Whitepaper

Software Defined Networking (SDN) mit Windows Server 2016 und System Center Virtual Machine Manager 2016 (SCVMM 2016)

© Gerhard Glenk
IT Consulting
Josef-Simon-Str. 35
90473 Nürnberg
E-Mail: gerhard.glenk@online.de

Version 1.0 – August 2018

Veröffentlicht bei:
Rachfahl IT-Solutions GmbH & CO. KG
Heiligenhaus 21
59969 Hallenberg
Website: https://www.hyper-v-server.de

Inhalt

1 Vorwort und Motivation

Der Microsoft Windows Server 2016 bietet eine Plattform zum Aufbau eines Software-Defined Data Centers (SDDC) basierend auf technischen Innovationen aus Microsoft Azure. Ein kritischer Bestandteil eines SDDC ist dabei das Thema Software Defined Networking (SDN). Microsoft Windows Server 2016 enthält eine Vielzahl an Neuerungen innerhalb des SDN-Stacks wie z.B. Netzwerk Virtualisierung mit dem VXLAN-Protokoll sowie Gateways, Load Balancer und Remote Access Dienste für virtualisierte Netze, welche sich durch den Einsatz von System Center 2016 Virtual Machine Manager effizient bereitstellen und verwalten lassen.

Auf der CDC 2018 in Hanau habe ich zusammen mit Petra Lipp einen Vortrag zum Thema "Software Defined Networking (SDN) mit Windows Server 2016 und System Center Virtual Machine Manager 2016 (SCVMM 2016)" gehalten. Mittlerweile haben wir einige Anfragen bekommen über Details zu den gezeigten Präsentationen und Demoszenarien.

Da keine Mitschnitte von der Veranstaltung existieren, habe ich beschlossen, die wichtigsten technischen Informationen sowie eine Beschreibung zum Aufbau der gezeigten Demoumgebung in einem Whitepaper zusammenzufassen.

1.1 Warum SDN?

In großen Rechenzentren war die Netzwerkverwaltung schon immer eine sehr personal- und zeitaufwendige und damit kostenintensive Aufgabe. Durch die immer stärker zunehmende Server-Virtualisierung entstehen weitere neue Herausforderungen: Netzwerkdienste müssen dynamisch und schnell bereitgestellt werden, die Verwaltung sollte über ein zentrales Werkzeug geschehen und über normierte Schnittstellen (APIs) programmierbar sein, so dass auch eine Automatisierung z.B. durch Skripte möglich wird. Außerdem sind Methoden notwendig, um den Netzwerkverkehr für verschiedene – möglicherweise sogar konkurrierende – Anwendergruppen sicher voneinander zu trennen und damit die jeweiligen Netzwerksegmente zu isolieren. Die Restriktionen der hierfür bislang verwendeten VLAN-Technik sollten beseitigt werden.

SDN Technologien versprechen, all diese Herausforderungen zu lösen.

1.2 Skript Download

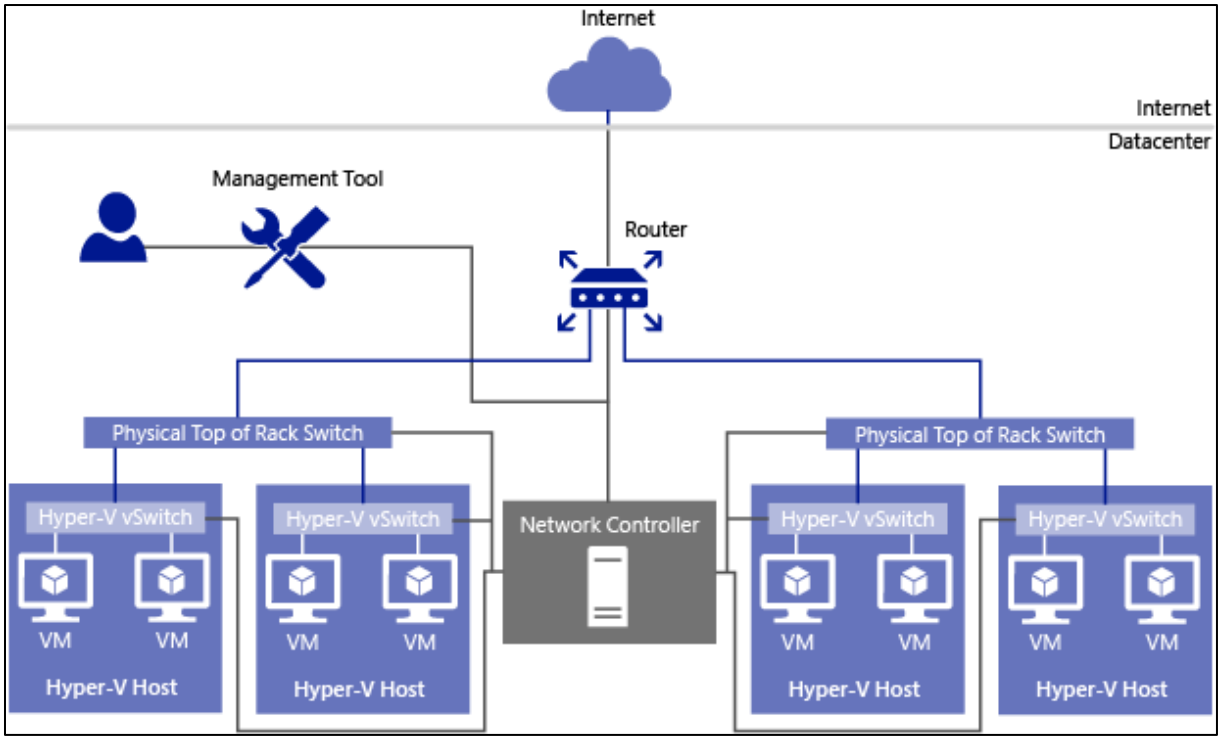
In diesem Whitepaper beschreibe ich einige PowerShell Skripte, die ich mir zusätzlich zu den von Microsoft bereitgestellten erstellt habe, um Aktionen, die mit vielen Mausklicks verbunden sind, etwas zu vereinfachen. Diese zusätzlichen Skripte habe ich in einer .ZIP-Datei zusammengestellt, die Sie hier downloaden können. Details zu den Skripten finden Sie am Ende dieses Dokuments.

2 Die neue Microsoft SDN-Architektur

Eigentlich gibt es SDN im Windows Server bereits seit der Version 2012R2. Dort hatte Microsoft die Technologie an verschiedenen Stellen im Betriebssystem und im SCVMM „versteckt“. Zur Isolation von Netzen wurde der GRE-Standard (Generic Routing Encapsulation) verwendet. GRE hat sich für diesen Zweck jedoch nicht so recht am Markt durchgesetzt. Die Mehrzahl von Netzwerk-Anbietern setzt mittlerweile auf den VXLAN-Standard (Virtual Extensible LAN), was zur Folge hatte, dass Microsoft dieses Protokoll sowohl in der Microsoft Azure Cloud implementierte und die entsprechenden Komponenten nun auch im Windows Server 2016 bereitstellt.

Anmerkung: VXLAN ist nun das Standard-Protokoll für SDN. Jedoch wird auch weiterhin das GRE-Protokoll unterstützt, so dass bisherige Installationen weiter genutzt werden können. Eine Migration von GRE auf VXLAN ist jedoch nicht vorgesehen und bedeutet eine Neuinstallation.

Werfen wir zunächst einen Blick auf das folgende Architekturbild, das aus der Microsoft TechNet Library stammt.



Im Zentrum der neuen SDN-Architektur steht eine neue Serverrolle, die nur in der Datacenter Edition von Windows Server 2016 verfügbar ist, der *Network Controller*. Wir benötigen dafür entweder physische oder auch virtuelle W2016 Datacenter Systeme, auf denen diese Serverrolle installiert und konfiguriert ist. Und warum die Mehrzahl? Technisch würde eine Instanz des *Network Controllers* genügen. Um aber die Hochverfügbarkeit zu gewährleisten, empfiehlt Microsoft für produktive Umgebungen aber wenigstens 3 Instanzen.

Der *Network Controller* besitzt zwei Schnittstellen (APIs), die *Northbound API* und die *Southbound API*. Über die *Southbound API* kommuniziert der *Network Controller* mit den physischen und virtuellen Netzwerkgeräten, also z.B. mit den physischen Switches im Server Rack (Top of Rack – TOR) oder mit den virtuellen Switches in den Hyper-V Hostsystemen. Über die *Northbound API* erhält der *Network Controller* von verschiedenen Netzwerk Management Tools Anweisungen bzw. Richtlinien,

wie die über die *Southbound API* verbundenen Netzwerkgeräte den Netzwerkverkehr behandeln sollen.

Für die Kommunikation zwischen dem *Network Controller* und den verschiedenen Netzwerkkomponenten muss in einer Microsoft SDN Plattform das in jedem Rechenzentrum ohnehin vorhandene (logische) Management Netz verwendet werden, das ich in diesem Artikel mit MGMT bezeichnen will.

Anmerkung: Die Einbettung des *Network Controllers* in das MGMT Netz hat nicht zur Folge, dass der Netzwerkverkehr virtueller (Kunden-) Maschinen ebenfalls über dieses Netz läuft. Hierfür werden wir noch weitere logische Netze einführen.

Die *Northbound API* ist als *Representational State Transfer (REST) API* ausgelegt, so dass beliebige Management Tools, die dieses Protokoll beherrschen, verwendet werden können, um dem *Network Controller* Richtlinien zu übermitteln. Darüber hinaus bietet diese API auch die Möglichkeit, das Netzwerk zu überwachen und bei Problemen Informationen fürs Troubbleshooting abzurufen.

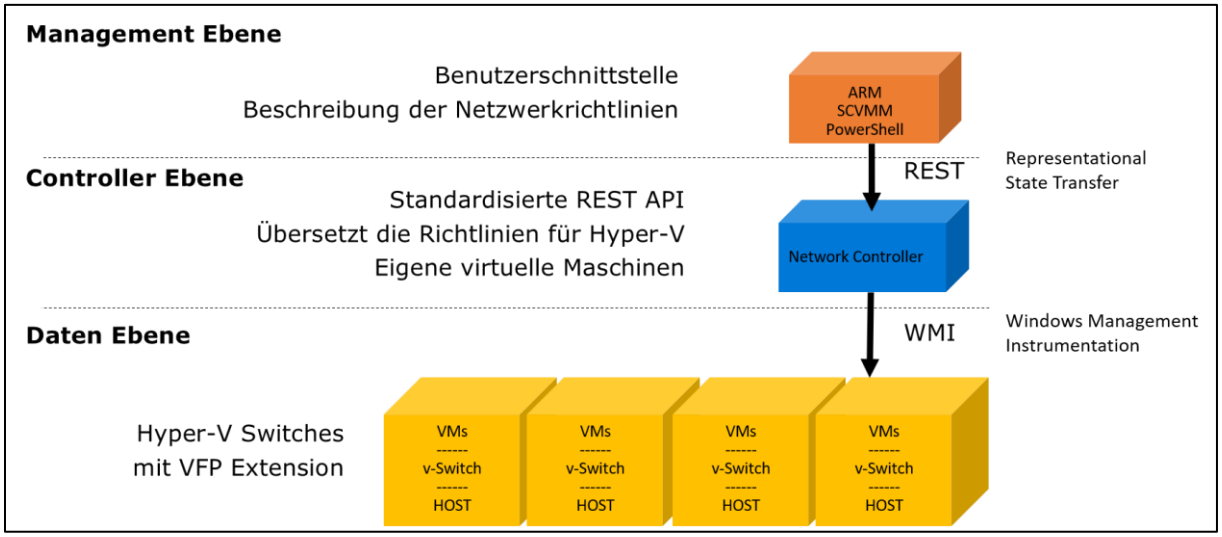
Als Management Tools für den *Network Controller* bietet Microsoft aktuell folgende Werkzeuge:

- PowerShell Cmdlets
- den *Azure Resource Manager (ARM)*
- den *System Center 2016 Virtual Machine Manager* (*SCVMM 2016*)

Insbesondere der SCVMM 2016 bietet mit seiner grafischen Benutzeroberfläche eine einfache Möglichkeit zum Konfigurieren des *Network Controllers* und stellt auch alle Funktionen über eigene PowerShell Cmdlets zur Verfügung. Wir werden uns dies später noch genauer ansehen.

Die über die Northbound API erhaltenen Richtlinien verteilt der Network Controller dann über seine Southbound API an die betroffenen Netzwerkgeräte. Bei Hyper-V Hosts ist dies eine neue Hyper-V Switch Erweiterung (Extension) mit dem Namen *Microsoft Azure VFP Switch Extension*. VFP steht für „Virtual Filtering Platform“.

Die Microsoft SDN-Architektur besteht also aus 3 Ebenen, wie die folgende Abbildung zeigt.



3 Netzwerk Controller Funktionalitäten

Mit dem Netzwerk Controller können nun folgende Funktionalitäten realisiert werden:

3.1 Netzwerk Virtualisierung

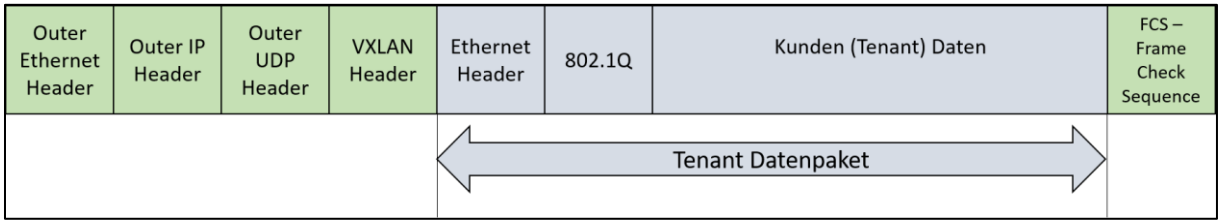
Es können für Gruppen von Kunden VMs (häufig auch als „Tenant“ VMs bezeichnet) eigene logische virtuelle Netze definiert werden. Diese Netze sind voneinander völlig getrennt, auch wenn sie identische Netzwerkparameter wie IP-Bereiche oder VLANIDs besitzen. Die Isolation des Datenverkehrs innerhalb eines solchen virtuellen Netzes erfolgt durch „Verpacken“ der Datenpakete nach dem VXLAN-Protokoll.

3.1.1 Das VXLAN-Protokoll

VMs in einem virtuellen Netz verwenden typischerweise kundenspezifische IP-Adressen („Customer Addresses“ – CAs), die nicht vom Rechenzentrumsbetreiber verwaltet und mit denen die Netzwerkgeräte im Rechenzentrum somit nichts anfangen können. Will nun eine Kunden-VM mit einer anderen kommunizieren, gibt sie ein Ethernet Netzwerkpaket über ihren Netzwerkadapter an einen Switch weiter, in einer Hyper-V Umgebung also an den Hyper-V Switch des Hosts, auf dem sie gerade läuft. Im Hyper-V Switch wird geprüft, ob die Ziel-VM im gleichen Host läuft. Falls ja, wird das Netzwerkpaket vom Switch direkt an die entsprechende VM weitergeleitet.

Falls die Ziel-VM jedoch auf einem anderen Host läuft, wird das Datenpaket in ein neues Netzwerkpaket mit vom RZ-Betreiber verwalteten IP-Adressen („Provider Addresses“ – PAs) gemäß dem VXLAN-Protokoll verpackt und an den Zielhost per UDP weitergeleitet. Dieser kann dann das ursprüngliche Datenpaket wieder auspacken und an die entsprechende VM weiterleiten.

Um die Zuordnung eines verpackten Kunden-Datenpakets zu gewährleisten, wird jedem virtuellen Kundennetz beim Anlegen über den Netzwerk Controller eine eindeutige VXLANID zugewiesen. Diese VXLANID hat eine Länge von 24 Bit. Somit können bis zu 16 Millionen Kundennetze identifiziert werden. Beim Verpacken eines Kunden-Datenpakets wird die VXLANID des Kundennetzes dem Kunden-Datenpaket als Header vorangestellt, so dass beim Auspacken der Daten im Zielhost diese in das entsprechende Kundennetz geleitet werden können.



Für die Kommunikation zwischen den Kunden-VMs benötigen wir also ein (und nur ein) zusätzliches vom Provider verwaltetes logisches Netz, über das die in VXLAN eingepackten Kunden-Datenpakete weitergeleitet werden. Für die weiteren im Netzwerk vorhandenen Geräte wie Switches oder Router sind die Datenpakete der Kunden-VMs also „unsichtbar“ und benötigen somit auch keine weiteren Management Aktivitäten.

Zum Vergleich: Bei der bisherigen Methode zur Netzwerk Virtualisierung mit VLANIDs muss hingegen jedes Kundennetz mit einer eindeutigen VLANID vom Provider definiert, verwaltet und an alle Netzwerkgeräte verteilt werden. Da die VLANIDs nur aus 12 Bits bestehen, ist die Anzahl von Kunden-Netzen auf knapp 4096 begrenzt.

Das logische Netzwerk zum Weiterleiten der mit VXLAN verpackten Datenpakete wird in einer Microsoft SDN-Umgebung üblicherweise als HNVPA (Hyper-V Netzwerk Virtualisierung mit Provider Adressen) Netz bezeichnet.

Die Netzwerk Virtualisierung mit VXLAN vereinfacht also drastisch das Netzwerkmanagement in einem Rechenzentrum, da sich der Provider nicht mehr explizit um die Isolation von Kundennetzen kümmern muss.

3.2 Load Balancing

Für VMs, die in einem virtuellen Kundennetzwerk den gleichen Workload bereitstellen (z.B. Webserver Cluster), kann per Software ein Lastenausgleich („Software Load Balancing“ – SLB) in Form eines SLB-Pools definiert werden. Das Load Balancing wird von eigenständigen über den Netzwerk Controller verwalteten VMs durchgeführt. Die SLB VMs ermöglichen dann auch den Zugriff von außen auf die SLB-Pools in einem virtuellen Kundennetzwerk.

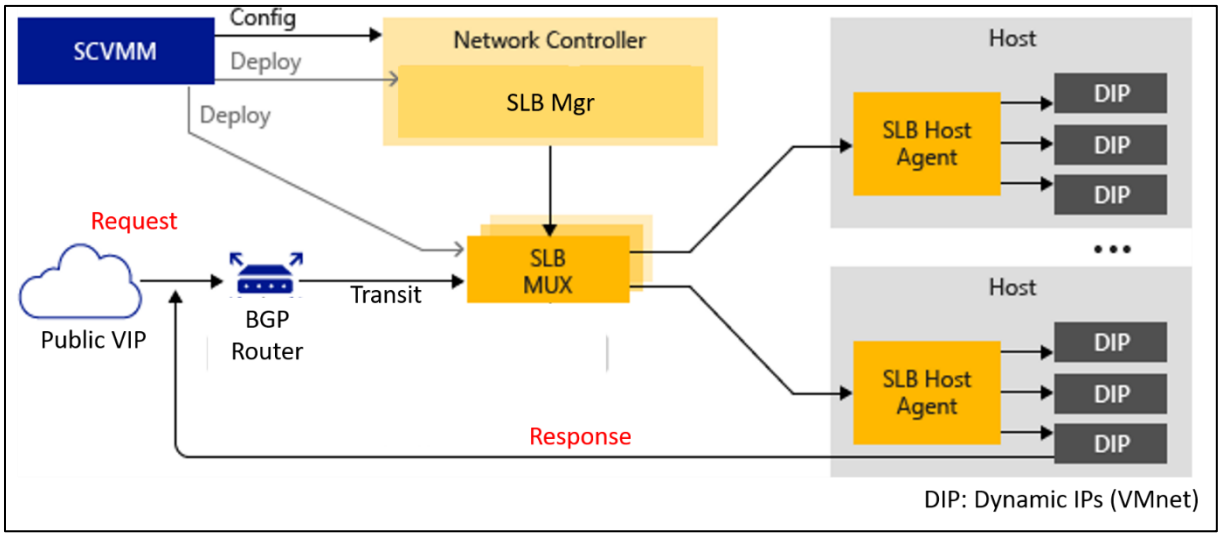
Ein SLB-Pool enthält die virtuellen IP-Adressen der für das Load Balancing vorgesehenen Kunden-VMs (hier als dynamische IP-Adressen – DIPs – bezeichnet). Dem Pool wird dann eine öffentliche IP-Adresse (PublicIP) zugeordnet. PublicVIPs sind typischerweise über das Internet erreichbar, so dass sich Anwender von außen damit verbinden können.

Der Netzwerk Controller übergibt diese Definitionen an die SLB VMs über das MGMT-Netz. Diese wiederum geben die PublicVIPs über das BGP-Protokoll bei einem zentralen BGP-Router bekannt, der die Verbindung zur Außenwelt bereitstellt. Die Kommunikation zwischen den SLB-VMs und dem BGP-Router wird typischerweise über ein eigenes logisches Netz mit dem Namen *Transit* abgewickelt.

Erreicht nun eine Anfrage den BGP-Router zu einer ihm bekannt gemachten öffentliche VIP-Adresse, leitet er diese über das Transit Netzwerk einer SLB-VM zur weiteren Bearbeitung weiter. Da alle SLB-VMs die gleichen Pool-Definitionen enthalten, bestehen also mehrere Wege für diese Weiterleitung. Wir sprechen deshalb hier auch von einem SLB Multiplexer oder SLB-MUX. Bei der Weiterleitung wird die Zieladresse der Anfrage mit der SLB-MUX Adresse im Transit Netz überschrieben.

Der SLB-MUX kann nun in Zusammenspiel mit den SLB Agents in den Hyper-V Switches entscheiden, an welche VM bzw. DIP die Anfrage zur Bearbeitung gegeben wird. In der Anfrage wird dazu die Zieladresse mit der ermittelten DIP überschrieben, das Datenpaket mit VXLAN verpackt und dem entsprechenden Hyper-V Host über das HNVPA-Netz übermittelt. Der Hyper-V Host entpackt das Paket und leitet es an die VM weiter.

Die VM erstellt nun eine Antwort, die als Ziel die ursprüngliche Client Adresse enthält und als Sender die eigene DIP. Der Switch im Hyper-V Host ersetzt diese DIP wieder mit der dem SLB-Pool zugeordneten PublicVIP und sendet das Paket dann über ein Gateway im öffentlichen Netz direkt an den Client, von dem die Anfrage kam. Der Client bekommt also nichts mit von den verschiedenen Weiterleitungsschritten innerhalb der SLB-MUX Umgebung.



3.3 Remote Access Service (RAS) Multitenant Gateways

In der vorstehenden Beschreibung für die Load Balancing Funktion habe ich kurz beschrieben, wie eine Anfrage eines einzelnen Clients aus dem Internet in einer SDN-Umgebung bearbeitet wird. In vielen Fällen wird es jedoch notwendig sein, nicht nur von einem einzelnen Client System eine Verbindung zu den VMs eines Tenants herzustellen, sondern ganze externe Netzwerke mit den virtuellen Systemen eines Tenants zu verbinden.

Hierzu stellt der Netzwerk Controller die Funktion Remote Access Service (RAS) Multitenant Gateway bereit. Es handelt sich dabei (ähnlich den SLB-MUX Systemen) um eigene VMs, die über den Netzwerk Controller verwaltet werden. Damit können dann folgende Remote Szenarien realisiert werden:

- Site-to-Site VPN
- Point-to-Site VPN
- GRE Tunneling
- Dynamic Routing mit dem Border Gateway Protocol (BGP)

In diesem Whitepaper werde ich nicht näher auf diese Themen eingehen, da wir dazu entsprechende Remote Systeme benötigen, die aktuell in unserer Lab-Umgebung nicht verfügbar sind.

4 SDN-Topologie

Nachdem wir nun einige grundlegende Merkmale der Microsoft SDN Architektur geklärt haben, wollen wir uns kurz anschauen, wie dies alles in eine physikalische Umgebung abgebildet werden kann.

4.1 Logische Netze

Fassen wir zusammen, welche logischen Netze in einer SDN-Umgebung notwendig sind:

- NC_Management – Dies ist das zentrale Netzwerk, über das alle Systeme in einem „Software Defined Datacenter“ (SDDC) verwaltet und konfiguriert werden, also die Netzwerk Controller mit den SDN-Komponenten wie SLB-MUXe und Gateways, die Hyper-V Hosts, aber auch weitere Infrastruktursysteme wie Active Directory, DNS und der SCVMM. Der Netzwerk Controller erhält und verteilt über dieses Netz auch die Anweisungen für die SDN-Komponenten.
- HNVPA – Über dieses Netz laufen die mit dem VXLAN-Protokoll verpackten Netzwerkpakete der Tenant VMs zwischen den Hyper-V Hosts.
- Transit – Dieses Netz wird verwendet für die Kommunikation zwischen den SLB-MUX Systemen und der Außenwelt, z.B. mit einem BGP-Router, der mit dem Internet verbunden ist.

Diese 3 logischen Netze bilden das Rückgrat von SDN. Sie müssen in allen beteiligten Hyper-V Hosts verfügbar und dort mit den virtuellen Switches verbunden sein. Diese Netze besitzen jeweils einen eigenen IP Adresspool. Zusätzlich können sie optional durch VLANs untereinander isoliert sein, was jedoch kein Muss ist (in der später noch zu beschreibenden Demoumgebung werde ich darauf verzichten).

Für spezielle Szenarien sind gegebenenfalls weitere logische Netze notwendig, die dann aber nicht in den Hyper-V Hosts verfügbar sein müssen:

- PublicVIP – Dieses Netz stellt das Tor zum und vom Internet dar. Bei den verwendeten IP-Adressen wird es sich also in der Praxis um öffentliche Adressen handeln und es gibt auch keine Isolation mit VLAN IDs.
- PrivateVIP – Dieses Netz enthält private Adressen, über die interne Clients wie z.B. die SLB Manager und andere interne Dienste miteinander kommunizieren können.
- GREVIP – Dieses Netz enthält virtuelle IP Adressen für Endpunkte von Site-to-Site (S2S) Verbindungen mit externen (Tenant) Netzen, bei denen das GRE-Protokoll verwendet wird.

Neben diesen SDN-spezifischen logischen Netzen wird es in einem Rechenzentrum weitere logische Netze z.B. für die Storage Anbindung geben, die wir hier aber nicht weiter betrachten wollen.

4.2 Hyper-V Hardware

Für den Aufbau einer SDN-Umgebung benötigen wir eine Reihe von Hyper-V Hosts mit der Windows Server 2016 Datacenter Edition. Es sollten wenigstens 3-4 Systeme verfügbar sein, die idealerweise in einem Hyper-V Failover Cluster zusammengefasst sind. In einer produktiven Umgebung wird man natürlich deutlich mehr Systeme einsetzen.

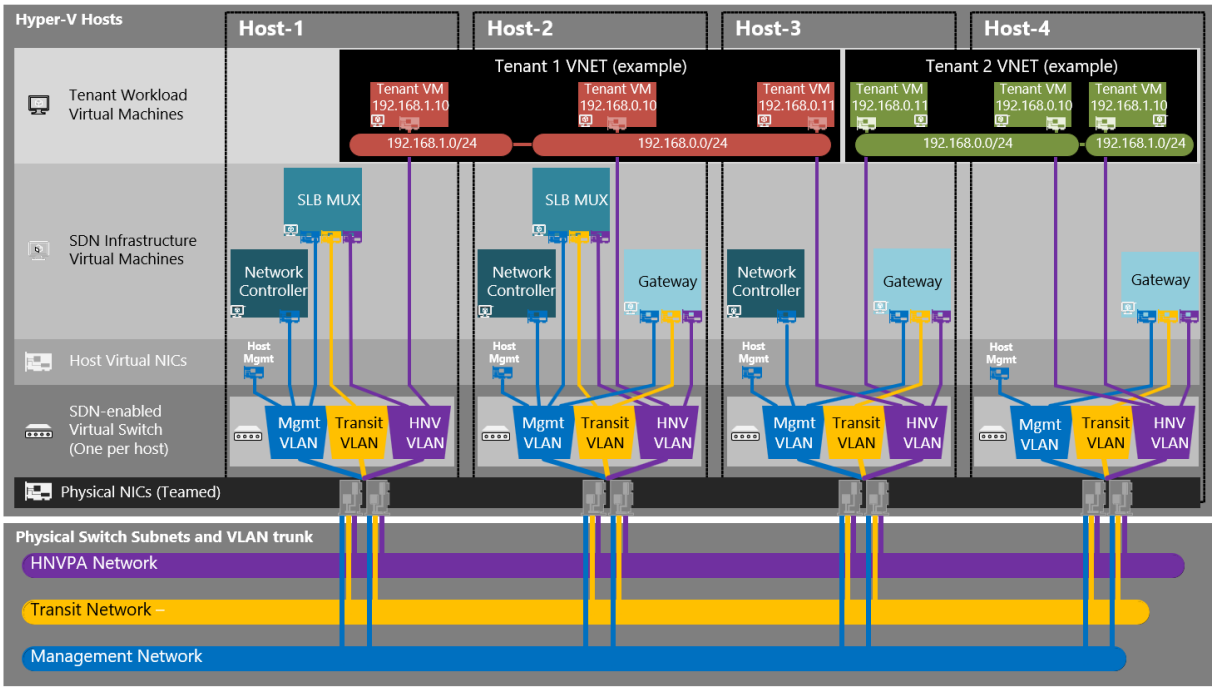
4.3 Netzwerk Hardware

Eigentlich stellt SDN keine speziellen Anforderungen an die Netzwerkhardware in den Hyper-V Hosts. Jedoch muss man berücksichtigen, dass der gesamte SDN-Netzwerkverkehr über die virtuellen Switches in den Hyper-V Hosts läuft und deshalb eine hohe Bandbreite und Verfügbarkeit erfordert. Deshalb sollte man über Teaming der in den Hyper-V Hosts verbauten Netzwerkadapter als Option nachdenken, was aber nicht zwingend ist.

Für Teaming kann sowohl das „klassische“ Teaming auf Basis des Windows Betriebssystems als auch das „Switch Embedded Teaming“ (SET) im Hyper-V Switch von Server 2016 zum Einsatz kommen. Letzteres ist natürlich vorzuziehen, wenn dies mit den vorhandenen Netzwerkadaptern möglich ist.

4.4 Putting It All Together

Wenn wir alle vorstehend beschriebenen Komponenten in einer SDN-Umgebung zusammenfassen, dann ergibt sich folgendes Bild für die SDN-Topologie:



Quelle: https://docs.microsoft.com/de-de/windows-server/networking/sdn/plan/plan-a-software-defined-network-infrastructure#network-controller-software-load-balancer-and-ras-gateway-deployment

Die unterste Ebene zeigt das physikalische Netz und die darin eingebetteten logischen SDN-Netze. Mit diesem Netz sind die Netzwerkadapter aller Hyper-V Hosts verbunden, gegebenenfalls auch als Team. Wenn Sie die logischen SDN-Netze als VLANs definiert haben, ist es wichtig, dass für die Ports an den physischen Switches, mit denen die Hyper-V Hosts verbunden sind, der Trunk Mode eingestellt ist. Nur dann sind in den virtuellen Hyper-V Switches alle SDN-Netze verfügbar.

Die virtuellen Hyper-V Switches präsentieren die SDN-Netze den im Host laufenden Infrastruktur VMs wie Netzwerk Controller, SLB-MUXe oder Gateways, so dass sie sich mit virtuellen Netzwerkadaptern damit verbinden können.

Die virtuellen Netzwerkadapter von Tenant VMs sind im Hyper-V Switch mit dem HNVPA Netz verbunden. Das Ver- und Entpacken der Tenant Daten mit dem VXLAN-Protokoll erfolgt dann im virtuellen Hyper-V Switch.

4.5 Weitere Infrastruktursysteme

Außer den Hyper-V Compute Systemen benötigen wir noch einige weitere Infrastruktursysteme:

- Active Directory mit DNS
- Ein System mit dem System Center Virtual Machine Manager (SCVMM) 2016 für das Management der SDN-Umgebung
- Einen BGP-Router, der uns die Verbindung zum öffentlichen Netz bereitstellt.

Auf diese Systeme werde ich später nochmals zurückkommen, wenn wir unsere Lab-Umgebung aufbauen.

5 SDN Deployment

Will man nun eine SDN-Umgebung aufbauen, bietet Microsoft verschiedene Methoden dafür:

- Manuelle Installation mit PowerShell

  Diese Methode ist sehr mühsam, kann aber sinnvoll sein, wenn man nur einzelne SDN-Technologien einsetzen will. Details siehe Deploy Software Defined Network Technologies using Windows PowerShell in der Technet Library.

- Installation mit Hilfe vorgefertigter PowerShell Skripte

  Diese Methode empfiehlt sich, wenn man eine SDN-Umgebung ohne den System Center Virtual Machine Manager (SCVMM) aufbauen und betreiben will, weil man andere Management Werkzeuge einsetzen will. Details siehe Deploy a Software Defined Network infrastructure using scripts in der Technet Library.

- Installation über die Netzwerkfabrik des System Center Virtual Machine Manager 2016 (SCVMM 2016)

  Dies ist meiner Meinung nach aktuell die bequemste und übersichtlichste Methode zur Installation einer SDN-Umgebung, insbesondere wenn der SCVMM 2016 sowieso als Management Werkzeug für die virtuelle Welt eingesetzt wird. Wir haben dabei sowohl die Möglichkeit der manuellen Installation über die VMM Konsole (Details siehe Set up a Software Defined Network (SDN) infrastructure in the VMM fabric in der Technet Library) als auch die automatisierte Installation mit PowerShell Skripten, die von Microsoft auf Github bereitgestellt werden.

Ich werde im Folgenden auf die automatisierte SCVMM Installation mit PowerShell Skripten näher eingehen und damit auch eine Lab-Umgebung aufbauen.

6 Die SDN Lab Umgebung

Legen wir also los mit dem Aufbau einer SDN Lab-Umgebung. Klären wir zunächst mal die Voraussetzungen.

