Deploying a Dedicated Hyper-V Management Guest using PowerShell and Dell EqualLogic Storage Tools

Dell Storage Engineering
May 2014

A Dell Deployment and Configuration Guide

Revisions

Date	Description
May 2014	Initial release

Table of contents

Executive summary

With increasing pressure to reduce business costs and increase service availability, virtualization is very prevalent in today's datacenters. Microsoft Windows Server Hyper-V virtualization is commonly used by businesses to minimize total cost of ownership (TCO), streamline management of both physical and virtual machines, and increase availability of business applications. Part of Microsoft's Hyper-V best practices is to deploy Hyper-V on Windows Server Core Hosts. This creates complications as in Server Core, the familiar Windows GUI tools are not available and administrator must use PowerShell or command line tools to setup and manage the environment. This paper will demonstrate to the reader how to set up an initial virtual machine on a configured Server Core Hyper-V host, and then convert that virtual machine to be dedicated for Hyper-V environment management.

Audience

This is paper is a technical solution guide intended for Information Technology administrators in the SMB business space. This paper focuses on how an administrator can manage and protect a Windows Server Hyper-V environment using PowerShell and Dell EqualLogic PS Series storage and tools. This paper assumes that the reader has:

- Previous Windows Server administration experience which has included installation and configuration
- Understanding of virtualization technologies
- Some familiarity with Microsoft PowerShell or some other scripting language

The examples in this guide will include extensive amounts of PowerShell commands. It is strongly recommended that the reader have some basic understanding of PowerShell prior to attempting the examples in the document. A detailed discussion on PowerShell and its capabilities goes beyond the scope of this paper, but it is vitally important that Windows and Hyper-V administrators get some understanding and familiarity with this powerful tool. For more information on PowerShell, consult the following resources:

- Dell Technical Report: *Windows Command-line Automation Techniques for Dell EqualLogic PS Series Arrays* at http://en.community.dell.com/dell-groups/dtcmedia/m/mediagallery/20421645.aspx
- Ed Wilson, *Windows PowerShell 3.0 Step by Step*, O'Reilly Media, Inc., Sebastopol, CA, 2013.
- Don Jones and Jeffery Hicks, *Learn Windows PowerShell 3 in a Month of Lunches*, Manning Publications Co., Shelter Island, NY, 2013.

Although Microsoft's System Center is a key component of any Hyper-V virtualization strategy, especially in larger environments, providing a detailed discussion on System Center is also beyond the scope of this paper. This paper focuses on small to medium Hyper-V deployments where System Center functionality is not yet required. For more information on Microsoft System Center and its integration with Dell EqualLogic Storage, consult the Dell Technical Report: *Automation and Integration with Microsoft System Center Virtual Machine Manager 2012 SP1 and Dell EqualLogic Storage* at http://en.community.dell.com/techcenter/extras/m/white_papers/20437936.aspx.

Software information

The following table shows the software and firmware used for the preparation of this Technical Report.

Vendor	Model	Software Revision
Dell	Host Integration Tools for Microsoft (HIT/ME)	V4.7
Dell	Dell EqualLogic Firmware	V7.0+
Microsoft	Windows Server 2012	
Microsoft	Windows Server 2012 R2	

Referenced documentation

The following table lists the documents referred to in this Technical Report. All Dell EqualLogic Technical Reports are available on the Customer Support site at eqlsupport.dell.com

Vendor	Document Title
Dell	Dell EqualLogic Host Integration Tools for Microsoft, Installation and User's Guide
Dell	Dell EqualLogic Auto-Snapshot Manager/Microsoft Edition User's Guide
Dell	Dell EqualLogic PowerShell Tools User Guide
Dell	Dell EqualLogic Host Integration Tools for Microsoft Release Notes
Dell	Dell EqualLogic Storage with Heterogeneous Virtualized Workloads on Microsoft Server 2012 with Hyper-V

Introduction: Creating the first virtual machine on the Server Core Hyper-V host

Once the Hyper-V host has been set up and properly configured, the administrator can begin building virtual machines (VMs or guests). On a Hyper-V server with a GUI installation, creating VMs is a simple task that can be done through wizards in the Hyper-V Manager from the Server Manager console. However, performing this on a Server Core host can be a little more challenging. When creating new VMs with Server Core, automation through PowerShell scripting and other tools becomes very important. It allows for more accurate and faster VM deployments similar to the abilities provided by the GUI and Wizards.

To successfully automate the creation of a VM using PowerShell, a script must be able to complete the following tasks from the Server Core Hyper-V host.

1. Gather user input via direct entry or through configuration files about VM parameters
2. Create the new VM and its virtual hard drive based on the gathered parameters
3. Modify (as needed) the new VM parameters based on user input
4. Add vNICs and attach the vNICs to the Hyper-V host virtual switches
5. Start the VM and perform an automated installation of the operating system using an answer file
6. Set the network configuration for the VM vNIC

When creating a VM in Hyper-V, either for management or application workloads, an administrator must take into account the same major hardware subsystem considerations (primarily the requirements for CPU, memory, storage, and networking) as if it were a traditional physical server. These four subsystems are important to understand in the general sense because not planning accordingly for each of them individually can lead to overall poor workload performance in the virtualized environment. These four hardware subsystem components and their impact on Hyper-V workload performance are discussed in detail in the Dell Technical Report *Dell EqualLogic Storage with Heterogeneous Virtualized Workloads on Microsoft Server 2012 with Hyper-V* . Many of the best practices detailed and discussed in the Heterogeneous Virtualized Workloads report were implemented in the example used in this document.

This document uses a nine-step process to create the first VM on the Server Core Hyper-V host. The first steps of this process are performed using PowerShell commands from the Hyper-V host. The steps in this document to create VMs build a workflow that lends itself to amending parts of (or the entire) process.

Steps 1 through 8 in Figure 1 involve mostly PowerShell commands to create the initial VM. Step 9 details the optional conversion of the initial VM to a dedicated management VM for the Hyper-V host.

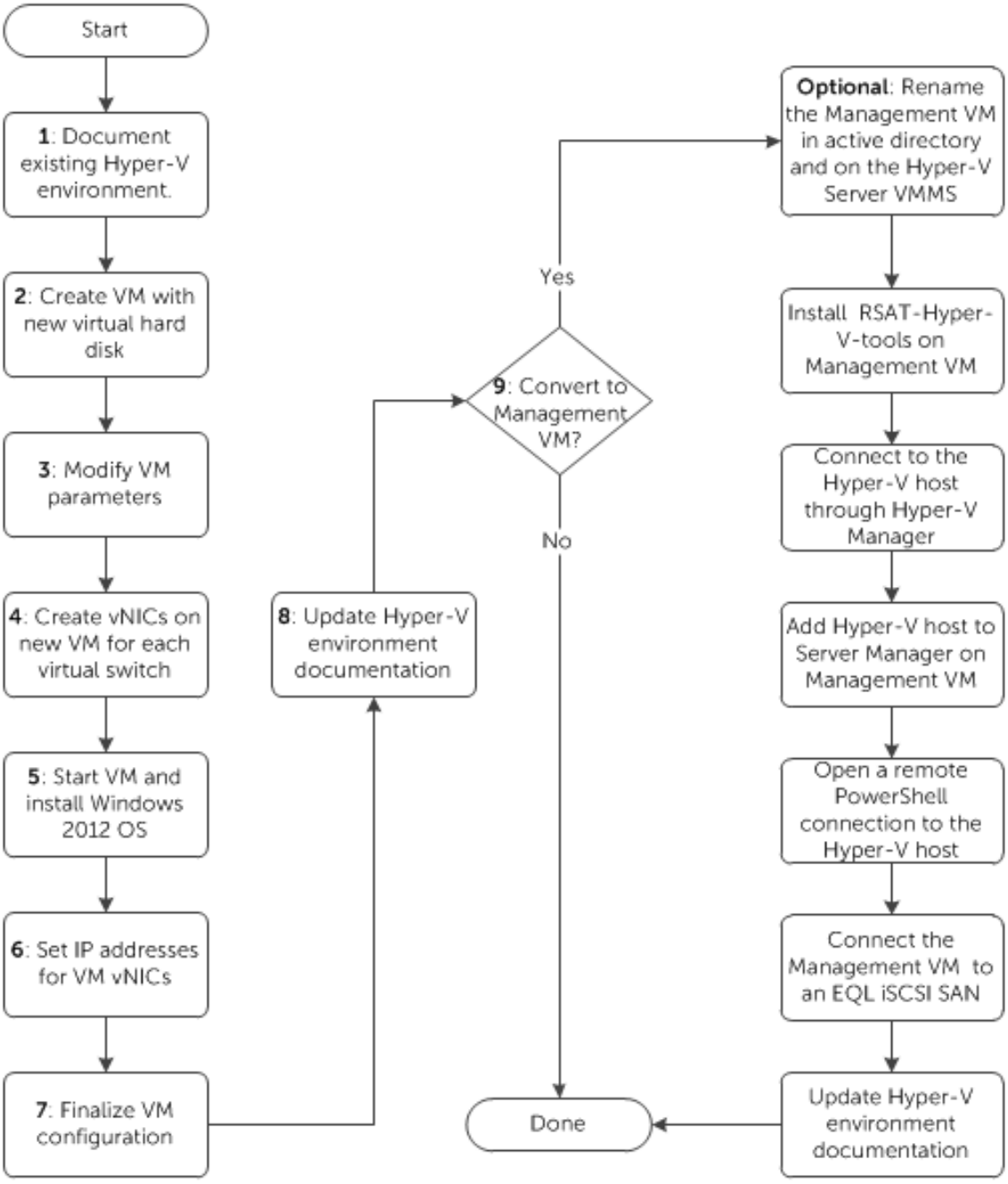


Figure 1 Creating a management guest process

1 Step 1: Document the current environment

Figure 2 provides an illustration of the sample Hyper-V host configuration layout used for the example in this document. This type of diagram is a great starting point for documenting the configuration of the initial VM.

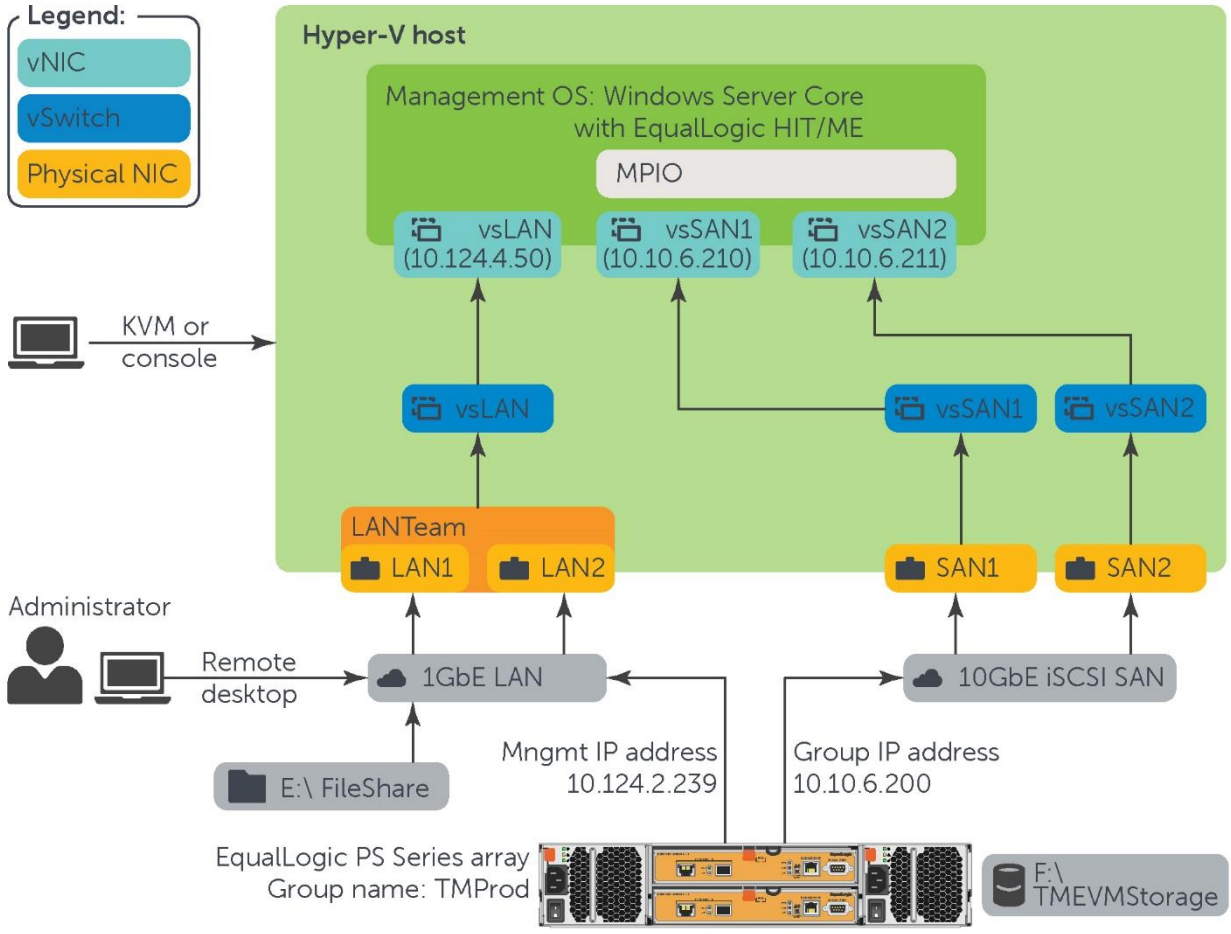


Figure 2 Initial configuration diagram

Table 1 Core Hyper-V host components in the diagram

Component	Description	Notes
LANTeam	NIC team of physical NICs LAN1 and LAN2	
SAN1	Physical NIC SAN1	
SAN2	Physical NIC SAN2	

Component	Description	Notes
vsLAN	External virtual switch associated to NIC team LANTeam	This switch allows access to Management OS. The vNIC, vsLAN, is for the Management OS; its assumed IP Address for the LANTeam is 10.140.6.50.
vsSAN1	External virtual switch associated to the physical NIC SAN1	This switch allows access to the Management OS. The vNIC, vsSAN1, is for the Management OS; its assumed IP Address for the physical NIC, SAN1, is 192.168.4.12.
vsSAN2	External virtual switch associated to Physical NIC SAN2	This switch allows access to the Management OS. The vNIC, vsSAN2, is for the Management OS; its assumed IP Address for the physical NIC, SAN2, is 192,168.4.13
C:\	Internal disk on Hyper-V host	
F:\	EqualLogic iSCSI volume TMEVMStorage	TMEVMStorage acts as a data store to house the VM, virtual hard disks and configuration files.
E:\	Network file share	The network file share stores utilities such as EqualLogic HIT Kit, PowerShell scripts, and Windows ISO files.

