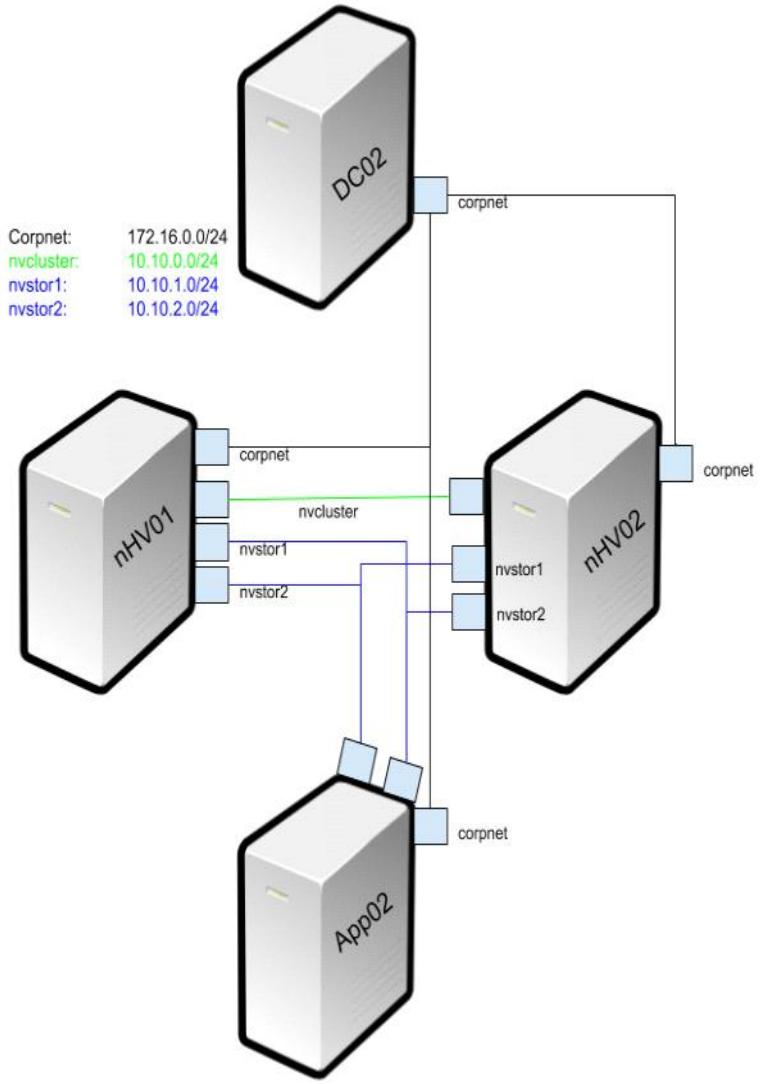
Beschreibung



Was soll erreicht werden?

Ziel ist es, ein Failovercluster mit Hyper-V Hosts als Knoten zu demonstrieren. Der Server 2016 bzw. Windows 10 beherrscht *Nested Virtualization*, d.h. man kann in einer VM nochmals die Hyper-V Rolle installieren und *Powershell-Direct*. Das machen wir uns zu Nutze.

Ausgangslage und Voraussetzungen

Vorhanden ist bereits ein Domaincontroller (DC02) der Domäne corp.howilab.local, sowie ein Mitgliedsserver (App02). Dieser wird als iSCSI-Zielserver für den Failovercluster dienen. Außerdem gibt es die die Server nhv01 und nhv02. Diese sind Workgroupmember und haben keine weiteren Rollen oder Feature installiert. Alle Server haben eine Netzwerkkarte, die dem Hyper-V Switch "Corpnet" zugewiesen sind.

Was ist zu tun?

1. Einrichtung neuer virtueller Switche für den iSCSI Storage (nvstor1, nvstor2) und den Cluster (nvcluster) auf dem Hyper-V Host
2. Hinzufügen und Konfigurieren der Netzwerkkarten zu nhv01, nhv02 und App02
3. nhv01 und nhv02 für Nested Virtualization vorbereiten, beide Server in die Domäne

aufnehmen und die notwendigen Rollen und Feature installieren.