6.1 Hardware

In meinem Testlabor habe ich nicht die notwendige Anzahl an Hardware-Systemen zur Verfügung, um eine vollständige SDN-Umgebung auszurollen. Ich werde deshalb das Ganze auf einem einzigen Hyper-V Host aufbauen und mit Nested Virtualisierung arbeiten.

Dieser Hyper-V sollte mindestens 128 GB RAM haben, für Windows Server 2016 Datacenter geeignet bzw. zertifiziert sein und natürlich Hyper-V unterstützen. Zur Not tut es auch ein System mit Windows 10 Pro (Laptop), das aber mindestens 64 GB RAM besitzen sollte. Falls Sie kein solches physisches System zur Verfügung haben, können Sie das Ganze auch in einer einzigen virtuellen Maschine umsetzen, die Sie z.B. in der Microsoft Azure Cloud oder bei einem anderen Service Provider anmieten.

6.2 Software

Sie benötigen folgende Softwarekomponenten für die Installation:

- Windows Server 2016 Data Center
- System Center Virtual Machine Manager (SCVMM) 2016
  einschließlich SQL Server 2014 und Windows Assessment and Deployment Kit
- SCVMM Service Templates und Powershell Skripte für das SDN Deployment

Da wir hier nur eine Testumgebung aufbauen wollen, genügen die Evaluierungs-Versionen der jeweiligen Komponenten.

Auf das Downloaden und die Verwendung der SCVMM Service Templates und PowerShell Skripte werde ich später noch genauer eingehen.

6.3 Die Lab Infrastruktur

Für die SDN-Umgebung benötigen wir einige virtuelle Maschinen sowie die Parameter der logischen SDN-Netzwerke.

6.3.1 Virtuelle Maschinen

Bei der Infrastruktur der SDN-Umgebung beschränke ich mich auf ein Minimum an virtuellen Maschinen:

- Domain Controller SDN-DC01

  Dieses System dient als Active Directory Controller mit integriertem DNS. Darüber hinaus stellt es über File Shares den Storage für unser Hyper-V Failover Cluster bereit.

- BGP-Router SDN-BGP

  Dieses System stellt den BGP-Router für unsere Lab-Umgebung dar. Es verwendet dazu die Windows Serverrolle *Remote Access*.

- SDN-VMM01

  Auf diesem System ist der System Center Virtual Machine Manager 2016 einschließlich der dafür notwendigen Basiskomponenten wie SQL-Server und Windows Assessment and Deployment Kit installiert. Von diesem System werden wir auch das Deployment der SDN-Umgebung steuern.

- SDN-HV01, SDN-HV02, SDN-HV03 und SDN-HV04

  Diese 4 Systeme bilden ein Hyper-V Failover Cluster, in das wir die virtuellen SDN-Systeme wie Network Controller, SLB-MUXs und Gateways sowie auch beispielhaft einige Tenant VMs ausrollen werden.

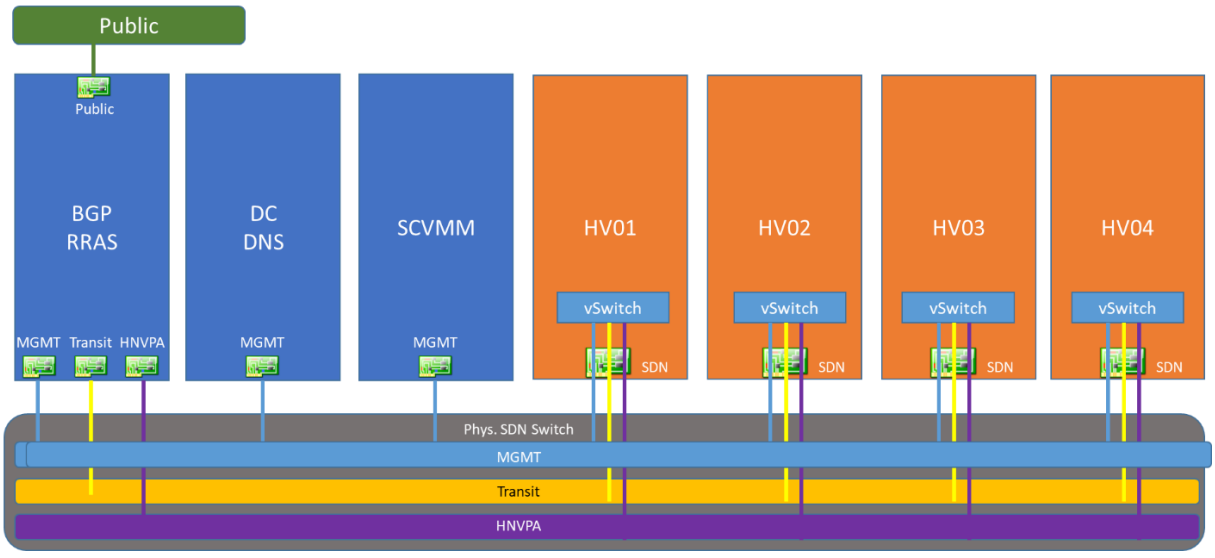
6.3.2 Logische SDN Netze

Zum Ausrollen der SDN-Umgebung müssen wir auch die Parameter der logischen SDN-Netze festlegen. Die nachstehende Tabelle zeigt die Parameter, wie wir sie später im VMM verwenden werden.

Log. Netz	Subnet	Netzmaske	VLAN (optional)	Gateway	DNS
NC_Management	192.168.80.0	255.255.255.0	0	192.168.80.1	192.168.80.10
	Verwendeter IP-Bereich: 192.168.80.230 – 192.168.80.254 Reserviert für andere Zwecke: 192.168.80.230 (IP-Adresse für die REST API)				
HNVPA	10.10.10.0	255.255.255.0	0	10.10.10.1	192.168.80.10
	Verwendeter IP-Bereich: 10.10.10.100 – 10.10.10.199				
Transit	10.10.20.0	255.255.255.0	0	10.10.20.1	192.168.80.10
	Verwendeter IP-Bereich: 10.10.20.100 – 10.10.20.199				
PrivateVIP	10.10.30.0	255.255.255.0	0	10.10.30.1	192.168.80.10
	Verwendeter IP-Bereich: 10.10.30.100 – 10.10.30.199 Reserviert für Load Balancer VIPs: 10.10.30.100 – 10.10.30.199				
GREVIP	10.10.40.0	255.255.255.0	0	10.10.40.1	192.168.80.10
	Verwendeter IP-Bereich: 10.10.40.100 – 10.10.40.199 Reserviert für Load Balancer VIPs: 10.10.40.100 – 10.10.40.199				
PublicVIP	10.10.50.0	255.255.255.0	0	10.10.50.1	192.168.80.10
	Verwendeter IP-Bereich: 10.10.50.100 – 10.10.50.199 Reserviert für Load Balancer VIPs: 10.10.50.100 – 10.10.50.199				

6.3.3 Ein grober Überblick

Die Infrastruktur der Lab-Umgebung hat also folgendes Aussehen:



6.4 Vorbereitungen

1. Installieren Sie auf dem physischen Hyper-V Host die Datacenter Edition von Windows Server 2016 und installieren Sie die Hyper-V Rolle. Für unsere Tests ist keine Windows Aktivierung erforderlich; ersatzweise können Sie auch Windows 10 Pro verwenden.
2. Erstellen Sie im Hyper-V Host die 3 folgenden virtuellen Switche – entweder über die GUI im Hyper-V Manager oder per PowerShell:

```
New-VMSwitch -Name "SDN" -SwitchType Internal
New-VMSwitch -Name "Public" -SwitchType Internal
New-VMSwitch -Name "Private" -SwitchType Internal
```

3. Erzeugen Sie Windows Server 2016 VHDX-Dateien für die verschiedenen zu erzeugenden VMs, z.B. mit dem PowerShell Skript *Convert-WindowsImage.ps1*, das Sie im Verzeichnis *<ISO>/NanoServer/NanoServerImageGenerator* der heruntergeladenen ISO Datei des Windows Server 2016 finden.

```
cd '<ISO>:\nanoserver\NanoServerImageGenerator'

. .\Convert-WindowsImage.ps1

# Erzeuge dyn. VHDX für Gen2 Hyper-V VM
# mit Windows Server 2016 Datacenter (= Edition4)
Convert-WindowsImage `
 -SourcePath '<ISO>:\sources\install.wim' `
 -Edition 4 `
 -VHDPath 'E:\Hyper-V\WS2016.vhdx' `
 -SizeBytes 80GB `
 -VHDFormat VHDX `
 -DiskLayout UEFI
```

4. Erzeugen Sie auf dem physischen Hyper-V Host nun folgende virtuelle Gen 2 Maschinen. Als Festplatte können Sie jeweils die im vorstehenden Schritt erzeugte VHDX-Datei verwenden. Statten Sie die VMs mit den angegebenen Netzwerkadaptern und gegebenenfalls mit den zusätzlichen Datenlaufwerken aus. Verbinden Sie die Netzwerkadapter mit dem jeweils angegebenen virtuellen Switch.

6.4.1 SDN-DC01

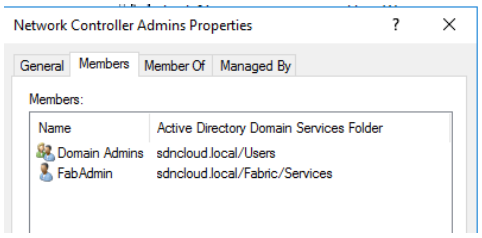
vCPU	2
Memory	4 GB
Netzwerkadapter	NC_Management Verbunden mit internen Switch SDN IP-Adresse 192.168.80.10 Gateway 192.168.80.1 DNS 192.168.80.10
Zusätzliches Laufwerk	1 virt. Festplatte, dynamisch erweiterbar Größe: >= 512 GB Shares: • *Hyper-V* – Storage für die HV01 – HV04 • *Cluster-Witness* – Synchronisationsdaten für das Hyper-V Cluster

Installieren und konfigurieren Sie folgende Serverrollen und Features einschließlich der jeweiligen Management Tools:

- Active Directory
  - Domainname: *SDNcloud.local*

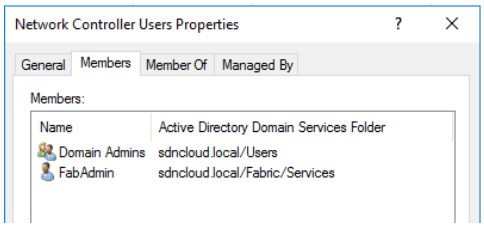
- Benutzerkonten:
  - *FabAdmin* – Fabric Admin: Klon des Benutzerkontos *Administrator* (optional). Ich arbeite normalerweise nur mit diesem Domain User. Damit kann keine Verwechslung mit dem lokalen *Administrator*-Konto entstehen beim Login an einem Member Server. Sie sollten dieses Konto jedoch auf allen Member Systemen in die lokale Gruppe *Administrators* aufnehmen, um dort auch alle Aufgaben durchführen zu können.
  - *VMM*-SVC – Service Account für den VMM Service
- Lokale Security Gruppen
  - *Network Controller Admins*

    Mitglieder dieser Sicherheitsgruppe dürfen die Netzwerk Controller Konfiguration verwalten (Kreieren, Löschen und Updaten von NC Komponenten). Für unsere Lab-Umgebung verwende ich die Standardgruppe *Domain Admins* und setze zusätzlich das oben beschriebene Benutzerkonto *FabAdmin* ein.



  - *Network Controller Users*

    Mitglieder dieser Sicherheitsgruppe dürfen nach dem NC-Deployment mit dem Netzwerk Controller über dessen REST-API kommunizieren. Für unsere Lab-Umgebung verwende ich die Standardgruppe Domain Admins und setze zusätzlich das oben beschriebene Benutzerkonto *FabAdmin* ein.



- DNS
  - Active Directory integriert
  - definieren Sie ggf. DNS-Forwarders, um Internet Namen auflösen zu können (optional; für einige spätere Testszenarien sinnvoll)

Stellen Sie sicher, dass Sie alle aktuellen Updates und Patches installiert haben.

6.4.2 SDN-BGP01

vCPU	2	
Memory	4 GB	
Netzwerkadapter	NC_Management	Verbunden mit internen Switch SDN
	IP-Adresse	192.168.80.50
	Gateway	192.168.80.1
	DNS	192.168.80.10
	HNVPA	Verbunden mit internen Switch SDN
	IP-Adresse	10.10.10.1
	Transit	Verbunden mit internen Switch SDN
	IP-Adresse	10.10.20.1
	Public	Verbunden mit internen Switch Public
	IP-Adresse	10.10.50.1

Joinen Sie das System als Member Server ins Active Directory.

Installieren und konfigurieren Sie folgende Serverrollen und Features einschließlich der jeweiligen Management Tools:

Remote Access

Es genügen die Rollen *Direct Access and VPN* sowie *Routing*

Aktivieren Sie die LAN-Router Funktion oder optional NAT (in meiner Lab-Umgebung arbeite ich mit NAT, da ich über das logische Netz *NC_Management* Internetzugriff habe).

Eine weitere Konfiguration ist zunächst nicht notwendig; sie erfolgt später.

Stellen Sie sicher, dass Sie alle aktuellen Updates und Patches installiert haben.

6.4.3 SDN-HV01 – SDN-HV04

vCPU	Je 4	
Memory	Je 16 GB – dyn. Memory darf nicht aktiviert sein!	
Netzwerkadapter	NC_Management	Verbunden mit internen Switch SDN
	IP-Adressen	SDN-HV01: 192.168.80.21
		SDN-HV02: 192.168.80.22
		SDN-HV03: 192.168.80.23
		SDN-HV04: 192.168.80.24
	Gateway	192.168.80.1
	DNS	192.168.80.10

- Joinen Sie alle 4 Systeme als Member Server ins Active Directory.

Installieren und konfigurieren Sie folgende Serverrollen und Features einschließlich der jeweiligen Management Tools auf allen 4 VMs:

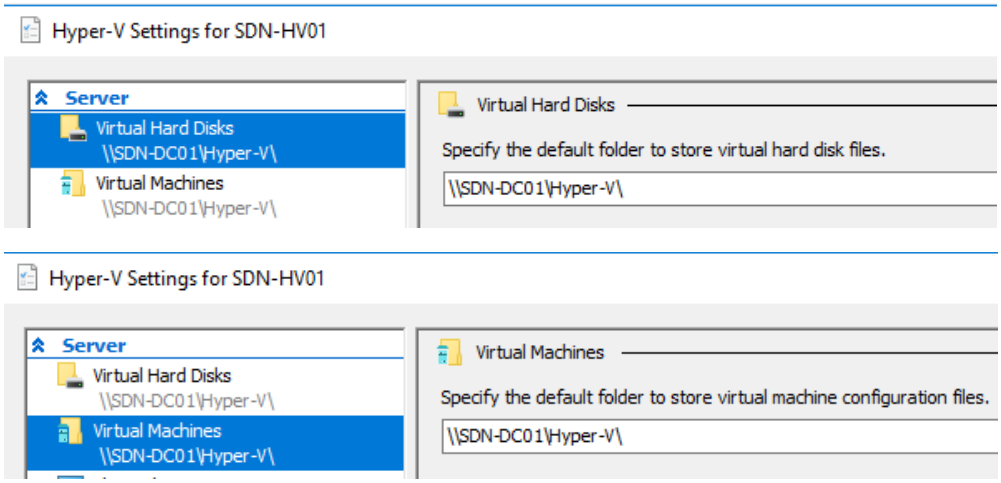
- Hyper-V

  Achtung: Aktivieren Sie Nested Virtualization und MAC Address Spoofing für alle 4 VMs mit PowerShell:

```
Get-VMProcessor -VMName SDN-HV01 | Set-VMProcessor -ExposeVirtualizationExtensions $true
Get-VMNetworkAdapter -VMName SDN-HV01 | Set-VMNetworkAdapter -MacAddressSpoofing On
```

Definieren Sie in den Hyper-V Systemen noch keinen virtuellen Switch. Wir werden dies später mit dem SCVMM erledigen.

Setzen Sie im Hyper-V Manager in den Hyper-V Einstellungen die Pfade für virtuelle Maschinen und virtuelle Hard Disks auf den Share, den Sie beim Kreieren des Systems SDN-DC01 angelegt haben.



- Failover Clustering

  Erzeugen Sie ein Hyper-V Failover Cluster mit den 4 Hyper-V Systemen. In meiner Lab-Umgebung verwende ich folgende Parameter:

  - Clustername: *SDN-HVCL*
  - Cluster IP: 192.168.80.20

Stellen Sie sicher, dass Sie alle aktuellen Updates und Patches installiert haben.

6.4.4 SDN-VMM01

vCPU	2
Memory	4 GB
Netzwerkadapter	NC_Management Verbunden mit internen Switch SDN IP-Adresse 192.168.80.31 / 24 Gateway 192.168.80.1 DNS 192.168.80.10
Zusätzliches Laufwerk	1 virt. Festplatte, dynamisch erweiterbar Größe: >= 512 GB Legen Sie Verzeichnisse für die VMM-Library (z.B. *D:\VMMLibrary*) und für die SDN Installationsskripte (z.B. *D:\Software*) an.

- Joinen Sie das System als Member Server ins Active Directory.

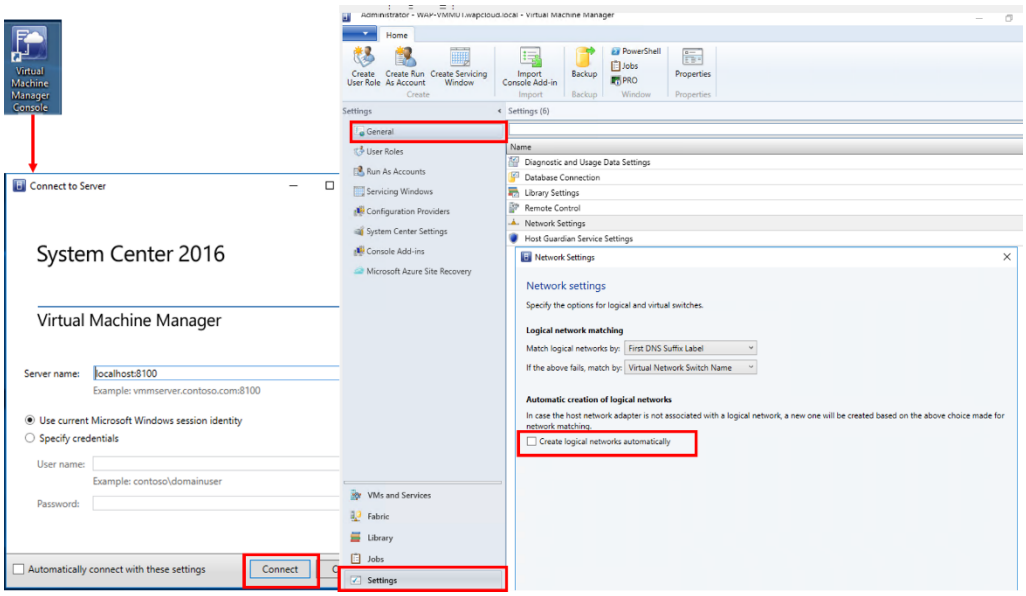
Downloaden und installieren Sie nun folgende Softwarekomponenten auf den SDN-VMM01:

- Windows Assessment and Deployment Kit
- SQL Server 2014 (oder neuer)
- System Center Virtual Machine Manager (SCVMM) 2016 einschließlich dem aktuellen Update Rollup UR5
  - Verwenden Sie als Benutzerkonto für den VMM Serverdienst den beim SDN-DC01 erstellten Service Account *VMM-SVC.*

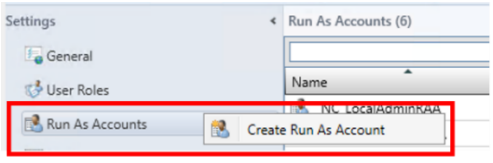
Stellen Sie sicher, dass Sie alle aktuellen Updates und Patches installiert haben.

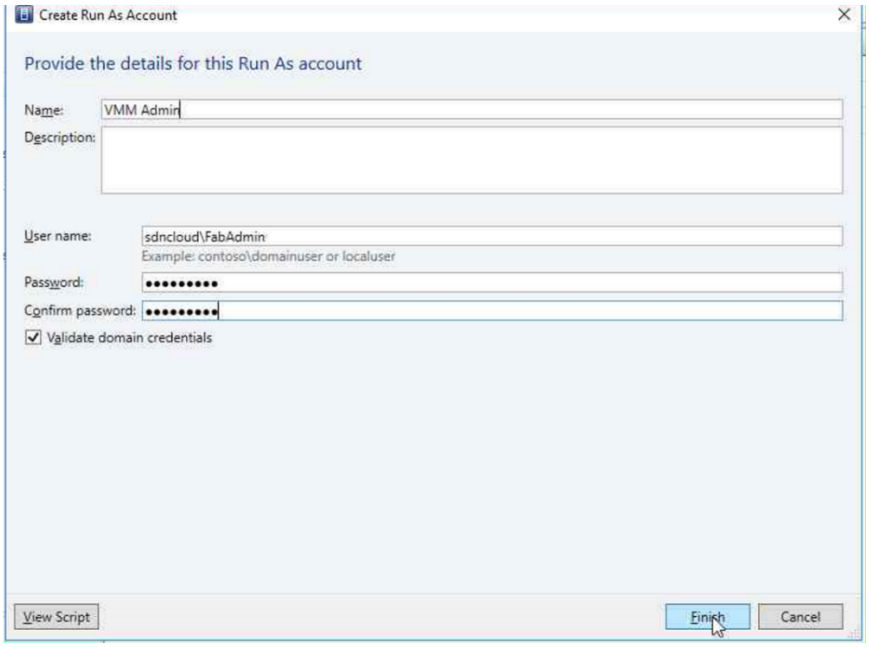
Starten Sie nach der vollständigen VMM Installation die VMM Konsole über das Icon auf dem Desktop und führen Sie folgende Basiskonfiguration durch:

- Rufen Sie im Arbeitsbereich *Settings* unter *General* die *Network Settings* auf und deaktivieren Sie das Kontrollkästchen *Create logical networks automatically*.



- Klicken Sie im Arbeitsbereich *Settings* mit der rechten Maustaste auf den Eintrag *Run As Accounts* und rufen den Befehl *Create Run As Account* auf. Im Dialogfenster geben Sie jetzt ein Benutzerkonto aus dem Active Directory einschließlich Kennwort an, das Domain Admin Rechte besitzt – im Beispiel der *FabAdmin* aus der Domain *sdncloud.*

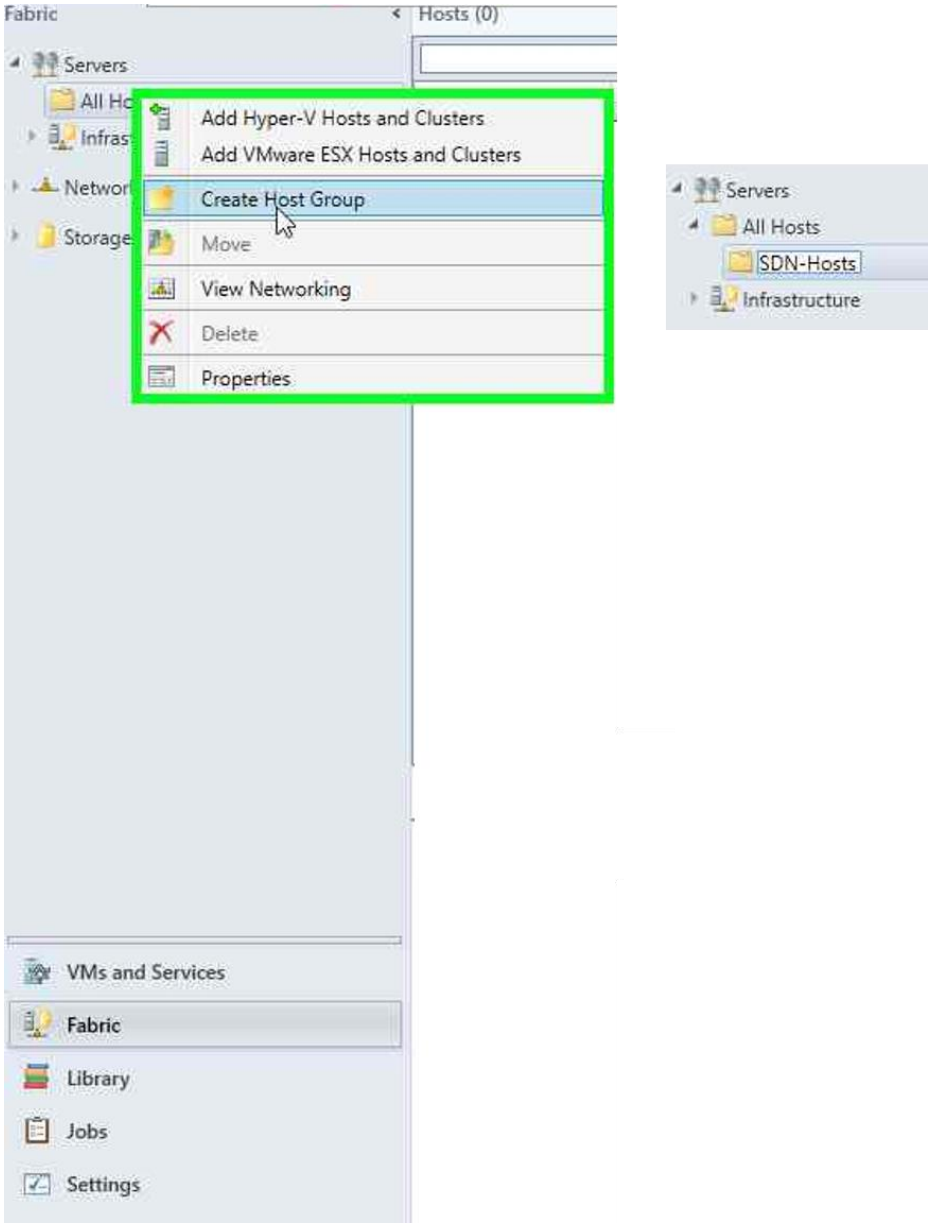




- Erstellen einer SDN-Host Gruppe im VMM

  Technisch ist dieser Schritt nicht unbedingt erforderlich. Man kann dann jedoch schnell sehen, welche Hosts zu der SDN-Umgebung gehören.

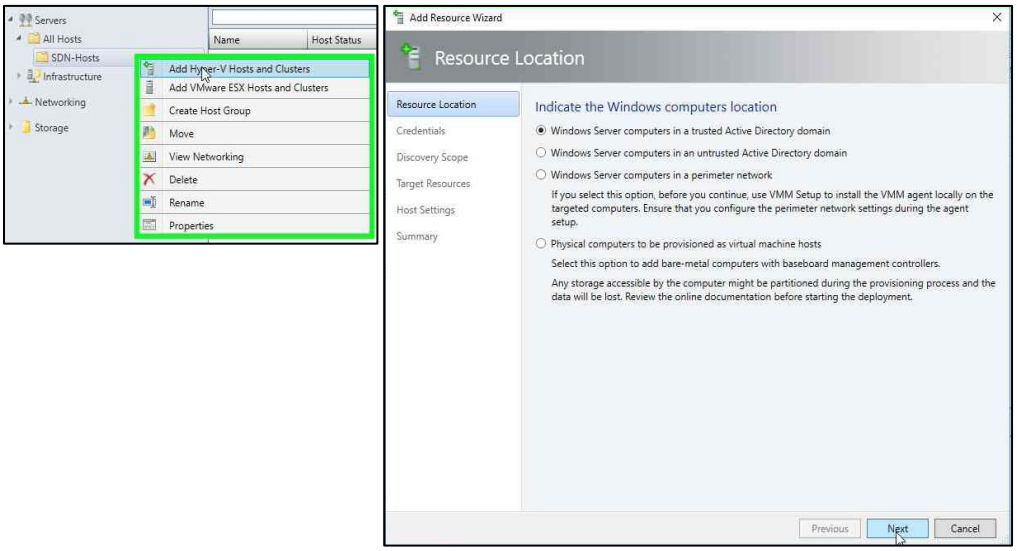
  Wechseln Sie in der VMM-Konsole in den Arbeitsbereich *Fabric,* klicken mit der rechten Maustaste unter *Servers* auf *All Hosts* und wählen Sie im Kontextmenü den Befehl *Create Host Group*. Geben Sie als Namen *SDN-Hosts* ein.



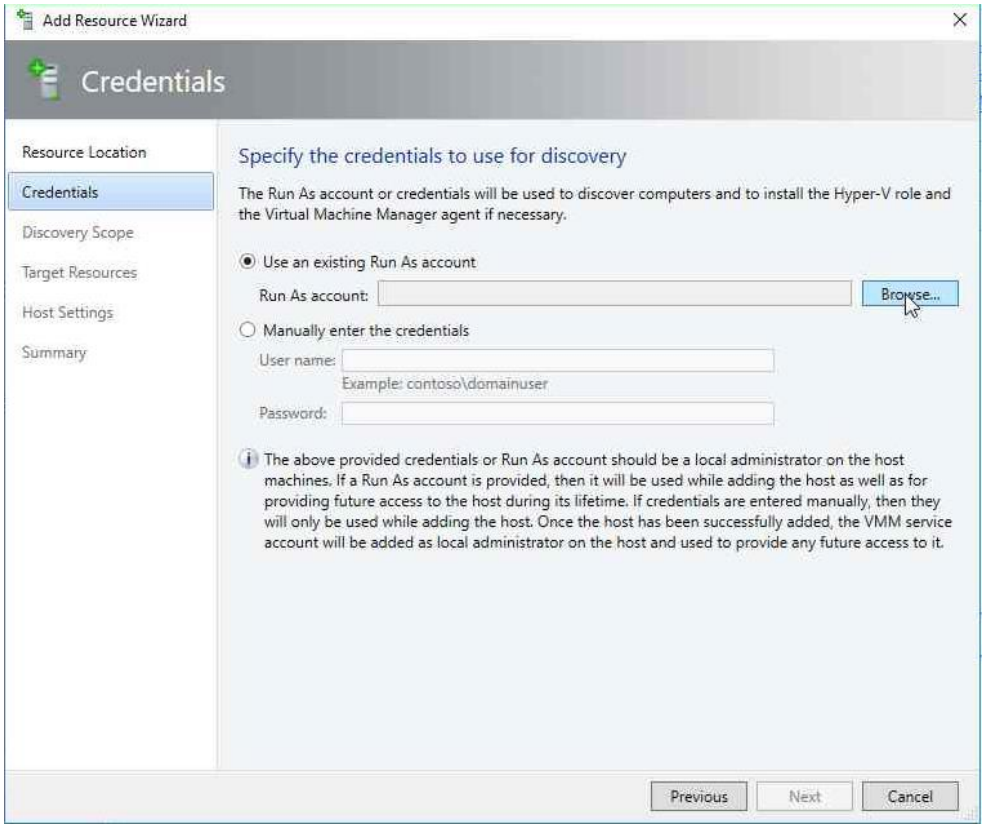
- Importieren der Hyper-V Hosts in den VMM

  Klicken Sie in der VMM-Konsole im Arbeitsbereich *Fabric* mit der rechten Maustaste auf die soeben erstellte Hostgruppe *SDN-Hosts* und wählen im Kontextmenü den Befehl *Add Hyper-V Hosts and Clusters*.

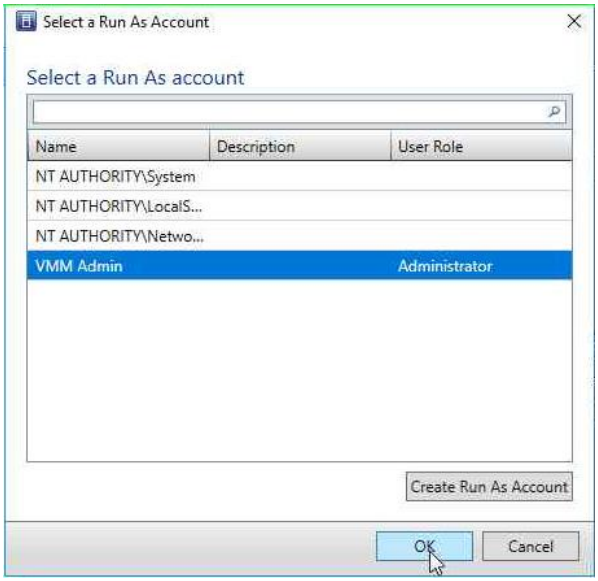
  Es öffnet sich der *Add Resource Wizard*. Wählen Sie dort die Option *Windows Server computers in a trusted Active Directory domain* und klicken Sie dann auf *Next*.



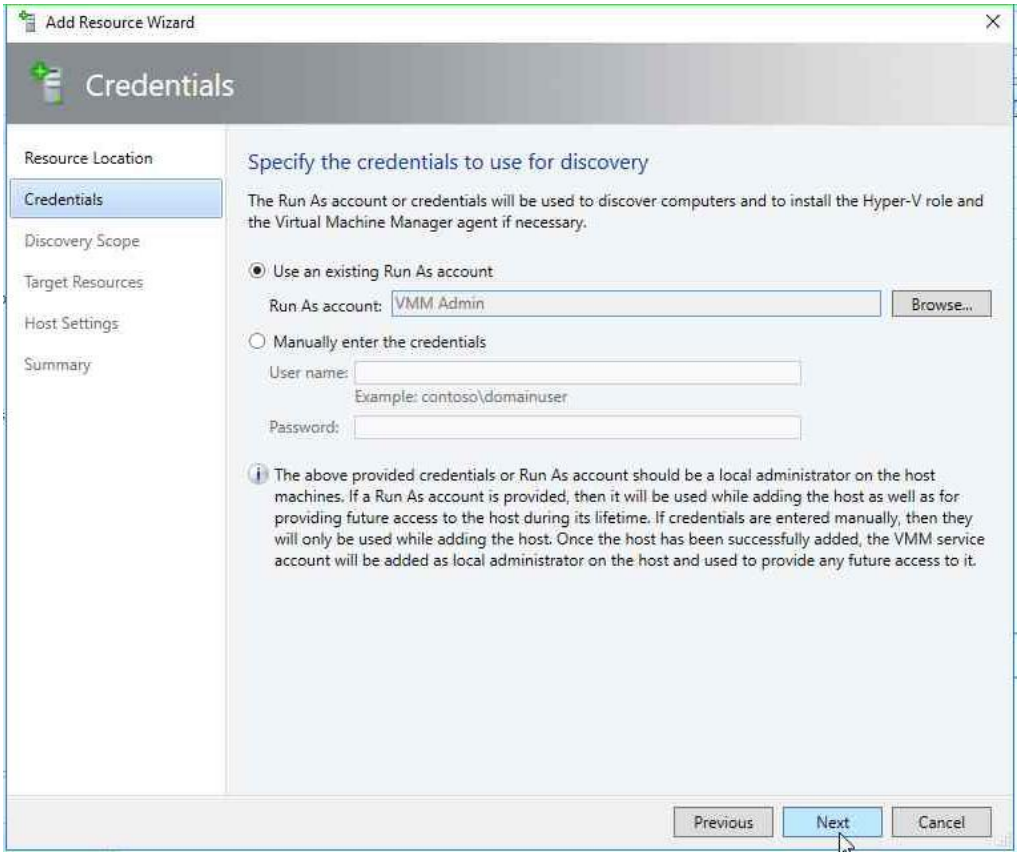
Wählen Sie den *Run As Account*, mit dem der Import durchgeführt werden soll, indem Sie auf *Browse* klicken.