Before building any VMs, document the details of the existing Hyper-V host environment as demonstrated in Figure 2 and Table 1. Give special attention to the network configuration used by the various network components in the virtualized environment. Without carefully documenting network details such as used IP address, gateways, and subnet masks, there is a high probability of network conflicts developing within the virtualized environment. This is especially true as more VMs are created deployed.

Table 2 Example structure for documenting the network configuration

Hyper-V Host	Component	vNIC name	Attached virtual switch	IP address	Gateway	Subnet mask (Prefix – address)	DNS servers
TMER4R805S30	Management OS	vsLAN	vsLAN	10.124.4.50	10.124.4.1	22 – 255.255.252.0	10.124.6.245, 8.8.8.8
		vsSAN1	vsSAN1	10.10.6.210	NA	16 – 255.255.0.0	NA
		vsSAN2	vsSAN2	10.10.6.211	NA	16 – 255.255.0.0	NA

2 Step 2: Create a new VM

Once the Hyper-V host environment is documented, an administrator can start creating Hyper-V VMs on the Server Core Hyper-V host using PowerShell. This is accomplished using the **new-vm** command. To get more information about the **new-vm** command, run **man new-vm –detailed** or go to http://technet.microsoft.com/library/hh848537.aspx.

2.1 Set up variables to hold VM information

Before creating a new virtual machine, set up some variables to hold the parameters for the **new-vm** command.

7. First create a variable for the name of the new VM. In this example, it is **New-VM**.

```
PS C:\> $VMName = "NewVM"
```

8. Set a variable to hold the location and size of the virtual hard disks on the Hyper-V server)

```
PS C:\> $VHDpath = (get-vmhost).virtualharddiskpath
PS C:\> $VHDsize = 60GB
```

9. Set variables to hold the number of virtual processors (vCPUs) and start up memory.

```
PS C:\> $vCPU = 2
PS C:\> $vRAM = 4GB
```

Note: The 4 GB represents the startup memory for the VM. In this example, it is setup with static memory. A more detailed discussion of using Dynamic Memory is in section 3.2.

10. Set variables for the network configuration information specified in the Hyper-V host network documentation.

```
PS C:\> $LANsubnetmask = "255.255.252.0"
PS C:\> $iSCSIsubnetmask = "255.255.0.0"
PS C:\> $LANdefaultgateway = "10.124.4.1"
PS C:\> $LANPrimaryDNS = "10.124.6.245"
```

2.1.1 Optional scripting tip: Solicit user input

The **VMName** variable can be entered into a script by user input with the **read-host** command.

```
PS C:\> $VMName = read-host "Enter the name of the new virtual machine"
```

2.1.2 Optional scripting tip: Dynamic variables

Instead of hard coding the network configuration in the variables, the parameters can be dynamic. The benefit this is that it makes the script using them more portable. For example, a script with the following dynamic variables can be used on different Hyper-V servers and subnets without modifying the script. The

following PowerShell commands enable the variable parameters to be directly mined from the Hyper-V host network configuration.

1. Begin with a Hyper-V server Management OS LAN vNIC IP address.

Note: If there is more than one LAN vNIC associated with the Management OS, the **| select –first 1** command returns the first address found in the output.

```
$HostLANIPAddress = (get-netipaddress | ? {$_.interfacealias -like
"*LAN*"}).ipaddress | select -first 1
```

2. Retrieve the IP address for a ManagementOS iSCSI SAN vNIC.

```
$HostSANIPaddress = (get-netipaddress | ? {$_.interfacealias -like
"*SAN*"}).ipaddress | select -first 1
```

3. Use the variables from steps 1 and 2 to get the associated subnetmasks for both the LAN and iSCSI SAN networks.

```
$LANsubnetmask = (get-wmiobject -computer . -class
"win32_networkadapterconfiguration" | Where-Object {$_.ipaddress -eq
"$HostLANIPaddress"}).IPsubnet

$iSCSIsubnetmask = (get-wmiobject -computer . -class
"win32_networkadapterconfiguration" | Where-Object {$_.ipaddress -eq
"$HostSANIPaddress"}).IPsubnet
```

4. Locate the Management OS LAN network default gateway and primary DNS server.

```
$LANdefaultgateway = ((get-netipconfiguration | ? {$_.interfacealias -like
"*LAN*"}).IPV4DefaultGateway).nexthop

$LANPrimaryDNS = ((get-netipconfiguration | ? {$_.interfacealias -like
"*LAN*"}).dnsserver).serveraddresses | select -first 1
```

2.2 Create a virtual machine and virtual hard disk

The following **new-vm** PowerShell command creates a new VM using the values specified in the stored variables.

```
PS C:\ new-vm –name $VMName –memorystartupbytes $vRAM –path $VHDPath –newVHDpath
"$VHDPath\$VMName\Disk0.vhdx" –newVHDSizeBytes $VHDSize
```

Sample output:

```
Name        State CPUUsage(%) MemoryAssigned(M) Uptime   Status
----        ----- ----------- ----------------- ------   ------
NewVM       Off   0           0                 00:00:00 Operating normally
```

2.2.1 A note about Hyper-V Generation 2 Virtual Machines

In Windows Server 2012 R2 there are two types to choose from when creating a new virtual machine:

- **Generation 1:** Provides the same virtual hardware to the virtual machine as in previous versions of Hyper-V.
- **Generation 2:** (New with 2012 R2) Provides the following new functionality on a virtual machine:
  - PXE boot by using a standard network adapter
  - Boot from a SCSI virtual hard disk
  - Boot from a SCSI virtual DVD
  - Secure Boot (enabled by default)
  - UEFI firmware support

The example in this paper uses Generation 1 virtual machines. When creating a Generation 2 virtual machine, there are particular requirements regarding the boot disk and ISO images which the administrator needs to be aware of. A full explanation of the pros / cons and requirements of using Generation 1 or Generation 2 virtual machines is beyond the scope of this paper. For a details about Generation 2 virtual machines, please consult the following Microsoft resources:

http://blogs.technet.com/b/jhoward/archive/2013/10/24/hyper-v-generation-2-virtual-machines-part-1.aspx

http://technet.microsoft.com/en-us/library/dn282285.aspx

3 Step 3: Modify the new VM parameters as needed

When the **new-vm** command completes, the output shows that it was successful and the VM is in the off state. Unless specified otherwise, the VM is created using the default parameters.

1. To examine the VM details run the PowerShell **get-vm** command.

   ```
   PS C:\> get-vm -name $VMName | fl *
   ```

   The sample output below shows that the new VM was created using a combination of default values (such as a single vCPU and static memory), in addition to the values specified in the **new-vm** command.

   ```
   VMName                        : NewVM
   ...
   ConfigurationLocation         : F:\Hyper-V\VirtualHardDisks\NewVM
   SnapshotFileLocation          : F:\Hyper-V\VirtualHardDisks\NewVM
   AutomaticStartAction          : StartIfRunning
   AutomaticStopAction           : Save
   AutomaticStartDelay           : 0
   ...
   ProcessorCount                : 1
   ...
   MemoryStartup                 : 4294967296
   DynamicMemoryEnabled          : False
   ...
   NetworkAdapters               : {Network Adapter}
   ...
   DVDDrives                     : {DVD Drive on IDE controller number 1 at
   location 0}
   HardDrives                    : {Hard Drive on IDE controller number 0 at
   location 0}
   ...
   ```

2. To change the values associated with the new virtual machine, use the **set-vm** command. In the following example, the **set-vm** command changes the number of virtual CPUs from one to two, adds an automatic startup delay of 30 seconds, and changes the stop action of the VM when the Hyper-V host shuts down from **save** to **shutdown**.

   ```
   PS C:\> set-vm -name $VMName -processorcount 2 -automaticstartdelay 30 -
   automaticstopaction shutdown
   ```

3. Once the set-vm command has run, run the **get-vm** command again to verify that the changes persisted.

```
PS C:\> get-vm -name $VMName | fl *
```

Sample output:

```
VMName                        : NewVM
...
AutomaticStartAction          : StartIfRunning
AutomaticStopAction           : ShutDown
AutomaticStartDelay           : 30
...
ProcessorCount                : 2
...
NetworkAdapters               : {Network Adapter}
...
```

Note: The **set-vm** command is a powerful command that reconfigures the core VM settings. Typically, the VM will need to be in the **off** state when this command is run. Making these changes to a VM while in production requires a scheduled outage unless the VMs are setup for high availability. For more information about configuring a Hyper-V VM for high availability, see the Microsoft Technet article at http://technet.microsoft.com/en-us/library/cc742396.aspx.

3.1 A note about Hyper-V vCPUs

Windows Server 2012 Hyper-V can support from one to 64 vCPUs per VM. It is important to note that assigning more vCPUs to a virtual machine does not guarantee better performance; in fact, in some cases it might hurt. Remember that when a VM performs an operation, it waits until the number of logical processors available equals the number of vCPUs for the virtual machine. On busy hosts with a number of running VMs, this could lead to significant wait times for virtual machines with a higher number of vCPUs. A Hyper-V performance counter that can be monitored to determine this wait time is the Hyper-V Hypervisor Virtual Processor\CPU Wait Time Per Dispatch counter. This counter displays the average time (in nanoseconds) spent waiting for a vCPU to be dispatched onto a logical processor.

Most major software applications will have their own vCPU recommendations. In absence of these recommendations, it is best to start with one or two vCPUs per Hyper-V VM, and then add more as needs dictate.

3.2 A note about Hyper-V Dynamic Memory

The most widely encountered bottleneck for the number of VMs that can run on a host is the amount of RAM installed on the host and how much of that RAM each VM is consuming. In many environments, the solution is to drastically increase the amount of RAM in the host. However, this can increase the cost of the host and the overall virtual environment. Dynamic Memory performs two primary tasks:

- Dynamic Memory is well suited for a virtualized environment because it treats host memory as a pool of resources to be shared across multiple VMs. It does this by giving each VM a startup amount of memory and optimizing the allocation of RAM. This allows them to expand to a maximum amount when needed, and then harvest the memory and place it back into a free memory resource pool when it is no longer being used. As the environment grows, additional VMs can tap the unused memory in the environment.
- Because Hyper-V Dynamic Memory allocates only the memory required, there is no guesswork involved about optimal memory allocation for a new VM. This minimizes the waste of precious memory resources on the host and makes managing the virtual memory environment much easier.

Note: Some memory intensive applications, like Microsoft SQL Server and Exchange, require special considerations when being run on VMs with Dynamic Memory enabled. Before enabling Dynamic Memory, check the application support documentation for support conditions and caveats about running on a Hyper-V VM with Dynamic Memory enabled. For more information about using Hyper-V Dynamic Memory with SQL Server go to: http://sqlmag.com/database-virtualization/using-hyper-v-dynamic-memory-sql-server.

To set up Dynamic Memory for a virtual machine, use the PowerShell **set-vmmemory** command. An example using the **$VMName** and **$vRAM** variables from the previous section are shown below.

```
PS C:\> set-vmmmemory –vmname $VMName –startupbytes $vRAM –dynamicmemoryenabled
$True –minimumbytes 100MB –maximumbytes 8GB –buffer 5 –priority 75
```

The parameters in the above **set-vmmemory** command are:

- **Startup Bytes**: The amount of memory allocated to the VM when it starts. This value is the default static memory value that is used by the virtual machine if dynamic memory is not enabled.
- **Dynamic Memory Enabled**: Determines whether the virtual machine will use dynamic memory. This parameter takes a Boolean true or false. The Hyper-V default is false. To enable Dynamic Memory, set this value to **$True**. This can only be done when the VM is created or when the created VM is off.
- **Minimum Bytes**: Hyper-V VMs require a minimum of 512 MB of RAM in order to boot up. However, when idle, they can be made to consume much less (as little as 8 MB). The **minimumbytes** parameter specifies the amount of memory the VM uses when idle. This parameter is very important when Hyper-V is being used for VDI.
- **Maximum Bytes**: This is the maximum amount of RAM that can be allocated to the VM (1 TB). Obviously, a VM can never be allocated more memory than what exists on the Hyper-V host. This parameter can be increased while the VM is running; meaning the VM does not require downtime to increase its memory.