4. App02 als iSCSI-Ziel mit zwei Disks einrichten
5. Redundante Anbindung der zukünftigen Clusterknoten an den gemeinsamen Speicher
6. Cluster Validierung und Einrichtung

1. Hyper-V Host vorbereiten

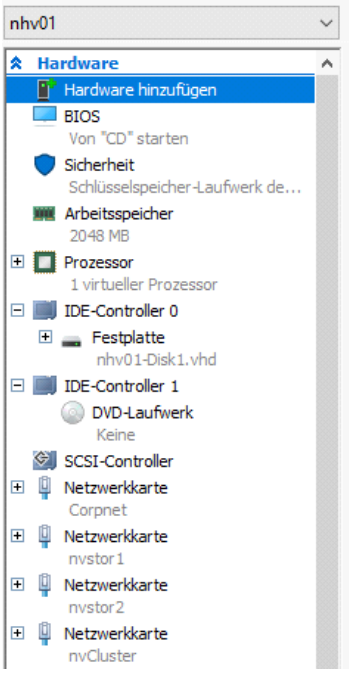
Im ersten Schritt werden auf dem Hyper-V Host neue Switche erstellt. Hierzu dient folgende Powershell Funktion:

```
# Virtuelle Switche erstellen
function SwitchTest ($SwitchName = "StandadrdSwitch") {

# Prüfen, ob ein Privater Switch namens "$SwitchName" schon vorhanden ist und der Variablen den Wert True oder False geben
$VirtualSwitchExists = ((Get-VMSwitch | where {$_.name -eq $SwitchName -and $_.SwitchType -eq "Private"}).count -ne 0)

# Falls kein Switch vorhanden ist wird er angelegt
if ($VirtualSwitchExists -like "False")
  {
   New-VMSwitch -SwitchName $SwitchName -SwitchType Private
   write-host "Privater Switch $SwitchName wurde erstellt"
  }
else
  {
    write-host "Privater Switch $SwitchName ist schon vorhanden"
  }

}

SwitchTest nvstor1
SwitchTest nvstor2
SwitchTest nvCluster
```

Nun werden den VMs nhv01, nhv02 und App02 zusätzliche Netzwerkarten hinzugefügt. Für diesen Schritt müssen die VMs ausgeschaltet sein.

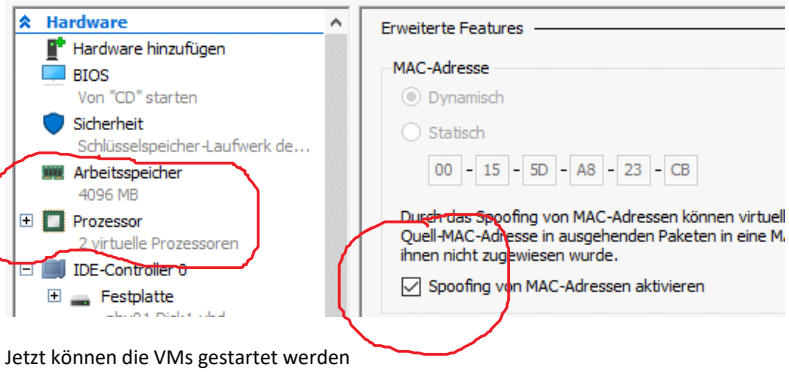
```
# zusatzliche NICs an die VMs hängen
Add-VMNetworkAdapter -VMName nhv01 -SwitchName "nvstor1"
Add-VMNetworkAdapter -VMName nhv01 -SwitchName "nvstor2"
Add-VMNetworkAdapter -VMName nhv01 -SwitchName "nvcluster"
Add-VMNetworkAdapter -VMName nhv02 -SwitchName "nvstor1"
Add-VMNetworkAdapter -VMName nhv02 -SwitchName "nvstor2"
Add-VMNetworkAdapter -VMName nhv02 -SwitchName "nvcluster"
Add-VMNetworkAdapter -VMName App02 -SwitchName "nvstor1"
Add-VMNetworkAdapter -VMName App02 -SwitchName "nvstor2"
```

Das Ergebnis für die VM nhv01 sieht so aus:

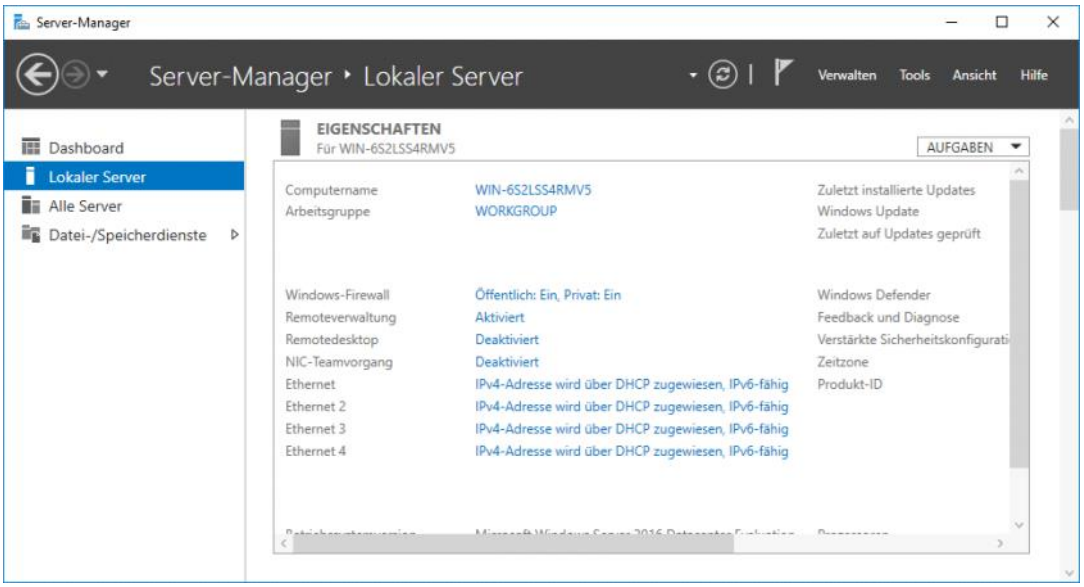


Zum Verwenden der *Nested Virtualization müssen einige Voraussetzungen erfüllt sein:*
Auf den Netzwerkkarten muss MacAdressspoofing eingeschaltet werden, der virtuelle Prozessor muss für die Virtualisierung vorbereitet werden und die VMs benötigen mindestens 4GB statischen Arbeitsspeicher.

```powershell
# MACAddressSpoofing einschalten
Set-VMNetworkAdapter -VMName nhv01 -MacAddressSpoofing on
Set-VMNetworkAdapter -VMName nhv02 -MacAddressSpoofing on

# Virtualisierung für VM konfigurieren
Set-VMProcessor -VMName nhv01 -ExposeVirtualizationExtensions $true -Count 2
Set-VMProcessor -VMName nhv02 -ExposeVirtualizationExtensions $true -Count 2

# 4GB statischen Arbeitsspeicher vergeben
Set-VMMemory -VMName nhv01 -DynamicMemoryEnabled $false -StartupBytes 4GB
Set-VMMemory -VMName nhv02 -DynamicMemoryEnabled $false -StartupBytes 4GB
```



Jetzt können die VMs gestartet werden



Noch sind Maschinen nhv01 und nhv02 in der Arbeitsgruppe "Workgroup", haben nicht den korrekten Namen, keine statischen IP-Adressen und es ist nicht zu erkennen an welchem Switch die Netzwerkkarten hängen. Kümmern wir uns zunächst um die Netzwerkkarten:

```
## Zuordnung der Switche zu den NICs in der VM $vmName prüfen und umbennen
function NicSwitch ($vmName) {
  $netnames = (Get-VMNetworkAdapter -VMName $vmName).SwitchName
  foreach ($netname in $netnames) {
  $adapterID =(Get-VMNetworkAdapter -VMName $vmName | where SwitchName -EQ $netname).AdapterId
  Get-VMNetworkAdapter -VMName $vmName | where Switchname -EQ $netname | Disconnect-VMNetworkAdapter
  Invoke-Command -VMName $vmName -Credential $credential {get-netadapter | where status -EQ Disconnected | Rename-NetAdapter -NewName $Using:netname}
  Get-VMNetworkAdapter -VMName $vmName | where AdapterId -eq $adapterID | Connect-VMNetworkAdapter -SwitchName $netname
  }
}

