Citrix PVS Store – Replikationsskript

Citrix Provisioning ist seit Jahren eine super Methode um Citrix Worker zu provisionieren. In der Konsole gibt es zwar seit ich denken kann die Möglichkeit den Replikationsstatus zu prüfen, jedoch bietet Citrix da keine eigenen Mechanismen innerhalb des PVS.

Wie viele andere auch haben wir seit Jahren auf einfache Skripte gesetzt, welche die lokalen Verzeichnisse zwischen den Servern kopiert. Mit dem Nachteil, dass bei mehreren Admins es zu grossen Kopierjobs kommen kann, wenn verschiedene vDisks gepflegt werden. Citrix hat dafür schon seit längerem das vDisk Replicator Utility veröffentlicht, welches wir jedoch nicht installieren wollten. Es ist für unsere Zwecke auch ein wenig überdimensioniert.

Ich habe mich dafür einmal hingesetzt und auf Basis unseres einfachen Skriptes ein neues mit Powershell kreiert, welches folgende Anforderungen erfüllen sollte:

- das Skript sollte evaluieren, welches der lokale und welches der Remote PVS ist
- das Skript sollte für mehrere Stores nutzbar sein – mit Auswahl
- das Skript sollte innerhalb des gewählten Stores die möglichen zu kopierenden vDisks als Auswahl bereitstellen

Die erste Anforderung war leicht und schnell implementiert:

```
# Script variables
$PVS01 = "pvs01.domain.pit"
$PVS02 = "pvs02.domain.pit"
```

```
# Enumerate the local and the remote PVS
$PVSLocal = $env:COMPUTERNAME + ".domain.pit"
If ($PVSLocal -eq $PVS01) {
    $PVSRemote = $PVS02
}
Else {
    $PVSRemote = $PVS01
}
```

Für die Store-Auswahl habe ich mich in dieser ersten Skript-Version mit einem statischen Array begnügt:

```
# Define PVS vDisk stores
# Create an array for each store and let the user chose which one should be checked
# If a store is added to the array, the option has to be added for the choice as well
$StoreArray = @()
$StoreArray                                                    += [pscustomobject]@{Store="VDI";StoreDisk="V";StorePath="\vDisks\VDI"}
$StoreArray                                                    += [pscustomobject]@{Store="RDSH";StoreDisk="R";StorePath="\vDisks\RDSH"}
$StoreArray                                                    += [pscustomobject]@{Store="Test";StoreDisk="T";StorePath="\vDisks\Test"}

# Prompt for Store choice
$StoreTitle = "PVS Store"
$StoreMessage = "Which PVS store will you replicate"
$Store1                    =                    New-Object System.Management.Automation.Host.ChoiceDescription   "&VDI Store", "VDI"
$Store2                    =                    New-Object System.Management.Automation.Host.ChoiceDescription   "&RDSH Store", "RDSH"
$Store3                    =                    New-Object System.Management.Automation.Host.ChoiceDescription   "&Test Store", "TEST"
$StoreOptions                                                   = [System.Management.Automation.Host.ChoiceDescription[]]($Store
```

```
1, $Store2, $Store3)
$Store2Replicate=$host.ui.PromptForChoice($StoreTitle,
$StoreMessage, $StoreOptions, 2)
```

Anhand der Auswahl werden die nächsten Variablen definiert und das lokale und remote Verzeichnis verglichen, ohne den WriteCache Ordner und allfällige .lok oder .i.vhdx Dateien:

```
# Prepare variables after the user entry
$StoreID = $Store2Replicate
$StoreDisk = $StoreArray[$StoreID].StoreDisk
$StorePath = $StoreArray[$StoreID].StorePath
$LocalStore = $StoreDisk + ":" + $StorePath + "\"
$RemoteStore = "\\" + $PVSRemote + "\" + $StoreDisk + "$" +
$StorePath + "\"
$LocalStorePath = $LocalStore + "*"
$RemoteStorePath = $RemoteStore + "*"

# Compare servers
$LocalStoreContent = Get-ChildItem -Path $LocalStorePath -
Exclude WriteCache,*.lok,*.i.vhdx
$RemoteStoreContent = Get-ChildItem -Path $RemoteStorePath -
Exclude WriteCache,*.lok,*.i.vhdx
$StoreDiff = Compare-Object -ReferenceObject
$LocalStoreContent -DifferenceObject $RemoteStoreContent -
Property Name, LastWriteTime
```

Die gefundenen Differenzen werden verglichen. Uns interessiert bei Ausführung des Skriptes jedoch nur, was auf unserem lokalen Server zum Kopieren ist (SideIndicator „<=“), sowie die eigentlichen vDisks (*.vhd*) ohne die Konfigurationsdateien. Aus diesen Informationen wird das nächste Array erstellt:

```
# Enumerate a list of vDisks
# Create an array with all disk names
$DiskArray = $null
$DiskArray = @()
$DiskID = 1
ForEach ($File in $StoreDiff) {
    If ($File.SideIndicator -eq "<="){
        $Filename = $File.Name
```

```
        If ($Filename -like "*.vhd*") {
            $DiskID = $DiskID+1
            $SplitChar = $Filename.IndexOf(".")
            $DiskName = $Filename.Substring(0, $SplitChar)
            $PVPName = $DiskName + ".pvp"
                                                $DiskArray   +=
[pscustomobject]@{DiskID=$DiskID;vDisk=$Filename;vDiskName=$Di
skName;PVPName=$PVPName}
        }
    }
}
```