Wählen Sie in der angezeigten Liste den vorhin angelegten *VMM Admin* aus und klicken Sie auf *OK*.

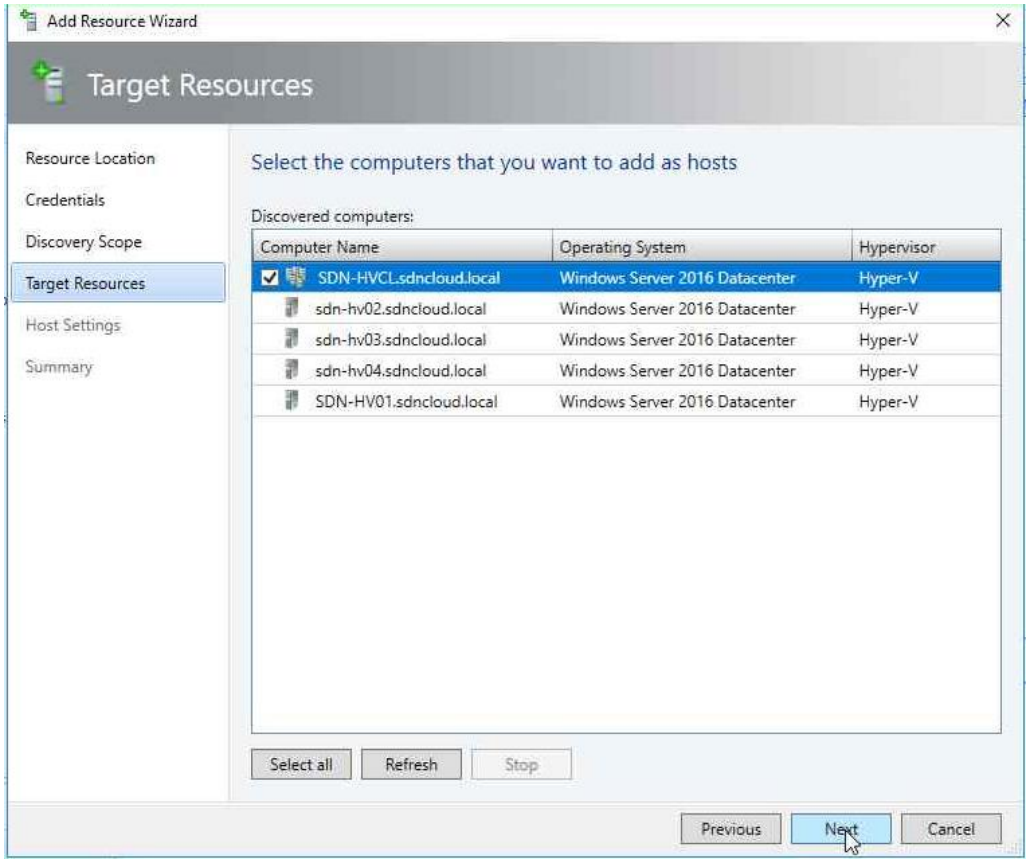


Zurück im *Add Resource Wizard* sehen Sie den ausgewählten *Run As* Account. klicken Sie auf *Next*.

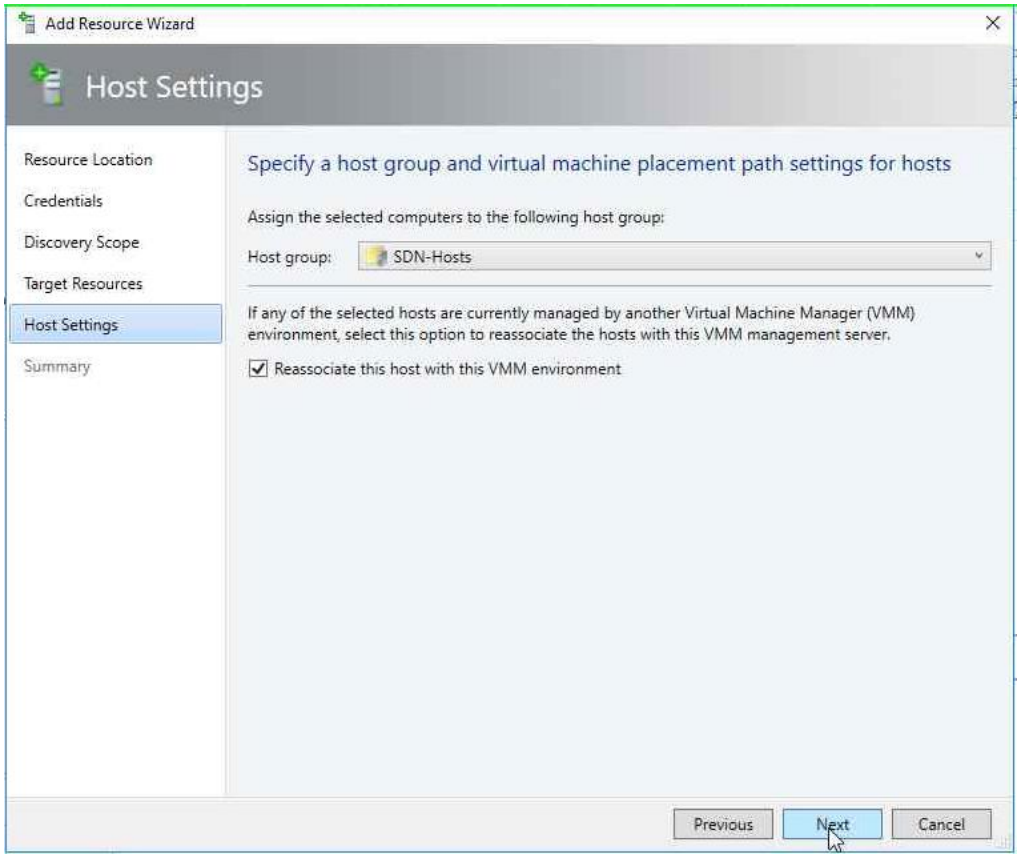


Im nächsten Fenster müssen Sie jetzt die Namen der zu importierenden Hyper-V Hosts bzw. Cluster eingeben, in unserem Fall also den Clusternamen *SDN-HVCL*. Klicken Sie dann wieder auf *Next*.

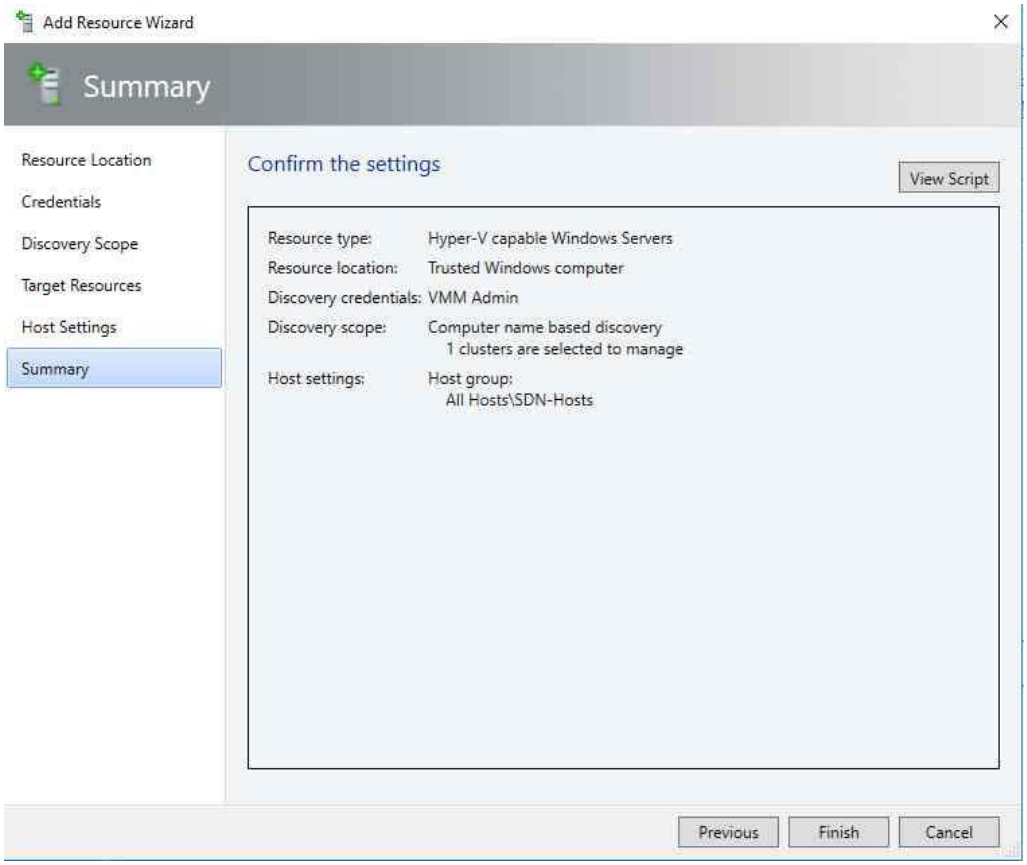
Jetzt wird Ihnen das Cluster mit den Mitgliedsservern angezeigt. Aktivieren Sie die Checkbox neben dem Clusternamen und klicken Sie auf *Next*.



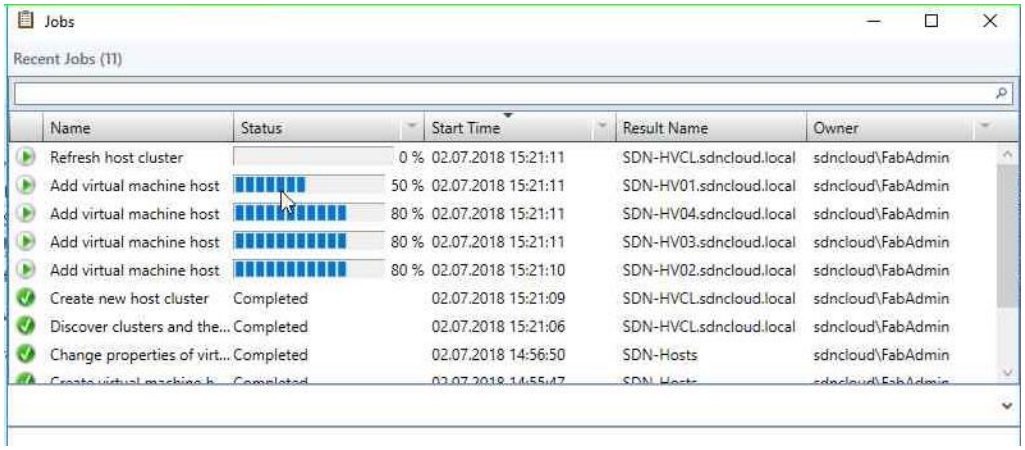
Wählen Sie jetzt die *Host Group* für den Import aus. Aktivieren Sie auch die Checkbox *Reassociate this host with this VMM environment*. Klicken Sie nochmals auf *Next*.



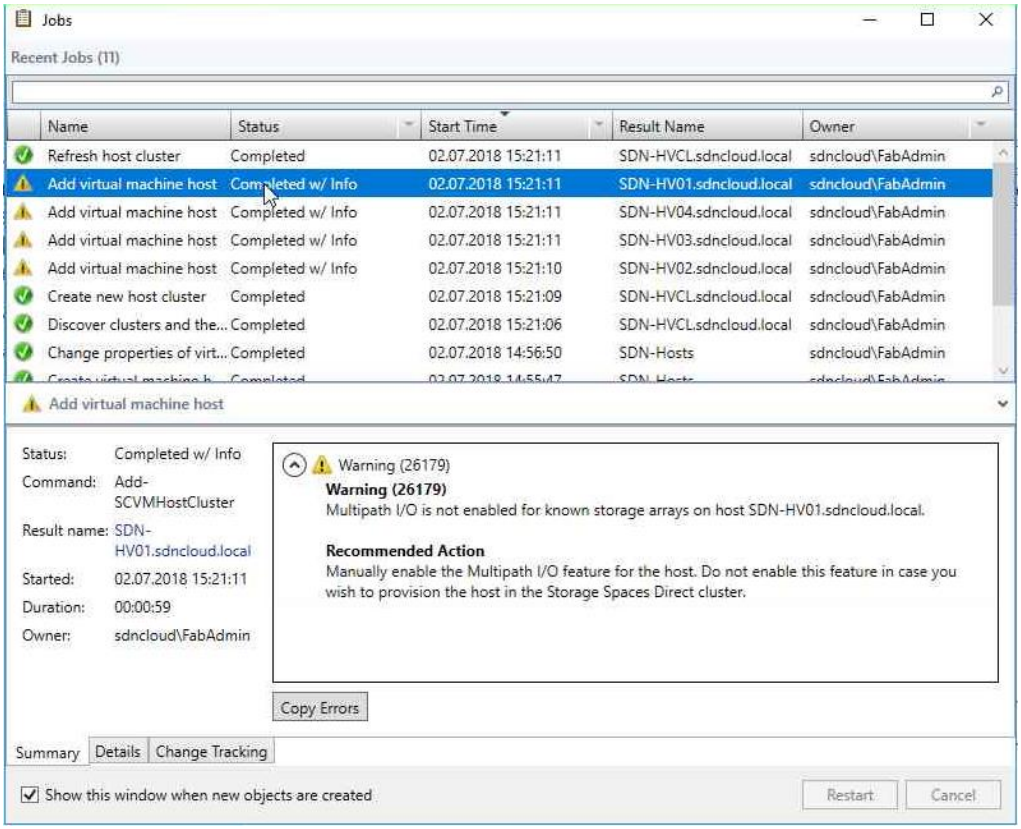
Prüfen Sie in der angezeigten Zusammenfassung nochmals Ihre Angaben und klicken Sie dann auf *Finish*.



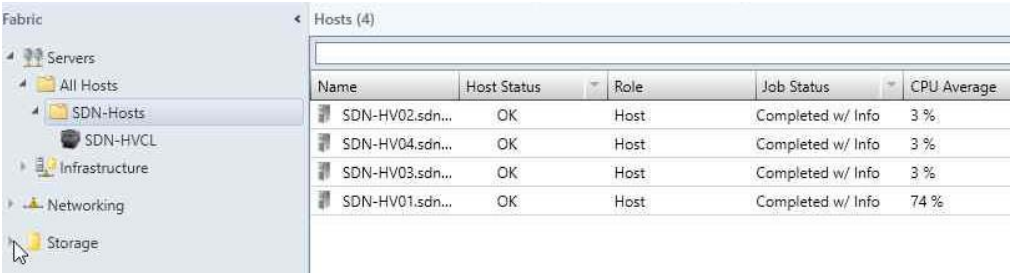
Der Importvorgang startet und Sie können den Fortschritt in der Jobliste des VMM mitverfolgen.



Warten Sie, bis alle Jobs erfolgreich abgeschlossen sind (Anzeige *Completed*). Die angezeigten Warnungen über das nicht aktivierte *Multipath IO* können wir hier ignorieren.

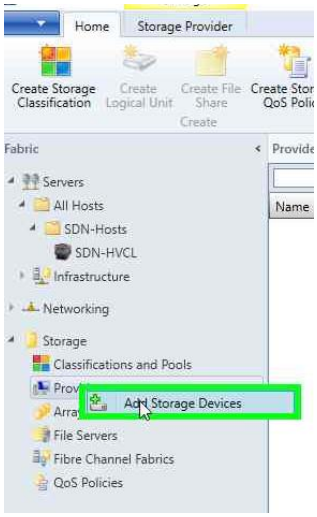


Schließen Sie das Jobfenster. In der VMM Konsole sind jetzt unser Cluster und die Hyper-V Hosts importiert.

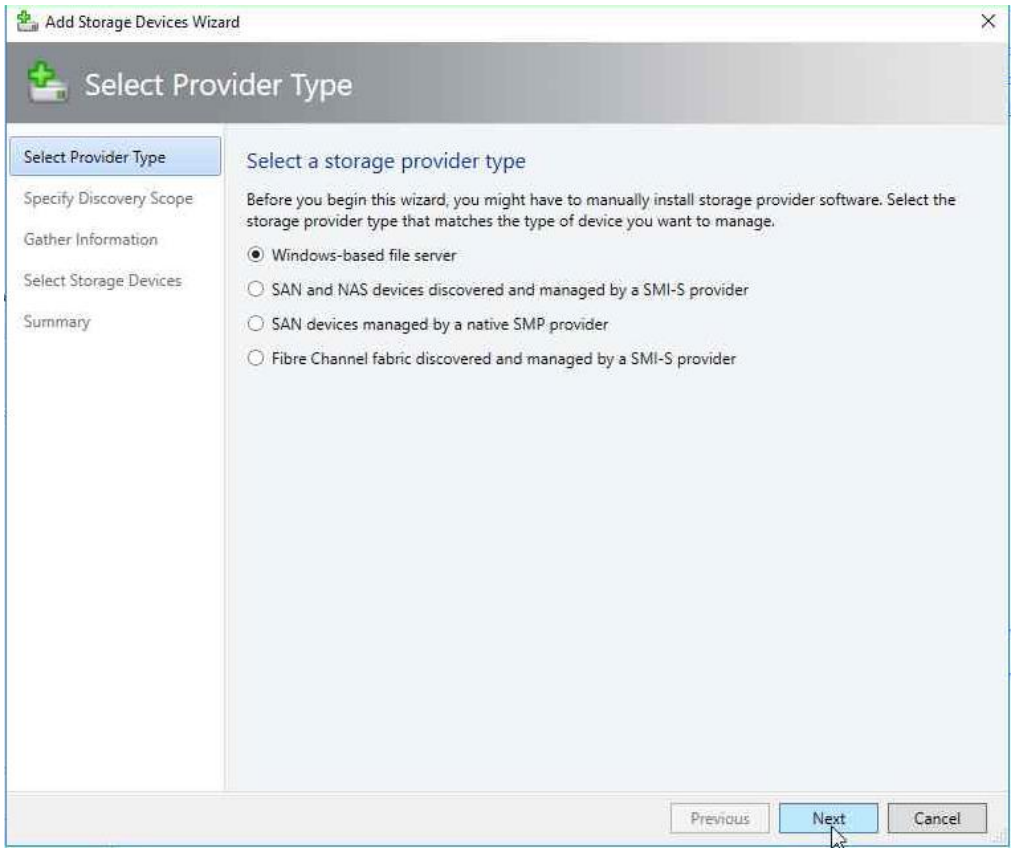


Im nächsten Schritt müssen wir dem VMM noch mitteilen, wo die Images für unsere zukünftigen virtuellen Maschinen abgelegt werden sollen.

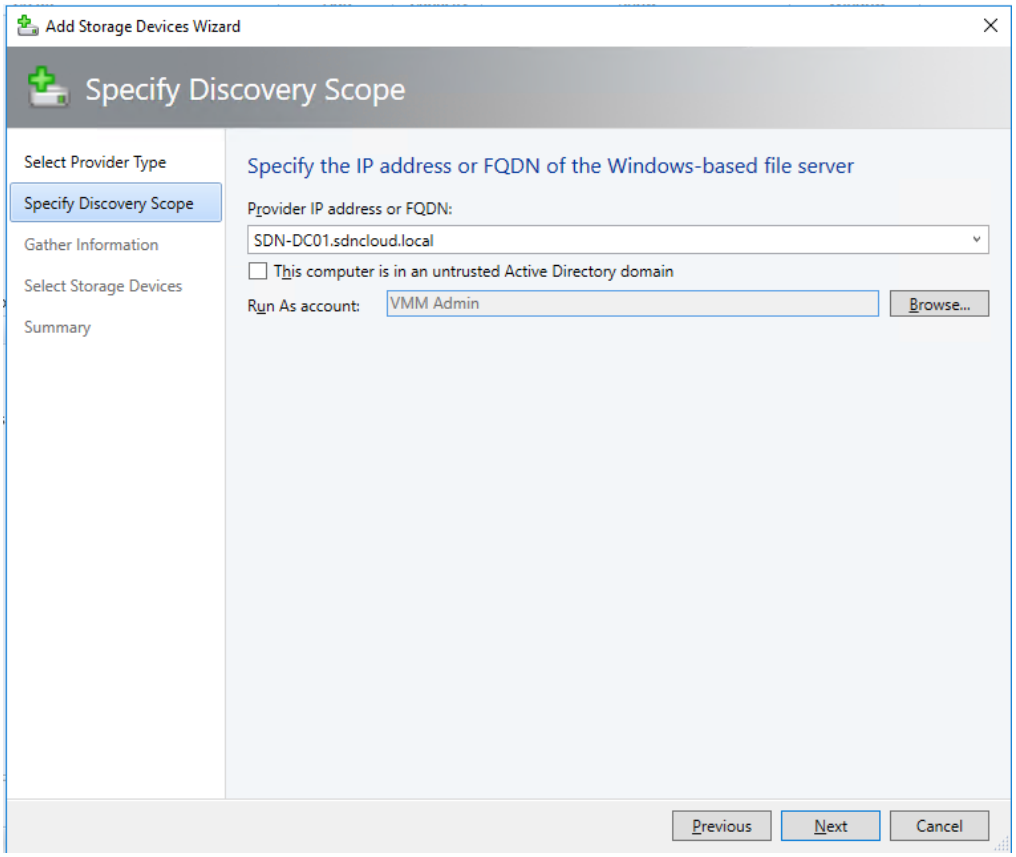
Wechseln Sie im *Fabric* Arbeitsbereich in die Kategorie *Storage*, klicken mit der rechten Maustaste auf *Providers* und wählen den Befehl *Add Storage Devices*.



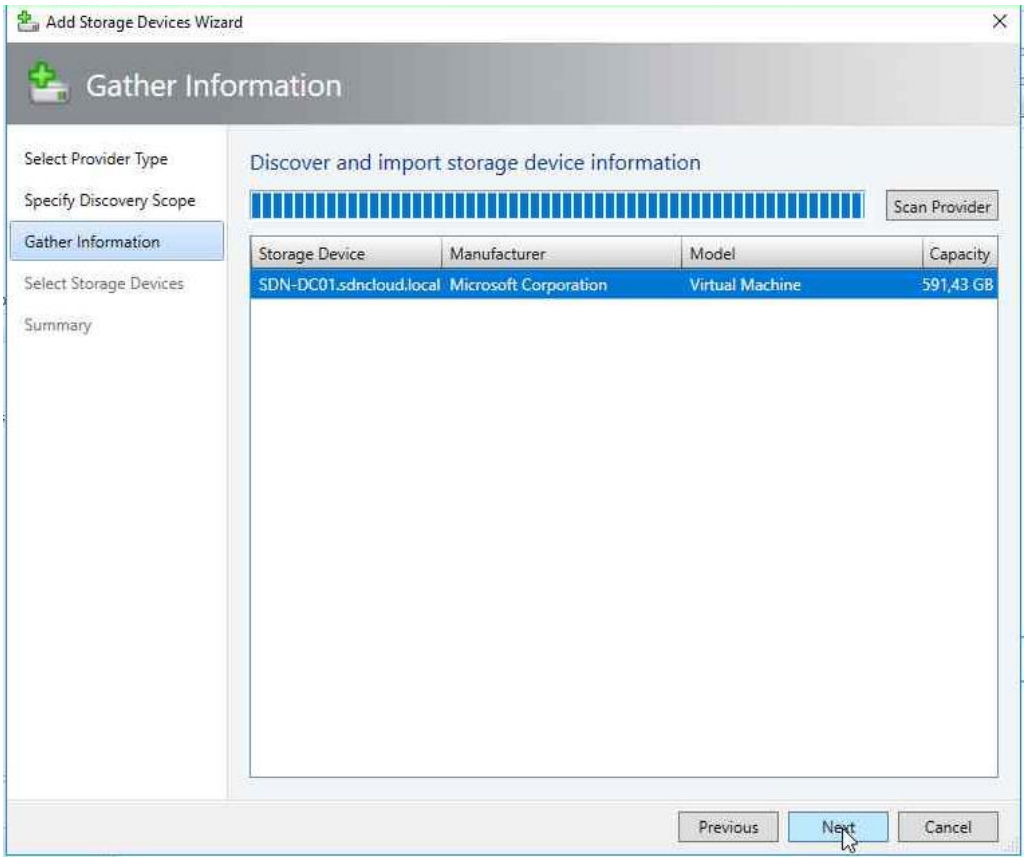
Es öffnet sich der *Add Storage Devices Wizard*. Da wir in dieser Lab-Umgebung für das Ablegen der VM-Images einen File Share mit dem Namen *Hyper-V* auf unserem Domain Controller *SDN-DC01* verwenden wollen, wählen wir als Provider Typ *Windows-based file server* und klicken auf *Next*.



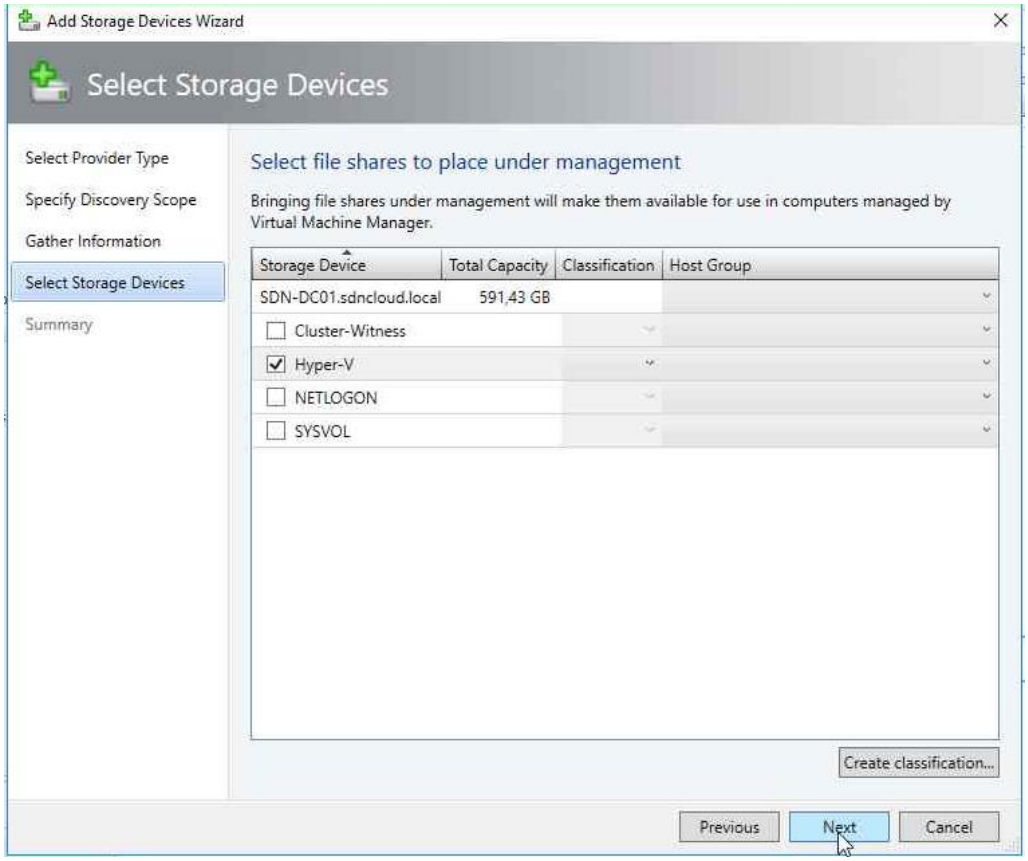
Jetzt müssen wir den Servernamen und einen *Run As account* angeben.



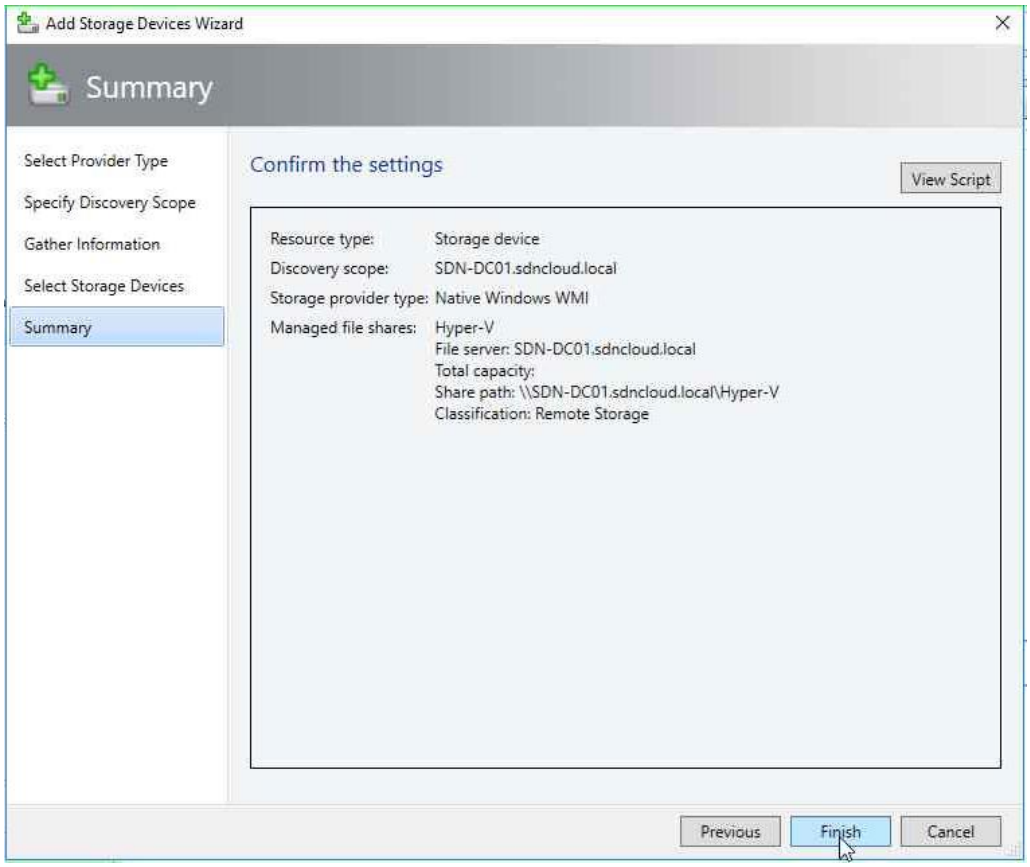
Nach einem Klick auf *Next* wird uns der *SDN*-DC01 als Storage Device angezeigt …



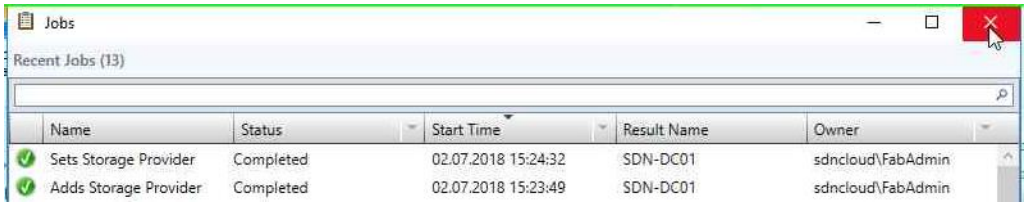
… und nach einem weiteren Klick auf *Next* die darauf verfügbaren Shares. Wählen Sie den *Hyper-V* Share aus und klicken Sie nochmals auf *Next*.



Es wird eine Zusammenfassung angezeigt. Wenn wir nun auf *Finish* klicken, wird unser File Share als Storage Device importiert.

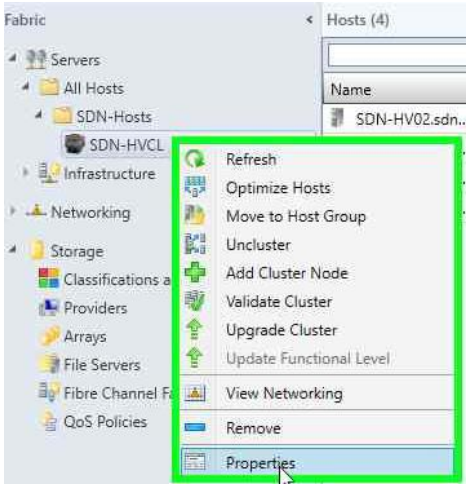


Über die Jobliste können wir den Vorgang mitverfolgen.