```

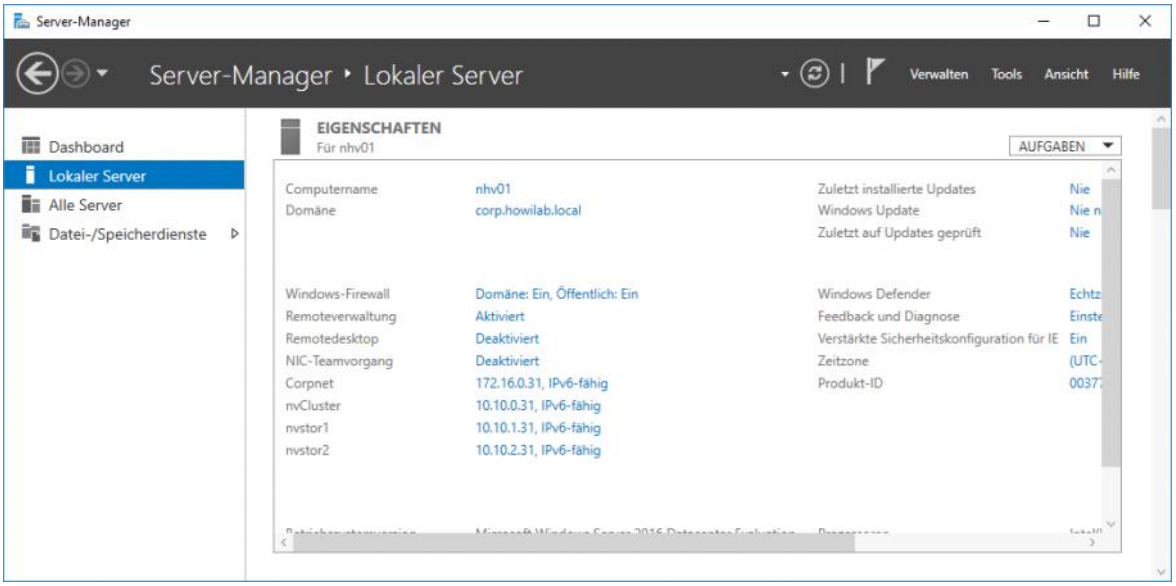
Die Funktion ermittelt als Erstes die Namen der virtuellen Switche (corpnet, nvcluster, nvstor1 und nvstor2). Jeder Netzwerkadapter besitzt eine eindeutige AdapterId, die wir zur korrekten Zuordnung später benötigen. Es wird erst der Switch von der Netzwerkarte getrennt, dann in der VM der getrernnte Adapter in den Switchnamen umbenannt und anschließend wieder verbunden.
Damit der invoke-command Befehl funktioniert müssen die Zugangsdaten in der Variablen $credential übergeben werden.

```
# Zugagngsdaten für NHV01 und NHV02 als Workgroup Mitglieder
$username = "Administrator"
$password = 'Pa$$w0rd' | ConvertTo-SecureString -asPlainText -Force
$credential = New-Object System.Management.Automation.PSCredential($username,$password)

## Zuordnung der Switche zu den NICs in der VM nhv01 prüfen und umbennen
NicSwitch -vmName nhv01

```

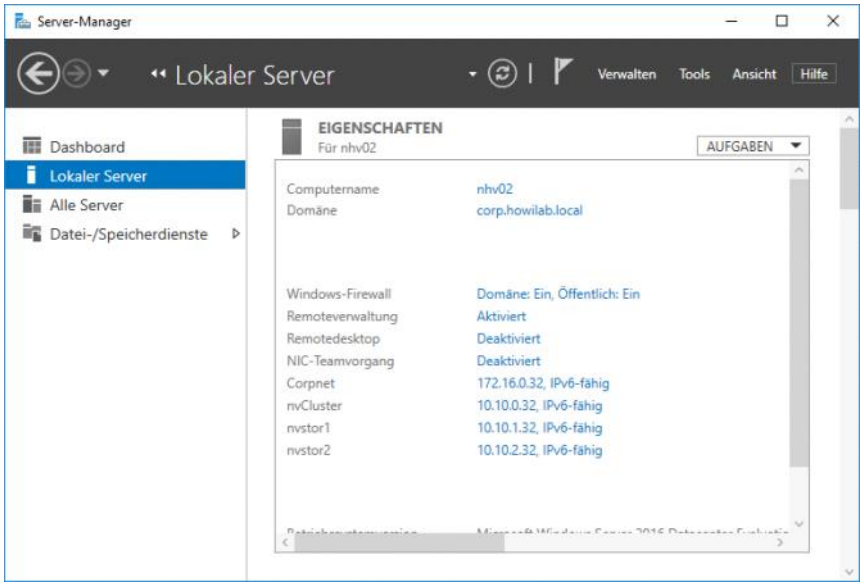
Der nächste Block setzt nun die korrekten IP-Adressen, nimmt den Rechner in die Domäne auf und benennt ihn um. Hier ist darauf zu achten Zeile für Zeile die Kommandos auszuführen, da zunächst die Powershell Session in die VM aufgebaut werden muss, bevor man die übrigen Kommandos ausführt

```
Enter-PSSession -VMName nhv01 -Credential $credential
# Per Powershell Direct die IP-Adressen auf NHV01 setzen und Domäne hinzufügen
$Netnames = @("corpnet","nvstor1","nvstor2","nvcluster")
foreach ($Netname in $Netnames) {
Set-NetIPInterface -InterfaceAlias $Netname -Dhcp Disabled
}
New-NetIPAddress -InterfaceAlias corpnet -IPAddress 172.16.0.31 -DefaultGateway 172.16.0.1 -PrefixLength 24
Set-DNSClientServerAddress -InterfaceAlias corpnet -ServerAddresses 172.16.0.11
New-NetIPAddress -InterfaceAlias nvcluster -IPAddress 10.10.0.31 -PrefixLength 24
New-NetIPAddress -InterfaceAlias nvstor1 -IPAddress 10.10.1.31 -PrefixLength 24
New-NetIPAddress -InterfaceAlias nvstor2 -IPAddress 10.10.2.31 -PrefixLength 24

$domain = "corp.howilab.local"
$username = "$domain\Administrator"
$password = 'Pa$$w0rd' | ConvertTo-SecureString -asPlainText -Force
$credential = New-Object System.Management.Automation.PSCredential($username,$password)
Add-Computer -DomainName $domain -Credential $credential
Rename-computer nhv01
Exit-PSSession
Restart-VM -Name nhv01 -force