Aus diesem Array heraus wird dann die nächste Auswahl generiert. Die Auswahl enthält als erstes und auch Standardauswahl die Option alles zu kopieren und die Variante abzubrechen. Aktuell ist vorgesehen, dass maximal fünf vDisks zur Auswahl gestellt werden können für einen einzelnen Kopier-Job:

```
# Prompt for replication choice
$ReplTitle = "vDisk replication"
$ReplMessage = "Choose what to replicate"
$vDiskAll                    =                    New-Object
System.Management.Automation.Host.ChoiceDescription   "&All
vDisks","Continue with all vDisks."
$CancelAll                   =                    New-Object
System.Management.Automation.Host.ChoiceDescription "&Cancel",
"Skip this operation and all subsequent operations."

# Generate choice output of each vDisk to replicate
$ReplOptionCount = 0
ForEach ($vDisk in $DiskArray) {
    $ReplOptionCount = $ReplOptionCount +1
    $ChoiceID = $vDisk.DiskID
    $ChoiceDisk = $vDisk.vDiskName
                         $ChoiceCmd      =      New-Object
System.Management.Automation.Host.ChoiceDescription
"&$ChoiceID $ChoiceDisk", "Replicate selected vDisk."
    New-Variable "ReplOption$ChoiceID" $ChoiceCmd
}
```

```
# Generate $ReplOptions variable for final choice output depending on the amount of options
Switch ($ReplOptionCount) {
                            0       {$ReplOptions       = [System.Management.Automation.Host.ChoiceDescription[]]($vDiskAll, $CancelAll)}
                            1       {$ReplOptions       = [System.Management.Automation.Host.ChoiceDescription[]]($vDiskAll, $CancelAll, $ReplOption2)}
                            2       {$ReplOptions       = [System.Management.Automation.Host.ChoiceDescription[]]($vDiskAll, $CancelAll, $ReplOption2, $ReplOption3)}
                            3       {$ReplOptions       = [System.Management.Automation.Host.ChoiceDescription[]]($vDiskAll, $CancelAll, $ReplOption2, $ReplOption3, $ReplOption4)}
                            4       {$ReplOptions       = [System.Management.Automation.Host.ChoiceDescription[]]($vDiskAll, $CancelAll, $ReplOption2, $ReplOption3, $ReplOption4, $ReplOption5)}
                            5       {$ReplOptions       = [System.Management.Automation.Host.ChoiceDescription[]]($vDiskAll, $CancelAll, $ReplOption2, $ReplOption3, $ReplOption4, $ReplOption5, $ReplOption6)}
}

$vDisk2Replicate=$host.ui.PromptForChoice($ReplTitle, $ReplMessage, $ReplOptions, 0)
```

Nach der Auswahl werden entweder alle vDisks oder die ausgewählte kopiert, oder das Skript abgebrochen:

```
If ($vDisk2Replicate -eq 0) {
    # Copy all found vDisks and PVP files
    ForEach ($Disk in $DiskArray) {
        $SourcevDisk = $null
        $SourcePVP = $null
        $DestvDisk = $null
        $DestPVP = $null
        $vDiskFile = $Disk.vDisk
        $PVPFile = $Disk.PVPName

        $SourcevDisk = $LocalStore + $vDiskFile
```

```
        $SourcePVP = $LocalStore + $PVPFile
        $DestvDisk = $RemoteStore + $vDiskFile
        $DestPVP = $RemoteStore + $PVPFile

        Copy-Item $SourcevDisk -Destination $DestvDisk
        Copy-Item $SourcePVP -Destination $DestPVP
    }
}
ElseIf ($vDisk2Replicate -eq 1) {
    # Cancel all
    return; break
}
Else {
    # Copy the selected vDisk
    # ID has to be reduced by 2 to match the array IDs
    $SourceID = $vDisk2Replicate - 2
    $SourcevDisk = $null
    $SourcePVP = $null
    $DestvDisk = $null
    $DestPVP = $null
    $vDiskFile = $DiskArray[$SourceID].vDisk
    $PVPFile = $DiskArray[$SourceID].PVPName

    $SourcevDisk = $LocalStore + $vDiskFile
    $SourcePVP = $LocalStore + $PVPFile
    $DestvDisk = $RemoteStore + $vDiskFile
    $DestPVP = $RemoteStore + $PVPFile

    Copy-Item $SourcevDisk -Destination $DestvDisk
    Copy-Item $SourcePVP -Destination $DestPVP
}
```

Für einen ersten Wurf find ich das Skript nicht schlecht. Sollte ich die Zeit finden, könnte ich mir vorstellen, dass ich die Store-Auswahl auch noch dynamischer gestalte, in dem ich mittels Powershell die PVS Konfiguration abfrage. Ebenfalls fehlt aktuell noch so etwas wie ein Statusbalken. Das Ende des Kopierens erkennt man nur am geschlossenen Skriptfenster.

Viel Spass beim Nachbauen :-)

ReplicatePVSStores.ps1_Herunterladen