- **Buffer %**: The additional percentage of memory that is allocated to a VM as spare memory. The default value is 20%. In the example above, if a VM requires 1000MB of memory, dynamic memory will allocate five percent (1050MB). It is used primarily as a cache of memory available to the VM in case dynamic memory cannot allocate fast enough.
- **Priority (or Memory Weight)**: A VM with a higher priority value receives memory before a VM with a lower priority. Unless specified all Hyper-V VMs are created with the default value of 50. The range for this parameter is 0 -100.

4 Step 4: Add vNICs to the VM

The next step in building a VM is to attach it to the virtual switches on the Hyper-V host. When the VM was created, a single vNIC was created by default (shown in sample **get-vm** output in section 3). Details of the default vNIC are shown by using the **get-vmnetworkadapter** command specifying the name of the new VM.

```
PS C:\> get-vmnetworkadapter -vmname $VMName
```

In this sample output, the vNIC **Network Adapter** is visible. The output shows that it is not part of the Management OS partition, it has no MAC address, its status is blank (disabled), it is not attached to any virtual switches, and it has no IP address assigned.

```
Name             IsManagementOs  VMName  SwitchName  MacAddress    Status  IPAddresses
----             --------------  ------  ----------  ----------    ------  -----------
Network Adapter  False           NewVM               000000000000  {}
```

4.1 Attach the default vNIC

The example in this document has three virtual switches on the Hyper-V host documented in the virtual environment.

1. Display the virtual switch names and the types in PowerShell.

   ```
   PS C:\> get-vmswitch | ft -property Name,SwitchType –AutoSize
   ```

   The following sample output displays the virtual switch representing the LAN network is vsLAN.

   ```
   Name    SwitchType
   ----    ----------
   vsSAN2    External
   vsSAN1    External
   vsLAN     External
   ```

2. Attach the default vNIC to the vsLAN virtual switch and rename the vNIC to also use the name vsLAN for easier identification.

   ```
   PS C:\> get-vmnetworkadapter -vmname $VMName| rename-vmnetworkadapter -
   newname vsLAN -passthru | connect-vmnetworkadapter -switchname vsLAN
   ```

   Command breakdown:

   a. The one vNIC associated with the new virtual machine is gathered with the **get-vmnetworkadapter** command.
   b. The vNIC object is passed through to the **rename-vmnetworkadapter** command and renamed to vsLAN.
   c. The vsLAN vNIC object is passed through to the **connect-vmnetworkadapter** command that then connects it to the specified virtual switch (the vsLAN virtual switch).

3. Display the renamed and connected vNIC.

```
PS C:\ > get-vmnetworkadapter -vmname $VMName| ft -autosize
```

Sample output:

```
Name   IsManagementOs  VMName  SwitchName  MacAddress    Status  IPAddresses
----   --------------  ------  ----------  ----------    ------  -----------
vsLAN  False           NewVM   vsLAN       000000000000          {}
```

4.2 Create and attach additional vNICs

At this point the new VM is ready to be attached to the iSCSI SAN virtual switches. In the example, the iSCSI SAN virtual switches are vsSAN1 and vsSAN2. The following PowerShell command creates two additional vNICs for the new VM and renames them to match the virtual machine.

```
PS C:\> foreach ($vswitch in (get-vmswitch | ? {$_.name –like "*SAN*"}).name)
{add-vmnetworkadapter -vmname $VMName –Name $vswitch -switchname $vswitch}
```

This command uses a **foreach** loop to get the name of each iSCSI SAN virtual switch and then store the values in the **vswitch** variable. This information is then used by the **add-vmnetworkadapter** command to add a new vNIC to the VM, rename it to the name stored in the **$vswitch** variable, and then connect the new vNIC to a virtual switch (also specified by the **$vswitch** variable). The command loops to the next iSCSI SAN virtual switch found and repeats the process.

Use the following get-vmnetworkadapter command to display the updated vNIC configuration for the new VM.

```
PS C:\> get-vmnetworkadapter -vmname $VMName | ft -autoSize
```

This sample output clearly shows that the new VM has three vNIC adapters and the virtual switch each one is connected to.

```
Name    IsManagementOs  VMName  SwitchName  MacAddress    Status  IPAddresses
----    --------------  ------  ----------  ----------    ------  -----------
vsSAN2  False           NewVM   vsSAN2      000000000000          {}
vsSAN1  False           NewVM   vsSAN1      000000000000          {}
vsLAN   False           NewVM   vsLAN       000000000000          {}
```

Note: The format table (**ft –autosize**) and format list (**fl–autosize**) commands have been used throughout this document. Autosize is primarily used to improve the output format as a table so that it fits on the screen. This helps when capturing output for documentation purposes. For full details on the **format-list** and **format-table** commands view the help pages (man ft, man fl) in PowerShell.

Updating the vNICs changes the virtual environment configuration as illustrated in Figure 3.

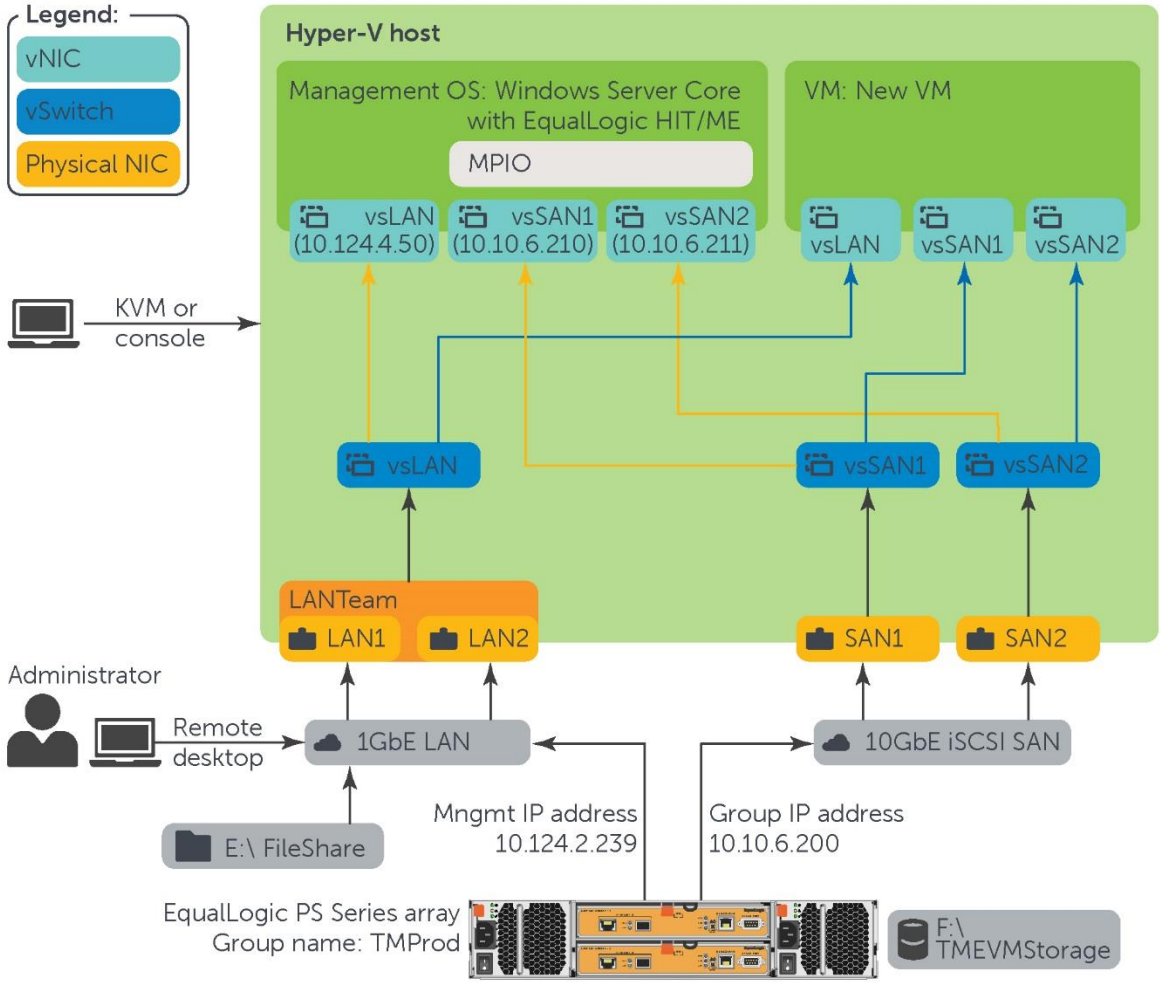


Figure 3 New vNICs attached to the iSCSI SAN virtual switches

Creating and attaching the vNICs to the iSCSI SAN virtual switches is essentially the virtual cabling to the network environment. However, the VM still does not have network connectivity because the network configuration (assigning of IP addresses, gateways, and subnets) has not been performed. The next section covers installing the operating system on the VM.

5 Step 5: Start the VM and install the operating system

5.1 Create Windows Server ISO for unattended installation

When working with Server Core, one of the most important factors for a successful implementation and management is the ability to automate and perform tasks remotely. This includes the installation of the OS on either the physical servers or virtual machines. When installing Windows on a VM that is on a Server Core Hyper-V host, the administrator does not have the luxury of monitoring the OS installation through the connect window in the Hyper-V Manager GUI tool. As a result, there is also an inability to provide user input at the installation prompts. An automated installation of the Windows OS on a virtual machine on a Server Core Hyper-V host is performed using an answer file. This answer file, autounattend.xml, is an xml formatted file. It stores and utilizes the various configuration information so that normal installation questions can be bypassed. When the xml file is injected into the Windows Server Core installation ISO, the install becomes a hands-free process.

The process of creating the autounattend.xml files is captured by Derek Seaman in a blog post from July 2012 at http://www.derekseaman.com/2012/07/windows-server-2012-unattended.html.

Aside from the normal installation configuration requirements and answers for the prompts, there are important roles and features in the autounattend.xml as additional packages to install for Hyper-V VMs on a Server Core host.

- Enable Remote Desktop Services on the Virtual Machine.
  Described on Technet at http://technet.microsoft.com/en-us/library/hh825710.aspx

- Configure the Firewall Settings on the Virtual Machine.
  Described on Technet at http://technet.microsoft.com/en-us/library/hh825676.aspx

Each customer environment has its own requirements that can be placed in the autounattend.xml file. However, enabling remote desktop and configuring the firewall settings during the OS installation allows the virtual machine to be accessible through a remote desktop after the network configuration is set from the Hyper-V host immediately after the OS installation completes.

Once the autounattend.xml is created with the desired custom configuration settings, embed the file into the Windows Server Core iso file using a tool like UltraISO. Name the file so it is easily identifiable from other Windows ISOs which might be stored at this location (for example, WindowsServer2012_auto.iso). Copy this Windows ISO to a network share, or other location, and keep a separate copy of the autounattend.xml file on the network share for later use.

Note: When working with XML files, an XML editor will provide better visualization of the file for easier editing. Below is a screenshot of an Autounattend.xml file open in XML Marker V1.1. Another editor available is Microsoft's XML Notepad.

5.2 Attach the hands-free Windows ISO to the vDVD Drive on the VM

Once the hands-free Windows ISO is created and stored on a network share or other accessible location, configure the virtual DVD drive of the virtual machine to mount this location at the initial start-up.

1. Locate the ISO on the network share.

   ```
   PS C:\> ls -r E:\ | ? {$_.name -like "*auto.iso"}
   ```

   Sample output:

   ```
       Directory: E:\WindowsISOs\WindowsServer2012


   Mode                 LastWriteTime     Length Name
   ----                 -------------     ------ ----
   -a---         9/26/2012   1:03 PM 3695179776 windows_server_2012_x64_auto.iso
   ```

2. Place the location of the ISO file into a variable.

   ```
   PS E:\> $winISOpath = (ls -r E:\ | ? {$_.name -like "*auto2.iso"}).fullname
   ```

3. Use this variable as the mount point for the virtual DVD drive on the virtual machine. This is done using the **set-vmdvddrive** PowerShell command.

   ```
   PS C:\> set-vmdvddrive -vmname $VMName -path $winISOpath
   ```

4. Finally, verify that the virtual DVD drive mount point has been successfully configured to point to the Windows ISO with the **get-dvddrive** command.

   ```
   PS C:\> get-vmdvddrive -vmname $VMname | ft -property VMName, Controllertype,
   Path -AutoSize
   ```

   Sample output:

   ```
   VMName     ControllerType Path
   ------     -------------- ----
   NewVM      IDE            E:\WindowsISOs\WindowsServer2012Auto\windows… _
   ```

5.3 Start the virtual machine to start the OS Installation

1. Once the Windows ISO is attached to the vDVD drive of the virtual machine, start it using the **start-vm** PowerShell command.

   ```
   PS C:\> start-vm -vmname $VMName
   ```

   The VM boots from the vDVD since the virtual hard disk is blank. This loads the Windows ISO image and starts the OS installation using the embedded autounattend.xml answer file.

2. Verify that the virtual machine started successfully by running the **get-vm** command.

   ```
   PS C:\> get-vm -vmname $VMName
   ```

The output of the **get-vm** command should show the VM state as **Running**, both CPU and Memory resources being consumed, and the status as **Operating normally** as shown in the sample output below.

```
Name    State       CPUUsage(%)   MemoryAssigned(M)   Uptime     Status
----    -----       -----------   -----------------   ------     ------
NewVM   Running 7   4096                              00:00:18   Operating normally
```

5.4 Monitor the OS installation on the VM from the Hyper-V host

Now that the new virtual machine is running, and the OS is being installed, it makes sense to monitor the installation progress. Unfortunately, since the Hyper-V host is running Server Core for the OS, there is no Hyper-V manager GUI to use to connect to the VM (vmconnect.exe). In addition, without an OS on the VM, there is not an IP address for connecting remotely to the new virtual machine. Until the OS successfully installs, the VM is unreachable.