```

Das Ergebnis sieht so aus:



Analog führt man die gleichen Schritte für die zweite VM aus.



Nachdem nun beide Maschinen in der Domäne sind werden die benötigten Rollen und Feature installiert, natürlich per Powershell:

```powershell
## Rollen und Feature auf nhv01 und nhv02 installieren
# Zugagngsdaten für NHV01 und NHV02 als Domain Member
$domain = "corp.howilab.local"
$username = "$domain\Administrator"
$password = 'Pa$$w0rd' | ConvertTo-SecureString -asPlainText -Force
$credential = New-Object System.Management.Automation.PSCredential($username,$password)

Enter-PSSession -VMName nhv01 -Credential $credential
Install-WindowsFeature -Name Hyper-V,Failover-Clustering,Multipath-IO -IncludeAllSubFeature -IncludeManagementTools -Restart

Enter-PSSession -VMName nhv02 -Credential $credential
Install-WindowsFeature -Name Hyper-V,Failover-Clustering,Multipath-IO -IncludeAllSubFeature -IncludeManagementTools -Restart
```

App02 als iSCSI-Ziel konfigurieren

Für den Failover Cluster benötigen wir gemeinsamen Speicher. Hierfür benutzen wir den App02 als iSCSI-Zielserver. Wie schon bei den anderen Servern wird zunächst die Zuordnung der Netzwerkkarten geprüft und statische Adressen vergeben. Danach wird der Rollendienst installiert. Schließlich bekommt der Server eine 1GB große Disk, die später als Quorumdisk im Cluster verwendet wird und eine 40GB große Disk für das CSV im Cluster. Zugriff auf diese Disks erhalten unsere nhv Rechner.

```
## Zuordnung der Switche zu den NICs in der VM App02 prüfen und umbennen
NicSwitch -vmName App02

## iscsi Target auf App02 installieren
Enter-PSSession -VMName App02 -Credential $credential

New-NetIPAddress -InterfaceAlias nvstor1 -IPAddress 10.10.1.22 -PrefixLength 24
New-NetIPAddress -InterfaceAlias nvstor2 -IPAddress 10.10.2.22 -PrefixLength 24
Install-WindowsFeature -Name FS-iSCSITarget-Server
New-IscsiVirtualDisk -Path c:\iscsiVirtualDisks\Quorum.vhdx -SizeBytes 1GB
New-IscsiVirtualDisk -Path c:\iscsiVirtualDisks\CSVData.vhdx -SizeBytes 40GB
New-IscsiServerTarget -TargetName nhvHosts
Set-IscsiServerTarget -TargetName nhvHosts -InitiatorIds IPAddress:10.10.1.31,IPAddress:10.10.2.31,IPAddress:10.10.1.32,IPAddress:10.10.2.32
Add-IscsiVirtualDiskTargetMapping -Path c:\iscsiVirtualDisks\Quorum.vhdx -TargetName nhvHosts
Add-IscsiVirtualDiskTargetMapping -Path c:\iscsiVirtualDisks\CSVData.vhdx -TargetName nhvHosts
Exit-PSSession
```