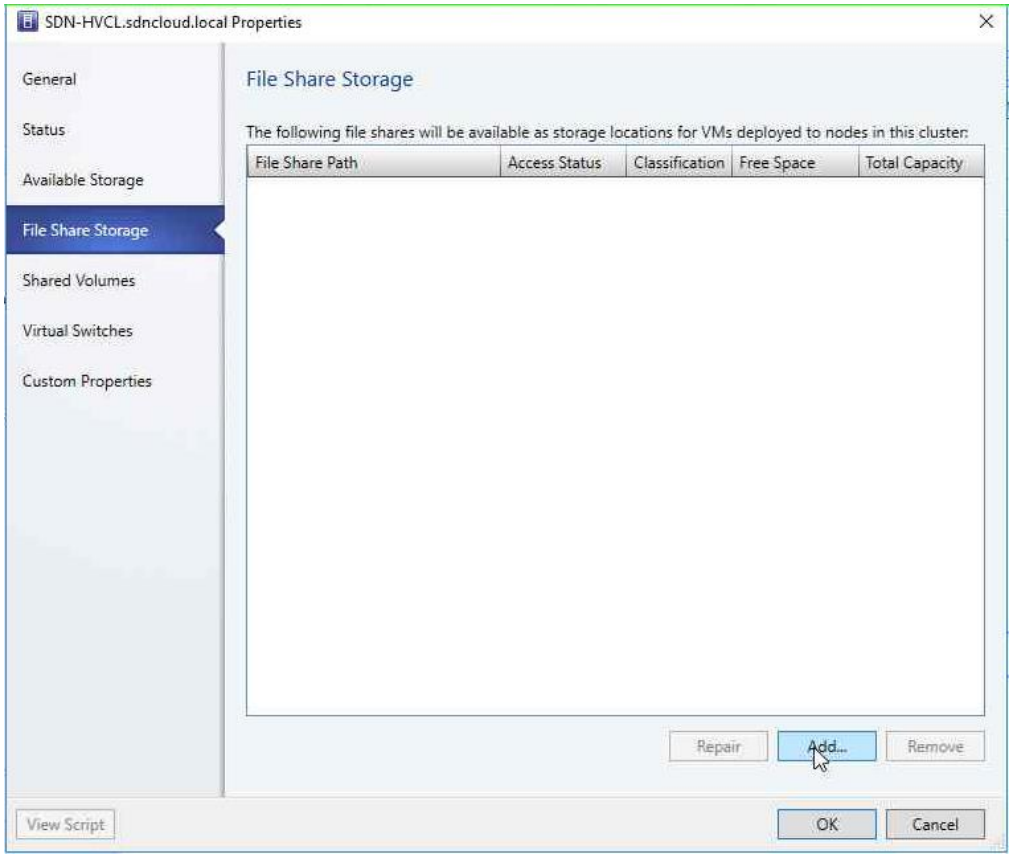


Jetzt müssen wir den gerade importierten Storage Provider noch in den *Properties* unseres Clusters registrieren, damit der VMM weiß, wo er VM Images für dieses Cluster ablegen kann.

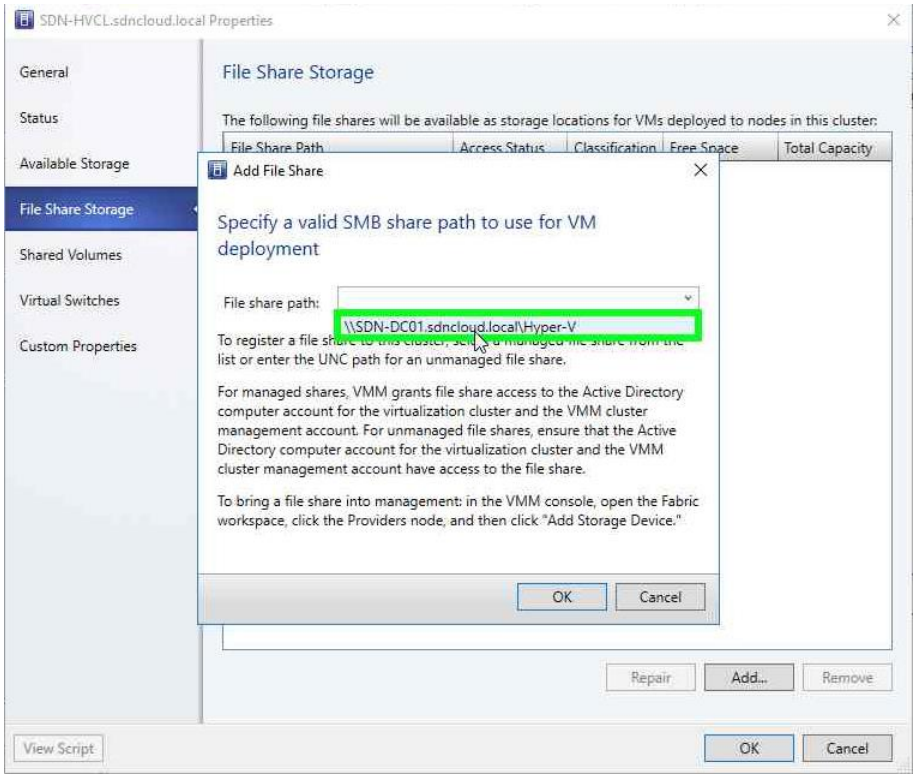
Klicken Sie mir der rechten Maustaste auf das Cluster im *Fabric* Arbeitsbereich unter der vorhin angelegten Hostgroup *SDN-Hosts* und rufen Sie im Kontextmenü die *Properties* auf.



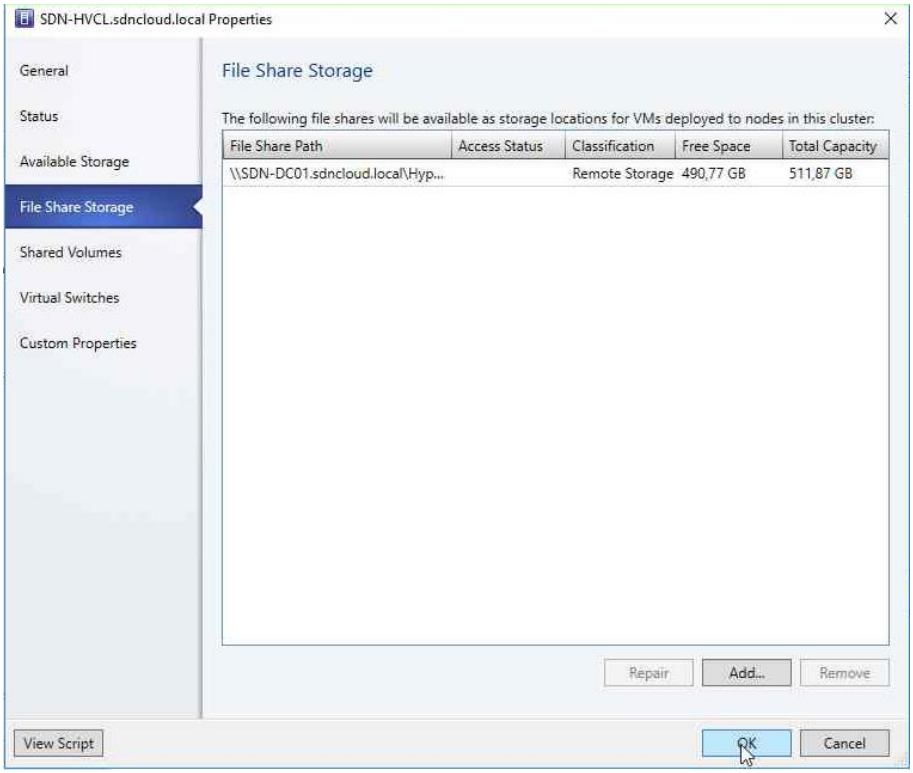
Klicken Sie auf der Registerkarte *File Share Storage* auf die Schaltfläche *Add…*



Im *Add File Share* Dialogfenster können Sie nun den vorher als Storage Provider definierten Pfad *\\SDN-DC01.sdncloud.local\Hyper-V* in der Klappbox auswählen. Klicken Sie auf *OK* …



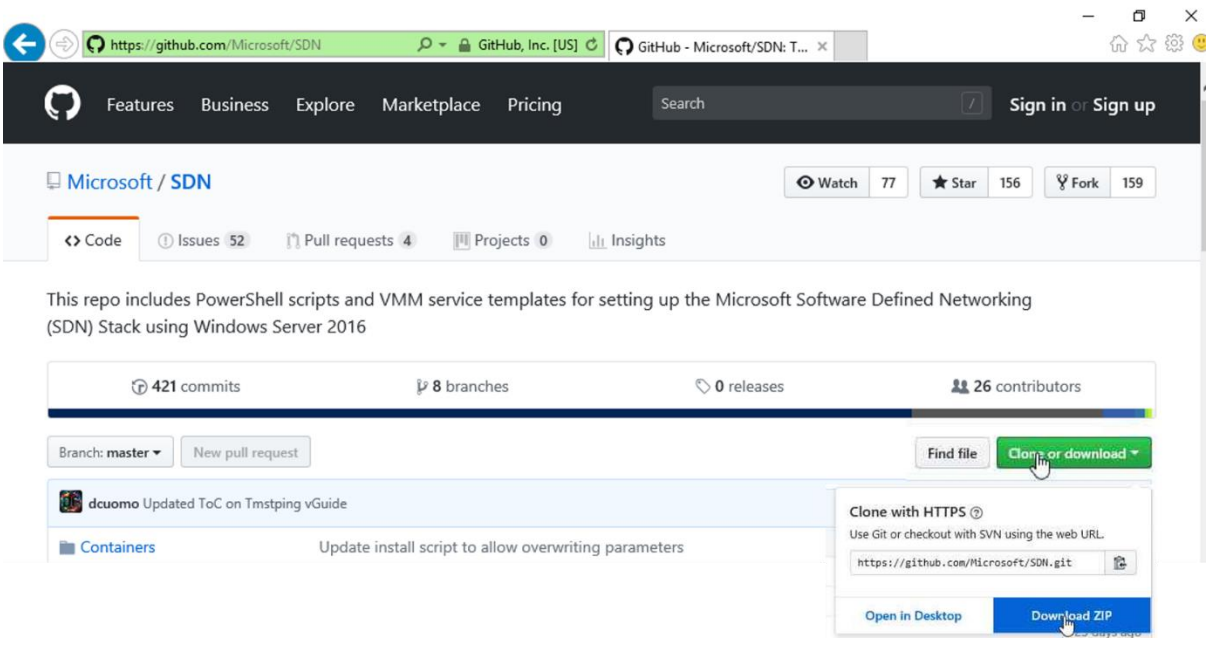
… und der Pfad wird in den *File Share Properties* unseres Clusters angezeigt. Durch einen weiteren Klick auf *OK* wird die Änderung übernommen.



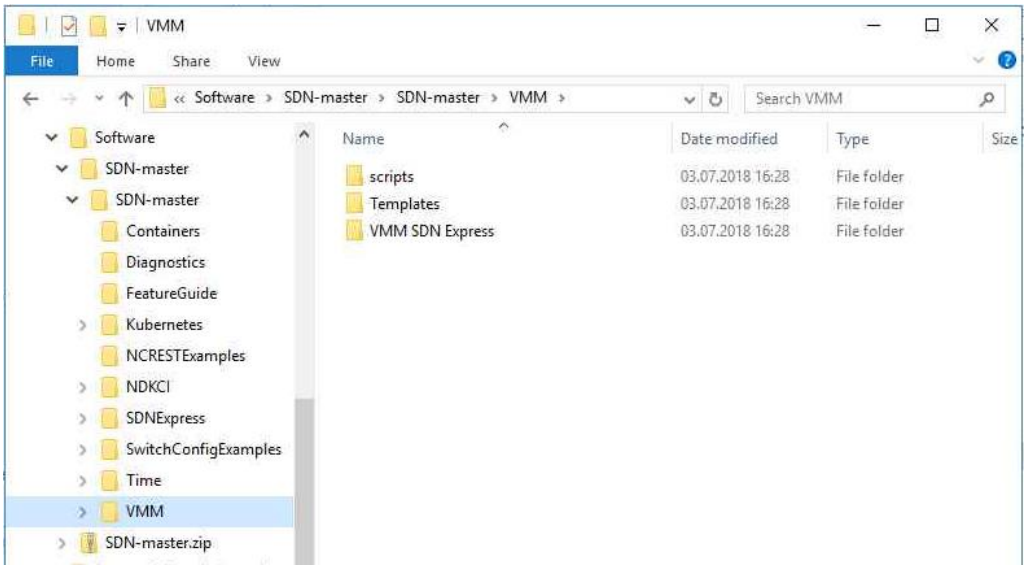
Damit ist die Basis-Installation und Konfiguration für unsere SDN Lab-Umgebung erstmal abgeschlossen.

7 Bereitstellen der Skripte für das SDN Deployment – VMM Express.ps1

Für das SDN Deployment mit dem SCVMM stellt Microsoft ein eigenes Repository auf GitHub mit Service Templates und Skripten zur Verfügung. Laden Sie dieses Paket als .ZIP-Datei in ein Verzeichnis auf dem Datenlaufwerk des System *SDN-VMM01* (z.B. *D:\Software*).



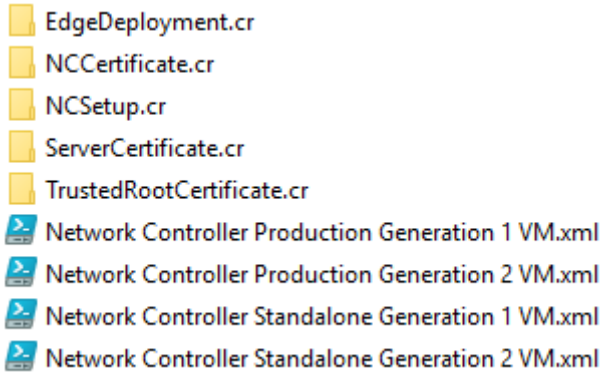
Entsperren und extrahieren Sie die heruntergeladene .ZIP-Datei. Sie erhalten dann folgende Verzeichnisstruktur:



Interessant für unsere weitere Arbeit ist der Unterordner VMM. Er enthält 3 Unterverzeichnisse:

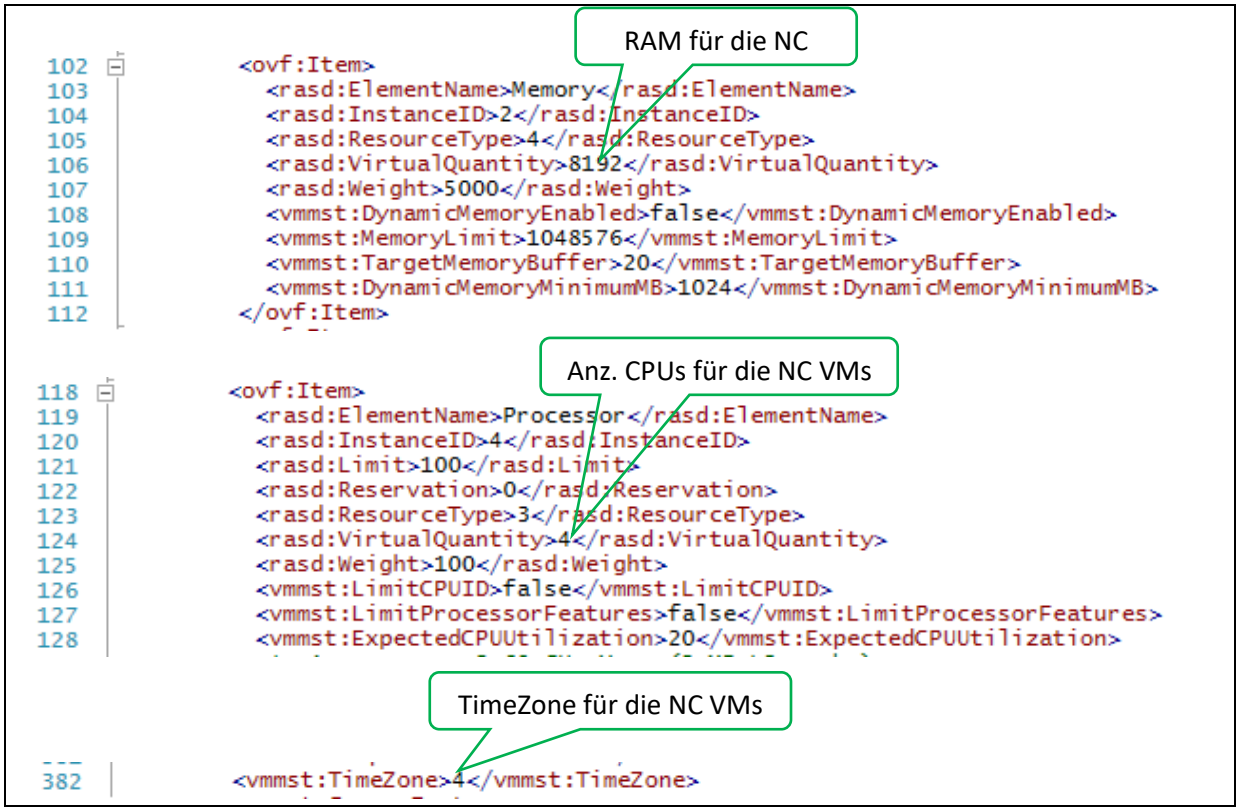
- *Scripts* – PowerShell Beispiel-Skripte für virtuelle Tenant IP-Adressen nach dem SDN Deployment
- *Templates* – VMM Service Templates für Network Controller, SLB und Gateway VMs
- *VMM SDN Express* – PowerShell Skripte für das SDN-Deployment

Werfen wir zunächst einen Blick in das *Templates* Verzeichnis. Es enthält 3 Ordner, je einen für die Netzwerk Controller (*NC*), für die Software Load Balancer(*SLB)* und für die Gateways (*GW*). Im *NC*-Verzeichnis finden wir 4 XML-Dateien für verschiedene Methoden zum Erzeugen der Netzwerk Controller VMs. Der Unterschied: Mit den *Production* Varianten wird ein virtuelles Guest Cluster mit 3 Network Controllern erzeugt, wohingegen die *Standalone* Varianten nur einen einzelnen Network Controller kreieren.



Für produktive Umgebungen sollten Sie auf jeden Fall die *Production* Varianten einsetzen. Wir werden in unserer Lab-Umgebung ebenfalls damit arbeiten und Gen2 VMs erzeugen.

Bei Bedarf können Sie in den Service Templates die Definitionen für die zu erzeugenden VMs noch anpassen. Öffnen Sie die jeweilige XML-Datei (hier: *Network Controller Production Generation 2 VM.xml*) in einem XML-Editor (z.B. PowerShell ISE). Sie können z.B. die Werte für RAM, Anzahl virtueller CPUs und die TimeZone anpassen:




```
102         <ovf:Item>
103           <rasd:ElementName>Memory</rasd:ElementName>
104           <rasd:InstanceID>2</rasd:InstanceID>
105           <rasd:ResourceType>4</rasd:ResourceType>
106           <rasd:VirtualQuantity>8192</rasd:VirtualQuantity>
107           <rasd:Weight>5000</rasd:Weight>
108           <vmmst:DynamicMemoryEnabled>false</vmmst:DynamicMemoryEnabled>
109           <vmmst:MemoryLimit>1048576</vmmst:MemoryLimit>
110           <vmmst:TargetMemoryBuffer>20</vmmst:TargetMemoryBuffer>
111           <vmmst:DynamicMemoryMinimumMB>1024</vmmst:DynamicMemoryMinimumMB>
112         </ovf:Item>

118         <ovf:Item>
119           <rasd:ElementName>Processor</rasd:ElementName>
120           <rasd:InstanceID>4</rasd:InstanceID>
121           <rasd:Limit>100</rasd:Limit>
122           <rasd:Reservation>0</rasd:Reservation>
123           <rasd:ResourceType>3</rasd:ResourceType>
124           <rasd:VirtualQuantity>4</rasd:VirtualQuantity>
125           <rasd:Weight>100</rasd:Weight>
126           <vmmst:LimitCPUID>false</vmmst:LimitCPUID>
127           <vmmst:LimitProcessorFeatures>false</vmmst:LimitProcessorFeatures>
128           <vmmst:ExpectedCPUUtilization>20</vmmst:ExpectedCPUUtilization>

382         <vmmst:TimeZone>4</vmmst:TimeZone>
```

Für unsere Lab-Umgebung reduziere ich den RAM-Wert auf 4096 und setze die Zeitzone auf den Wert 110 (W. Europe Standard Time). Anm.: Eine Tabelle mit den Indexwerten für Zeitzonen finden Sie z.B. hier.

Analog können Sie auch die Service Templates für die SLBs und Gateways anpassen.

Um die Location Einstellungen der VMs (wie z.B. Tastatur Layout und Datum / Uhrzeit Anzeige in Deutsch) anzupassen, habe ich mir eine eigene *Unattend.xml* Datei erstellt, die ich während des Deployments der Service Templates dem SCVMM „unterjuble“.

Nachdem wir die Service Templates gegebenenfalls angepasst haben, können wir uns dem eigentlichen Deployment der SDN-Umgebung zuwenden.

Dazu wechseln wir in das Verzeichnis *VMM SDN Express*:

In diesem Verzeichnis finden wir eine PowerShell Skriptdatei *VMMExpress.ps1* sowie einige PowerShell Datendateien (.PSD1).

- *VMMExpress.ps1*

  Dieses PowerShell Skript enthält die komplette Logik zum Ausrollen einer SDN-Umgebung. Es installiert die notwendigen virtuellen Switches in den Hyper-V Systemen, kreiert die weiter oben beschriebenen logischen Netze und erzeugt dann die VMs für die SDN-Komponenten (Network Controller, Software Load Balancer und Gateways).

Das Skript wird über eine Konfigurationsdatei (PowerShell Datendatei – .PSD1-Datei) gesteuert, die beim Aufruf von *VMMExpress.ps1* über den Parameter *-ConfigurationDataFile* angegeben wird. Diese Konfigurationsdatei enthält sowohl Parameter für die SDN-Umgebung wie logische Netze und VMs als auch Ablaufsteuerungsschalter. Letztere ermöglichen entweder einen gesamten Durchlauf für alle Komponenten oder ein schrittweises Deployment der einzelnen Komponenten. So kann festgelegt werden, dass zunächst nur die Netzwerk Controller Instanzen erzeugt werden. In einem weiteren Aufruf von *VMMExpress.ps1* werden dann die Software Load Balancer erzeugt und in einem dritten Aufruf schließlich die Gateways. Diese Methode für das schrittweise Deployment der diversen SDN-Komponenten werde ich für unsere Lab-Umgebung verwenden.

Das Verzeichnis *VMM SDN Express* enthält 2 Beispiele für Konfigurationsdateien. Die Datei *Fabricconfig.psd1* stellt ein „leeres“ Muster dar. *Fabricconfig_Example.psd1* enthält die Parameter für eine von Microsoft erstellte Beispielkonfiguration.

Für unsere SDN Lab-Umgebung habe ich eine eigene Konfigurationsdatei mit den weiter oben beschriebenen Parametern erstellt – *Fabricconfig-SDNcloud-Production.psd1.*

- *Fabricconfig-SDNcloud-Production.psd1*

Nachstehend finden Sie die vollständige Konfigurationsdatei, um in unserer SDN Lab-Umgebung zunächst nur den Netzwerk Controller und die logischen SDN-Netze zu erzeugen. Speichern Sie diese Datei im gleichen Verzeichnis wie *VMMExpress.ps1* unter dem Namen *Fabricconfig-SDNcloud-Pro1duction.psd1*. Auf die notwendigen Änderungen für das Deployment der SLB- und Gateway-Komponenten werde ich später noch eingehen.

```
# This is a sample configuration file for VMM Express. All the paremeters should be
# modified according to your setup for correct deployment of VMM Express.

# Start VMMExpress.ps1:
# change into the directory with VMMExpress.ps1 and then start the following command
# .\VMMExpress.ps1 -ConfigurationDataFile .\Fabricconfig-SDNcloud-Production.psd1

@{

  AllNodes =
  @(
    @{


      ############################
      #  VM Creation variables
      ############################

      # Name of the VHD or VHDX to use for VM creation. Must Exist in the
      # VMM Library
      VHDName="WS2016_EN.vhdx"

      # VMM Library share to be used for keeping the resources.
      VMMLibrary="\\SDN-VMM01\MSSCVMMLibrary"

      # Product key Can be blank if using a volume license VHD or VHDX, or you are
      # deploying in eval mode.  (Don't forget to press "skip" while VM creation).
      # This key is the AVMA for Windows Server 2016 Datacenter Edition
      ProductKey="TMJ3Y-NTRTM-FJYXT-T22BY-CWG3J"

      #Generation of VM to be used for deployment : Gen1 or Gen2
      Generation="Gen2"
```

```
#Type of Deployment. The values are :
#Standalone : For single Node
#Production : For 3-node
DeploymentType="Production"

#Higly Available VM. Do you want the infrastructural VMs to be deployed on
#Clustered Host and being higly Available ? If yes pass $true else $false
HighlyAvailableVMs = $true

StorageClassification = "Remote Storage"

#leave it if you want default IPvAddressType to be taken which is static
# else change it to "Dynamic"
IPv4AddressType=""

#Host Group to be Managed by Network Controller
NCHostGroupName="SDN-Hosts"

 ###############################################################################
# Section for deploying Logical switch and Logical Network for NC.
#
# Specify IsLogicalSwitchDeployed = $false and
#        IsManagementVMNetworkExisting = $false
# if VMMExpress.ps1 should try to create a SET switch.
#
# If SET cannot be used you have to deploy Logical Switch and
# Management Network by yourself.
#
# NOTE : This script assumes either you have both logical switch and
# logical Network created and deployed or else you will use the script
# to deploy both.
###############################################################################

#Do you have an existing logical switch and the switch is deployed on all
#the hosts you wish to Manage by NC
IsLogicalSwitchDeployed = $false
# IsLogicalSwitchDeployed =  $true

#if above is true give the name of logical switch
LogicalSwitch  = "NC_LogicalSwitch"

# Do you have existing Management Network that you would like to use
IsManagementVMNetworkExisting = $false
# IsManagementVMNetworkExisting = $true

#if above is true give the name of ManagementVMNetwork
ManagementVMNetwork = "NC_Management"

 #Uplink Port Profile to be used
UplinkPortProfile = "HV Uplink Port"

#================================================================================
# The below set of Parameters are required for creation of Management Logical Network
# and other Logical Networks Managed by NC.
# NOTE : If you already have Management Logical Network Created and switch deployed,
# you don't need to specify any parameters for "NC_Management" LN
#================================================================================
 LogicalNetworks = @(
                    @{
      Name = "HNVPA"
      Subnets = @(
        @{
          # VLANID = 255
          VLANID = 0
```

```
            AddressPrefix = "10.10.10.0/24"
            DNS = @("192.168.80.10")
            Gateways = "10.10.10.1"
            PoolStart = "10.10.10.100"
            PoolEnd = "10.10.10.199"
          }
        )
      },
                                  @{
        Name = "Transit"
        Subnets = @(
          @{
            # VLANID = 254
            VLANID = 0
            AddressPrefix = "10.10.20.0/24"
            DNS = @("192.168.80.10")
            Gateways = "10.10.20.1"
            PoolStart = "10.10.20.100"
            PoolEnd = "10.10.20.199"
          }
        )
      },
      @{
        #The first IP address (PoolStart) for this logical network is
        #automatically assigned to the SLB Manager. Other addresses such
        #as the GatewayPublicIPAddress will start after that.
        Name = "PublicVIP"
        Subnets = @(
          @{
            VLANID = 0
            AddressPrefix = "10.10.50.0/24"
            DNS = @("192.168.80.10")
            Gateways = "10.10.50.1"
            PoolStart = "10.10.50.100"
            PoolEnd = "10.10.50.199"
            IsPublic = $true
          }
        )
      },
      @{
        #The first IP address (PoolStart) for this logical network is
        #automatically assigned to the SLB Manager. Other addresses such
        #as the GatewayPublicIPAddress will start after that.
        Name = "PrivateVIP"
        Subnets = @(
          @{
            # VLANID = 253
            VLANID = 0
            AddressPrefix = "10.10.30.0/24"
            DNS = @("192.168.80.10")
            Gateways = "10.10.30.1"
            PoolStart = "10.10.30.100"
            PoolEnd = "10.10.30.199"
            IsPublic = $false
          }
        )
      },
      @{
        #This is used for onboarding Gateway
        Name = "GREVIP"                    # Don't change this. There should be no LN with this name in VMM
        Subnets = @(
          @{
            VLANID = 0
            AddressPrefix = "10.10.40.0/24"
```

```
            DNS = @("192.168.80.10")
            Gateways = "10.10.40.1"
            PoolStart = "10.10.40.100"
            PoolEnd = "10.10.40.199"
            IsPublic = $false
          }
        )
      }
      @{
        #if Management VM Network is not deployed give the ManagementVMNetwork information. Skip
        # this if you already have this created.
        Name = "NC_Management"
        Subnets = @(
          @{
            # VLANID = 811
            VLANID = 0
            AddressPrefix = "192.168.80.0/24"
            DNS = @("192.168.80.10")
            Gateways = "192.168.80.1"
            PoolStart = "192.168.80.230"
            PoolEnd = "192.168.80.254"
            ReservedIPset =  "192.168.80.230"              #This IP will be used for NC Rest API
          }
        )
      }
)

#=============================================================================================
# The following set of paremeters are required for importing VMM service Template,
# configuring the Service Template and Deploying the service Template.
#=============================================================================================

# Make this true if self signed certificate is to be used
# Example : $True , $False
IsCertSelfSigned = $true

#The password for server certificate. This sertificate will be installed on the Host
ServerCertificatePassword="Passw0rd!"

# The following are service settings required for configuring and
# deploying the service template imported client security Group Name
ClientSecurityGroupName= "SDNcloud\Network Controller Users"

# Local Admin credentials
# The local admin user name will be .\Administrator
LocalAdminPassword= "Passw0rd!"

# Management Domain Account Which will be used for NC Deployment
ManagementDomainUser="SDNcloud\FabAdmin"
ManagementDomainUserPassword="Passw0rd!"

 # This is the domain which NC VMs will join
ManagementDomainFDQN="SDNcloud.local"

#Managemet Security Group Name
ManagementSecurityGroupName="SDNcloud\Network Controller Admins"

# Prefix to be added to infrastructural VMs created. Put the prefix such
# that it makes VM name unique as this is the machine name of VM and should be unique.
ComputerNamePrefix = "SDN"

# RestName = "SDN-NCREST.SDNcloud.local"
RestName = "192.168.80.230"
```

```
        ###################################
        #  Deoloyment Control Switches
        ###################################

        # Do you want to deploy NC
        # DeployNC = $false
        DeployNC = $true

        #Do you want to create NC managed HNVPA and Transit networks.
        #These are required if SLB and GW needs to be deployed
        # createNCManagedNetworks = $false
        createNCManagedNetworks = $true

        #Do you want to Deploy SLB. Values are $true , $false
        # DeploySLB =   $true
        DeploySLB =   $false

        #Do you want to deploy GW. Values are $true , $false
        # DeployGW = $true
        DeployGW = $false
    };

  );
}
```

7.1 Schrittweises Deployment der SDN-Komponenten

Ich habe bereits erwähnt, dass die Datei *Fabricconfig.psd1* auch Schalter zur Steuerung der von *VMMExpress.ps1* durchzuführenden Aktionen enthält. Ich werde dies für das Deployment unserer SDN Lab-Umgebung nutzen.

Ich sehe damit den Riesenvorteil, dass man jede Komponentenkategorie separat ausrollen und validieren kann. So kann man z.B. vor der Installation der Software Load Balancer sicherstellen, dass die Netzwerk Controller Umgebung korrekt arbeitet und die Netzwerkvirtualisierung mit dem VXLAN-Protokoll funktioniert.

Folgende Schalter stehen zur Verfügung. Sie finden Sie am Ende der Datei *Fabricconfig.psd1*:

DeployNC = $true | $false

Durch Angabe von *$true* weisen Sie das Skript an, die Netzwerk Controller Instanzen zu erzeugen. Ändern Sie diesen Wert auf *$false*, wenn Sie nach einer erfolgreichen Installation des Netzwerk Controllers nur die Software Load Balancer bzw. Gateway Instanzen erzeugen wollen.

createNCManagedNetworks = $true | $false

Durch Angabe von *$true* werden durch *VMMExpress.ps1* die für die SDN-Umgebung notwendigen logischen Netze wie *HNVPA*, *Transit*, *PublicVIP*, *GREVIP* und *PrivateVIP* im VMM erzeugt und an die SDN-Switches in den Hyper-V Hosts verteilt. Diese Netze müssen vorhanden sein, um die SLB und Gateway Komponenten erzeugen zu können.

DeploySLB = $false | $true

Damit legen Sie fest, dass die Software Load Balancer Komponenten erzeugt werden. Wenn Sie *$true* angeben, muss sichergestellt sein, dass zuvor der Netzwerk Controller und die logischen SDN-Netze erfolgreich erzeugt wurden.

DeployGW = $false | $true

Damit legen Sie fest, dass die Gateway Komponenten erzeugt werden. Wenn Sie *$true* angeben, muss sichergestellt sein, dass zuvor der Netzwerk Controller, die logischen SDN-Netze sowie die SLB-Komponenten erfolgreich erzeugt wurden.

8 Starten des SDN-Deployments

Wenn Sie die Konfigurationsdatei *Fabricconfig.psd1* mit den zu Ihrer Umgebung passenden Parametern erstellt haben, speichern Sie diese im gleichen Verzeichnis wie das Skript *VMMExpress.ps1* mit einem passenden Namen, z.B. für unsere Lab-Umgebung unter *Fabricconfig-SDNcloud-Production.psd1*.

Dann können Sie das SDN-Deployment mit folgendem Aufruf in einer Powershell Konsole starten:

```
.\VMMExpress.ps1 -ConfigurationDataFile .\Fabricconfig-SDNcloud-Production.psd1
```

Tipp: Laden Sie *VMMExpress.ps1* in ein PowerShell ISE Fenster und arbeiten Sie im Debug Modus. Ich werde gleich darauf zurückkommen.

9 Erfahrungen, Tipps und Tricks

9.1 Hyper-V Switch Deployment

In den meisten Fällen kann die Logik in *VMMExpress.ps1* verwendet werden, um in den Hyper-V Hosts einen SDN-kompatiblen Switch zu erzeugen. Dieser wird mit SET (Switch Embedded Teaming) angelegt. Das Skript prüft dazu die in den Hyper-V Hosts vorhandenen physischen Netzwerkadapter und bindet den SDN-Switch an den ersten Adapter, der im Trunk Mode mit dem physischen Netz verbunden ist und dem noch kein logisches Netz zugeordnet ist.