Through experimentation during this project, a work around was established to enable notification on the Hyper-V host when the VM OS installation completes. This method involves using the **get-vmnetworkadapter** command to check the status of the vNICs on the virtual machine.

1. After the virtual machine has started and while the OS is being installed, the status of the vNICs on the virtual machine will be in the **NoContact** state.

   ```
   PS C:\> get-vmnetworkadapter -vmname $VMName
   ```

   Sample output:

```
Name    IsManagementOs  VMName  SwitchName  MacAddress    Status        IPAddresses
----    --------------  ------  ----------  ----------    ------        -----------
vsSAN2  False           NewVM   vsSAN2      00155D04325C  {NoContact}   {}
vsSAN1  False           NewVM   vsSAN1      00155D04325D  {NoContact}   {}
vsLAN   False           NewVM   vsLAN       00155D04325E  {NoContact}   {}
```

2. Once the OS is installed on the virtual machine and it has rebooted, the vNICs attempt to connect to the virtual switches on the Hyper-V host. If successful, their status changes to **Ok**.

   ```
   PS C:\> get-vmnetworkadapter -vmname $VMName
   ```

   The following sample output shows the VM **Ok** status. Also, note that the DHCP has automatically assigned IP addresses, but the virtual machine is still unreachable because there is still no subnet or gateway information in the vNIC configuration.

```
Name     IsManagementOs   VMName   SwitchName   MacAddress     Status   IPAddresses
----     --------------   ------   ----------   ----------     ------   -----------
vsSAN2   False            NewVM    vsSAN2       00155D04325C   {Ok}     {169.254.63.203, fe80::… }
vsSAN1   False            NewVM    vsSAN1       00155D04325D   {Ok}     {169.254.20.198, fe80::… }
vsLAN    False            NewVM    vsLAN        00155D04325E   {Ok}     {10.124.5.159, fe80::… }
```

3. A script that includes a **while** loop uses this status change as a trigger to inform the administrator that the OS installation has completed and that the vNICs have successfully connected to the virtual switches.

   ```
   PS C:\> while ((get-vmnetworkadapter -vmname $VMName | ? {$_.name -eq
   "vsLAN"}).status -ne "Ok") {write-host ">" -NoNewLine; start-sleep -seconds
   10};write-host "`nOS Installation complete on $VMName. Ready for IP
   Configuration"
   ```

   This command runs in the foreground of the PowerShell session and presents a series of angled brackets (>) every ten seconds while the status of the vNICs is not equal to **Ok**. Once this status change occurs, a new line displays that the OS installation on the VM is complete and ready for the IP configuration. When the command loop ends, it returns the command prompt back to the PowerShell window.

   ```
   >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>... >>>>>
   OS Installation complete on NewVM. Ready for IP Configuration
   PS C:\
   ```

It is important to note that there are other ways to accomplish VM install monitoring. If a GUI is required, Microsoft recommends the following options for installing VM's on a core system.

- Temporarily install the Minimal Server Interface or the full GUI using PowerShell on the Server Core Hyper-V host. This can be done with the following PowerShell command:

  ```
  PS C:\> install-windowsfeature server-gui-mgmt-infra, Server-GUI-Shell –
  restart
  ```

This installs the Hyper-V Management tools including the vmconnect.exe file. Once the MSI or GUI is installed, the VM can be connected through Hyper-V Manager. To connect to the new virtual machine using the vmconnect utility from the Hyper-V host command line type:

```
PS C:\> vmconnect localhost $VMName
```

The syntax for the **vmconnect** command is **vmconnect** *Hyper-V Server VM Name*.

Once the need for the GUI has been removed, the Hyper-V host can be converted back to a Server Core configuration by typing:

```
PS C:\> Uninstall-windowsfeature server-gui-mgmt-infra, Server-GUI-Shell –
restart
```

- Dedicate a separate physical server with a full Windows Server Core GUI to be a Hyper-V Management Server.

6 Step 6: Set the VM vNIC Network Configuration

Another challenge that accompanies the inability to launch a VM console from Server Core Hyper-V host, is configuring the network settings for the virtual machine's VNICs. On a Hyper-V host with the GUI installed, this could be done from within the Hyper-V Manager VM console window. However, on Server Core, the vmconnect utility and required GUI components are not installed, so a PowerShell method is needed. Unfortunately, there is not an equivalent PowerShell command like VMWare's PowerCLI's **set-vmGuestNetworkInterface** command to perform this function.

Microsoft recommends a method of creating and managing Hyper-V virtualized environments by using System Center Virtual Machine Manager (SCVMM). From within this tool, setting the network configuration (IP addresses, DNS, Subnet, Gateway) for a new virtual machine is easy. However, without the use of the Hyper-V Manager GUI console (VMConnect), SCVMM, or PowerShell command, the best option is using SetGuestNetworkAdapterConfiguration. This is a method within the Windows Management (WMI) class, Msvm_VirtualSystemManagementService (V2)". An individual named Ravikanth Chianti has created a PowerShell function called "set-VMNetworkConfiguration" which uses this WMI class and method to set the IP address, Subnet, DNS Server, and Default Gateway for virtual machines vNICs. The script fulfills the requirements to automate Hyper-V VM creation by configuring vNIC IP addresses from within a PowerShell script on the Hyper-V host. After completing the vNIC network configuration using the **set-vmnetworkconfiguration** command, a remote desktop has direct access to the virtual machine. This function is described in detail at http://www.ravichaganti.com/blog/?p=2766.

6.1 Create the set-vmnetworkconfiguration PowerShell module

To use the described function, complete the following steps.

1. Copy the text in Appendix A of this document and paste it into a PowerShell module (.psm1) file named, "set-vmnetworkconfiguration.psm1".
2. Place set-vmnetworkconfiguration.psm1 into a directory on the Hyper-V host at C:\Users\Administrator\Documents\WindowsPowerShell\Modules.
   Note: Create this directory if it does not exist using:

   ```
   PS C:\> mkdir C:\Users\Administrator\Documents\WindowsPowerShell\Modules
   ```

3. Once this file is in the modules directory, it can be imported directly into the PowerShell session or into the script.
   a. Place the path to the **set-vmnetworkconfiguration** module into a variable.

   ```
   PS C:\> $ModulePath =
   "C:\Users\Administrator\Documents\WindowsPowerShell\Modules"
   ```

   b. Import the module into the PowerShell session or script by using the **import-module** PowerShell Command.

   ```
   PS C:\> import-module $ModulePath\set-vmnetworkconfiguration.psm1
   ```

6.2 Set the network configuration for the VMs LAN vNIC

Once the module is imported, it can be used to set the vNIC IP address, subnet, gateway, and DNS server with the returned vNIC object from the **get-vmnetworkadapter** command as pipeline input. The following example commands use a hard coded IP address and the variables from section 2.1 for the subnet masks, default gateway, and Primary DNS Server.

```
PS C:\> $LANsubnetmask = "255.255.252.0"
PS C:\> $iSCSIsubnetmask = "255.255.0.0"
PS C:\> $LANdefaultgateway = "10.124.4.1"
PS C:\> $LANPrimaryDNS = "10.124.6.245"
```

These variables are used in the following command string to set the VM LAN vNIC (vsLAN).

```
PS C:\> get-vmnetworkadapter -vmname $VMName -name vsLAN |
set-vmnetworkconfiguration –ipaddress 10.124.4.51 –subnet $LANsubnetmask
–defaultgateway $LANdefaultgateway –DNSServer $LANPrimaryDNS
```

6.3 Set the network configuration for the VMs SAN vNICs

Use the following command strings to set the two VM iSCSI vNICs.

```
PS C:\> get-vmnetworkadapter -vmname $VMName -name vsSAN1 |
set-vmnetworkconfiguration –ipaddress 10.10.6.214 –subnet $iSCSIsubnetmask

PS C:\> get-vmnetworkadapter -vmname $VMName -name vsSAN2 |
set-vmnetworkconfiguration –ipaddress 10.10.6.215 –subnet $iSCSIsubnetmask
```

Note: In the example, the iSCSI SAN network is non-routable, therefore the gateway and DNS Server values are ommitted from the command string.

6.4 Verify the new network configuration for the VM vNICs

After the network configuration of the virtual machines vNICs is complete, verify that the changes have taken effect by re-running the **get-vmnetworkadapter** adapter command.

```
PS C:\> get-vmnetworkadapter -vmname $VMname | ft -AutoSize
```

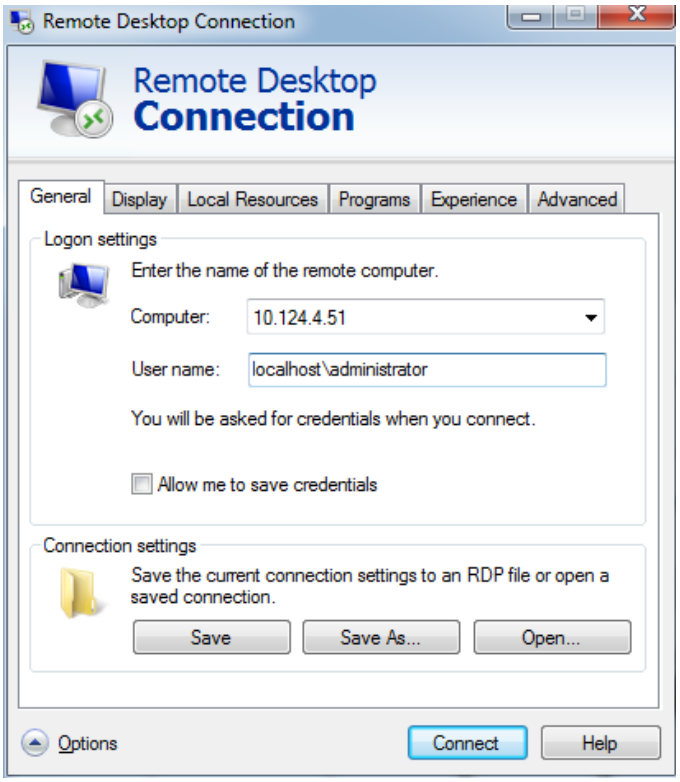
The sample output below shows that the new network configuration has been applied to the virtual machines vNICs making the virtual machine accessible through a remote desktop.

```
Name     IsManagementOs   VMName   SwitchName   MacAddress     Status   IPAddresses
----     --------------   ------   ----------   ----------     ------   -----------
vsSAN2   False            NewVM    vsSAN2       00155D04325C   {Ok}     {10.10.6.215, fe80::…}
vsSAN1   False            NewVM    vsSAN1       00155D04325D   {Ok}     {10.10.6.214, fe80::…}
vsLAN    False            NewVM    vsLAN        00155D04325E   {Ok}     {10.124.4.51, fe80::… }
```

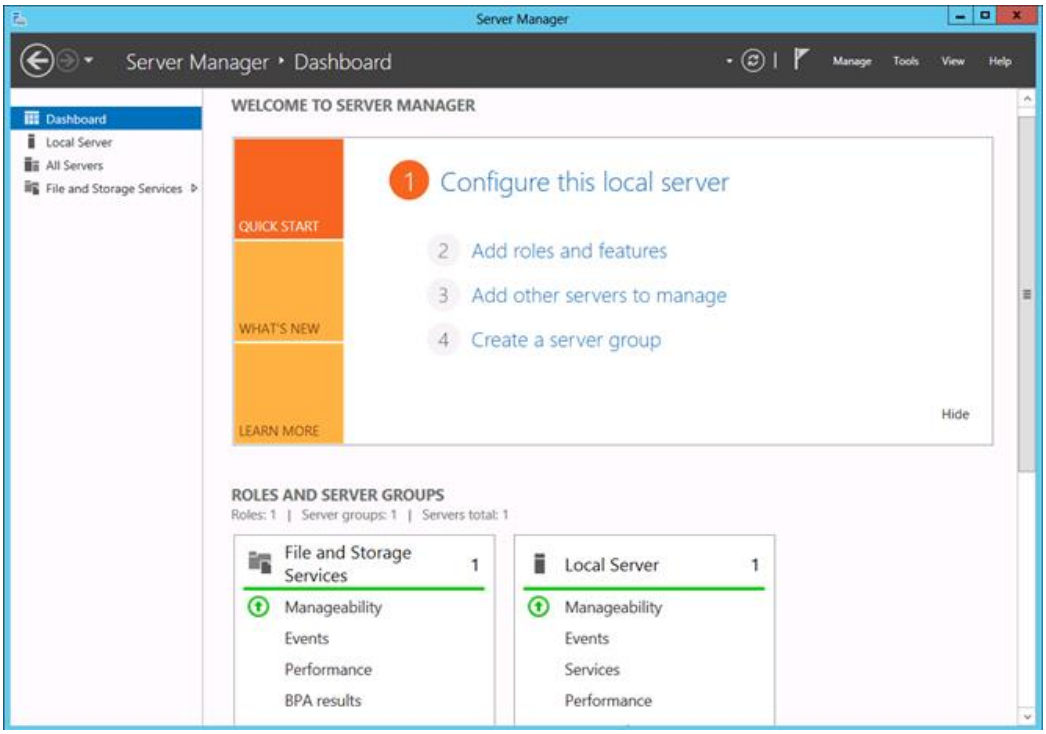
7 Step 7: Finalize new virtual machine configuration

After the virtual machine is built and the OS is installed, it can be accessed through remote desktop, if it was enabled. From the remote desktop session, the final configuration tasks can be performed for the new virtual machine.