VIRTUELLE iSCSI-DATENTRÄGER

Alle virtuellen iSCSI-Datenträger | 2 insgesamt

Filter

Pfad	Status	Status des virtuellen Datenträgers	Zielname	Zielstatus	Initiator-ID	Größe
App02 (2)						
c:\iscsiVirtualDisks\Quorum.vhdx		Nicht verbunden	nhvHosts	Nicht verbunden	IPAddress:10.10.1.31; IPAddress:10.10.1.32; IPAddress:10.10.2.31; IPAddress:10.10.2.32	1,00 GB
c:\iscsiVirtualDisks\CSVData.vhdx		Nicht verbunden	nhvHosts	Nicht verbunden	IPAddress:10.10.1.31; IPAddress:10.10.1.32; IPAddress:10.10.2.31; IPAddress:10.10.2.32	40,0 GB

iSCSI-ZIELE

c:\iscsiVirtualDisks\Quorum.vhdx auf App02

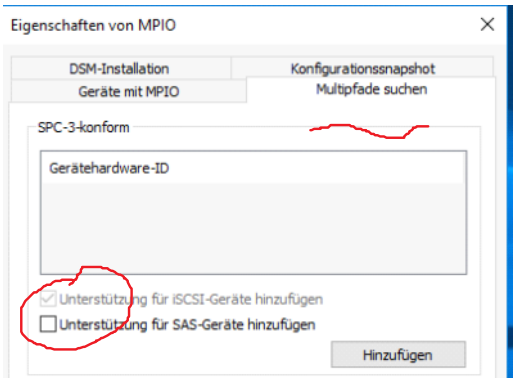
Filter

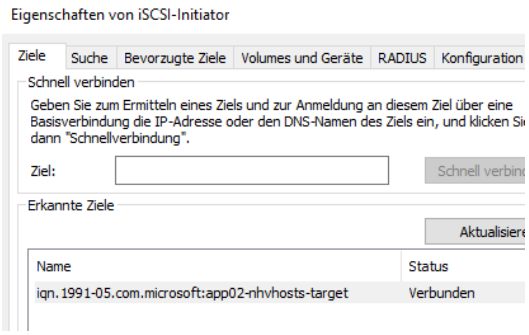
Name	Servername	Ziel-IQN	Zielstatus	Initiator-ID	Le
nhvHosts	App02	iqn.1991-05.com.microsoft:app02-nhvhosts-target	Nicht verbunden	IPAddress:10.10.1.31; IPAddress:10.10.1.32; IPAddress:10.10.2.31; IPAddress:10.10.2.32	

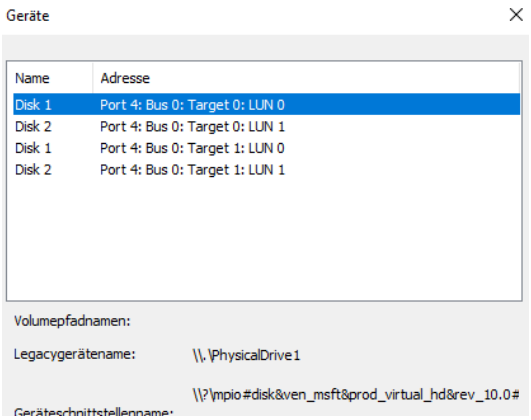
Clustervorbereitung und -Einrichtung

Der Storage muss nun redundant mit den zukünftigen Clusterknoten verbunden werden. Hierzu nutzen wir MPIO und den iSCSI Initiator. Beginnen wir mit dem nhv01. Nachdem der Dienst des iSCSI Initiators für den automatschen Start konfiguriert und gestart wurde, müssen ein paar Firewallregeln und in MPIO die Unterstützung für iSCSI-Geräte aktiviert werden. Danach können wir den Rechner mit den Geräten auf dem Zielserver über zwei unabhängige Wege verbinden.

```
# Konfigurieren desiSCSI Initiators mit MPIO auf nhv01
Enter-PSSession -VMName nhv01 -Credential $credential
Set-Service -Name msiscsi -StartupType Automatic
Start-Service msiscsi
Get-NetFirewallServiceFilter -Service msiscsi | Enable-NetFirewallRule
Enable-MSDSMAutomaticClaim -BusType iSCSI
New-IscsiTargetPortal -TargetPortalAddress 10.10.1.22
New-IscsiTargetPortal -TargetPortalAddress 10.10.2.22
Get-IscsiTarget | Connect-IscsiTarget  -IsPersistent $True -IsMultipathEnabled $True -InitiatorPortalAddress 10.10.1.31
Get-IscsiTarget | Connect-IscsiTarget  -IsPersistent $True -IsMultipathEnabled $True -InitiatorPortalAddress 10.10.2.31