Skript-Ausschnitt Zeile 738:

```
$NetworkAdapter = @(Get-SCVMHostNetworkAdapter -VMHost $VMHost | `
        where {$_.VLanMode -eq "Trunk" -and `
            $_.ConnectionState -eq "Connected" -and `
            $_.LogicalNetworkMap.count -eq  0})
```

Wird kein geeigneter Adapter gefunden ($NetworkAdapter.count -eq 0), gibt das Skript eine Warnung aus. In diesem Fall müssen Sie den logischen SDN-Switch über die VMM-Konsole selbst erzeugen und an die Hyper-V Hosts verteilen.

Ebenso müssen Sie den logischen SDN-Switch selbst erzeugen, wenn Sie SET nicht verwenden wollen (z.B. weil keine geeigneten Netzwerkadapter vorhanden sind oder weil Sie mit „klassischem“ Teaming auf Betriebssystemebene in den Hyper-V Hosts arbeiten wollen).

Anmerkung: SET in Verbindung mit geeigneter Hardware bringt zwar deutliche Performance Vorteile, ist aber keine zwingende Voraussetzung für SDN!

Selbst erzeugte Hyper-V Switches müssen vor dem Start von *VMMExpress.ps1* definiert und verteilt sein. Setzen Sie in diesem Fall in der Konfigurationsdatei *Fabricconfig.psd1* dann folgende Werte:

```
 #Do you have an existing logical switch and the switch is deployed on all
#the host you wish to Manage by NC
IsLogicalSwitchDeployed = $true

#if above is true give the name of logical switch
LogicalSwitch  = " NC_LogicalSwitch "

# Do you have existing Management Network that you would like to use
IsManagementVMNetworkExisting = $true
```

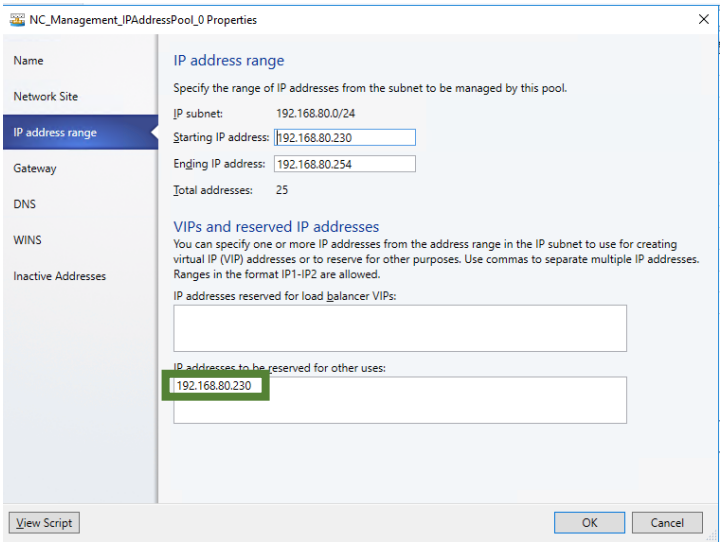
```
#if above is true give the name of ManagementVMNetwork
ManagementVMNetwork = " NC_Management "

#Uplink Port Profile to be used
UplinkPortProfile = "HV Uplink Port"
```

9.2 REST IP

Wenn Sie die *Production* Variante des Netzwerk Controllers ausrollen wollen, müssen Sie für die REST API des NC eine IP-Adresse in der Konfigurationsdatei angeben. Diese muss eine Adresse aus dem IP-Pool für das Management Netz sein und darf nicht für andere Zwecke verwendet werden. Geben Sie in der Definition für das Managementnetz diese IP-Adresse beim Parameter *ReservedIPset* an:

```
ReservedIPset = "192.168.80.230"
```

oder über die VMM Konsole:



Damit ist sichergestellt, dass der VMM diese Adresse nicht einem anderen Objekt zuweisen kann.

In der Konfigurationsdatendatei geben Sie dann diese Adresse beim Parameter *RestName* an:

```
RestName = "192.168.80.230"
```

Beim Erzeugen der NC VMs wird dann von *VMMExpress.ps1* diese Adresse für die REST API des NC Guest Clusters konfiguriert.

Bitte verwenden Sie keinen DNS-Namen; dies wird im Deployment für unsere Lab-Umgebung nicht unterstützt.

Wenn Sie für das NC Deployment die *Standalone* Variante definiert haben, werden diese Angaben ignoriert. Als IP-Adresse für die REST API wird dann direkt die IP-Adresse der NC VM verwendet.

9.3 Undo Funktion

VMMExpress.ps1 hat eine eingebaute *Undo*-Funktion, die aufgerufen wird, wenn während der Ausführung des Skripts ein Fehler auftritt. Sie sorgt dafür, dass alle zuvor angelegten Objekte in der richtigen Reihenfolge wieder aus dem VMM entfernt werden. Das funktioniert aber nur, solange der Netzwerk Controller noch nicht im VMM als Netzwerk Service registriert ist. Danach müssen Sie manuell die Umgebung wieder bereinigen. Hilfe dazu erhalten Sie in der Technet-Library.

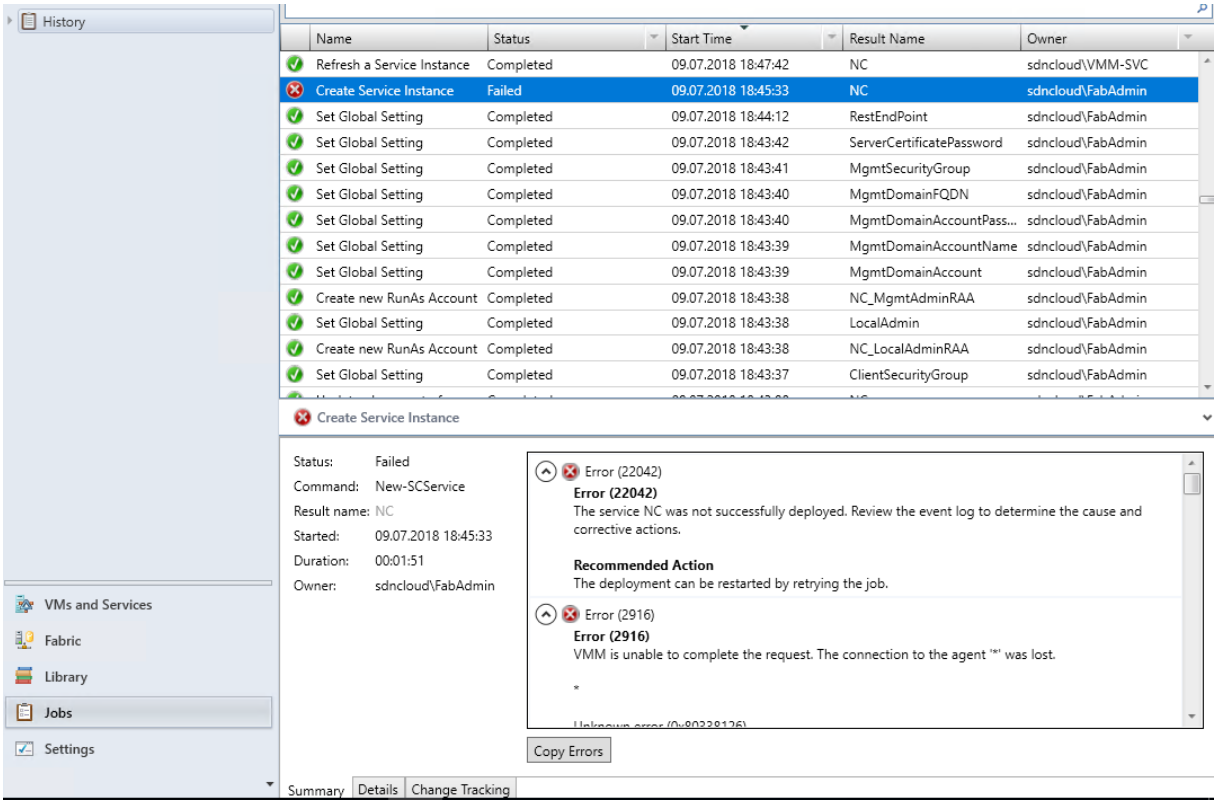
Tipp: Führen Sie *VMMExpress.ps1* im Debug Modus aus und setzen Sie einen Breakpoint auf den Start der *UndoNCDeployment*-Funktion (ab Zeile 513). Dann können Sie im Fehlerfall die einzelnen Schritte beim Beseitigen fehlerhafter Objekte genauer mitverfolgen und gegebenenfalls manuell eingreifen.

```
function undoNCDeployment
{
        param([Object] $node)

  if ($NetworkControllerOnBoarder -eq $false)
    ....
```

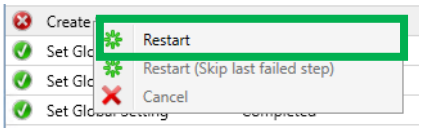
9.4 VMMExpress im Debugger ausführen

Im Zusammenhang mit der *Undo*-Funktion von *VMMExpress.ps1* habe ich Ihnen gerade empfohlen, *VMMExpress.ps1* im Debug Modus der PowerShell ISE auszuführen. Nach meiner Erfahrung hat dies den Vorteil, dass man die einzelnen Schritte bzw. Funktionsblöcke genauer mitverfolgen kann. *VMMExpress.ps1* arbeitet intern mit den CmdLets des PowerShell Moduls für den Virtual Machine Manager. Diese CmdLets starten jeweils einen *Job* im VMM Server, was im *Job*-Fenster der VMM-Konsole mitverfolgt werden kann.

Der folgende Screenshot zeigt eine Situation, als während des Erzeugens der VMs für den Netzwerk Controller ein Problem im Hyper-V Cluster auftrat (Zeile 505 in *VMMExpress.ps1*).



Oftmals können Sie die Ursache des Problems sofort beheben und dann einen Restart des fehlgeschlagenen Jobs versuchen (im Kontextmenü des Jobs).



Ich empfehle Ihnen an folgenden Stellen Breakpoints zu setzen und dann die jeweiligen Befehle im Einzelschrittmodus auszuführen:

- NC Deployment

  Funktion *OnBoardNetworkController* Zeile 252:

  An dieser Stelle können Sie den Connection String für die REST API des Netzwerk Controllers in der Variablen $ConnectionString überprüfen.

  Funktion *importServiceTemplate* Zeile 359:

  Dies wäre eine geeignete Stelle, um über den Service Designer in das importierte NC Service Template noch eine eigene *Unattend.xml* einzubringen. Achtung: Der VMM erzeugt selbst eine *Unattend.xml* Datei und mischt sie mit der angegebenen benutzerspezifischen Variante später beim Kreieren der VMs zusammen.

  Funktion *configureAndDeployService* Zeile 467:

  ```
  if($ServiceUpdate.deploymenterrorlist -ne $null)
  ```

  Überprüfen Sie, ob die Variable wirklich $null enthält. Nur dann war die Verteilung der NC VMs auf die Hyper-V Hosts erfolgreich und das Skript kann weiterlaufen. Korrigieren Sie gegebenenfalls das Placement der VMs über den Service Designer in der VMM-Konsole und weisen der Variablen den Wert $null zu.

  Funktion *configureAndDeployService* Zeile 505:

  ```
  $sc= New-SCService -ServiceConfiguration $ServiceConfig
  ```

  Hier können Sie im Service Designer der VMM-Konsole nochmals alle Parameter zum Erzeugen der NC VMs überprüfen.

- SLB Deployment

  Funktion *ImportSLBServiceTemplate* Zeile 878:

  Dies wäre eine geeignete Stelle, um über den Service Designer in das importierte SLB Service Template noch eine eigene *Unattend.xml* einzubringen. Achtung: Der VMM erzeugt selbst eine *Unattend.xml* Datei und mischt sie mit der angegebenen benutzerspezifischen Variante später beim Kreieren der VMs zusammen.

  Funktion *ConfigureAndDeploySLBService* Zeile 917:

  ```
  if($ServiceUpdate.deploymenterrorlist -ne $null)
  ```

  Überprüfen Sie die Variable wieder, ob sie $null enthält und korrigieren Sie eventuell das Placement der VMs.

  Funktion *ConfigureAndDeploySLBService* Zeile 937:

  Hier können Sie im Service Designer der VMM-Konsole nochmals alle Parameter zum Erzeugen der SLB VMs überprüfen.

- Gateway Deployment

  Funktion *importGatewayTemplate* Zeile 1050:

Dies wäre eine geeignete Stelle, um über den Service Designer in das importierte Gateway Service Template noch eine eigene *Unattend.xml* einzubringen. Achtung: Der VMM erzeugt selbst eine *Unattend.xml* Datei und mischt sie mit der angegebenen benutzerspezifischen Variante später beim Kreieren der VMs zusammen.

Funktion *ConfigureAndDeployGatewayService* Zeile 1091:

```
if($ServiceUpdate.deploymenterrorlist -ne $null)
```

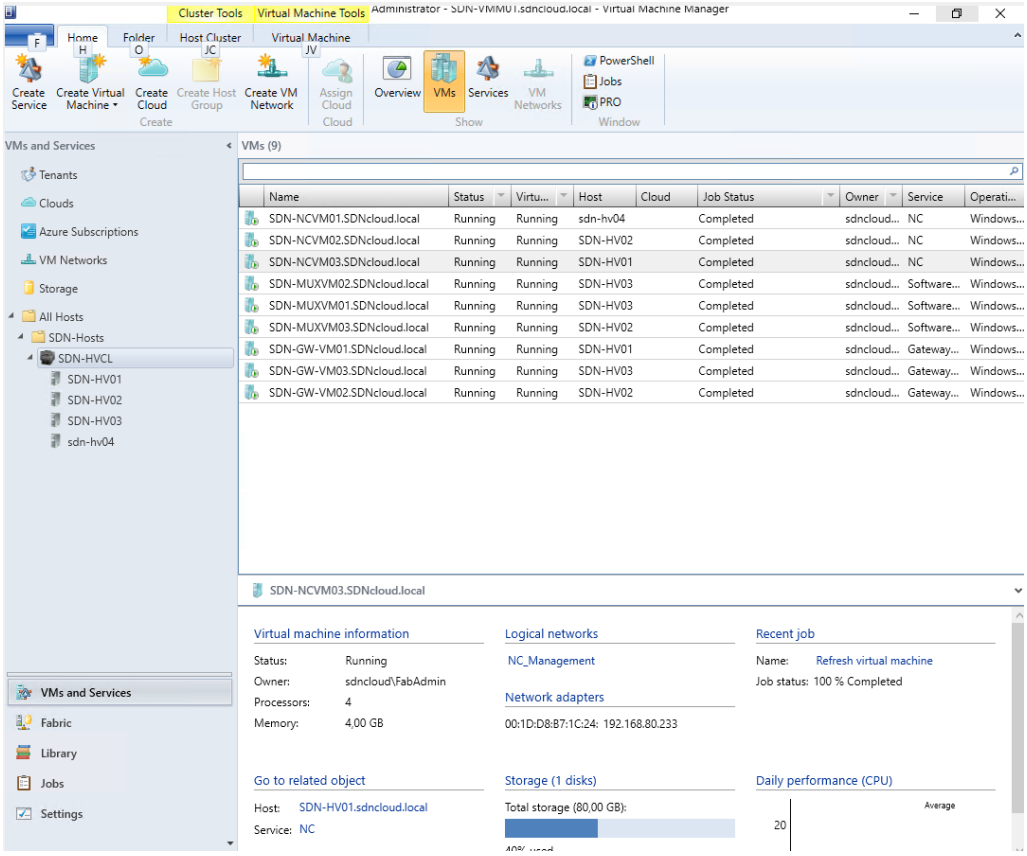
Überprüfen Sie die Variable wieder, ob sie $null enthält und korrigieren Sie eventuell das Placement der VMs.

Funktion *ConfigureAndDeployGatewayService* Zeile 1112:

Hier können Sie im Service Designer der VMM-Konsole nochmals alle Parameter zum Erzeugen der Gateway VMs überprüfen.

10 Post NC Deployment Schritte und Validierung

Wenn Sie das Skript *VMMExpress.ps1* erfolgreich ausgeführt haben, finden Sie zwar alle SDN-Komponenten als virtuelle Maschinen und Services im VMM (der folgende Screenshot zeigt unsere vollständige LAB-Umgebung).



Damit ist aber die SDN-Konfiguration noch nicht abgeschlossen. Vielmehr sind für die einzelnen SDN-Komponenten noch manuelle Schritte notwendig. Und was natürlich nicht fehlen darf, ist das Durchspielen einiger Testszenarien.

10.1 Tenant Testumgebungen

Wenn der Netzwerk Controller (*Standalone* Variante) bzw. das NC Guest Cluster (*Production* Variante) ausgerollt und als Netzwerk Service im VMM registriert sind, sollten Sie als nächstes 2 Tenant Testumgebungen anlegen. Jede Tenant Umgebung enthält ein *VM Network* mit jeweils 2 Subnetzen. In jedem Tenant *VM Network* gibt es dann jeweils 2 VMs.

10.1.1 Tenant VM Netze

- Erzeugen Sie 2 Tenant VM Netzwerk Umgebungen („*Red VMNet*“ und „*Green VMNet*“) mit identischen Netzwerkdefinitionen:

  Subnetz 0: „*Red VMnet_0*“ bzw. „*Green VMnet_0*“ – 192.168.0.0/24
  IP-Pool 0: “*Red IP Pool 0*” bzw. “*Green IP Pool 0*” – 192.168.0.4 - 192.168.0.254
  Gateway in den IP Pools: jeweils 192.168.0.1

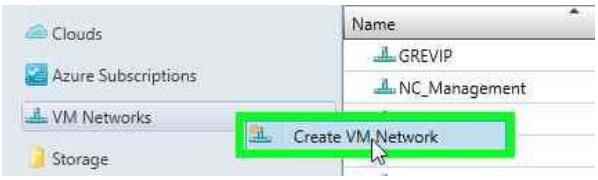
  Subnetz 1: „*Red VMnet_1*“ bzw. „*Green VMnet_1*“ – 192.168.1.0/24
  IP-Pool 1: “*Red IP Pool 1*” bzw. “*Green IP Pool 1*” – 192.168.1.4 - 192.168.1.254
  Gateway in den IP Pools: jeweils 192.168.1.1

DNS in allen IP Pools: 192.168.80.10 (der Standard DNS-Server in unserer Lab-Umgebung auf dem System SDN-DC01

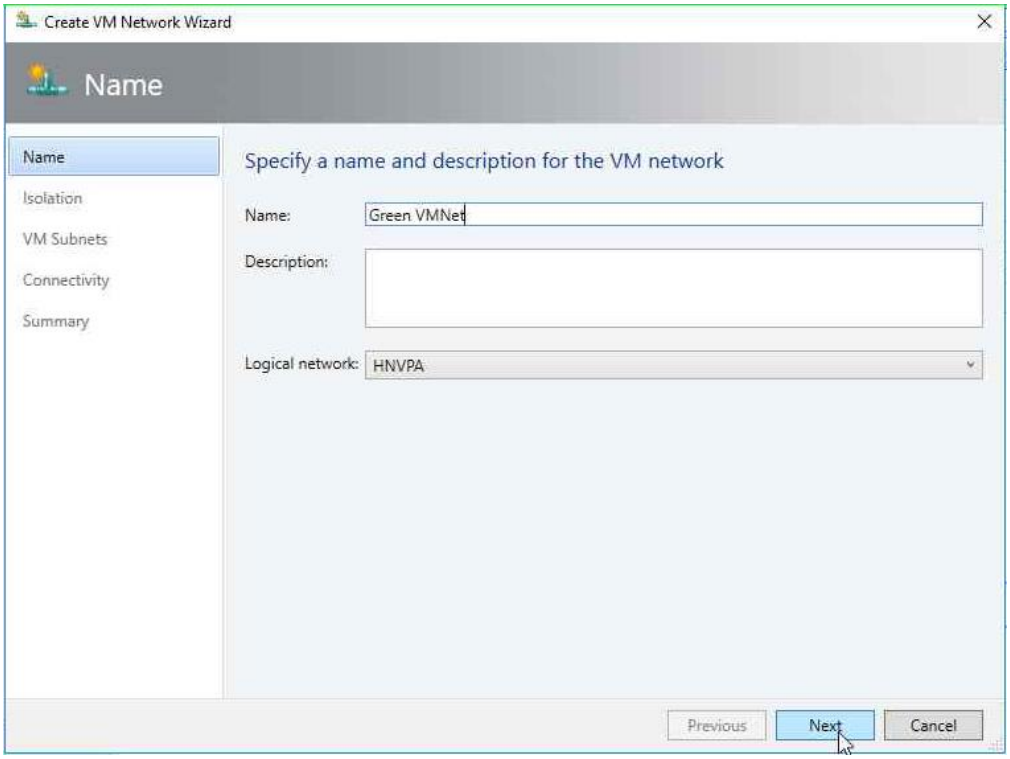
Sie können dies über die VMM-Konsole oder mit dem weiter unten folgenden PowerShell Skript erledigen.

VMM-Konsole

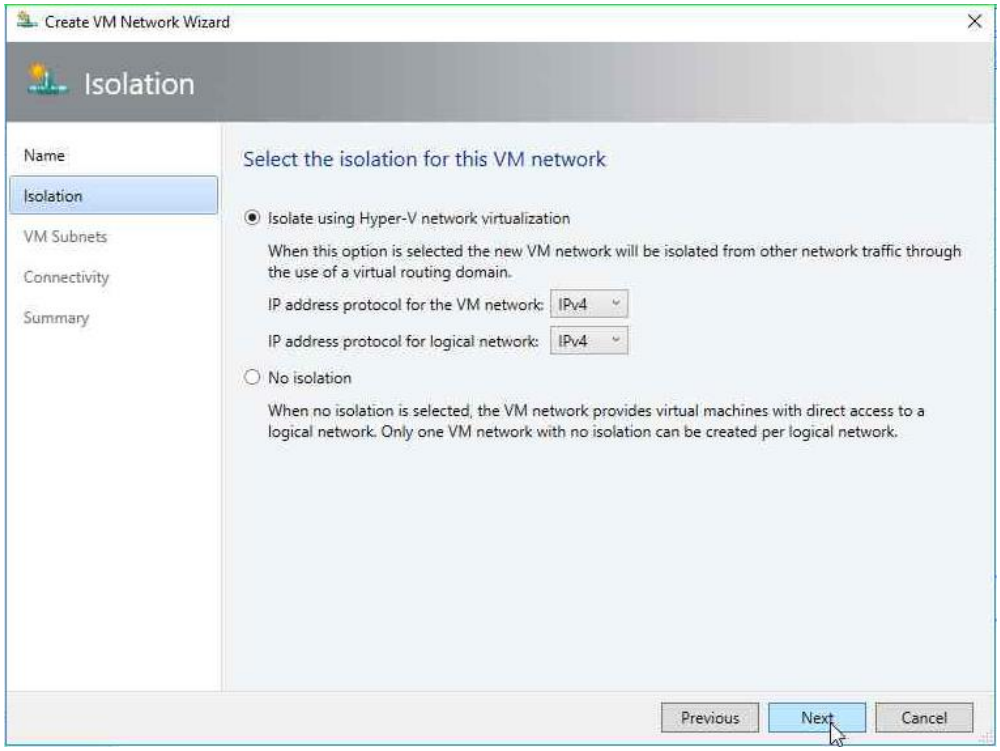
Klicken Sie im VMM Arbeitsbereich *VMs and Services* mit der rechten Maustaste auf die Kategorie *VM Networks* und wählen Sie im Kontextmenü *Create VM Network.*



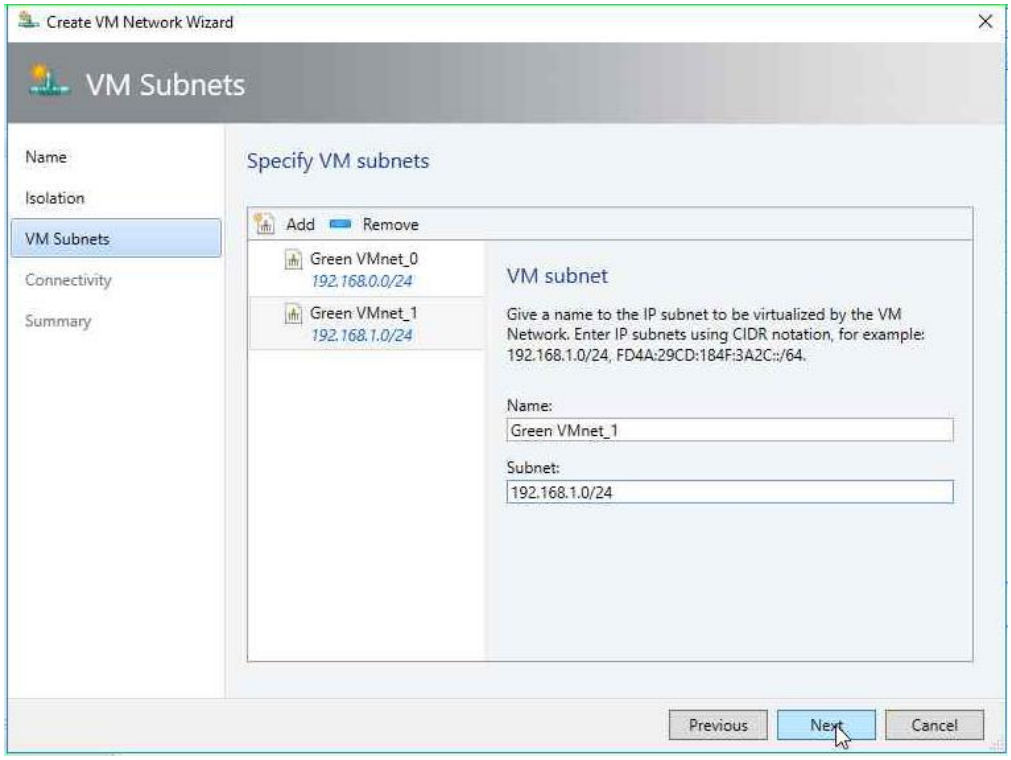
Geben Sie den Namen des anzulegenden VM Netzwerks an und achten Sie darauf, dass in der Klappliste *Logical Network* das HNVPA Netz ausgewählt ist. Klicken Sie auf *Next.*



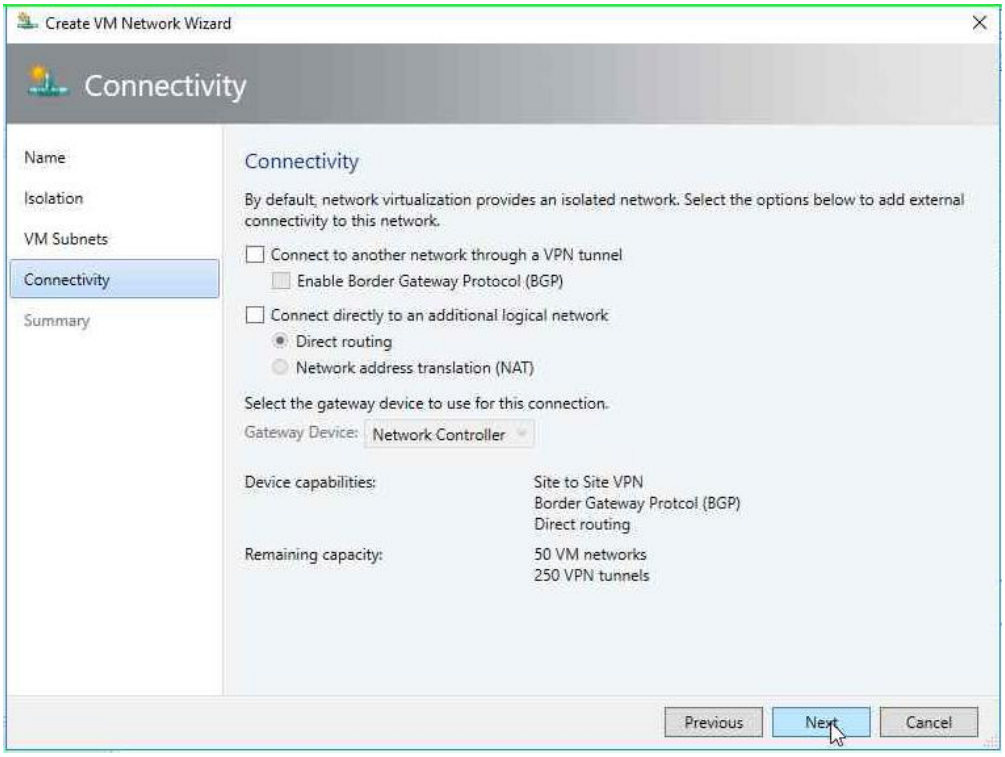
Lassen Sie im Dialogfenster *Isolation* die Voreinstellung *Isolate using Hyper-V network virtualization* und klicken Sie auf *Next.*



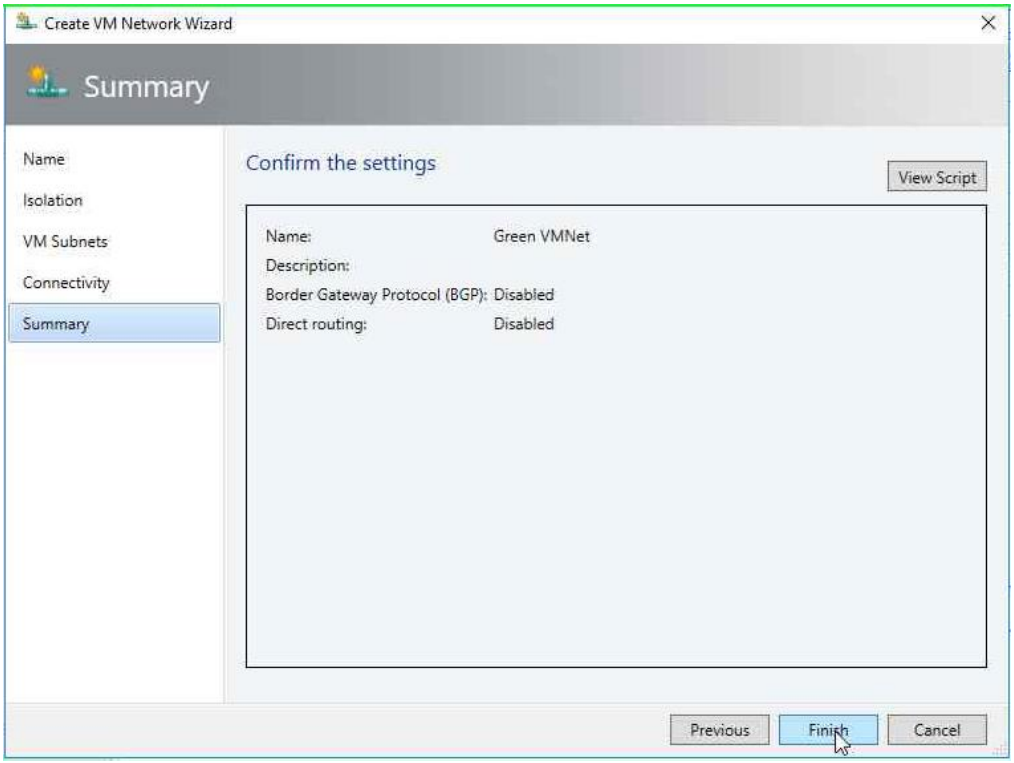
Tragen Sie im Dialogfenster *VM Subnets* die Parameter für die beiden Subnetze ein. Klicken Sie dann wieder auf *Next*.



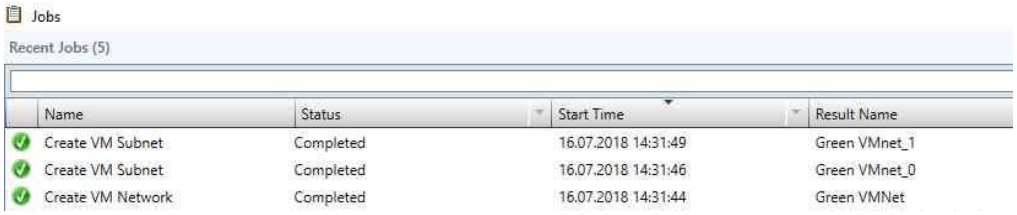
Im Dialog *Connectivity* lassen Sie die Voreinstellungen und klicken auf *Next*.



Sie sehen nun eine Zusammenfassung Ihrer Eingaben.

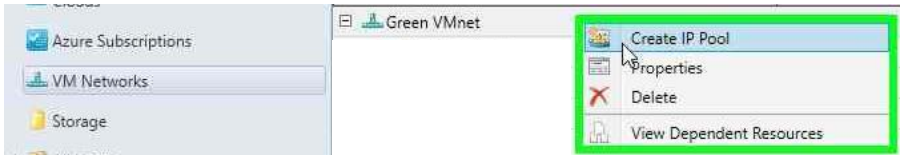


Durch einen Klick auf *Finish* können Sie nun das Anlegen des VM Networks starten. In der *Job*-Liste des VMM können Sie das mitverfolgen.