1. Connect to the new virtual machine using remote desktop.
   If remote desktop was enabled properly in the autounattend.xml file used in the Windows OS ISO, and if the network configuration of the virtual machine vNICs were successful, then the virtual machine can be accessed by opening an RDP connection to the LAN vNIC IP address.

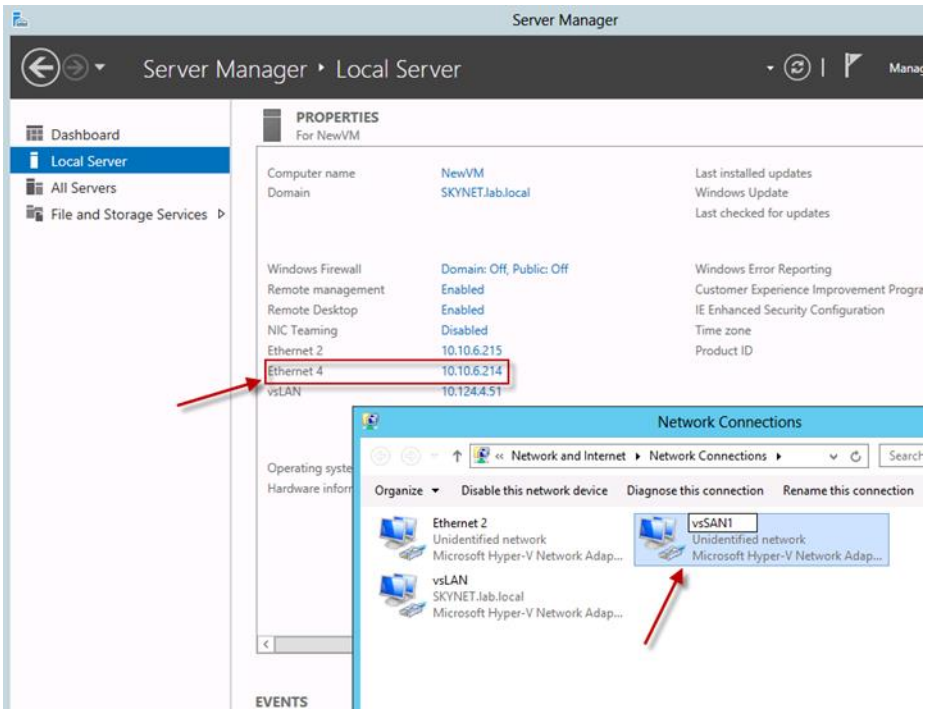


2. Rename, add to a domain, and define other settings with the Server Manager.
   If the GUI version of Windows Server was installed, at the time it was connected to the virtual machine, the familiar Server Manager GUI is visible.



3. The final configuration of the virtual machine can be performed with either Server Manager, the sconfig utility, or PowerShell. Configuration tasks typically performed with Server Manager are:
   - Renaming the virtual machine and adding it to the local domain (results in a VM reboot)
   - Configuring Windows Firewall and IE enhanced security configurations
   - Setting the time, time zone, and date

4. Modify the default virtual machine vNIC names. When the OS installs on the virtual machine, the vNICs is assigned a default name (Ethernet, Ethernet 1, Ethernet 2, etc.). The names assigned to the virtual machines vNICs on the Hyper-V host do not carry forward to populate the names of the vNICs in Server Manager on the virtual machine. As a best practice, take the extra step required to rename the vNICs to match the virtual machine vNIC names associated with the Hyper-V host for easier identification and synchronization.



Note: There is a wealth of documentation that describes performing management operations through Server Manager on Microsoft TechNet at http://technet.microsoft.com/en-us/library/hh801900.aspx.

Once the vNICs have been renamed, continue with the remainder of the steps to finalize the configuration.

5. Place a copy of the autounattend.xml file (created in section 5.1) in the C:\Windows\System32\Sysprep directory on the new virtual machine.
6. Optional: Change the PowerShell command prompt to reflect the name of the virtual machine. When running with multiple PowerShell session windows open on different servers and virtual machines, it is a good idea to change the PowerShell prompts to include the name of the host running the command. This is done by typing:

```
PS C:\>function prompt {"[$env:computername] PS $(get-location)>"}

[NEWVM] PS C:\>
```

7. Enable Jumbo Frames on the iSCSI vNICs, vsSAN1 and vsSAN2
   Enabling Jumbo Frames for the iSCSI vNICs is easier using PowerShell than through the Server Manager GUI.

   a. This can be done for both of the virtual machines iSCSI vNICs from the example (vsSAN1 and vsSAN2) with the command:

```
[NEWVM] PS C:\>set-netadapteradvancedproperty -name *vsSAN* -displayname
"Jumbo Packet" -displayvalue "9014 Bytes"
```

b. Verify that the change was successful.

```
[NEWVM] PS C:\>get-netadapteradvancedproperty -name *vsSAN* -displayname
"Jumbo Packet","Flow Control" | ft -property Name, Displayname, Displayvalue,
validdisplayvalues -AutoSize
```

Sample output:

```
Name   Displayname  Displayvalue validdisplayvalues
----   -----------  ------------ ------------------
vsSAN1 Jumbo Packet 9014 Bytes   {Disabled, 4088 Bytes, 9014 Bytes}
vsSAN2 Jumbo Packet 9014 Bytes   {Disabled, 4088 Bytes, 9014 Bytes}
```

c. Also verify that a jumbo packet can be transferred without fragmentation from the virtual machine fto the EqualLogic PS Array group IP address.

```
[NEWVM] PS C:\>ping -f -l 8972 10.10.6.200
```

Sample output:

```
Pinging 10.10.6.200 with 8972 bytes of data:
Reply from 10.10.6.200: bytes=8972 time=1ms TTL=255
Reply from 10.10.6.200: bytes=8972 time<1ms TTL=255
...
```

8. Optional: Disable IPV6 on virtual machine using PowerShell.
Disabling unused protocols and services is a good way to conserve resources and increase security. These same practices should apply to the virtual machines as well. On the virtual machine with a GUI installation, this can be done through **Server Manager** > **Network Connections** > **Properties,** or through PowerShell. Doing this with PowerShell is more simple because IPV6 can be disabled on all of the vNICs for the virtual machine with the same single command.

```
[NEWVM] PS C:\ foreach ($name in (get-netadapter | where-object {$_.status -
eq "up"}).name) {disable-netadapterbinding -name $name -componentid
ms_tcpip6}
```

9. Verify that the command successfully disabled IPV6 for the virtual machine vNICs with the **get-netadapter** command. Transfer the basic vNIC information returned as an object to the **get-netipconfiguration** command. The output should show only IPV4 addresses for the vNICs.

```
[NEWVM] PS C:\>get-netadapter | get-netipconfiguration
```

Sample output:

```
InterfaceAlias       : vsSAN1
InterfaceIndex       : 15
InterfaceDescription : Microsoft Hyper-V Network Adapter #4
NetProfile.Name      : Unidentified network
```

```
IPv4Address          : 10.10.6.214
IPv4DefaultGateway   :
DNSServer            :

InterfaceAlias       : vsSAN2
InterfaceIndex       : 13
InterfaceDescription : Microsoft Hyper-V Network Adapter #2
NetProfile.Name      : Unidentified network
IPv4Address          : 10.10.6.215
IPv4DefaultGateway   :
DNSServer            :

InterfaceAlias       : vsLAN
InterfaceIndex       : 12
InterfaceDescription : Microsoft Hyper-V Network Adapter
NetProfile.Name      : SKYNET.lab.local
IPv4Address          : 10.124.4.51
IPv4DefaultGateway   : 10.124.4.1
DNSServer            : 10.124.6.245
                       8.8.8.8
```

10. For Windows 2012 R2 Virtual Machines - Activate Windows using Automatic Virtual Machine Activation (AVMA)

    AVMA allows for the installation of virtual machines on a properly activated Windows server without having tomanage product keys for each individual virtual machine, even in disconnected environments. AVMA binds the virtual machine activation to the licensed virtualization server and activates the virtual machine when it starts up. AVMA also provides real-time reporting on usage and historical data on the license state of the virtual machine. Reporting and tracking data is available on the virtualization server.

    To activate Windows on a virtual machine, type the following command:

    ```
    C:\> slmgr /ipk <AVMA key>
    ```

    Table 3 AVMA keys

Edition	AVMA key
Datacenter	Y4TGP-NPTV9-HTC2H-7MGQ3-DV4TW
Standard	DBGBW-NPF86-BJVTX-K3WKJ-MTB6V
Essentials	K2XGM-NMBT3-2R6Q8-WF2FK-P36R2

8 Step 8: Update the Hyper-V environment documentation

Once the virtual machine is built and configured, update all existing Hyper-V documentation to reflect the changes in the environment. The virtualized environment in this example is illustrated in the following diagram.

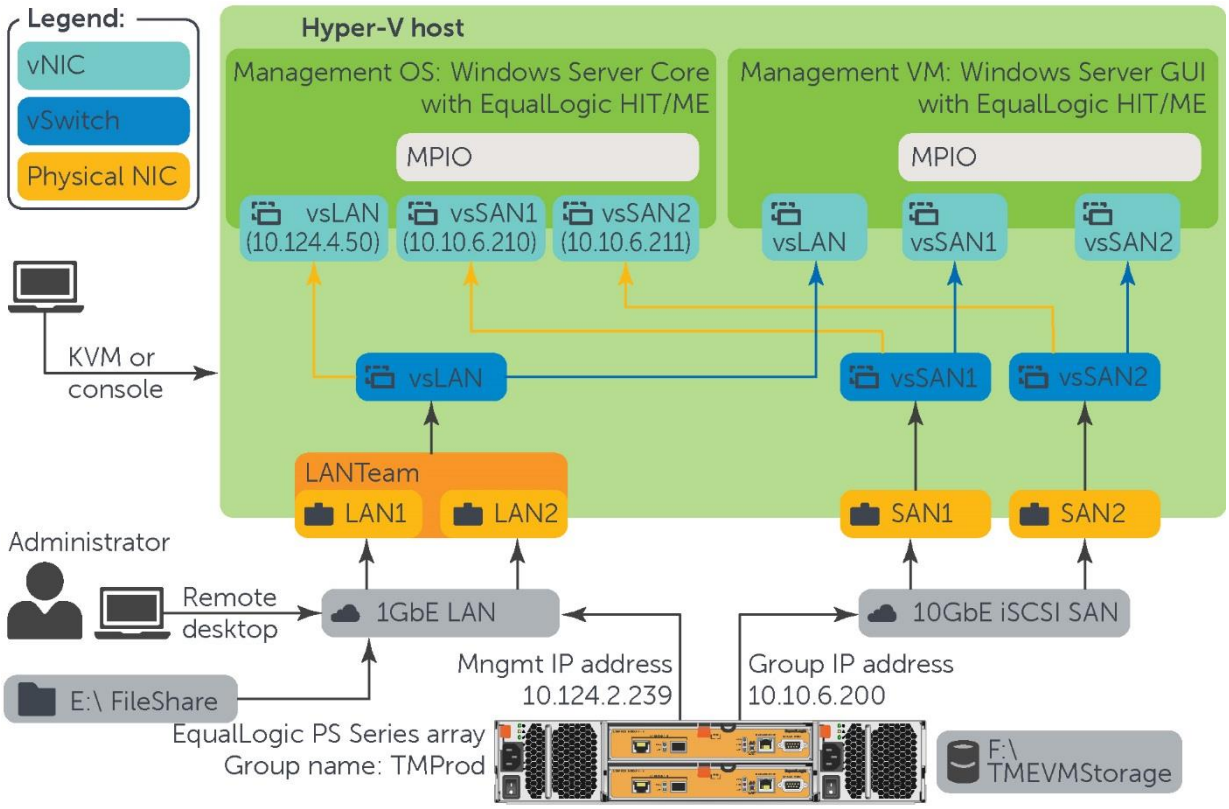


Figure 4 Management OS, the first guest VM and associated network configuration

Table 4 Virtualized environment network configuration

Hyper-V host	Component	vNIC name	Attached virtual switch	IP address	Gateway	Subnet mask (prefix – address)	DNS servers
TMER4R805S30	Management OS	vsLAN	vsLAN	10.124.4.50	10.124.4.1	22 – 255.255.252.0	10.124.6.245, 8.8.8.8
		vsSAN1	vsSAN1	10.10.6.210	NA	16 – 255.255.0.0	NA
		vsSAN2	vsSAN2	10.10.6.211	NA	16 – 255.255.0.0	NA
	NewVM	vsLAN	vsLAN	10.124.4.51	10.124.4.1	22 – 255.255.252.0	10.124.6.245, 8.8.8.8
		vsSAN1	vsSAN1	10.10.6.214	NA	16 – 255.255.0.0	NA
		vsSAN2	vsSAN2	10.10.6.215	NA	16 – 255.255.0.0	NA

The new virtual machine can be installed with application software or converted into a Hyper-V management guest.

9 Step 9: Optional – Covert VM to a dedicated management guest

For Hyper-V running on Server Core hosts, creating a Hyper-V management guest VM provides the following benefits.

- A management VM allows centralized management and administration of the Hyper-V host from a separate and independent interface. This provides greater platform stability since management tasks performed on the Management VM run independently. The result is that the potential impact on the Hyper-V host Management OS and its other associated virtual machines is minimized. The management VM can be highly available when used with Microsoft Failover Clustering.
- The Management VM can be installed with a full Windows Server GUI installation so that all of the graphical management tools are available to the administrator without adding overhead to the Management OS. Again, a primary goal of installing Server Core on the Hyper-V host is to provide a bare minimum installation so that the majority of Hyper-V server resources can be dedicated to running the Hyper-V role.
- The management VM provides a single access point to administer the Hyper-V host. Done properly, this increases security of specific profiles so that selected users have sole administrative privileges on the Management VM. This method is a simple way to reduce the attack footprint.

