFile
      $Pagefile.put()
   }

Write-Host "Setting Page File..." -ForegroundColor Green
Set-PageFile

Write-Host "Setting timezone to Copenhagen timezone..." -ForegroundColor Green
Set-TimeZone -name "Romance Standard Time"

Write-Host "Allowing traffic through the Windows Firewall Domain Profile..." -ForegroundColor Green
Set-NetFirewallProfile -Name Domain -DefaultInboundAction Allow

Write-Host "Changing the DVD drive to B:\..." -ForegroundColor Green
$cd = $NULL
$cd = Get-WMIObject -Class Win32_CDROMDrive -ComputerName $env:COMPUTERNAME -ErrorAction Stop
if ($cd.Drive -eq "E:")
{
  Write-Output "Changing CD Drive letter from E: to B:"
  Set-WmiInstance -InputObject ( Get-WmiObject -Class Win32_volume -Filter "DriveLetter = 'E:'" ) -Arguments @{DriveLetter='B:'}
}



#################################################################################################

# Step 2: Network

# Rework this into getting the IP configuration from IMDS and assigning the IPs that way through.

$logicalIPs = '10.193.28.104'
$logicalHostNames = "sapqb3cs01","sapqb3cs01.vestas.net"
#Get-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Lsa\MSV1_0' -Name "BackConnectionHostNames"
#$logicalHostNames = "sapqb3cs02","sapqb3cs02.vestas.net"
#Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Lsa\MSV1_0' -Name "BackConnectionHostNames" -Value $logicalHostNames

$interface = Get-NetAdapter | ?{$_.Name -match 'Ethernet*' -and $_.InterfaceDescription -match 'Microsoft Hyper-V Network Adapter*'}
$PrimaryIP = (Get-NetIPAddress -InterfaceIndex $interface.ifIndex | ?{$_.AddressFamily -eq 'IPv4'}).IPAddress
$defaultGateway = ((Get-NetIPConfiguration -InterfaceIndex $interface.ifIndex).IPv4DefaultGateway).nextHop
Set-NetIPInterface -InterfaceIndex $interface.ifIndex -Dhcp Disabled
if (($defaultGateway -eq '10.192.96.1') -or ($defaultGateway -eq '10.71.48.1') -or ($defaultGateway -eq '10.193.28.1')) {
New-NetIPAddress -InterfaceIndex $interface.ifIndex -AddressFamily IPv4 -IPAddress $PrimaryIP -PrefixLength 23 -DefaultGateway $defaultGateway
foreach ($IPs in $logicalIPs) {
New-NetIPAddress -InterfaceIndex $interface.ifIndex -AddressFamily IPv4 -IPAddress $IPs -PrefixLength 23 -SkipAsSource $true
}
}
else {
New-NetIPAddress -InterfaceIndex $interface.ifIndex -AddressFamily IPv4 -IPAddress $PrimaryIP -PrefixLength 24 -DefaultGateway $defaultGateway
foreach ($IPs in $logicalIPs) {
New-NetIPAddress -InterfaceIndex $interface.ifIndex -AddressFamily IPv4 -IPAddress $IPs -PrefixLength 24 -SkipAsSource $true
}
}

Set-DnsClientServerAddress -InterfaceIndex $interface.ifIndex -ServerAddresses "10.0.10.80", "10.0.10.81" -PassThru

# Configure IP addresses for Logical Host Names, those should always be set to SkipAsSource

# Change the values here based on the number of Logical Host Names defined

New-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Lsa\MSV1_0' -Name "BackConnectionHostNames" -Value $logicalHostNames -PropertyType MultiString
New-ItemProperty -Path 'HKLM:\SYSTEM\CurrentControlSet\Services\lanmanserver\parameters' -Name "DisableStrictNameChecking" -Value "1" -PropertyType DWord

#Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\MSV1_0



##########################################################################################################

# Step 3
# Disk Config for SQL DB
# Get-PhysicalDisk -canPool $true | ft DeviceId, Size, PhysicalLocation -a

$disk = @(
#disk1
[pscustomobject]@{
driveLetter = 'E';
storagePoolName = 'SQL_BIN';
driveName = 'SQL_BIN';
deviceID = '2';
sqlDisk = 'True'
}
#disk2
[pscustomobject]@{
driveLetter = 'G';
storagePoolName = 'MsSQLlog';
driveName = 'MsSQLlog';
deviceID = '5';
sqlDisk = 'True'
}
#disk3
[pscustomobject]@{
driveLetter = 'H';
storagePoolName = 'MsSQLtemp';
driveName = 'MsSQLtemp';
deviceID = '6';
sqlDisk = 'True'
}