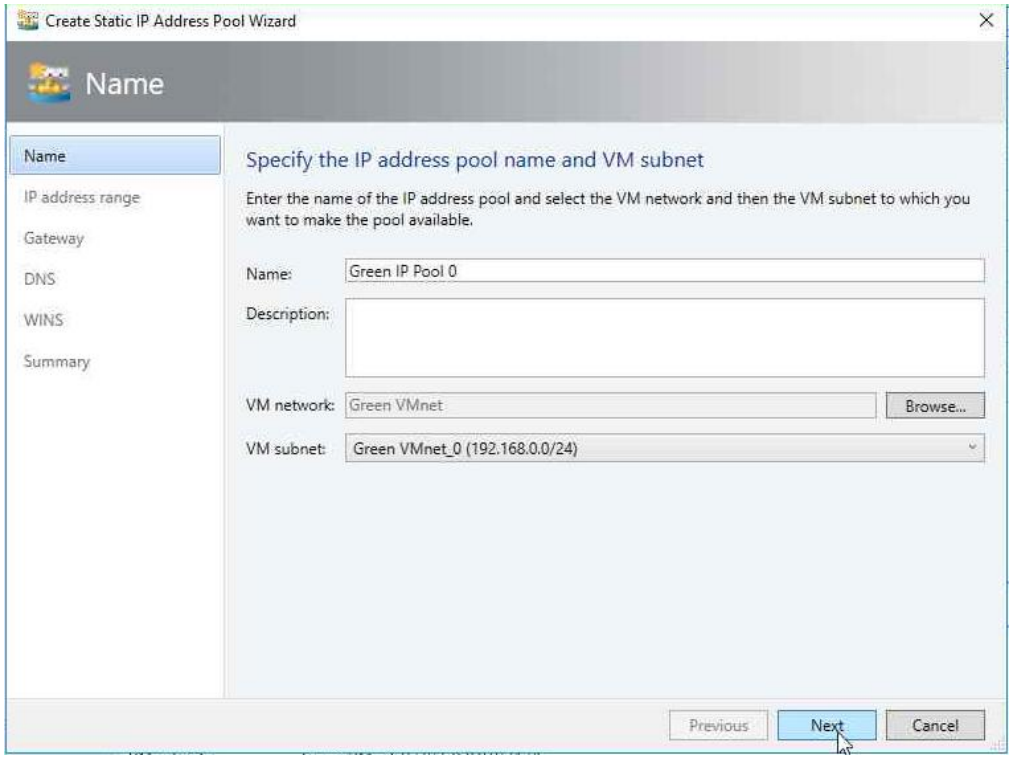


Nun müssen Sie für jedes Subnetz noch den zugehörigen IP Pool anlegen.

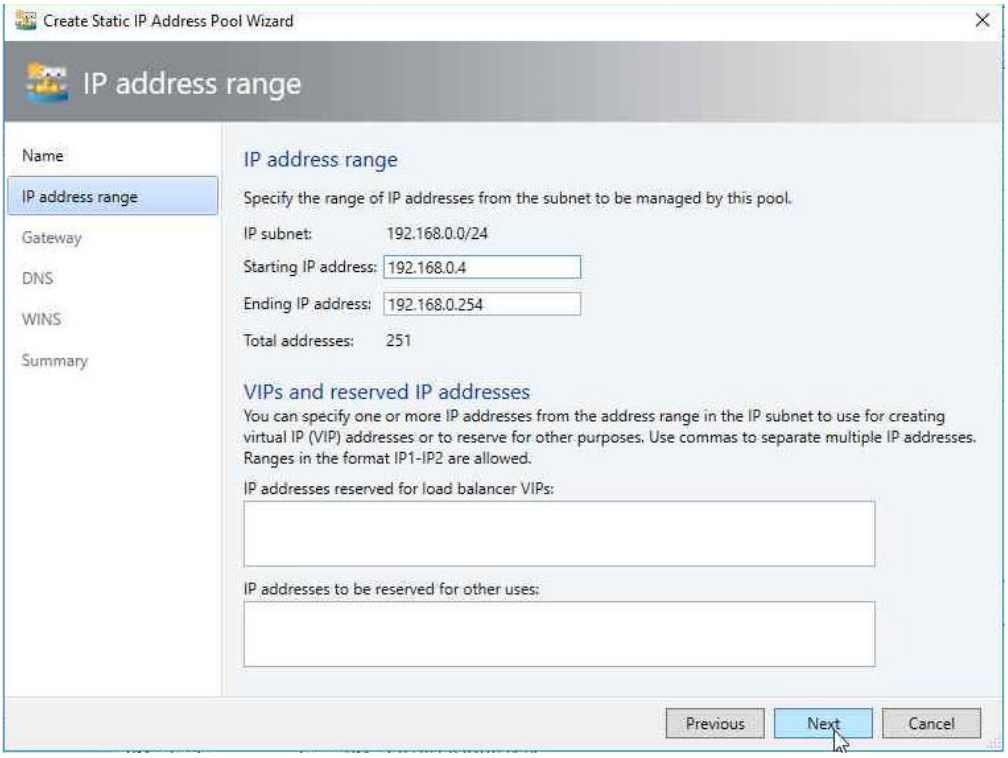
Klicken Sie mit der rechten Maustaste auf das soeben erzeugte VM Network (*Green VMnet*) und wählen Sie im Kontextmenü die Aktion *Create IP Pool*.



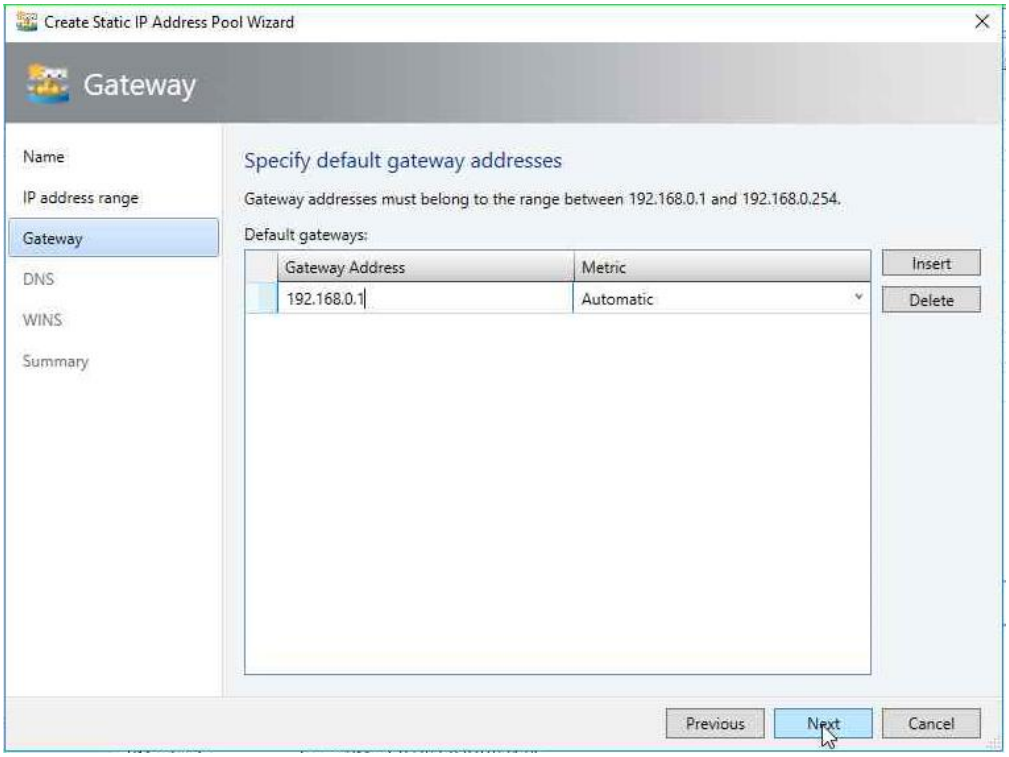
Geben Sie den Namen des IP Pools an. Stellen Sie sicher, dass im Feld *VM network* das richtige Tenant VM-Netz und im Feld *VM Subnet* das gewünschte Subnetz ausgewählt sind. Klicken Sie dann auf *Next*.



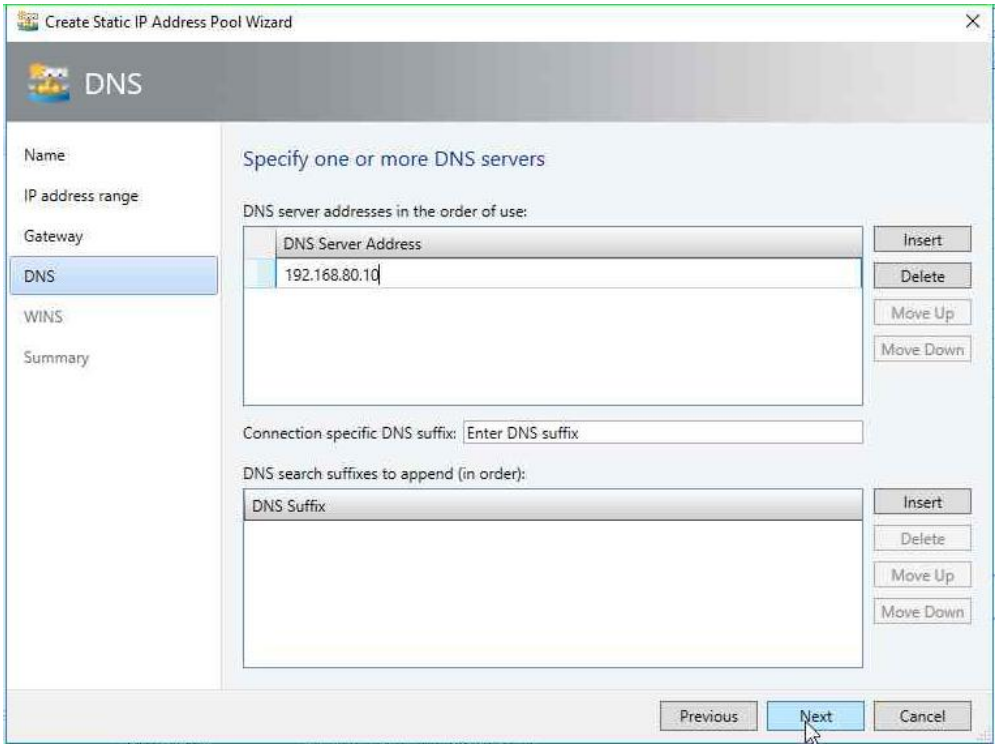
Die im Dialog *IP address range* angezeigten Werte können Sie für unsere Test-Umgebung durch einen Klick auf Next übernehmen.



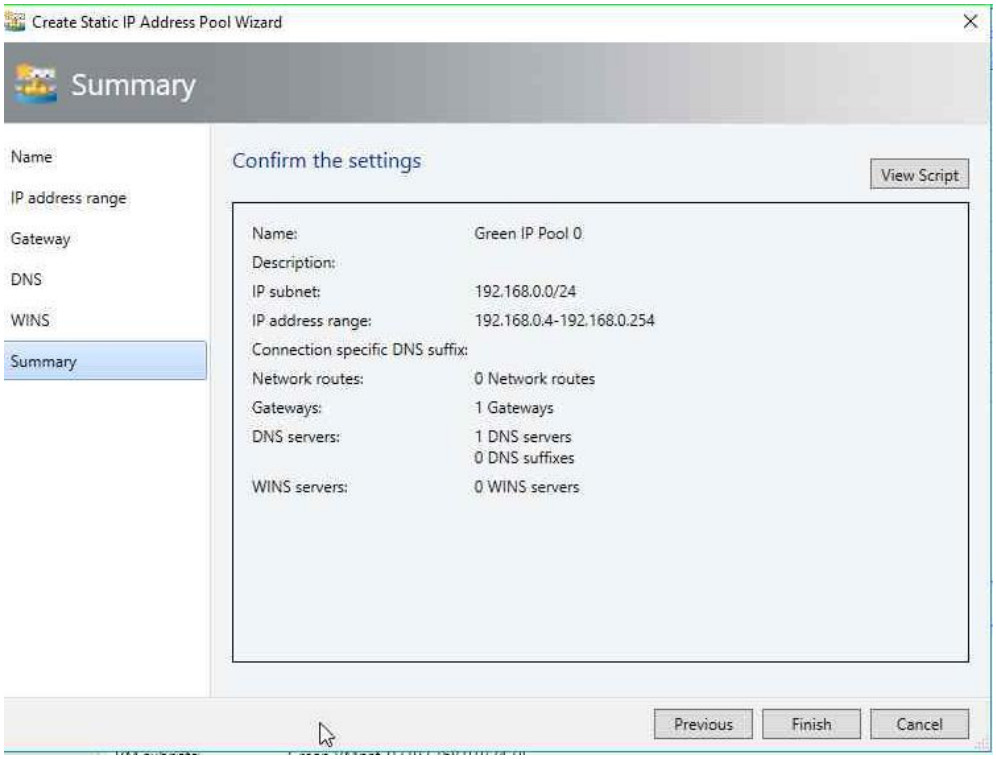
Geben Sie im *Gateway* Dialog die Adresse des jeweiligen Gateways ein und klicken Sie auf *Next*.



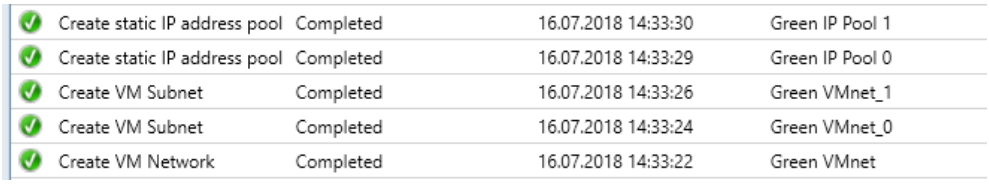
Im *DNS* Dialog geben Sie die Adresse des DNS-Servers unserer Lab-Umgebung (192.168.80.10) ein und klicken auf *Next*.



Den *WINS* Dialog können Sie mit *Next* überspringen. Überprüfen Sie in der Zusammenfassung nochmals Ihre Eingaben und klicken Sie dann auf *Finish*, um den IP Pool zu erstellen.



In der VMM Jobliste können Sie den Vorgang wieder mitverfolgen.



Erzeugen Sie analog die IP Pools für die anderen Subnetze der Tenants.

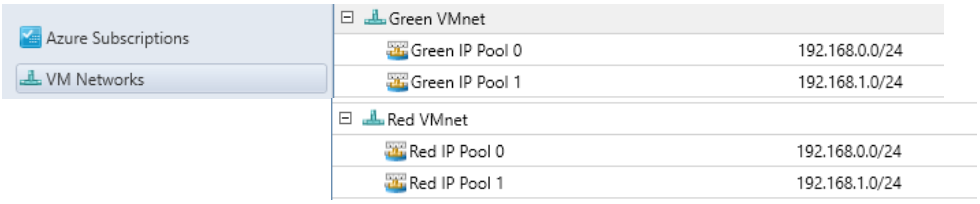
PowerShell

Etwas schneller und komfortabler geht das Ganze mit dem nachstehenden PowerShell Skript. Weisen Sie der Variablen $tenantname den gewünschten Tenant Namen zu und starten das Skript in einer PowerShell Sitzung mit Administrator Rechten.

```
# Create VM network for tenant
$tenantname = "Red"
# $tenantname = "Green"

$subnet0 = "192.168.0.0/24"
$IPrangeStart0 = "192.168.0.4"
$IPrangeEnd0 = "192.168.0.254"
$gateway0 = "192.168.0.1"

$subnet1 = "192.168.1.0/24"
$IPrangeStart1 = "192.168.1.4"
$IPrangeEnd1 = "192.168.1.254"
$gateway1 = "192.168.1.1"

$DNSserver = "192.168.80.10"


$logicalNetwork = Get-SCLogicalNetwork -Name "HNVPA"

# tenant VM net
$vmNetwork = New-SCVMNetwork -Name "$tenantname VMnet" `
        -LogicalNetwork $logicalNetwork -IsolationType "WindowsNetworkVirtualization" `
        -CAIPAddressPoolType "IPV4" -PAIPAddressPoolType "IPV4"

# tenant subnets
$subnetVLan0 = New-SCSubnetVLan -Subnet $subnet0
$VMsubnet0 = New-SCVMSubnet -Name "$tenantname VMnet_0" `
        -VMNetwork $vmNetwork -SubnetVLan $subnetVLan0
$subnetVLan1 = New-SCSubnetVLan -Subnet $subnet1
$VMsubnet1 = New-SCVMSubnet -Name "$tenantname VMnet_1" `
        -VMNetwork $vmNetwork -SubnetVLan $subnetVLan1

# VMnet_0

# Gateways
$allGateways = @()
$allGateways += New-SCDefaultGateway -IPAddress $gateway0 -Automatic
# DNS servers
$allDnsServer = @($DNSserver)
# DNS suffixes
$allDnsSuffixes = @()
# WINS servers
$allWinsServers = @()

New-SCStaticIPAddressPool -Name "$Tenantname IP Pool 0" -VMSubnet $VMsubnet0 `
   -Subnet $subnet0 -IPAddressRangeStart $IPrangeStart0 -IPAddressRangeEnd $IPrangeEnd0 `
   -DefaultGateway $allGateways -DNSServer $allDnsServer -DNSSuffix "" `
   -DNSSearchSuffix $allDnsSuffixes

# VMnet_1

# Gateways
$allGateways = @()
$allGateways += New-SCDefaultGateway -IPAddress $gateway1 -Automatic
# DNS servers
```

```
$allDnsServer = @($DNSserver)
# DNS suffixes
$allDnsSuffixes = @()
# WINS servers
$allWinsServers = @()

New-SCStaticIPAddressPool -Name "$Tenantname IP Pool 1" -VMSubnet $VMsubnet1 `
   -Subnet $subnet1 -IPAddressRangeStart $IPrangeStart1 -IPAddressRangeEnd $IPrangeEnd1 `
   -DefaultGateway $allGateways -DNSServer $allDnsServer -DNSSuffix "" `
   -DNSSearchSuffix $allDnsSuffixes
```

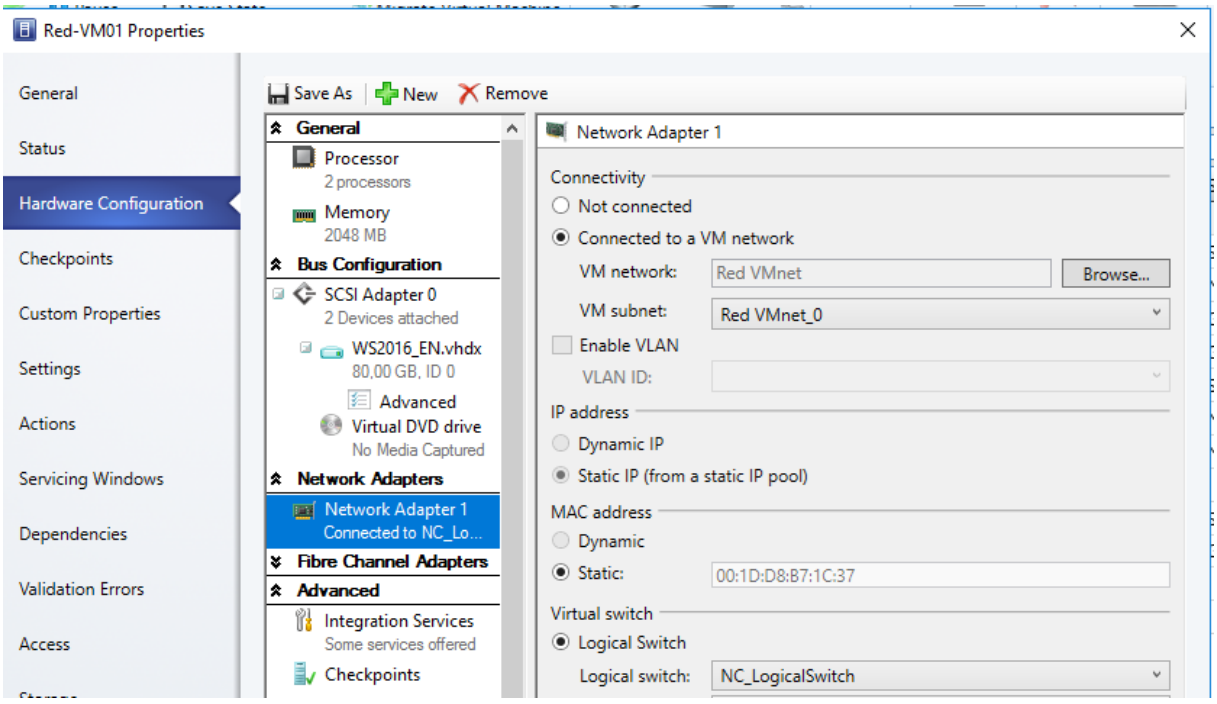
Den Ablauf des Skripts können Sie wieder über die Jobliste im VMM verfolgen. Als Ergebnis erhalten Sie dann das jeweilige Tenant VM Netzwerk einschließlich der IP Pools.



10.1.2 Tenant Test VMs

Erzeugen Sie für jeden der beiden Tenants 2 VMs und verbinden Sie diese wie folgt mit den Subnetzen der beiden VM Netze:

- *Red-VM01* – verbunden mit Subnetz 0 von VM Netzwerk *Red VMnet*



analog:

- *Red-VM02* – verbunden mit Subnetz 1 von VM Netzwerk *Red VMnet*
- *Green-VM01* – verbunden mit Subnetz 0 von VM Netzwerk *Green VMnet*
- *Green-VM02* – verbunden mit Subnetz 1 von VM Netzwerk *Green VMnet*

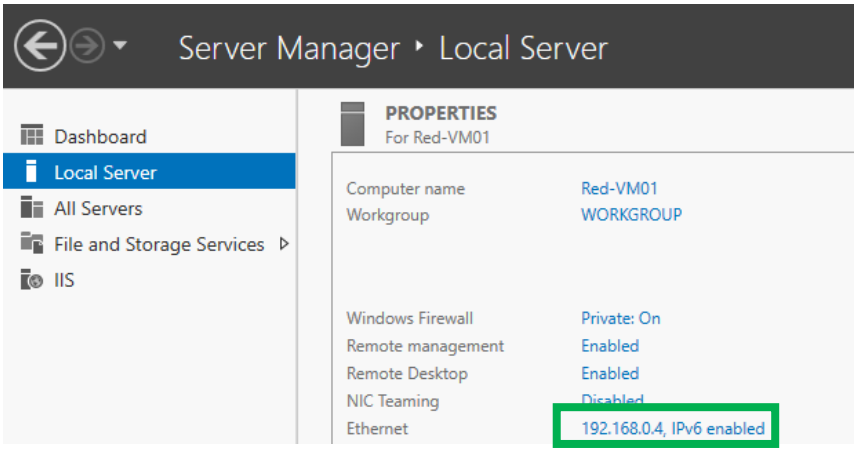
Installieren Sie in jeder dieser VMs die Serverrolle *Web Server* mit Standardeinstellungen. Modifizieren Sie anschließend das Bild der IIS-Startseite – Datei *iisstart.png* im Verzeichnis *C:\inetpub\wwwroot* – so dass sie den Namen der VM zeigt, z.B.

Tipp: Das können Sie ganz einfach mit *Paint* aus dem *Windows Zubehör* durchführen.

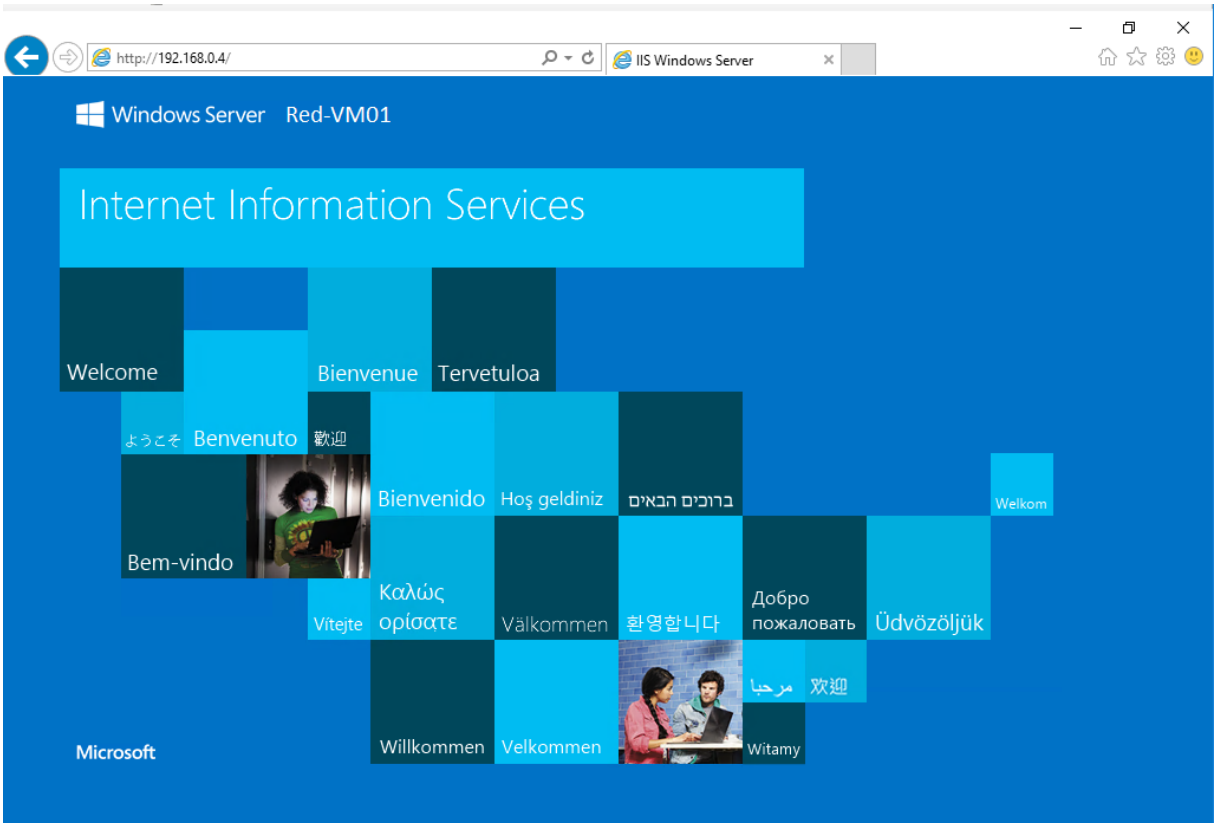
10.2 Validierung der Netzwerkisolation mit dem VXLAN-Protokoll

Nun können Sie folgende Szenarien zur Netzwerkisolation mit dem VXLAN-Protokoll durchspielen.

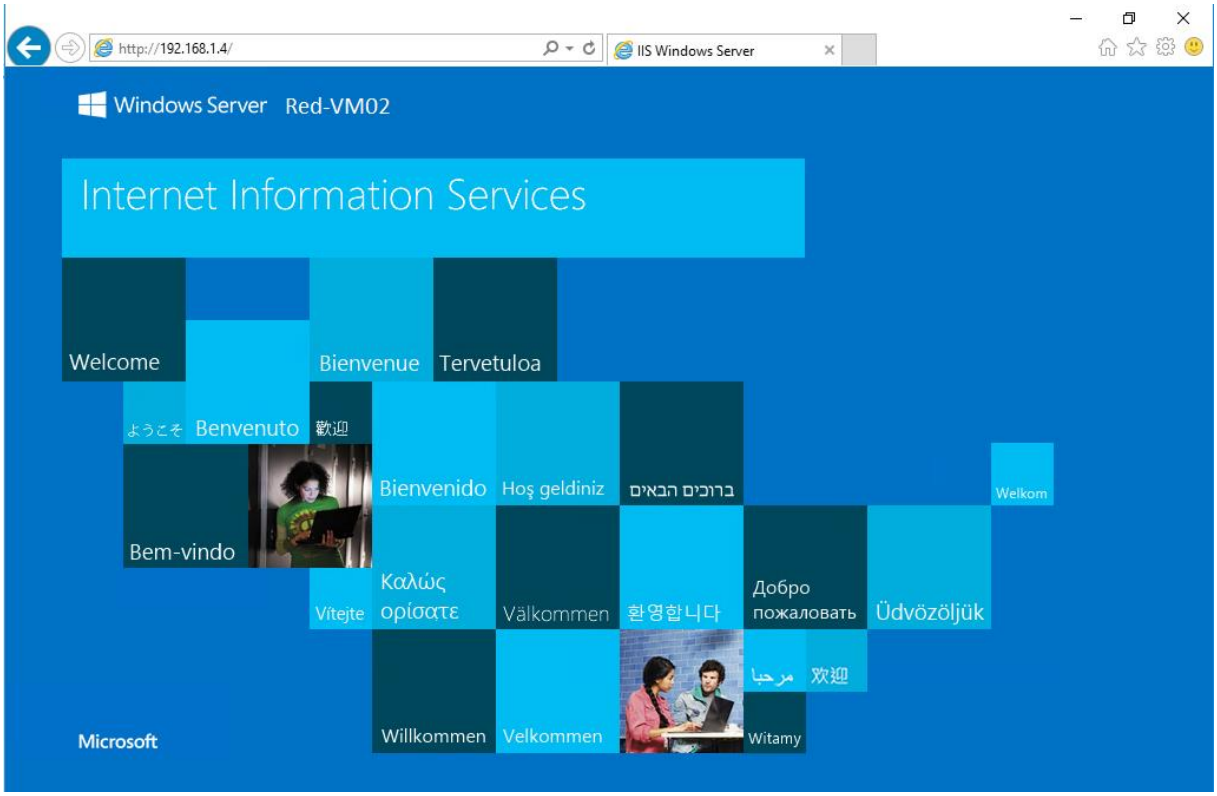
Melden Sie sich auf dem Tenant System *Red-VM01* an. Im Server Manager können Sie die zugewiesene IP-Adresse überprüfen:



Melden Sie sich jetzt auf dem Tenant System *Red-VM02* an. Der Server Manager wird Ihnen die Adresse 192.168.1.4 anzeigen. Starten Sie den Internet Explorer und geben Sie in die Adresszeile folgende Adresse ein: *http://192.168.0.4* – Ergebnis:



Natürlich können Sie auch die umgekehrte Richtung wählen – vom System *Red-VM01* (192.168.0.4) aus die Startseite von *Red-VM02* (192.168.1.4) aufrufen. Entsprechend erhalten Sie:



Die gleichen Spielchen können Sie jetzt auch zwischen den Systemen im *Green-VMnet* ausführen. Sie werden immer nur die Systeme im gleichen VMnet erreichen, können aber nicht auf Systeme in anderen VM Netzen zugreifen.

Erkenntnisse:

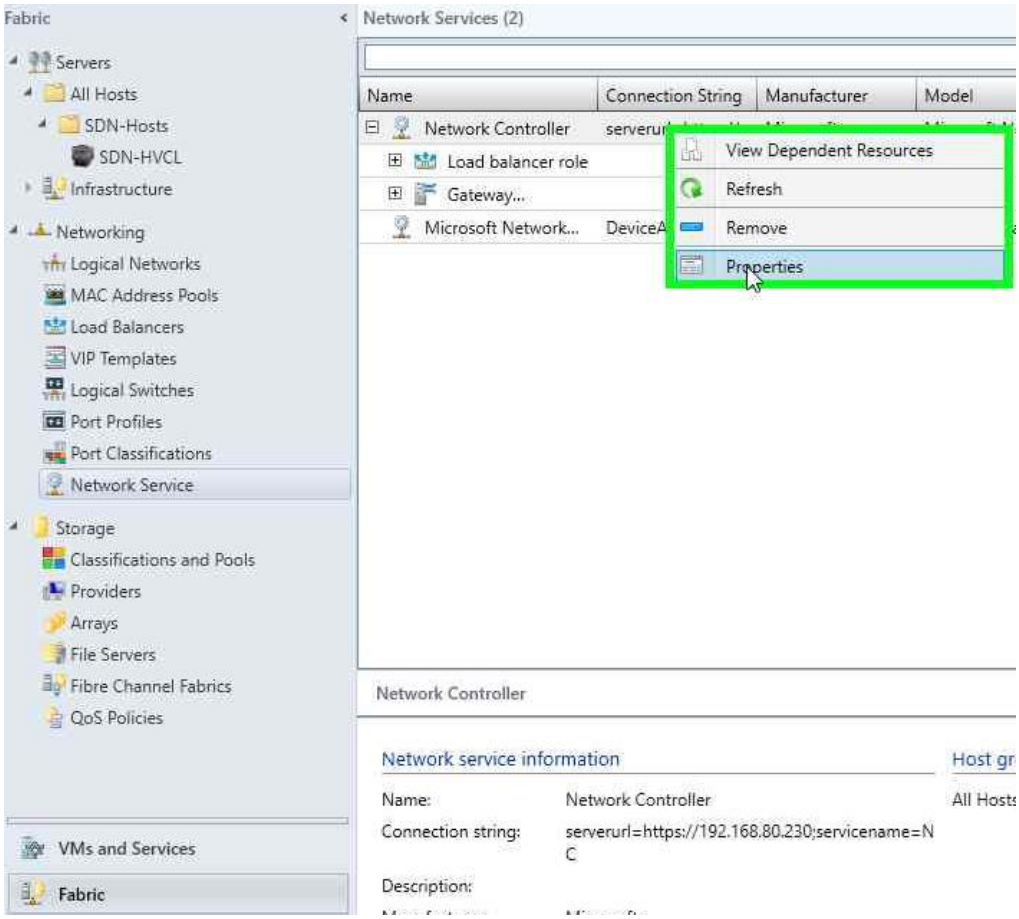
Innerhalb eines Tenant VM Netzes werden IP-Pakete zwischen den definierten Subnetzen geroutet. Eine Verbindung zu Systemen in andere Tenant VM Netze ist jedoch nicht möglich, auch wenn die IP-Bereiche identisch sind oder sich überlappen.

10.3 Konfiguration der Load Balancer Service Instanzen

Vom Skript *VMMExpress.ps1* wurden zwar SLB Instanzen als VMs erzeugt. Diese müssen wir aber noch konfigurieren.

- Wechseln Sie in der VMM-Konsole in den *Fabric* Arbeitsbereich und markieren Sie die Kategorie *Network Service*. Klicken Sie mit der rechten Maustaste auf den Eintrag *Network Controller* und rufen die *Properties* auf.

  Wechseln Sie in der VMM-Konsole in den *Fabric* Arbeitsbereich und markieren Sie die Kategorie *Network Service*. Klicken Sie mit der rechten Maustaste auf den Eintrag *Network Controller* und rufen die *Porperties* auf.