9.1 Optional: Rename the VM to ManagementVM

If a virtual machine is to be used for Hyper-V management functions, it is a good idea to rename it in Active Directory and its name in Hyper-V host VMMS so it is easily identifiable as the dedicated management virtual machine.

9.1.1 Renaming the virtual machine in Active Directory

1. Retitling the host name of the virtual machine can be done from the virtual machine either through Server Manager or through PowerShell using the **rename-computer** command. This command requires a domain administrator credential when entered.

   ```
   [NEWVM] PS C:\>rename-computer -newname ManagementVM -domaincredential
   skynet.lab.local\administrator –restart
   ```

2. Enter the domain administrator password in the credential window that is displayed.
3. Click **OK** to reboot the virtual machine.
4. When the virtual machine reboots, open a new PowerShell session window and change the prompt to reflect the new name of for the virtual machine.

   ```
   PS C:\> function prompt {"[$env:computername] PS $(get-location)>"}

   [MANAGEMENTVM] PS C:\ >
   ```

9.1.2 Rename the virtual machine in the Hyper-V host VMMS

Once the virtual machine name is changed in Active Directory, open a remote PowerShell session to the Hyper-V Server and change the name of the virtual machine in VMMS

1. Creating a remote PowerShell session to the Hyper-V host is done by first creating a new PowerShell session to the remote server and storing the session information into a variable.

   ```
   [MANAGEMENTVM] PS C:\> $session = New-PSSession -ComputerName TMER4R805S30 -
   Credential skynet.lab.local\administrator
   ```

2. Enter the administrator password in the credential window that opens.
3. Click **OK** and the new remote session can be entered with the following command.

   ```
   [MANAGEMENTVM] PS C:\>enter-PSSession -session $session
   ```

4. Once in the remote session, the command prompt changes to the name of the Hyper-V server.

   ```
   [TMER4R805S30]: PS C:\>
   ```

5. Rename the virtual machine by using the **get-vm** command and specifying the vm name with the **–vmname** parameter.
6. Transfer the returned output to the **set-vm** command specifying the new vm name.

   ```
   [TMER4R805S30] PS C:\>get-vm -vmname NewVM | set-vm -newvmname ManagementVM
   ```

7. Run **get-vm** again to verify that the name change for the virtual machine was successful.

   ```
   [TMER4R805S30] PS C:\>get-vm
   ```

   Sample output:

```
Name           State     CPUUsage(%)   MemoryAssigned(M)   Uptime     Status
----           -----     -----------   -----------------   ------     ------
ManagementVM   Running   0             4000                00:39:13   Operating normally
```

Note: From this point forward in the example, the management guest virtual machine will be referred to as the Management VM.

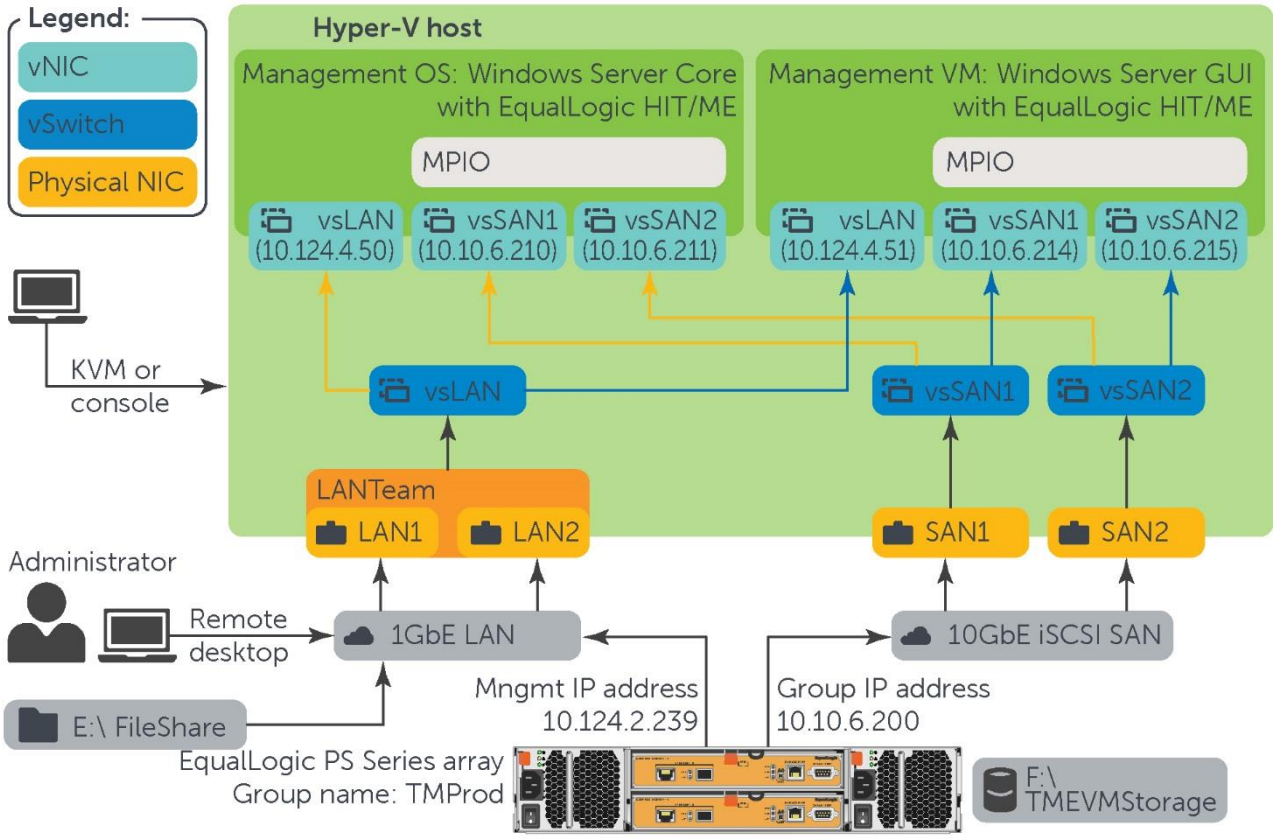


Figure 5 New VM is now Management VM

9.2 Install Hyper-V Remote Server Administration Tools

One of the primary purposes of creating a management virtual machine is to remotely manage the Hyper-V core server through the Hyper-V Management Console GUI. The Hyper-V Management Console and associated Hyper-V PowerShell commands can be installed on the management VM as the Windows feature, Hyper-V Remote Server Administration Tools (RSAT).

1. To install this feature, first verify that it is available using the following PowerShell command.

```
[MANAGEMENTVM] PS C:\> get-windowsfeature | ? {$_.name -like "RSAT-Hyper*"}
```

This sample output shows that the Hyper-V RSAT feature is available for installation.

```
Display Name                                Name                        Install State
------------                                ----                        -------------
[X] Hyper-V Management Tools                RSAT-Hyper-V-Tools              Available
```

2. To install this feature, run the **add-windowsfeature** command.

```
[MANAGEMENTVM] PS C:\>add-windowsfeature -name RSAT-Hyper-V-Tools
```

Sample output:

```
Success Restart Needed Exit Code   Feature Result
------- -------------- ---------   --------------
True    No             Success     {Hyper-V Module for Windows PowerShell, Hy...
```

3. Once the Hyper-V RSAT feature installs, re-run the **get-windows** feature and examine all of the installed features on the Management VM.

```
[MANAGEMENTVM] PS C:\> get-windowsfeature | where installed
```

Sample output:

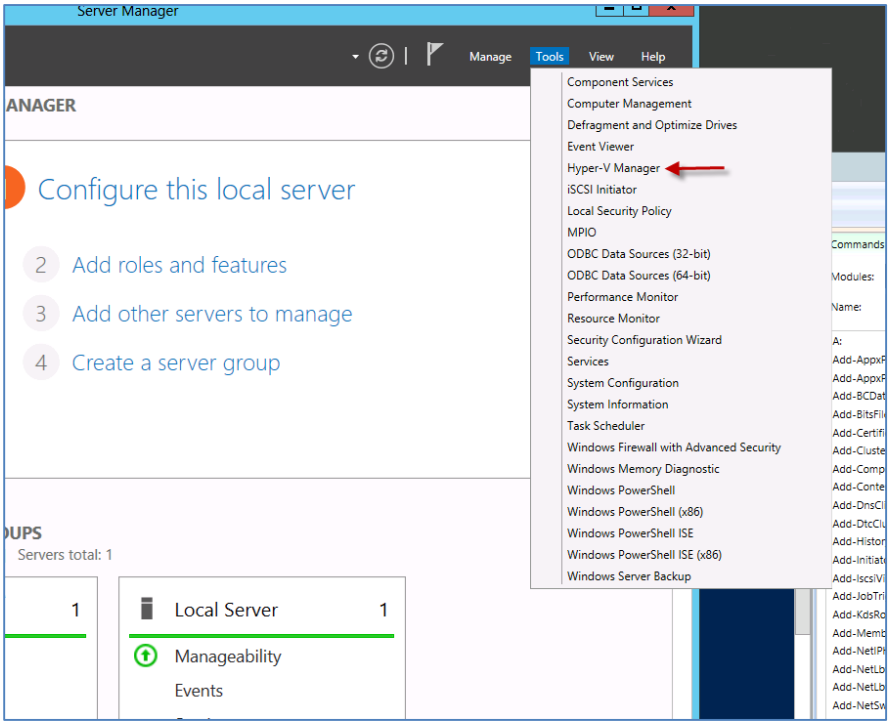
```
Display Name                                            Name                    Install State
------------                                            ----                    -------------
[X] File And Storage Services                           FileAndStorage-Services Installed
    [X] Storage Services                                Storage-Services        Installed
[X] .NET Framework 4.5 Features                         NET-Framework-45-Fea...  Installed
    [X] .NET Framework 4.5                              NET-Framework-45-Core    Installed
    [X] WCF Services                                    NET-WCF-Services45       Installed
        [X] TCP Port Sharing                            NET-WCF-TCP-PortShar...  Installed
[X] Multipath I/O                                       Multipath-IO            Installed
[X] Remote Server Administration Tools                  RSAT                    Installed
    [X] Role Administration Tools                       RSAT-Role-Tools         Installed
        [X] Hyper-V Management Tools                    RSAT-Hyper-V-Tools      Installed
            [X] Hyper-V GUI Management Tools            Hyper-V-Tools           Installed
            [X] Hyper-V Module for Windows PowerShell   Hyper-V-PowerShell      Installed
[X] User Interfaces and Infrastructure                  User-Interfaces-Infra   Installed
    [X] Graphical Management Tools and Infrastructure   Server-Gui-Mgmt-Infra   Installed
    [X] Server Graphical Shell                          Server-Gui-Shell        Installed
[X] Windows PowerShell                                  PowerShellRoot          Installed
    [X] Windows PowerShell 3.0                          PowerShell              Installed
    [X] Windows PowerShell ISE                          PowerShell-ISE          Installed
[X] WoW64 Support                                       WoW64-Support           Installed
```

Installing the Hyper-V RSAT feature initiates the installation of the Hyper-V GUI Manager and the Hyper-V PowerShell commands.

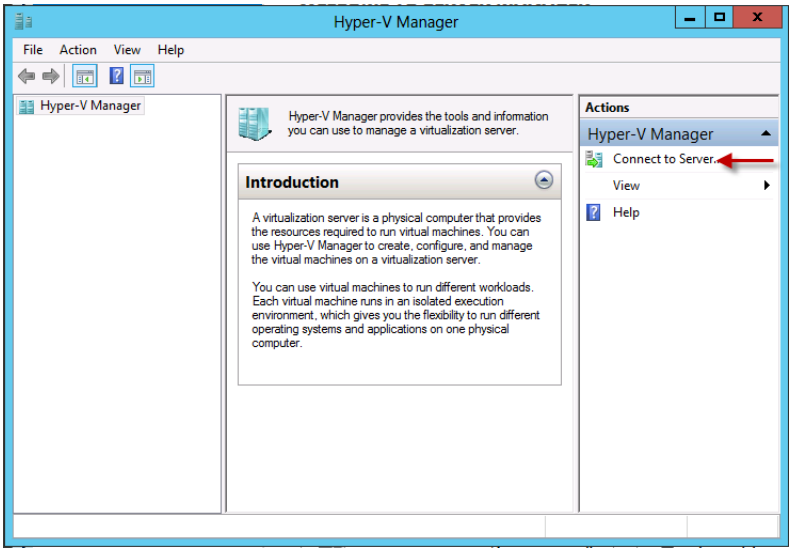
9.3 Connect to the Hyper-V Host via Hyper-V Manager

Once Hyper-V RSAT is installed on the Management VM, the Hyper-V Management Console GUI can be accessed through Server Manager.

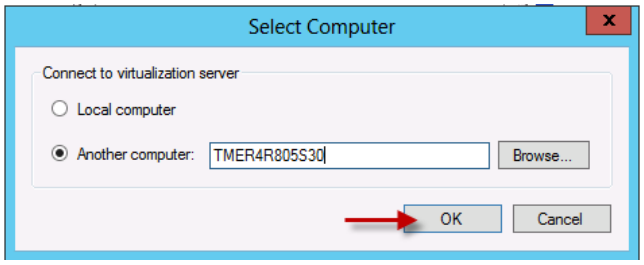
1. Click **Tools** > **Hyper-V Manager**.