)
foreach($d in $disk)
{
if ($d.sqlDisk -eq 'True'){
$disks = Get-PhysicalDisk -CanPool $true -DeviceNumber $d.deviceID
New-StoragePool -FriendlyName $d.storagePoolName -PhysicalDisks $disks -ResiliencySettingNameDefault Simple -
StorageSubSystemFriendlyName "Windows Storage*" -ProvisioningTypeDefault Fixed |
New-VirtualDisk -FriendlyName $d.storagePoolName -UseMaximumSize -ResiliencySettingName Simple |
Initialize-Disk -PartitionStyle GPT -PassThru |
New-Partition -DriveLetter $d.driveLetter -UseMaximumSize |
Format-Volume -NewFileSystemLabel $d.driveName -AllocationUnitSize 65536 -UseLargeFRS
}
else
{
$disks = Get-PhysicalDisk -CanPool $true -DeviceNumber $d.deviceID
New-StoragePool -FriendlyName $d.storagePoolName -PhysicalDisks $disks -ResiliencySettingNameDefault Simple -
StorageSubSystemFriendlyName "Windows Storage*" -ProvisioningTypeDefault Fixed |
New-VirtualDisk -FriendlyName $d.storagePoolName -UseMaximumSize -ResiliencySettingName Simple |
Initialize-Disk -PartitionStyle GPT -PassThru |
New-Partition -DriveLetter $d.driveLetter -UseMaximumSize |
Format-Volume -NewFileSystemLabel $d.driveName
}
}

#Disk config for disk pool
$storagePoolName = "MsSQLData"
$driveLetter = "F"
#Get-PhysicalDisk -CanPool $true | where{($_.DeviceId -eq '3') -or ($_.DeviceId -eq '4')}
$disks = Get-PhysicalDisk -CanPool $true | where{($_.DeviceId -eq '3') -or ($_.DeviceId -eq '4')}
New-StoragePool -FriendlyName $storagePoolName -PhysicalDisks $disks -ResiliencySettingNameDefault Simple -
StorageSubSystemFriendlyName "Windows Storage*" -ProvisioningTypeDefault Fixed |
New-VirtualDisk -FriendlyName $storagePoolName -UseMaximumSize -ResiliencySettingName Simple |
Initialize-Disk -PartitionStyle GPT -PassThru |
New-Partition -DriveLetter $driveLetter -UseMaximumSize |
Format-Volume -NewFileSystemLabel $storagePoolName -AllocationUnitSize 65536 -UseLargeFRS
```

```
###########################################################################################################################

# Step 3
# Disk Config for app server
$VolumeName = "Local Disk E_Data"
$storagePoolName = "SAP Application"


$disks = Get-PhysicalDisk -CanPool $true
New-StoragePool -FriendlyName $storagePoolName -PhysicalDisks $disks -ResiliencySettingNameDefault Simple -StorageSubSystemFriendlyName "Windows Storage*" -
ProvisioningTypeDefault Fixed |
New-VirtualDisk -FriendlyName $storagePoolName -UseMaximumSize -ResiliencySettingName Simple |
Initialize-Disk -PartitionStyle GPT -PassThru |
New-Partition -DriveLetter E -UseMaximumSize |
Format-Volume -NewFileSystemLabel $VolumeName
```

```
---Check Disk Details
Get-PhysicalDisk -canPool $true | select DeviceId, @{n='LUN';e={$_.PhysicalLocation.Split(":")
[4]}}, @{n='Size(Gb)';e={[int]($_.Size/1GB)}}

----Pang validate ng disk stripping
$virtualDisks = Get-VirtualDisk
foreach ($vDisk in $virtualDisks) {
    $numberofColumns = $vDisk.numberofColumns
    $volumes = $vDisk | Get-Disk | Get-Partition | Get-Volume
    [PSCustomObject]@{
        FriendlyName = $volumes.FileSystemLabel
        DriveLetter = $volumes.DriveLetter
        StripingColumns = $numberofColumns
    }
}

move-clustergroup "available storage" -node azsapqcs22 -wait 0
move-clustergroup "available storage" -node azsapqcs21 -wait 0
```

Tier1-SRV-ADM-azsapq
SAP_QE1_GlobalAdmin; SAP_SMD_GlobalAdmin

Deploy IaC Build

1. az login
2. az account set --subscription "vestas-sap-ea-westeurope-prd-01"
3. az deployment sub create --template-file C:\Users\RENZ\Documents\Vestas\SQL\Test\template.json --location "westeurope" --confirm