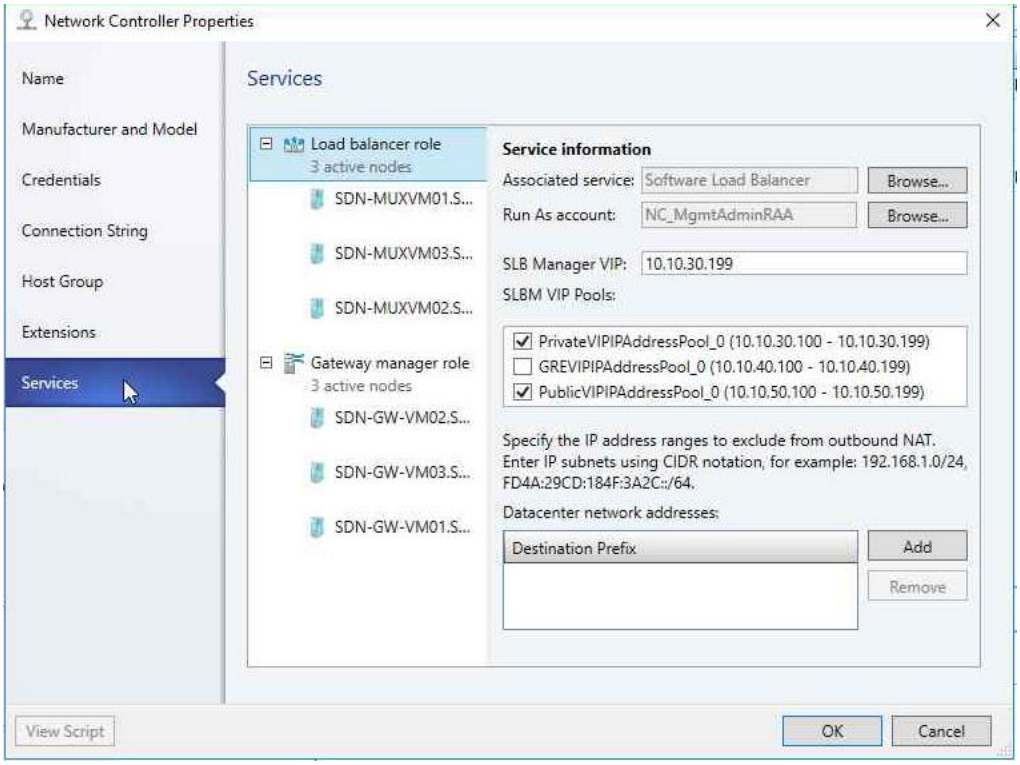
- Wechseln Sie in den *Network Controller Properties* auf die Registerkarte *Services* und markieren die *Load Balancer Role*.

  Im Feld *Associated Service* wählen Sie über die *Browse…* Schaltfläche unseren *Software Load Balancer* Service aus.

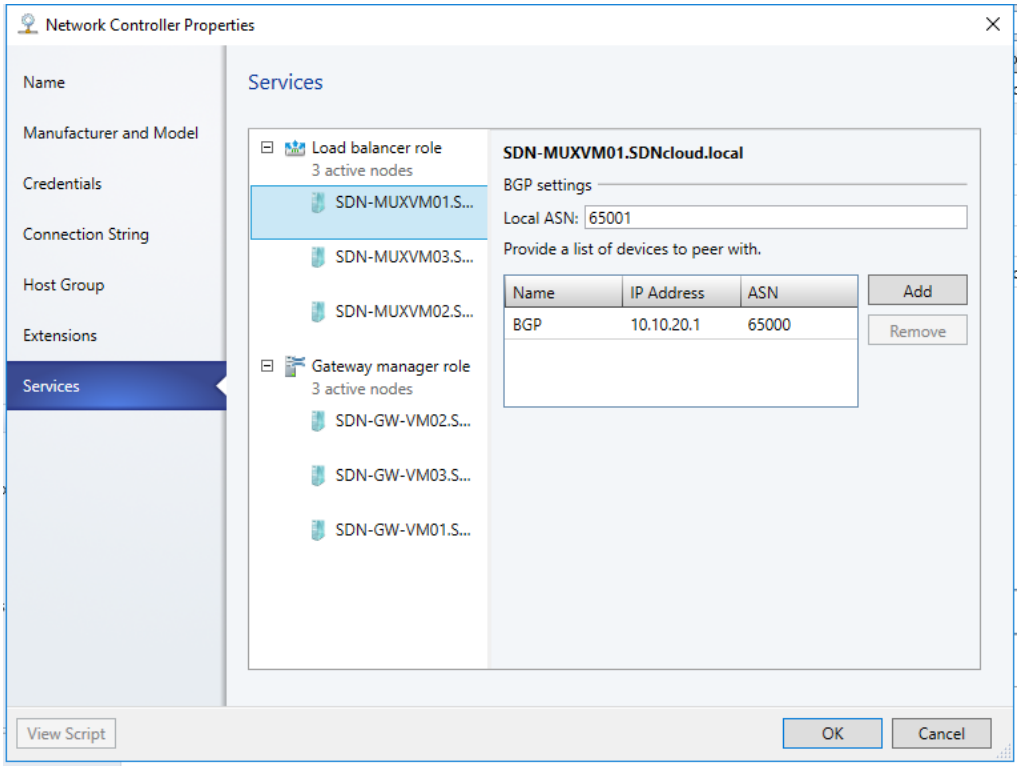
  Wählen Sie für das Feld *Run As Account* einen passenden Account aus (hier: *NC_MgmtAdminRAA*)

  In das Feld *SLB Manager VIP* tragen Sie die letzte IP-Adresse aus dem vom Skript *VMMExpress.ps1* erzeugten IP-Pool des logischen Netzes *PrivateVIP* ein (hier also: 10.10.30.199).

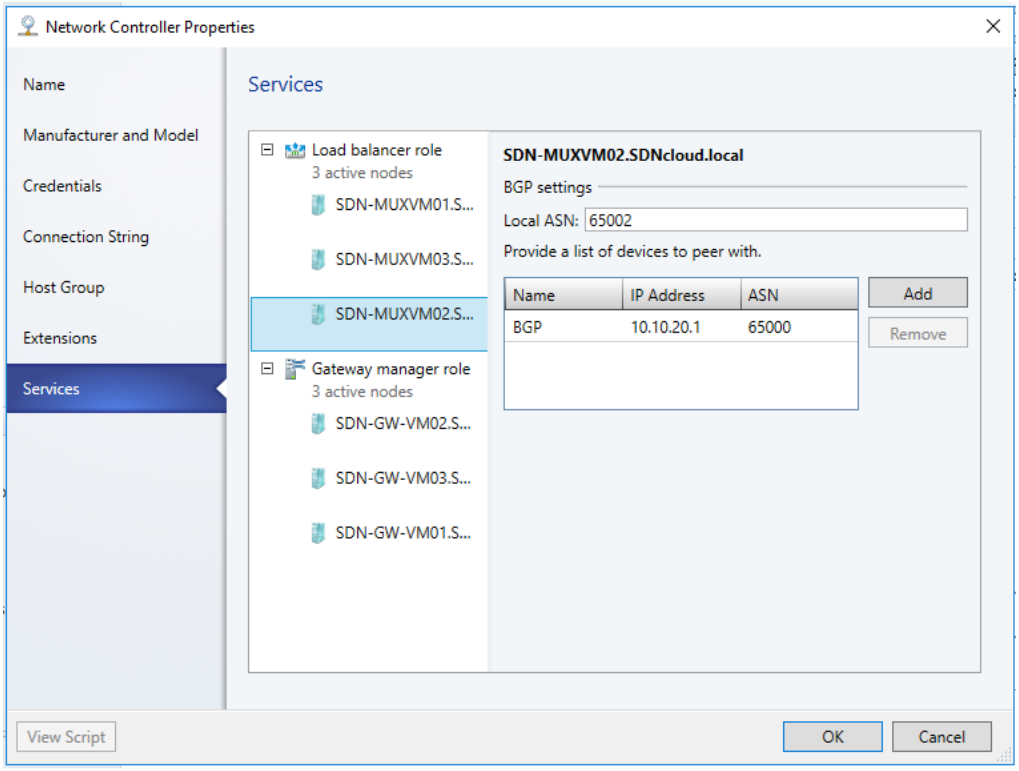
  Selektieren Sie außerdem in der Liste *SLBM VIP Pools* die IP Pools *PublicVIP_IPAddressPool_0* und *PrivateVIP_IPAddressPool_0*. Diese beiden IP Pools werden dann vom VMM dem SLB Manager zur Verfügung gestellt.

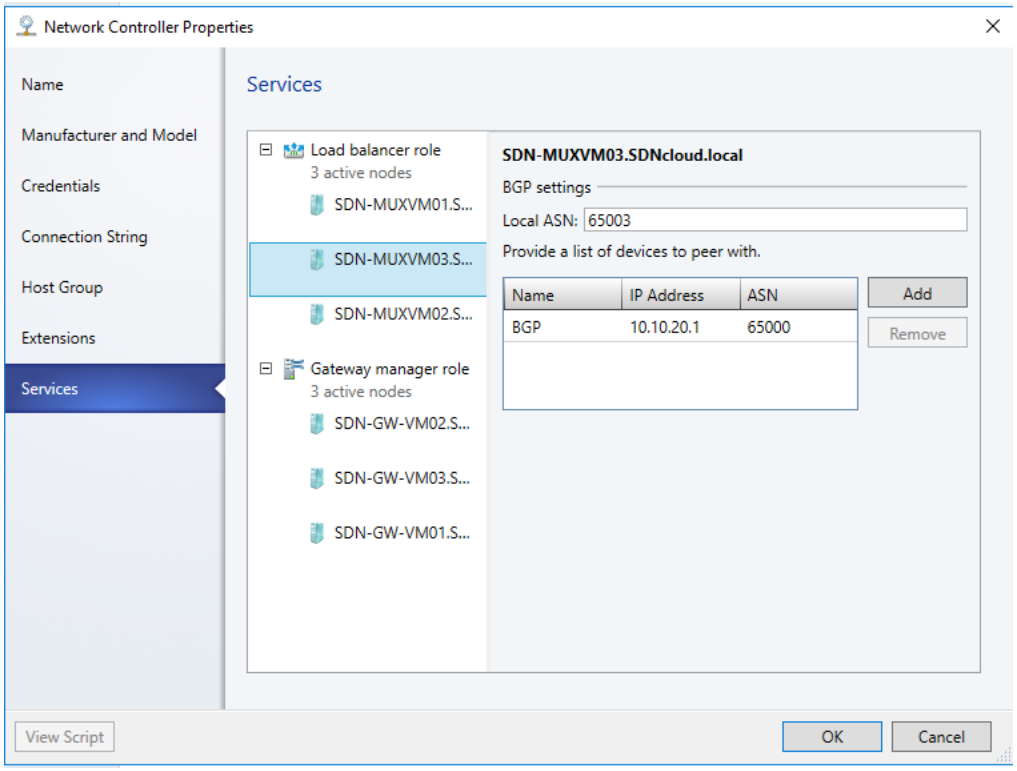


- Klicken Sie auf die erste SLB/MUX Instanz *SDN-MUXVM01*. Geben Sie in das Feld *Local ASN* eine für Ihre Umgebung geeignete ASN (*Autonome System Nummer* im Sinne des BGP-Protokolls) ein, hier 65001. Über die *Add* Schaltfläche fügen Sie nun die Daten des BGP-Routers ein, mit dem diese SLB-Instanz kommunizieren soll. Unser System *SDN-DC01* soll ja der BGP-Router werden und er soll die ASN 65000 erhalten. Für die Kommunikation verwenden wir das *Transit* Netzwerk, in dem das System *SDN-DC01* die IP-Adresse 10.10.20.1 besitzt.



- Wiederholen Sie diese Schritte für die restlichen SLB-Instanzen





Damit ist die Konfiguration der SLB Instanzen abgeschlossen.

10.4 Konfiguration des BGP Routers

Bei der Installation unseres Domain Controllers *SDN-DC01* haben wir unter anderem bereits die Serverrolle *RRAS* installiert, aber nicht weiter konfiguriert. Das werden wir nun nachholen, indem wir die BGP-Funktion dieser Serverrolle aktivieren. Dies muss über PowerShell Befehle geschehen.

Führen Sie auf dem *SDN-DC01* in einer PowerShell Sitzung mit Administratorberechtigung folgenden Befehl aus:

```
Add-BgpRouter -BgpIdentifier 10.10.20.1 -LocalASN 65000
```

Hiermit wird die BGP-Router Funktion aktiviert. Dabei wird auch die *lokale ASN* (*Autonome System Nummer*) des BGP-Routers festgelegt. Dies kann ein beliebiger Integer-Wert sein. Üblicherweise werden in SDN-Umgebungen ASNs ab 65000 verwendet, da diese nicht zu Konflikten mit ASNs physischer Geräte führen können. Diese sind typischerweise kleiner 65000.

Als *BgpIdentifier* geben wir die IP-Adresse des BGP-Routers an - in unserem Fall die Adresse des *SDN-DC01* im *Transit* Netz (also 10.10.20.1).

Jetzt können wir die *Peers* definieren, die sich mit dem BGP-Router verbinden und Routen austauschen können. Führen Sie für unsere Lab-Umgebung folgende PowerShell Befehle aus. Bitte vergleichen Sie vorher die Kombinationen *PeerASN* / *PeerIPAddress* mit den Parametern, die Sie bei der Konfiguration der SLB Service Instanzen festgelegt haben.

```
Add-BgpPeer -Name MUX001 -LocalIPAddress 10.10.20.1 -PeerIPAddress 10.10.20.101 `
    -LocalASN 65000 -PeerASN 65001 -OperationMode Mixed -PeeringMode Automatic
Add-BgpPeer -Name MUX002 -LocalIPAddress 10.10.20.1 -PeerIPAddress 10.10.20.100 `
    -LocalASN 65000 -PeerASN 65002 -OperationMode Mixed -PeeringMode Automatic
Add-BgpPeer -Name MUX003 -LocalIPAddress 10.10.20.1 -PeerIPAddress 10.10.20.102 `
    -LocalASN 65000 -PeerASN 65003 -OperationMode Mixed -PeeringMode Automatic
```

Warten Sie, bis sich alle *Peers* mit dem BGP-Router verbunden haben (*ConnectivityStatus = Connected*). Dies kann unter Umständen geraume Zeit dauern. Den Verbindungsstatus können Sie mit folgendem Kommando prüfen:

```
Get-BgpPeer
```

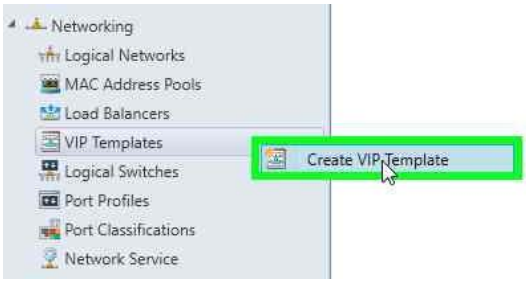
```
PS C:\Windows\system32> Get-BgpPeer

PeerName LocalIPAddress PeerIPAddress PeerASN OperationMode ConnectivityStatus
-------- -------------- ------------- ------- ------------- -----------------
MUX001   10.10.20.1     10.10.20.101  65001   Mixed         Connected
MUX002   10.10.20.1     10.10.20.100  65002   Mixed         Connected
MUX003   10.10.20.1     10.10.20.102  65003   Mixed         Connected
```

10.5 Validierung des Software Load Balancing

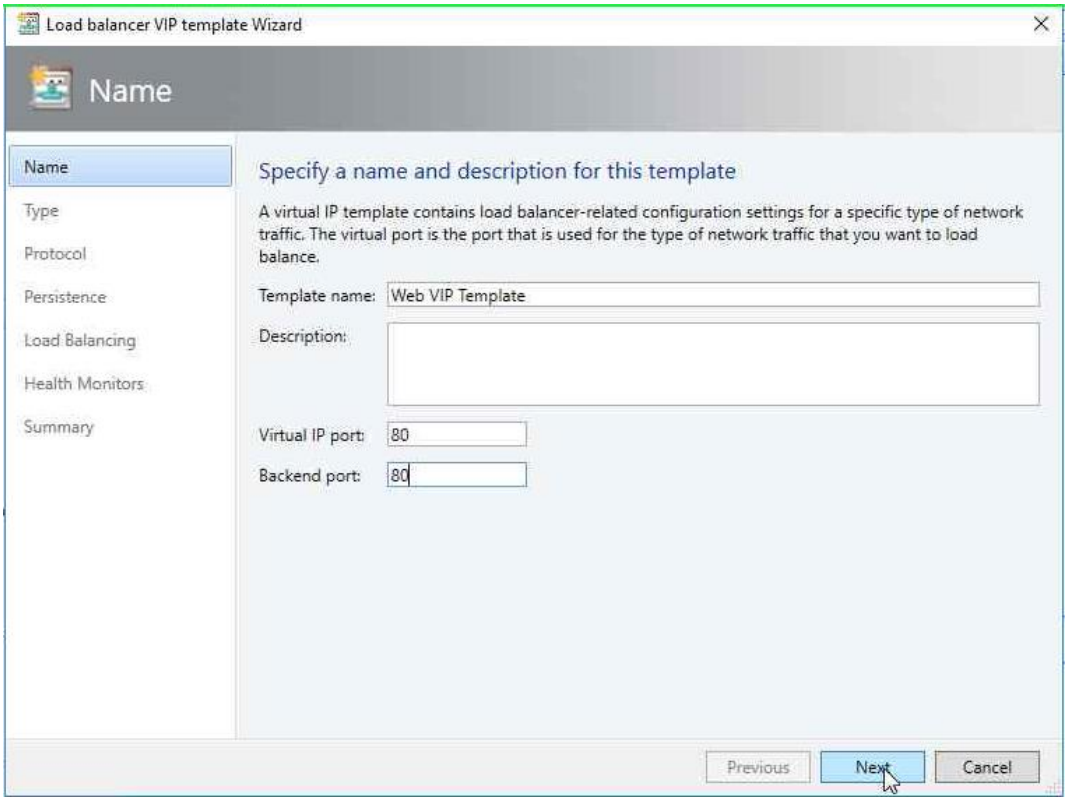
Um den Zugriff auf Tenant VMs mit Hilfe des Software Load Balancing in unserer Lab-Umgebung testen zu können, müssen wir öffentliche virtuelle Adressen (*PublicVIPs)* definieren.

10.5.1 Erstellen eines *VIP Templates*

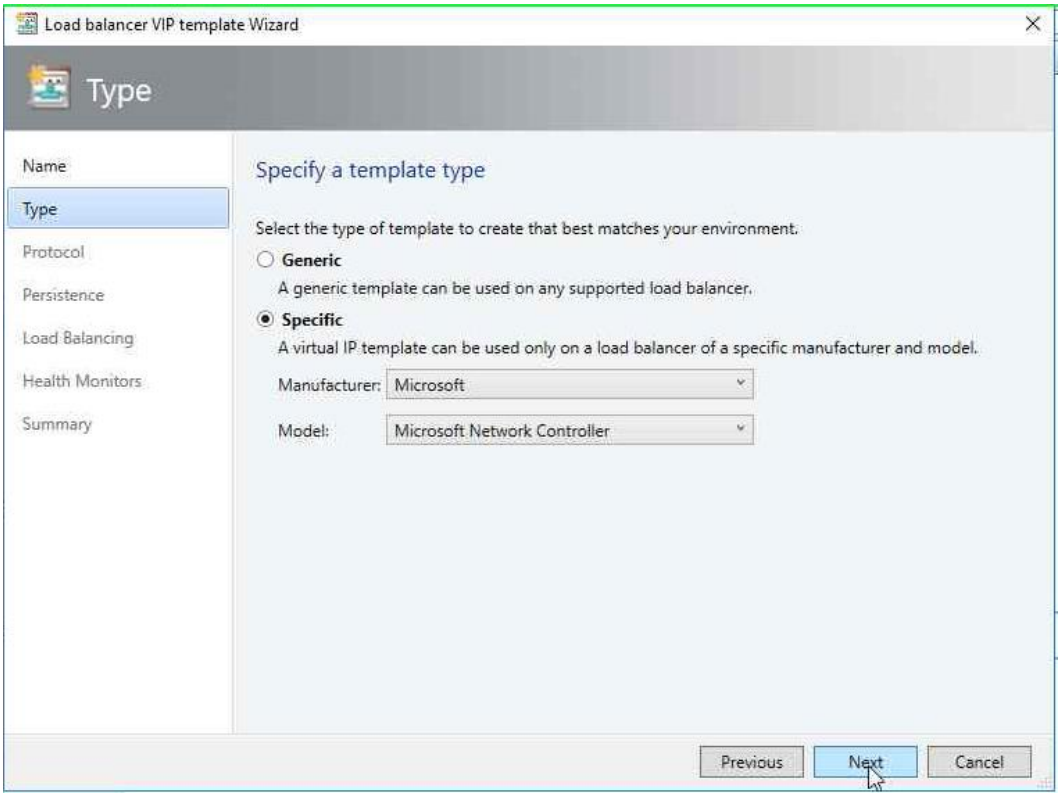
In der VMM-Konsole erstellen wir zunächst ein *VIP Template*. Dazu wechseln Sie in den *Fabric* Arbeitsbereich, öffnen die Kategorie *Networking* und klicken mit der rechten Maustaste auf *VIP Templates*. Wählen Sie im Kontextmenü *Create VIP Template*.



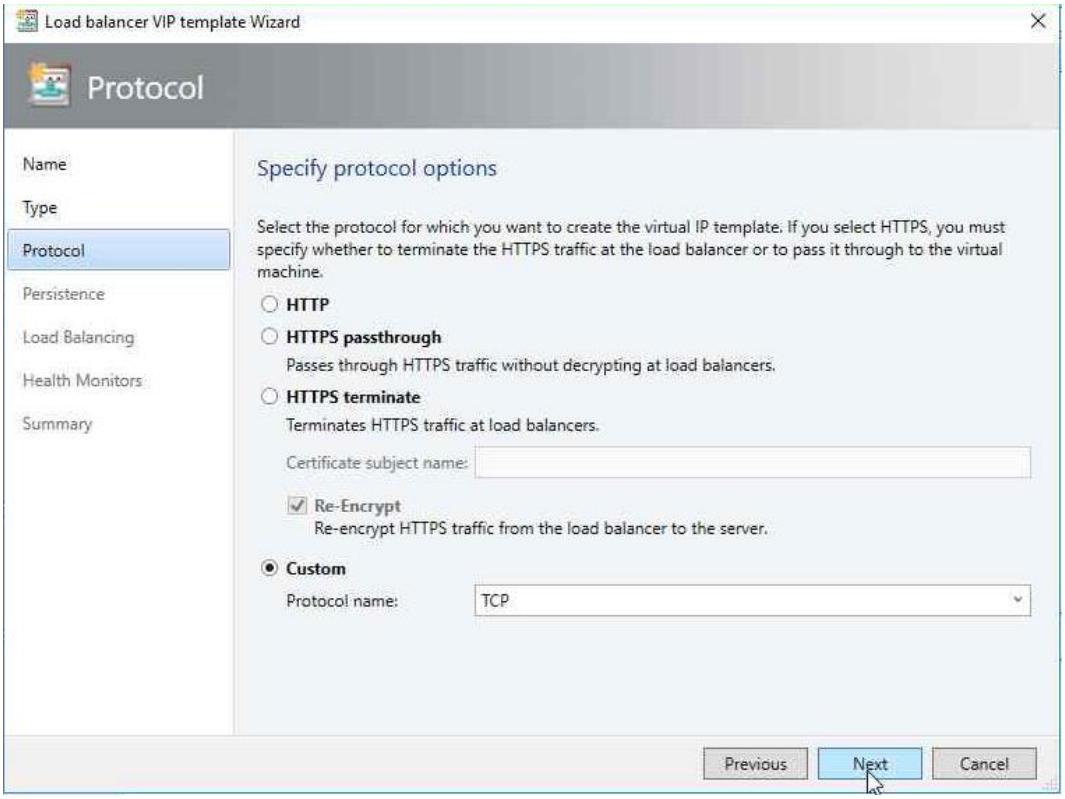
Geben Sie einen Namen für das VIP Template ein (z.B. *Web VIP Template*) sowie die Ports, zwischen denen die Netzwerkdaten ausgetauscht werden sollen (hier: jeweils Port 80). Klicken Sie auf *Next*.



Wählen Sie als Template Typ *Specific* und selektieren Sie in der *Manufacturer* Liste *Microsoft* sowie in der *Model* Liste *Microsoft Network Controller*. Klicken Sie auf *Next*.

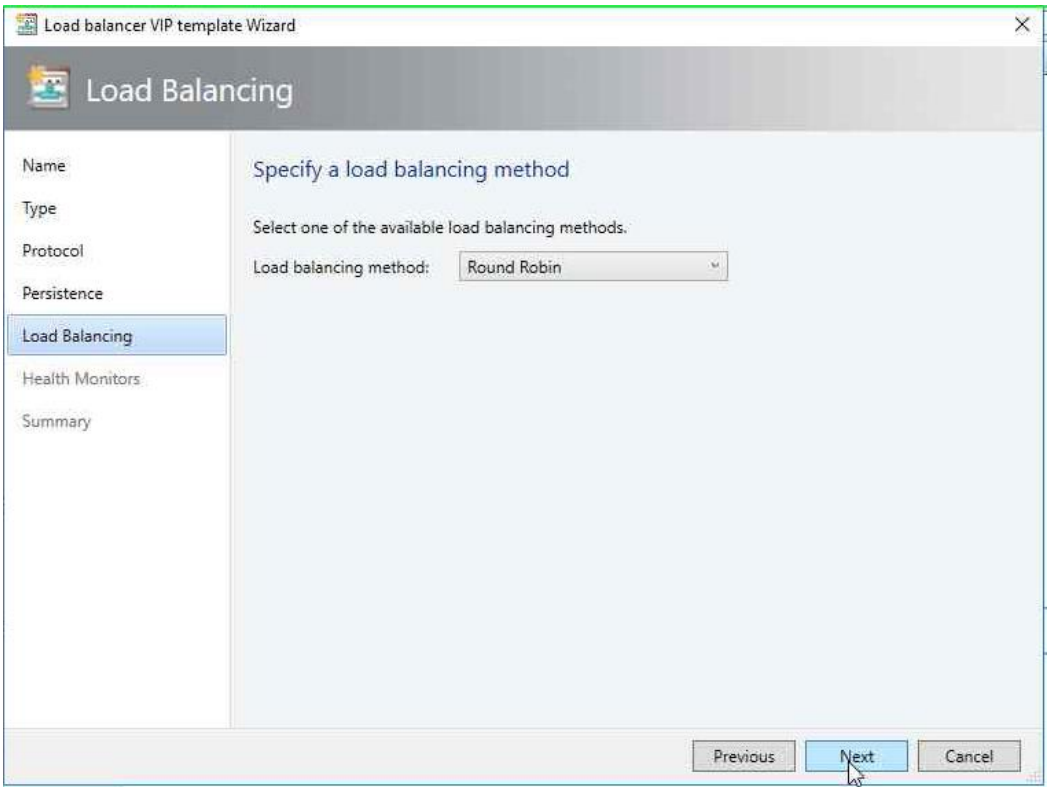


Im *Protocol* Dialog wählen Sie die Option *Custom* und geben als *Protokoll Name* den Text *TCP* ein. Klicken Sie wieder auf *Next*.

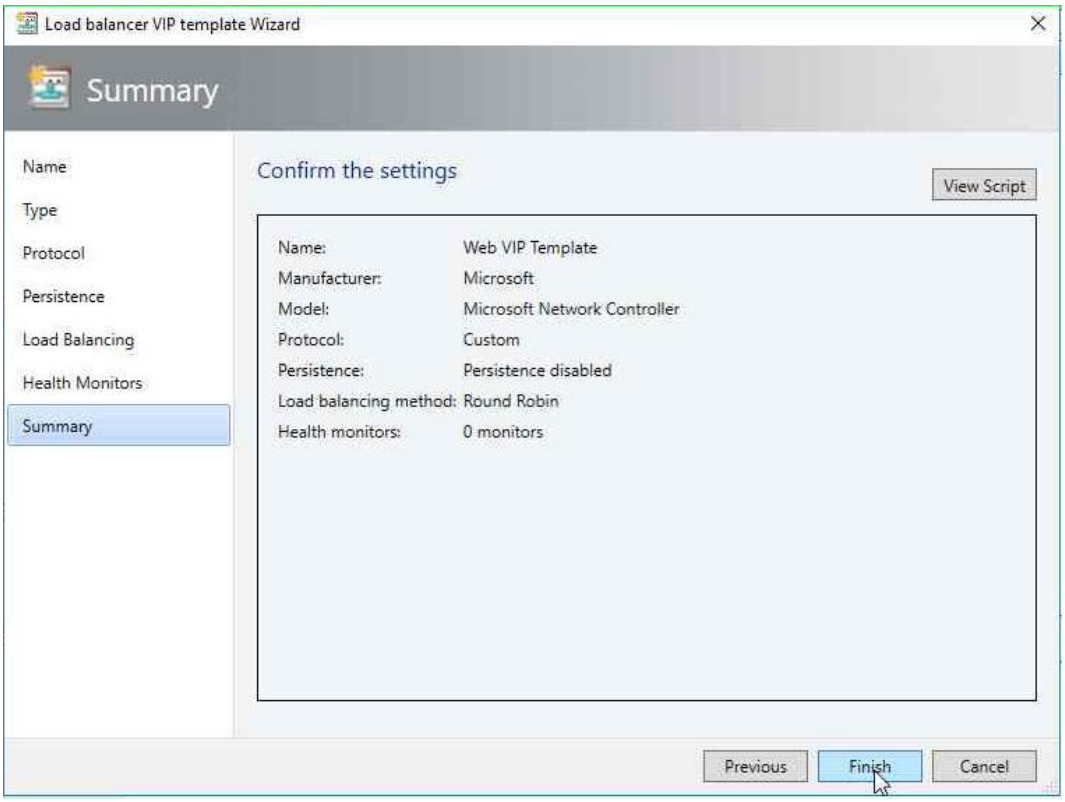


Die Seite *Persistance* können wir direkt mit *Next* überspringen.

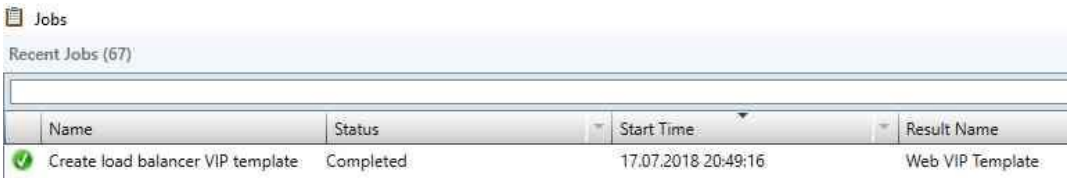
Auf der Seite *Load Balancing* wählen Sie aus der Liste *Load Balancing Method* den Eintrag *Round Robin* und klicken dann auf *Next*.



Die Seite *Health Monitors* überspringen wir mit *Next* und gelangen auf die *Summary* Seite.



Nach dem Klick auf *Finish* wird nun das VIP Template angelegt.



10.5.2 Erstellen einer öffentlichen virtuellen IP Adresse (*PublicVIP*) für ein Tenant VM Netz

Um eine *PublicVIP* für eines unserer *Tenant VM Netze* zu erstellen, müssen wir wieder die PowerShell verwenden. Führen Sie folgendes Skript aus:

```
param(

[Parameter(Mandatory=$false)]
# Name of the Network Controller Network Service
# This value should be the name you gave the Network Controller service
# when you on-boarded the Network Controller to VMM
$LBServiceName = "Network Controller",

[Parameter(Mandatory=$false)]
# Name of the VM instances to which you want to assign the VIP
$VipMemberVMNames =  @("Red-VM01", "Red-VM02"),

[Parameter(Mandatory=$false)]
# VIP address you want to assign from the VIP pool.
# Pick any VIP that falls within your VIP IP Pool range.
$VipAddress = "10.10.50.100",

[Parameter(Mandatory=$false)]
# Name of the VIP VM Network
$VipNetworkName = "PublicVIP",
```

```
[Parameter(Mandatory=$false)]
# The name of the VIP template you created via the VMM Console.
$VipTemplateName = "Web VIP Template",

[Parameter(Mandatory=$false)]
# Arbitrary but good to match the VIP you're using.
$VipName = "Red-VIP"
)

Import-Module virtualmachinemanager

$lb = Get-scLoadBalancer | where { $_.Service.Name -eq $LBServiceName};
$vipNetwork = get-scvmnetwork -Name $VipNetworkName;

$vipMemberNics = @();
foreach ($vmName in $VipMemberVMNames)
{
$vm = get-scvirtualmachine -Name $vmName;
#   if ($vm.VirtualNetworkAdapters[0].VMNetwork.ID -ne $vipNetwork.ID)
#   {
#      $vm.VirtualNetworkAdapters[0] | set-scvirtualnetworkadapter -VMNetwork $vipNetwork;
#   }

$vipMemberNics += $vm.VirtualNetworkAdapters[0];
}

$existingVip = get-scloadbalancervip -Name $VipName
if ($existingVip -ne $null)
{
#   foreach ($mem in $existingVip.VipMembers)
#   {
#      $mem | remove-scloadbalancervipmember;
#   }

$existingVip | remove-scloadbalancervip;
}

$vipt = get-scloadbalancerviptemplate -Name $VipTemplateName;

$vip = New-SCLoadBalancerVIP -Name $VipName -LoadBalancer $lb -IPAddress $VipAddress -LoadBalancerVIPTemplate
$vipt -FrontEndVMNetwork $vipNetwork -BackEndVirtualNetworkAdapters $vipMemberNics;
Write-Output "Created VIP " $vip;

$vip = get-scloadbalancervip -Name $VipName;
Write-Output "VIP with members " $vip;
```

Anmerkung: Die Parameter dieses Skripts sind so vorbelegt, dass Sie direkt die *PublicVIP* 10.10.50.100 anlegen können, um ein Load Balancing zwischen den VMs *Red-VM01* und *Red-VM02* zu definieren.