2. Select **Connect to Server** in the actions pane on the right to connect to the Hyper-V host.

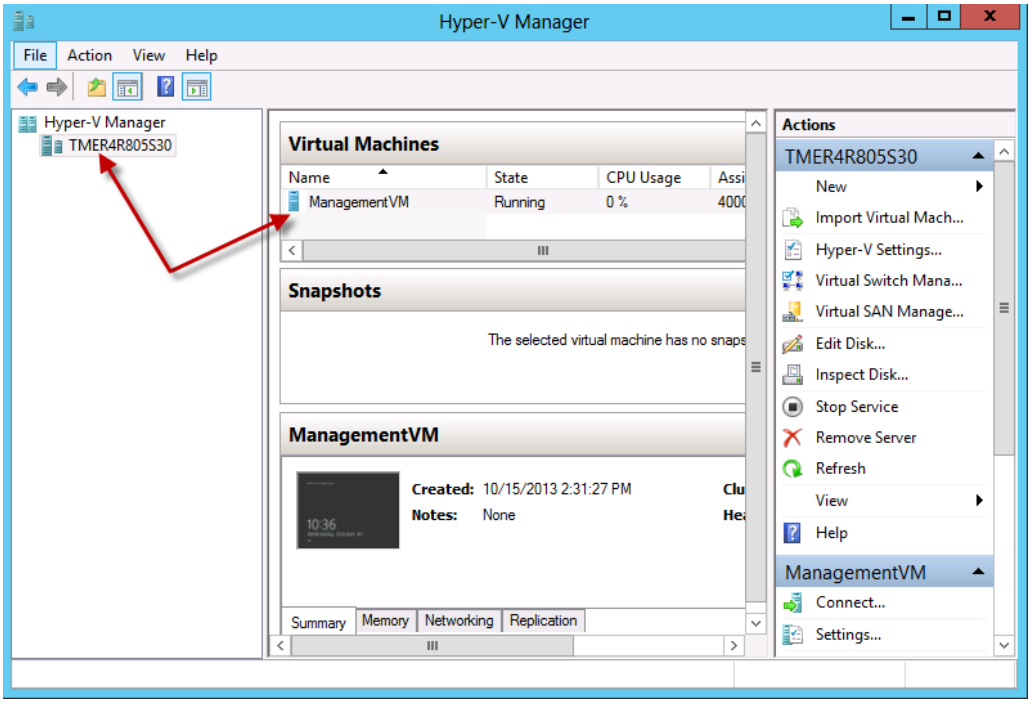


3. Enter the name or IP address of the Hyper-V server in **Another Computer**, and click **OK**.



The Hyper-V Management Console makea a remote connection to the named Hyper-V server and imports the Hyper-V information. the next console window displays the name of the Hyper-V server in the left hand pane and all of the virtual machines in the center window.

4. Select the Hyper-V server and the virtual machines to manage them through the Hyper-V Manager on the Management VM.



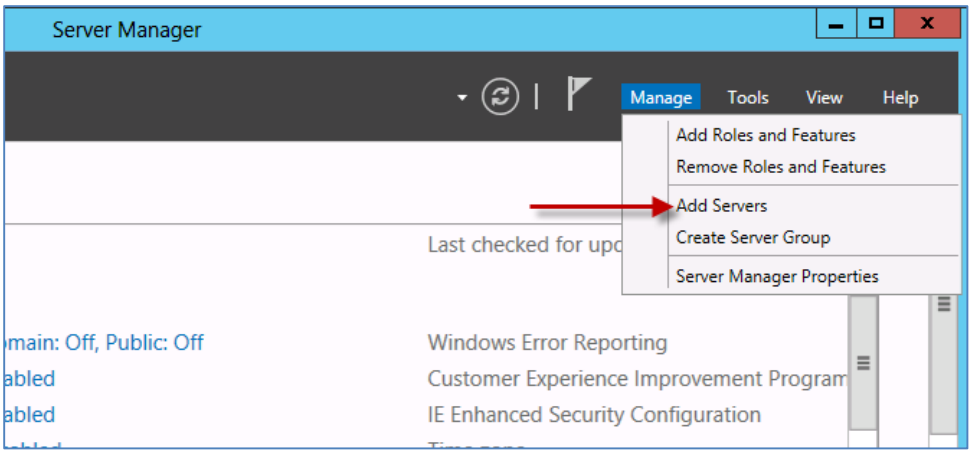
5. Once inside Hyper-V Manager, management functions such as those listed are available.
   - Examining and changing Hyper-V properties, such as the default location for storing virtual hard disk files
   - Creating new virtual machines
   - Performing virtual machine live and storage migrations
   - Starting and stopping virtual machines
   - Launching a connection window (**vmconnect**) into a virtual machine without a configured network
   - Creating and configuring Hyper-V virtual switches

In summary, Hyper-V Manager is a very powerful tool which is available to Hyper-V administrators at no additional cost from Microsoft. This tool along with PowerShell Hyper-V commands will form the primary toolkit that administrators will use to manage the Hyper-V environment on a day to day basis.

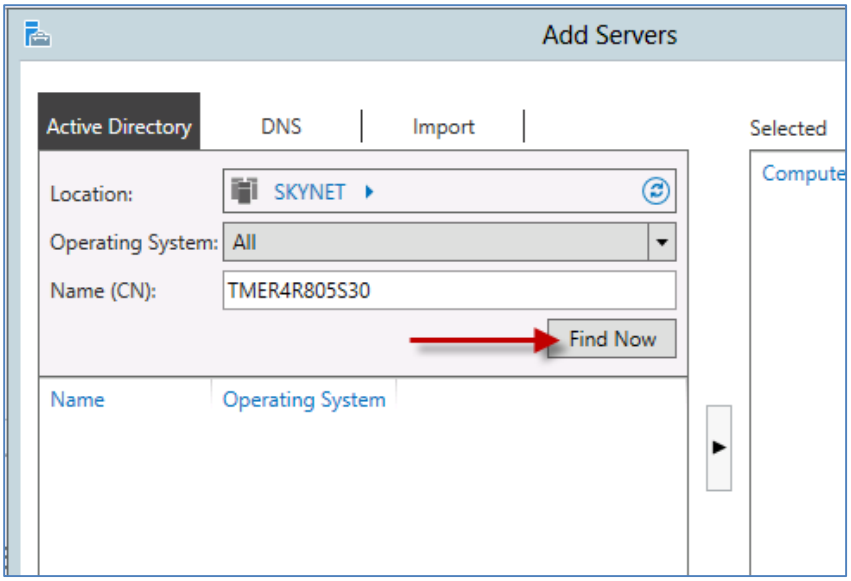
9.4 Add the Hyper-V host to the Server Manager console

One of the primary benefits of creating a dedicated management virtual machine, installed with the Windows Server GUI version, is the ability to use Server Manager as a management interface to a Server Core Hyper-V host. This concept works well as a use case for a dedicated management virtual machine since the Hyper-V host can be added as a remote server into the Server Manager configuration. Using Server Manager on the management virtual machine provides a graphical method to manage the Hyper-V Server Core host as an alternative to strictly relying on PowerShell commands. For novice or new Windows administrators this provides a significant ease-of-use benefit.

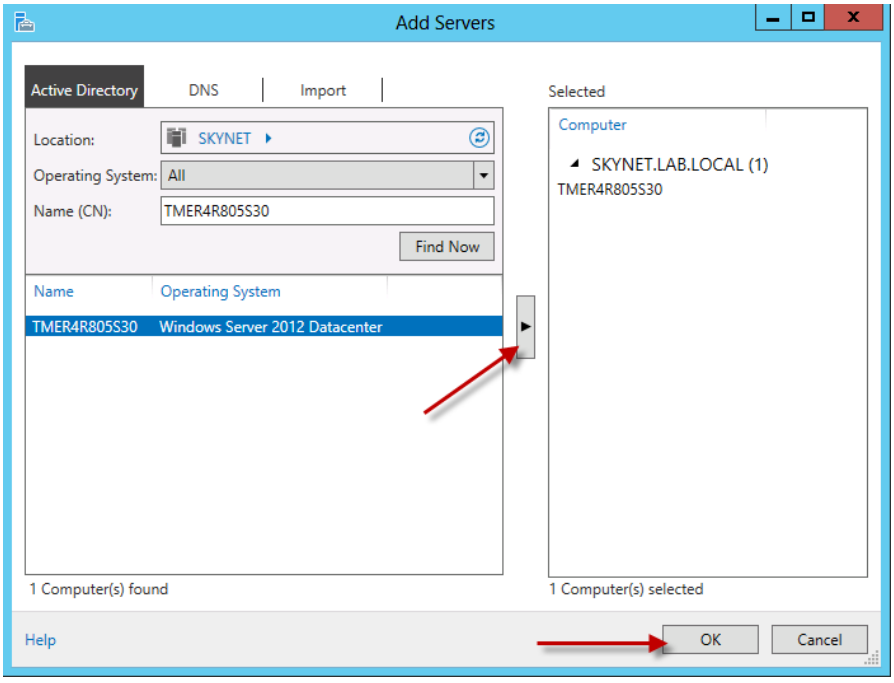
1. To add the Hyper-V Server Core host into the Server Manager configuration on the management virtual machine, open Server Manager and select **Manage** > **Add Servers** to open the **Add Servers** wizard.



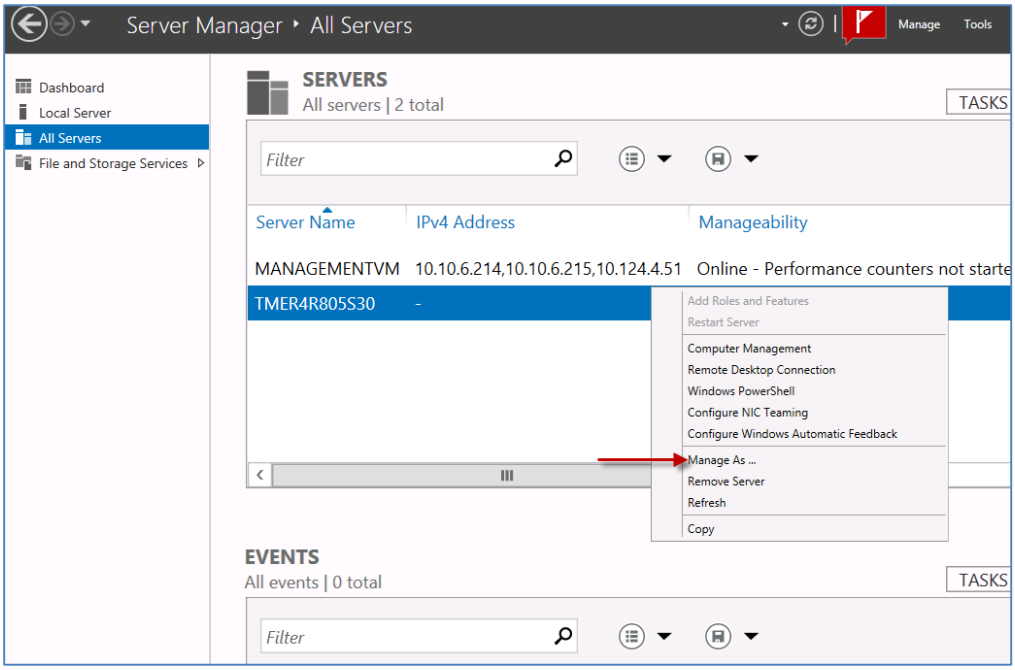
2. Type in the name or IP address of the Hyper-V Server Core host (in the example the name is TMER4R805S30) and click **Find Now**.



The wizard will scan the network for the name of the host entered and will display the results in the window below the **Find Now** button.

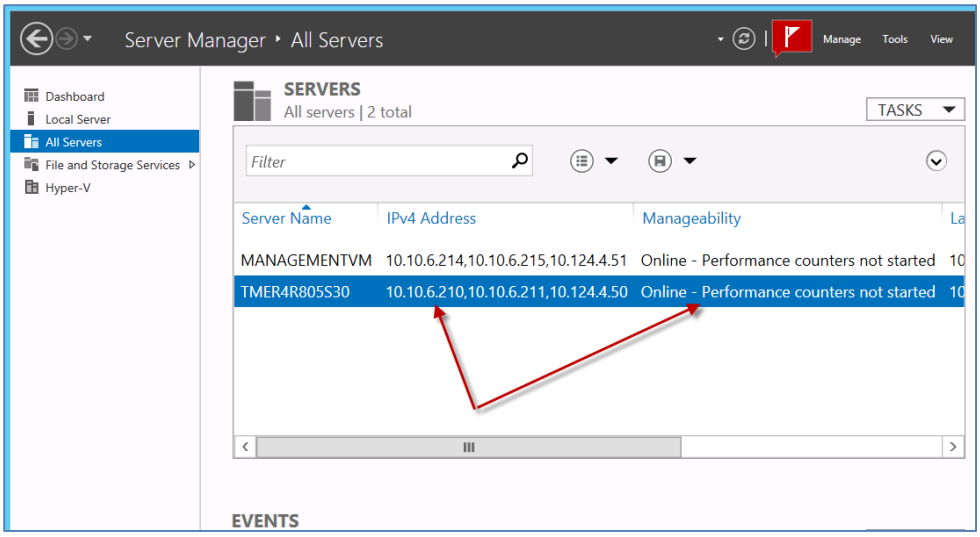


3. Highlight the server and select the right arrow button to add it to the **Selected** window. Click **Ok** to add it to the Server Manager configuration.