```







Nachdem die Disks nun verbunden sind, werden sie noch initialisiert und formatiert

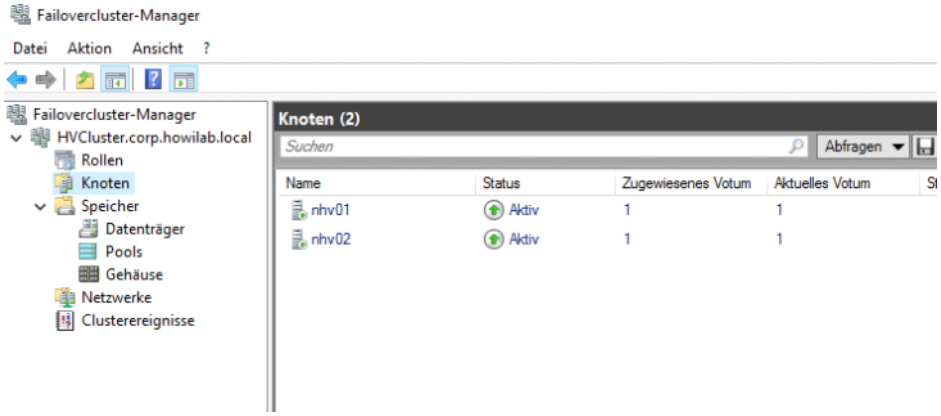
```
get-disk | where Size -EQ 1GB | Initialize-Disk -PartitionStyle MBR
get-disk | where Size -EQ 40GB | Initialize-Disk -PartitionStyle MBR
get-disk | where Size -EQ 1GB | New-Partition -UseMaximumSize -AssignDriveLetter | Format-Volume -FileSystem NTFS -NewFileSystemLabel "Quorum"
get-disk | where Size -EQ 40GB | New-Partition -UseMaximumSize -AssignDriveLetter | Format-Volume -FileSystem NTFS -NewFileSystemLabel "CSVData"

Exit-PSSession

```

Analog verfährt man auf dem nhv02, wobei hier die Disks nur noch Online geschaltet werden müssen.

```powershell
# Konfigurieren desiSCSI Initiators mit MPIO auf nhv02
Enter-PSSession -VMName nhv02 -Credential $credential
Set-Service -Name msiscsi -StartupType Automatic
Start-Service msiscsi
Get-NetFirewallServiceFilter -Service msiscsi | Enable-NetFirewallRule
Enable-MSDSMAutomaticClaim -BusType iSCSI
New-IscsiTargetPortal -TargetPortalAddress 10.10.1.22
New-IscsiTargetPortal -TargetPortalAddress 10.10.2.22
Get-IscsiTarget | Connect-IscsiTarget  -IsPersistent $True -IsMultipathEnabled $True -InitiatorPortalAddress 10.10.1.32
Get-IscsiTarget | Connect-IscsiTarget  -IsPersistent $True -IsMultipathEnabled $True -InitiatorPortalAddress 10.10.2.32

get-disk | where Size -EQ 1GB | set-disk -IsOffline:$false
get-disk | where Size -EQ 40GB | set-disk -IsOffline:$false

Exit-PSSession
```

Jetzt sind es nur noch wenige Schritte bis zum fertigen Cluster. Auf dem nhv01 wird der Validierungstest ausgeführt und danach ein neuer Cluster erstellt.

```powershell
## Cluster erstellen auf NHV01
Enter-PSSession -VMName nhv01 -Credential $credential
# Validierungstest
Test-Cluster -Node NHV01,NHV02
# Cluster erstellen
New-Cluster -Name HVCluster -Node NHV01,NHV02 -StaticAddress 172.16.0.40
```



Das Skript

nvcluster