Analog können Sie auch eine *PublicVIP* 10.10.50.101 für ein Load Balancing zwischen den Tenant VMs *Green-VM01* und *Green.VM02* zu definieren.

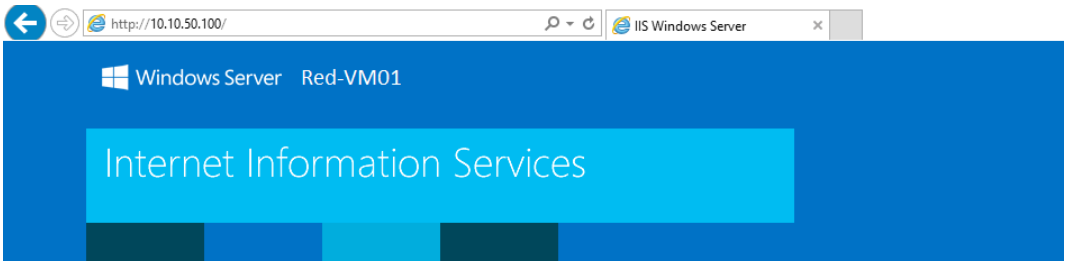
Mit dem CmdLet *Get-SCLoadBalancerVIP* können Sie sich nun eine Liste aller definierten *PublicVIPs* anzeigen lassen.

```
PS C:\Windows\system32> Get-SCLoadBalancerVIP


Name                            : Green-VIP
Address                         : 10.10.50.101
IPAddress                       : 10.10.50.101
LoadBalancer                    : https://192.168.80.230
Protocol                        : TCP
Port                            : 80
IsPersistenceEnabled            : False
Persistence                     :
LoadBalancingMethod             : RoundRobin
HealthMonitors                  : {}
VIPMembers                      : {192.168.1.4, 192.168.0.4}
state                           : Deployed
LoadBalancerConfigurationName   :
FrontEndVMNetwork               : PublicVIP
ServerConnection                : Microsoft.SystemCenter.VirtualMachineManager.Remoting.ServerConnection
ID                              : 6e0e01ff-76e2-4436-80b8-439a8f802b4d
IsViewOnly                      : False
ObjectType                      : LoadBalancerVIP
MarkedForDeletion               : False
IsFullyCached                   : True

Name                            : Red-VIP
Address                         : 10.10.50.100
IPAddress                       : 10.10.50.100
LoadBalancer                    : https://192.168.80.230
Protocol                        : TCP
Port                            : 80
IsPersistenceEnabled            : False
Persistence                     :
LoadBalancingMethod             : RoundRobin
HealthMonitors                  : {}
VIPMembers                      : {192.168.0.4, 192.168.1.4}
state                           : Deployed
LoadBalancerConfigurationName   :
FrontEndVMNetwork               : PublicVIP
ServerConnection                : Microsoft.SystemCenter.VirtualMachineManager.Remoting.ServerConnection
ID                              : 5a029f09-e199-41c2-b885-e689ec0441ae
IsViewOnly                      : False
ObjectType                      : LoadBalancerVIP
MarkedForDeletion               : False
IsFullyCached                   : True
```

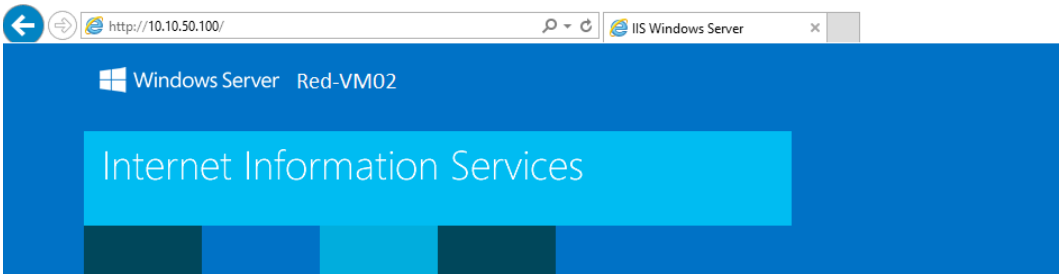
10.5.3 Load Balancing Test

Nachdem wir nun für unsere Tenant VMs öffentliche IP-Adressen definiert haben, können wir das Load Balancing einmal ausprobieren.

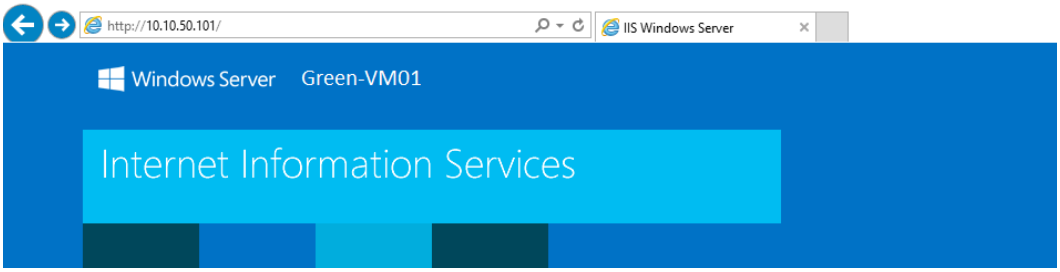
Starten Sie auf einem System, das Zugriff auf das *PublicVIP* Netz hat – z.B. unser Domain Controller *SDN-DC01* – einen Browser und geben in die Adresszeile *http://10.10.50.100* (die VIP des Red Tenants) ein. Sie sollten folgende Anzeige erhalten:

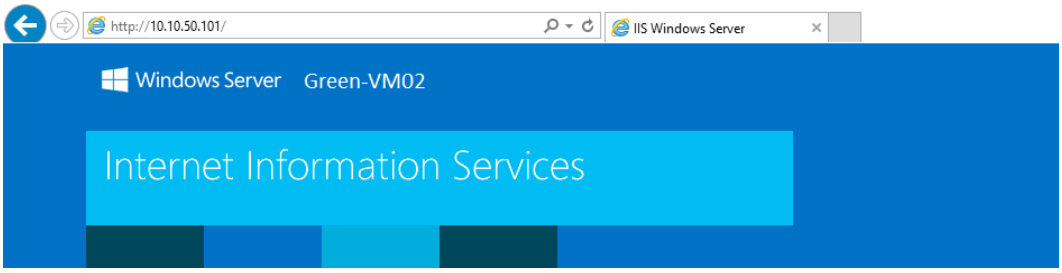


Starten Sie eine weitere Browser Instanz und geben die gleiche Adresse ein. Ergebnis:



Geben Sie in die Adressleisten der Browser die *PublicVIP* des Green Tenants ein (*http://10.10.50.101*) und Sie erhalten folgende Anzeige:

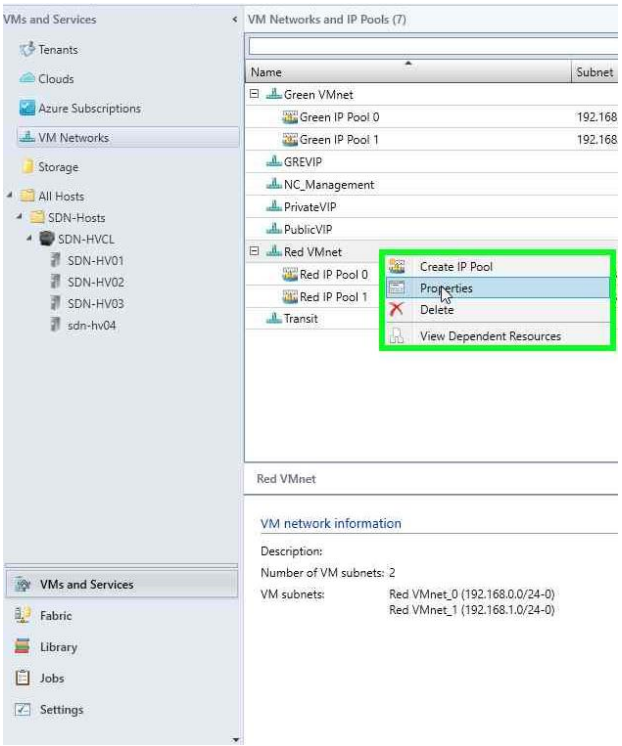




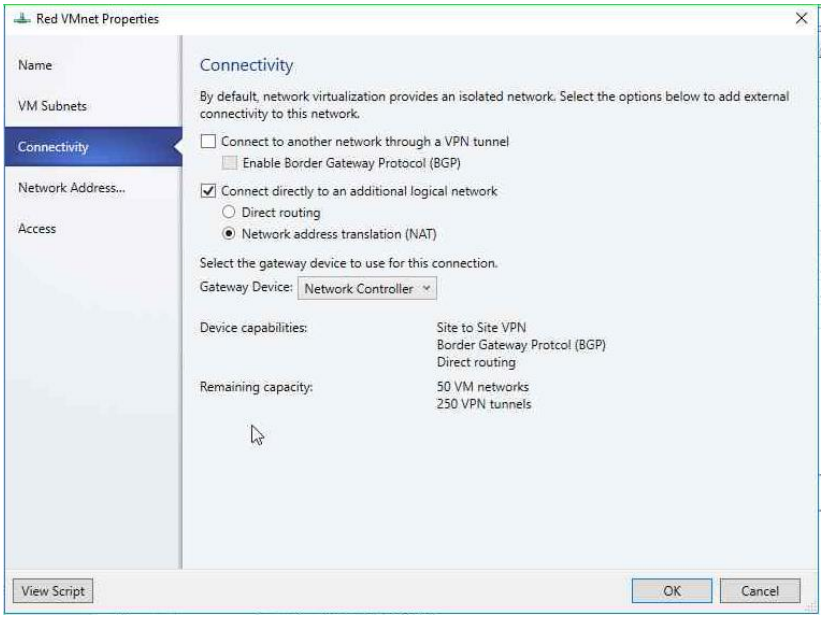
10.5.4 NAT konfigurieren

Bislang können die Tenant VMs nur in ihrem eigenen VMnet kommunizieren. Nachdem wir die SLB-Instanzen konfiguriert haben, können wir ihnen nun auch den Zugriff auf externe Ressourcen per NAT (Netzwerk Address Translation) einrichten.

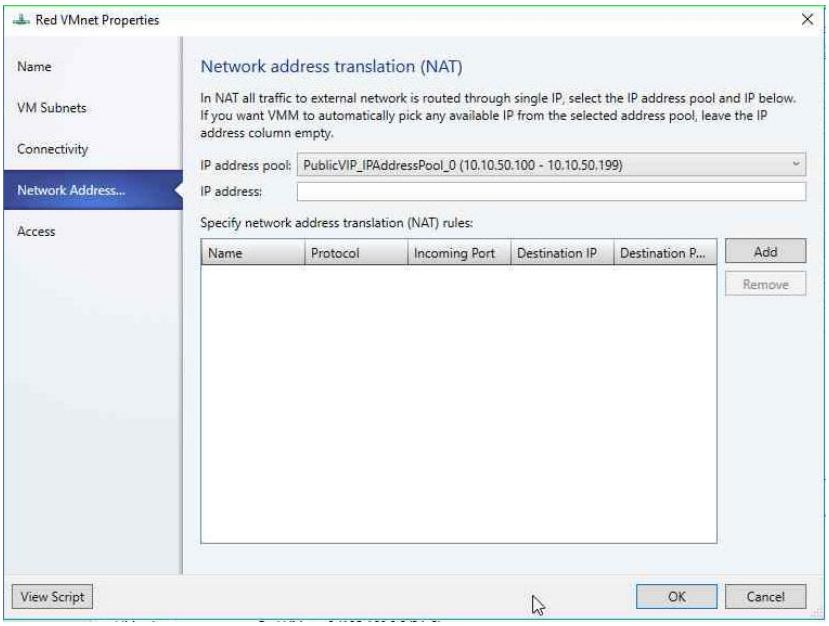
Wechseln Sie in der VMM-Konsole in den Arbeitsbereich *VMs and Services*, wählen die Kategorie *VM Networks* und klicken Sie mit der rechten Maustaste auf das VMnet, für das Sie NAT aktivieren wollen. Rufen Sie die *Properties* auf.



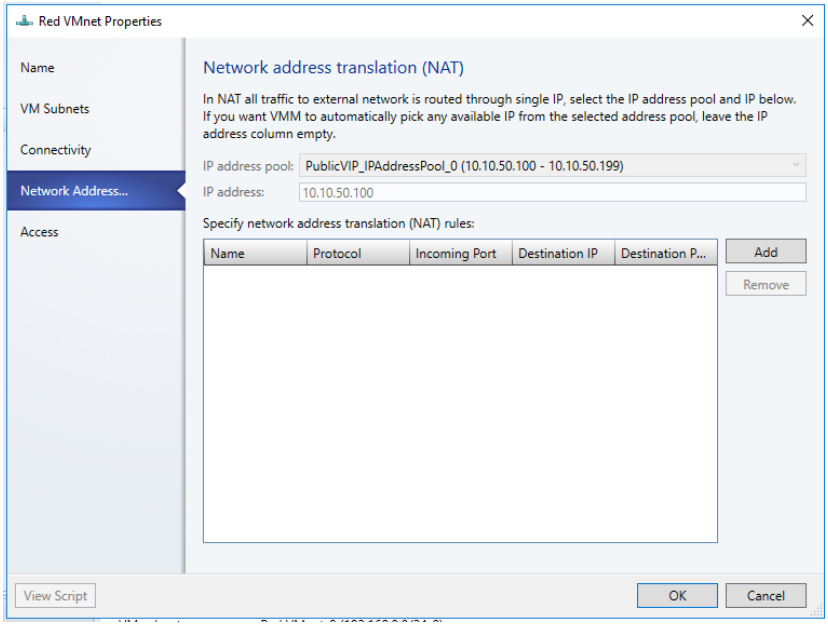
Wählen Sie die Registerkarte *Connectivity*. Aktivieren Sie das Kontrollkästchen *Connect directly to an additional logical network* und wählen Sie darunter die Option *Network Address Translation (NAT)* – im linken Bereich erscheint dabei ein Reiter für eine weitere Registerkarte *Network Address…* Stellen Sie sicher, dass im Feld *Gateway Device* der *Network Controller* eingestellt ist.



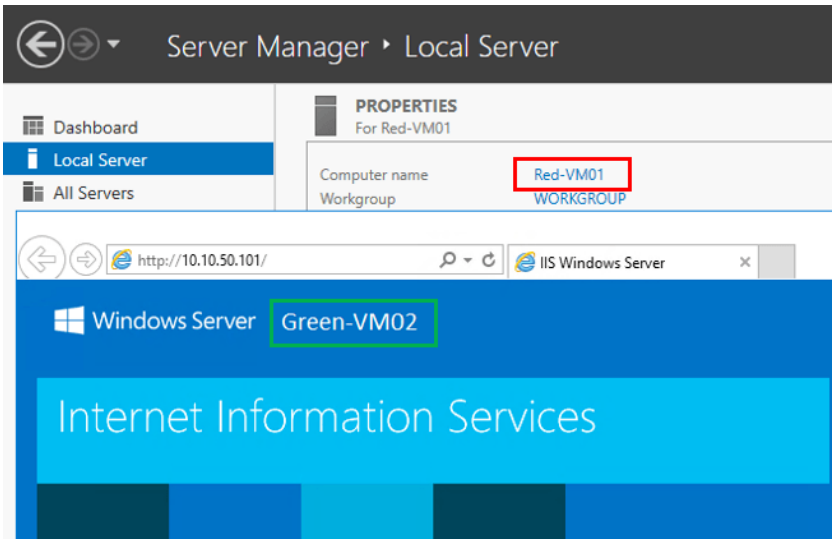
Klicken Sie jetzt auf den neuen Reiter für die Registerkarte *Network Address…* Im Feld *IP Address Pool* wählen Sie den *PublicVIP_IPAddressPool_0*. Das Feld *IP Address* können Sie freilassen. Vom VMM wird dann eine geeignete IP-Adresse aus dem gewählten Pool eingesetzt. Klicken Sie auf *OK*.



Zur Kontrolle können Sie nochmals die *Properties des VMnets* aufrufen. Auf der Registerkarte *Network Address* ist jetzt das Feld *IP Address* ausgefüllt (hier: 10.10.50.100).



Jetzt könnten Sie von einer VM im *Red VMNet* eine externe Webseite aufrufen. In unserer Lab-Umgebung fehlt jedoch die Verbindung zwischen dem *PublicVIP* Netz und dem öffentlichen Internet. Deshalb können Sie keine externen Ressourcen im Internet ansprechen. Sie können aber beispielsweise versuchen, über die öffentliche VIP eines anderen Tenant Netzes die IIS-Startseite einer VM zu erreichen, z.B.



Jetzt können wir auch noch NAT-Regeln für eingehende Netzwerkpakete von unserer öffentlichen IP-Adresse definieren.

Öffnen Sie wieder die Properties des gewünschten Tenant VMnets und wechseln Sie auf die Registerkarte *Network Address...*

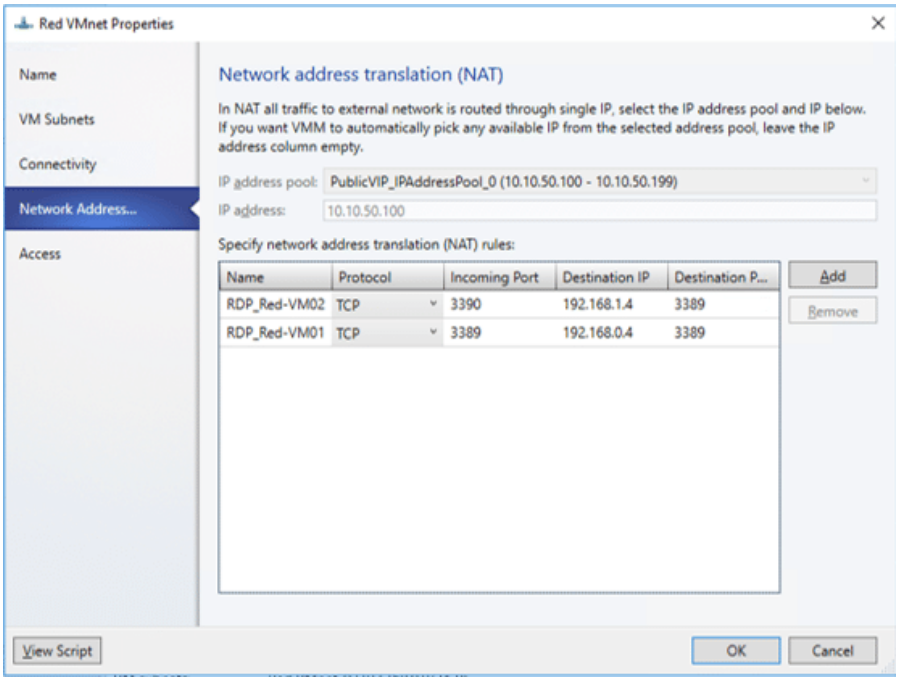
Über die Schaltfläche *Add* können Sie jetzt Regeln für den eingehenden Netzwerkverkehr konfigurieren. Um z.B. eine Remote Desktop Verbindung (RDP) von einem externen System zum System Red-VM01 herstellen zu können, geben Sie folgende Parameter ein:

Name	Geben Sie einen Namen für die eingehende NAT-Verbindung ein, z.B. *RDP_Red-VM01*
Protocol	Wählen Sie das Protokoll (**TCP** oder UDP) der eingehenden Verbindung.

Incoming Port	Geben Sie die Portnummer für die eingehenden Netzwerkdaten an, hier **3389** (Standard RDP Port)
Destination IP	Geben Sie die IP-Adresse der Tenant VM ein, an die die Daten weitergeben werden sollen, z.B. für die *Red-VM01* die Adresse 192.168.0.4
Destination Port	Portnummer innerhalb der Tenant VM als Ziel der Datenweiterleitung (RDP-Standard 3389)

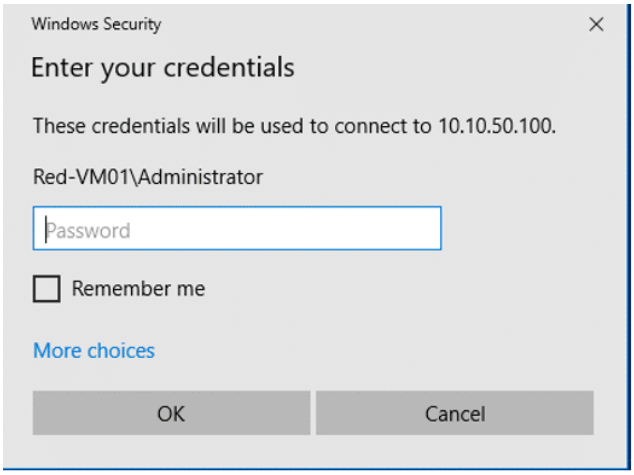
Und wenn Sie gerade schon dabei sind, konfigurieren Sie auch gleich eine NAT-Regel für RDP zum System *Red-VM02*. Diesen Datenverkehr erwarten wir auf dem Port 3390 und leiten ihn an den Port 3389 der Ziel-VM weiter.

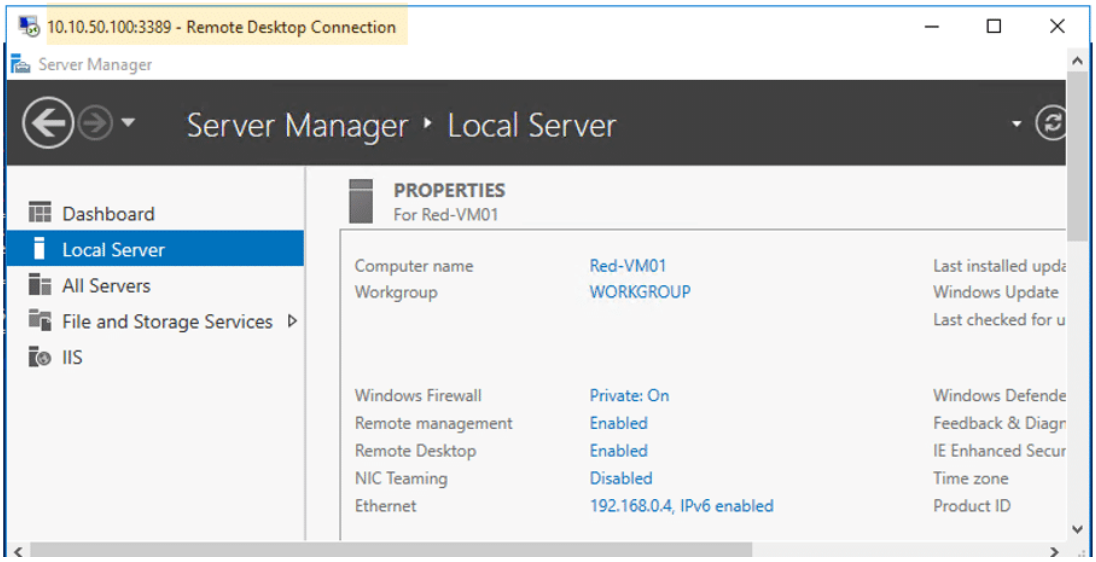
Name	Geben Sie den Namen der eingehenden NAT-Verbindung ein, z.B. *RDP_Red-VM02*
Protocol	Wählen Sie das Protokoll (**TCP** oder UDP) der eingehenden Verbindung.
Incoming Port	Geben Sie die Portnummer für die eingehenden Netzwerkdaten an, hier **3390**
Destination IP	Geben Sie die IP-Adresse der Tenant VM ein, an die die Daten weitergeben werden sollen, z.B. für die *Red-VM02* die Adresse 192.168.1.4
Destination Port	Portnummer innerhalb der Tenant VM als Ziel der Datenweiterleitung (RDP-Standard 3389)



Jetzt können Sie von einem System in unserem *PublicVIP* Netz RDP-Verbindungen zu den Tenant VMs herstellen, z.B. zu *Red-VM01*:

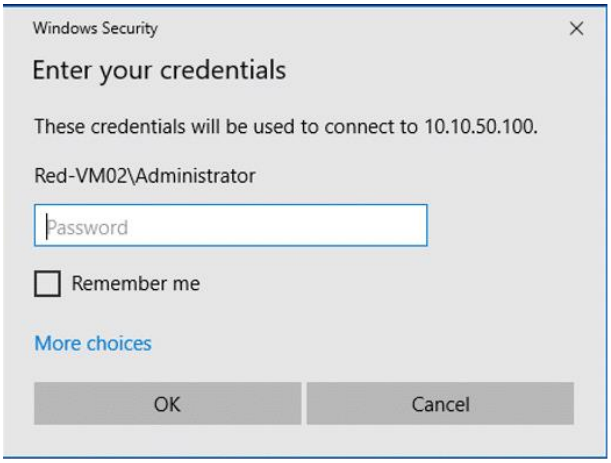
```
mstsc /v 10.10.50.100:3389
```

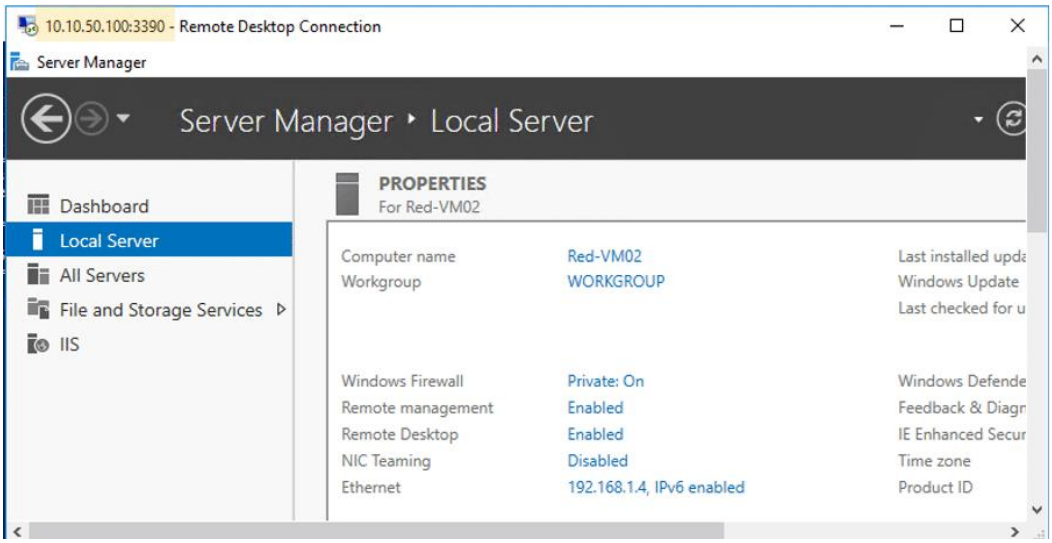




Für eine RDP-Verbindung zu *Red-VM02* geben Sie die in der NAT-Regel hinterlegte Portnummer 3390 an:

```
mstsc /v 10.10.50.100:3390
```





Die NAT-Konfiguration und Regeln für ein VMnet können Sie sich auch mit der PowerShell anzeigen lassen:

```
$vnet = Get-SCVMNetwork -Name "Red VMnet"
Write-Output "NAT Connections and Rules for VM Network: $vnet"
$NATconn = Get-SCNATConnection -VMNetwork $vnet
Write-Output $NATconn
$NATrules = Get-SCNATRule -NATConnection $NAT
Write-Output $NATrules
```

```
NAT Connections and Rules for VM Network: Red VMnet


Name                          : Red VMnet_NatConnection
Description                   :
Status                        : Enabled
MaximumBandwidthInboundKbps   :
MaximumBandwidthOutboundKbps  :
VMNetworkGateway              :
VMNetwork                     : Red VMnet
Rules                         : {RDP_Red-VM02, RDP_Red-VM01, 10.10.50.100}
ServerConnection              :
Microsoft.SystemCenter.VirtualMachineManager.Remoting.ServerConnection
ID                            : cab624bb-781c-4507-b858-9841e07db1c5
IsViewOnly                    : False
ObjectType                    : NATConnection
MarkedForDeletion             : False
IsFullyCached                 : True

Name              : RDP_Red-VM02
ExternalIPAddress : 10.10.50.100
```

```
ExternalPort      : 3390
InternalIPAddress : 192.168.1.4
InternalPort      : 3389
Protocol          : TCP
NATConnection     : Red VMnet_NatConnection
ServerConnection  : Microsoft.SystemCenter.VirtualMachineManager.Remoting.ServerConnection
ID                : b77e7714-c034-44d5-9abe-680f94139c0f
IsViewOnly        : False
ObjectType        : NATRule
MarkedForDeletion : False
IsFullyCached     : True

Name              : RDP_Red-VM01
ExternalIPAddress : 10.10.50.100
ExternalPort      : 3389
InternalIPAddress : 192.168.0.4
InternalPort      : 3389
Protocol          : TCP
NATConnection     : Red VMnet_NatConnection
ServerConnection  : Microsoft.SystemCenter.VirtualMachineManager.Remoting.ServerConnection
ID                : 0821898a-5aee-4f53-b70c-92187a331cca
IsViewOnly        : False
ObjectType        : NATRule
MarkedForDeletion : False
IsFullyCached     : True

Name              : 10.10.50.100
ExternalIPAddress : 10.10.50.100
ExternalPort      : 0
InternalIPAddress :
InternalPort      :
Protocol          : TCP
NATConnection     : Red VMnet_NatConnection
ServerConnection  : Microsoft.SystemCenter.VirtualMachineManager.Remoting.ServerConnection
ID                : 7dc93b3e-fd97-408f-b517-b20e9f4f0b9d
IsViewOnly        : False
ObjectType        : NATRule
MarkedForDeletion : False
IsFullyCached     : True
```

11 Nächste Schritte

In den vorstehenden Abschnitten habe ich versucht, einige der Testszenarien, die Petra Lipp und ich bei unserer Session auf der CDC 2018 präsentiert haben, etwas detaillierter zu beschreiben.

Nun ist das nur ein Bruchteil dessen, was die Microsoft SDN-Technologie bietet. Was wir hier überhaupt nicht weiter betrachtet haben, sind Remote Szenarien, um z.B. ein Unternehmensnetz mit einem *VMnet* in unserer SDN-Umgebung zu verbinden.

Eine weitere Herausforderung wäre auch, wie Service Provider SDN-Technologien ihren Kunden über Self Service Portale anbieten können. Eine Lösung dafür wäre z.B. das gute alte Windows Azure Pack, das sich hervorragend in eine SDN-Umgebung integrieren lässt. Petra und ich haben auf der CDC2018 auch dazu eine Session gehalten und die Implementierung eines Kunden vorgestellt.

Hierzu müssten wir aber unsere Lab-Umgebung jedoch um weitere Infrastruktur- und Remote-Systeme erweitern, was aber irgendwann unsere Ressourcen und den Rahmen dieses Whitepapers sprengen würde.

Bei Bedarf können wir gerne in weiteren Dokumenten und persönlichen Beratungsgesprächen darauf eingehen. Im Moment kann ich Sie nur auf die offizielle Microsoft Dokumentation in der TechNet-Library verweisen, die zugegebenermaßen zumindest in der englischen Originalfassung mal relativ ausführlich und gut verständlich ist. Die (maschinelle) deutsche Übersetzung sollten Sie mit Vorsicht genießen.

Neben dieser offiziellen Dokumentation gibt es auch noch eine Reihe sehr empfehlenswerter Blog-Beiträge verschiedener Microsoft Gurus und MVPs, die mir beim Erstellen und dem Aufbau der Lab-Umgebung wertvolle Informationen geliefert haben. Ohne Anspruch auf Vollständigkeit möchte ich hier auf folgende Artikelserien verweisen (in Englisch):

Schumann Ge's Blog: Deploying SDN on One single physical host using VMM

Larryexchange Blog:

- Step-by-step for deploying a SDNv2 using VMM – Part 1-4
- Configure WAP to support new SDN stack on Windows Server 2016

Und die Entwicklung wird weitergehen. Im Herbst dieses Jahres wird die nächste Version von Windows Server unter dem Namen *Windows Server 2019* erscheinen und voraussichtlich einige Neuerungen zum Thema SDN bringen, über die wir natürlich zeitnah berichten werden.

Ich hoffe, dass dieses Whitepaper Ihnen einen Überblick geben konnte zu den Microsoft SDN Technologien mit Windows Server 2016 und dem SCVMM 2016 und Ihnen auch zeigen konnte, wie Sie eine SDN-Umgebung mit Hilfe der von Microsoft auf GitHub bereitgestellten PowerShell Skripte ausrollen können.

Für weitere Fragen, Anregungen und Kommentare stehen wir gerne zur Verfügung.

Anhang: Skript Download Details

In diesem Whitepaper habe ich einige PowerShell Skripte beschrieben, die ich mir zusätzlich zu den von Microsoft auf GitHub bereitgestellten erstellt habe, um Aktionen, die mit vielen Mausklicks verbunden sind, etwas zu vereinfachen. Diese zusätzlichen Skripte habe ich in einer .ZIP-Datei zusammengestellt, die Sie hier downloaden können. Nachstehend finden Sie Details zum Inhalt dieser .ZIP-Datei.

Dateiname: *VMM-SDN.zip* – Entpacken Sie den Inhalt dieser Datei in das *SDN-Master* Verzeichnis, das Sie nach dem Download der Microsoft Skripte aus GitHub erstellt haben.

Inhalt:

VMM\scripts\CreateTenantVMnet-Red.ps1 – erzeugt das *Red VMnet*
VMM\scripts\CreateTenantVMnet-Green.ps1 – erzeugt das *Green VMnet*
VMM\scripts\CreateTenantVMnet-Red.ps1 – erzeugt VIP im *PublicVIP*-Netz für den *Red Tenant*
VMM\scripts\CreateTenantVMnet-Green.ps1 – erzeugt VIP im *PublicVIP*-Netz für den *Green Tenant*
VMM\VMM SDN Express\Fabricconfig-SDNcloud-Production.psd1 –
Konfigurationsdatendatei für das Deployment der Lab-Umgebung mit *VMMexpress.ps1*