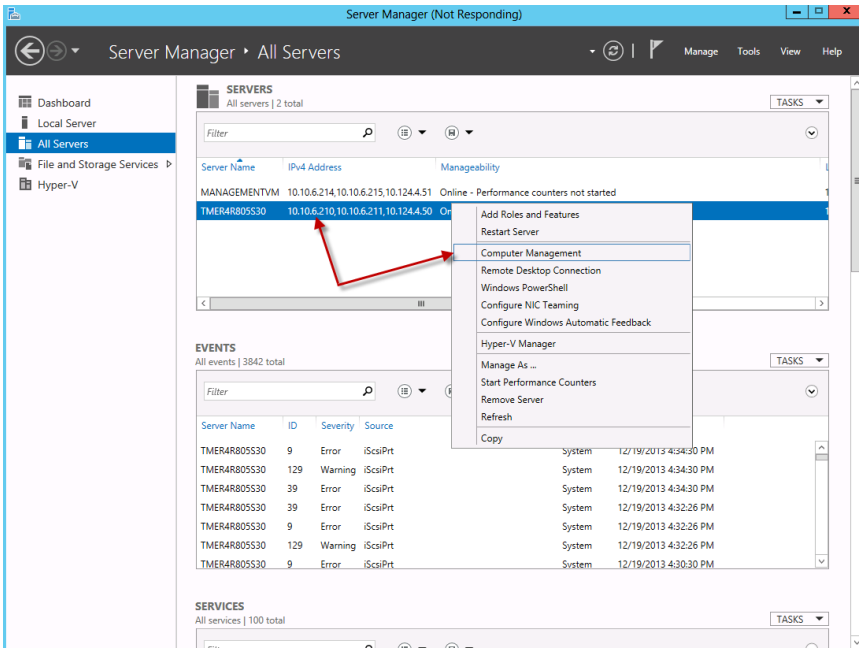


Once the Hyper-V Server is selected and added into the Server Manager Configuration, it will show up in the **All Servers Window**.

4. Right-click on the server and select **Manage As...** to enter the administrator credentials for the server. This allows it to be brought under the administrative control of the management VM server manager interface.
5. In the Windows Security pop-up, enter a user and password that enables administrative rights to the Hyper-V host and click **Ok**.
   Once the user name and password are entered and validated, the Hyper-V host is brought online in server management window with the IP addresses and manageability status displayed. In the below output, it is shown that the manageability is online without the windows performance counters being started.

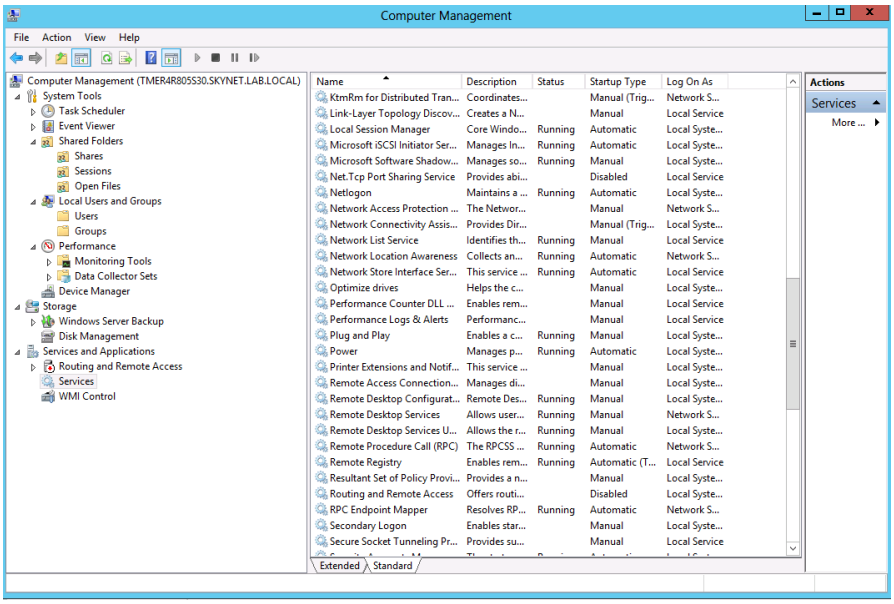


6. Right-click the name of the Hyper-V server to display the available management tasks.



Some of the available functions are **Start Performance Counters**, **Remote Desktop Connection**, **Configure NIC Teaming**, and **Computer Management**.

7. Select **Computer Management** to open the computer management wizard where performance, services, event logs, and storage device information can be obtained.



For more information on how to setup and configure Microsoft Server Manager go to:
http://technet.microsoft.com/en-us/library/hh831456.aspx

9.5 Open a remote PowerShell connection to the Hyper-V Host from the Management VM

Next, open a remote PowerShell connection to the Hyper-V host from the Management VM.

1. Add the Hyper-V host into the trusted hosts file on the Management VM.

```
[ManagementVM]: PS C:\> set-item wsman:localhost\client\trustedhosts
TMER4R805S30
```

2. Run the following **get-item** command to verify that the Hyper-V host was added to the list.

```
[ManagementVM]: PS C:\> get-item wsman:localhost\client\trustedhosts
```

Sample Output:

```
   WSManConfig: Microsoft.WSMan.Management\WSMan::localhost\Client

Type             Name                           SourceOfValue    Value
----             ----                           -------------    -----
System.String    TrustedHosts                                    TMER4R805S30
```

Note: Use a fully qualified domain name for the Hyper-V server. Any remote session from the client machine must use the name as it is exactly in the trusted host list. In domains where there is no name resolution provided by a DNS server, an IP address could be used as shown below:
`[ManagementVM]: PS C:\> set-item wsman:localhost\client\trustedhosts 10.124.4.50`

3. Once verified, open a remote PowerShell session to the Hyper-V host

```
[ManagementVM]: PS C:\> New-PSSession -ComputerName TMER4R805S30 -Credential
TMER4R805S30\administrator | enter-PSsession
```

The new remote PowerShell session opens the credential window where the Hyper-V server administrator password is entered. With the correct credentials, the command string opens the remote PowerShell session to the Hyper-V server from the Management VM using the credential of the Hyper-V server administrator account. The session opens in the default home directory of the Administrator user.

```
[TMER4R805S30]: PS C:\Users\Administrator\Documents>
```

9.6 Connect the Management VM to an EqualLogic iSCSI SAN

In this section, the management virtual machine will be connected to an EqualLogic iSCSI SAN. This process involves installing the EqualLogic HIT kit on the virtual machine as a push install from the Hyper-V host. Installing the HIT kit on the virtual machine initiates installing the many useful tools including the EqualLogic PowerShell tools, the Auto-Snapshot Manager GUI, and the Dell EqualLogic Multipath I/O DSM. Once the HIT kit is installed on the virtual machine, a new connection to the EqualLogic PS Group (TMProd) is made and EqualLogic PowerShell commands is run against it to confirm connectivity.

Note: It is a best practice recommendation to push the Dell EqualLogic HIT Kit to all virtual machines running on the Hyper-V host. This allows a management virtual machine or host (installed with the ASM/ME GUI) to access and protect the iSCSI volumes that are directly attached to the virtual machines themselves. Push the EqualLogic HIT from the Hyper-V host to the Management VM

9.6.1 Push the EqualLogic Host Integration Tools from the Hyper-V host to the Management VM

In the example, the Host Integration Tools was installed on the Hyper-V host from a setup file called Setup64.exe located on a network share (E :\). This file and location are used again for the push installation from the Hyper-V host to the virtual machine.

1. The first step in the installation process is to log into the Hyper-V host, locate this file, and note the path.

```
[TMER4R805S30] PS C:\>(ls -r E:\ | where-object {$_.name -like
"Setup64.exe"}).fullname
```

```
E:\EQLSoftware\HIT_46_GA\Setup64.exe
```

2. With the Setup64.exe file location noted, change the directory on the Hyper-V server to the EqualLogic software installation directory.
3. Locate the HitRemoteInstall.ps1 file and execute it.

```
[TMER4R805S30] PS C:\>cd 'C:\Program Files\EqualLogic\bin'

[TMER4R805S30] PS C:\Program Files\EqualLogic\bin>./HitRemoteInstall.ps1
```

4. Once this file is executed, a prompt appears requesting two pieces of information. First, the remote computer name where the software resides (ManagementVM), and then the path to the Setup64.exe file (E:\EQLSoftware\HIT_46_GA).

```
Remote Computer Names (separate with spaces): ManagementVM
MSI Location: E:\EQLSoftware\HIT_46_GA\
```

After the prompts, the installation will begin.

```
Installing E:\EQLSoftware\HIT_46_GA\\Setup64.exe on ManagementVM
Setup64.exe has been copied to ManagementVM and installation will now start
Started Setup64.exe with process ID 1636
Waiting for installation to complete on ManagementVM...
...
Waiting for installation to complete on ManagementVM...
Install process has finished on ManagementVM

Setup64.exe has been successfully installed on ManagementVM

Installation is complete on all servers
Installation log files have been moved to C:\ProgramData\EqualLogic\Log
```

The command prompt returns when the installation is complete.

```
[TMER4R805S30] PS C:\Program Files\EqualLogic\bin
```

5. After the installation completes, log into the Management virtual machine and verify that the EqualLogic services have been installed using the **get-service** command searching for services with **Equal*** in the display name.

```
[MANAGEMENTVM] PS C:\>get-service | ? {$_.displayname -like "Equal*"} | ft -
AutoSize
```

Sample output:

```
Status  Name          DisplayName
------  ----          -----------
Running EHCMService   EqualLogic Host Connection Management Service
Running EqlASMAgent   EqualLogic Auto-Snapshot Manager Agent
Running EqlLogd       EqualLogic Trace Logging Service
```

```
Running EqlReqService EqualLogic VSS Requestor
Running EqlSMPHost    EqualLogic SMP Host Service
Stopped EqlVdsHwPrv   EqualLogic VDS Hardware Provider
Running EqlVss        EqualLogic VSS Service
```

6. Finally, verify that the HIT installation has included MPIO on the virtual machine using the **get-windowsfeature** command.

   ```
   [MANAGEMENTVM] PS C:\>get-windowsfeature | ? installed | ft –autosize
   ```

   Sample output:

```
Display Name                                            Name                       Install State
------------                                            ----                       -------------
[X] File And Storage Services                           FileAndStorage-Services    Installed
    [X] Storage Services                                Storage-Services           Installed
[X] .NET Framework 4.5 Features                         NET-Framework-45-Features  Installed
    [X] .NET Framework 4.5                              NET-Framework-45-Core      Installed
    [X] WCF Services                                    NET-WCF-Services45         Installed
        [X] TCP Port Sharing                            NET-WCF-TCP-PortSharing45  Installed
[X] Multipath I/O                                       Multipath-IO               Installed
[X] User Interfaces and Infrastructure                  User-Interfaces-Infra      Installed
    [X] Graphical Management Tools and Infrastructure   Server-Gui-Mgmt-Infra      Installed
    [X] Server Graphical Shell                          Server-Gui-Shell           Installed
[X] Windows PowerShell                                  PowerShellRoot             Installed
    [X] Windows PowerShell 3.0                          PowerShell                 Installed
    [X] Windows PowerShell ISE                          PowerShell-ISE             Installed
[X] WoW64 Support                                       WoW64-Support              Installed
```

9.6.2 Register and connect an EqualLogic PS Series array to the management VM with PowerShell

Installing the EqualLogic HIT kit loaded both the EqualLogic Storage Management Provider (SMP) as well as the EqualLogic PowerShell cmdlets on the new virtual machine. The following command sequence is run from the new virtual machine.

1. To connect and register to the EqualLogic PS Series array on the new virtual machine through PowerShell, first import the EqualLogic PowerShell tools as shown below.

   ```
   [MANAGEMENTVM] PS C:\>import-module -name "c:\program
   files\equallogic\bin\EqlPSTools.dll"
   ```

2. Next, use the **new-eqlgroupaccess** command to register the PS Group.

   ```
   [MANAGEMENTVM] PS C:\>new-eqlgroupaccess -groupname TMProd -groupwkaddress
   10.10.6.200
   ```

3. Use the **connect-eqlgroup** command to open a management session to the PS Group. Note that in the example, the management IP address is used.

   ```
   [MANAGEMENTVM] PS C:\>connect-eqlgroup -groupaddress 10.124.2.239 –credential
   (get-credential) -ignoreSavedCredentials
   ```

4. Once the connection is verified, run various commands from the EqualLogic PowerShell tool kit such as **get-eqlgroupaccess** which displays the list of registered EqualLogic PS groups.

   ```
   [MANAGEMENTVM] PS C:\> get-eqlgroupaccess
   ```

   Sample output:

   ```
   SessionId           : 6019CBF194DA71B4F7221504000020C3
   GroupAddress        : 10.10.6.200
   GroupWKAddress      : 10.10.6.200
   GroupMKAddress      :
   GroupName           : TMProd
   UseSSO              : no
   ...
   ```

5. For a full listing of these commands, use **get-command** and specify **EQLPSTools** with the **module** parameter.

   ```
   [MANAGEMENTVM] PS C:\ > get-command -module EQLPStools
   ```

Detailed information is provided about these commands in the *Dell EqualLogic PowerShell Tools User Guide* found in the c:\program files\equallogic\doc directory or in the resources section of the Dell EqualLogic customer support Web site

9.7 Update the Hyper-V environment documentation

It is a best practice to update the Hyper-V environment documentation after an initial configuration or changes. The following diagram and table below reflect the configuration of the example used in this document after completing the changes.

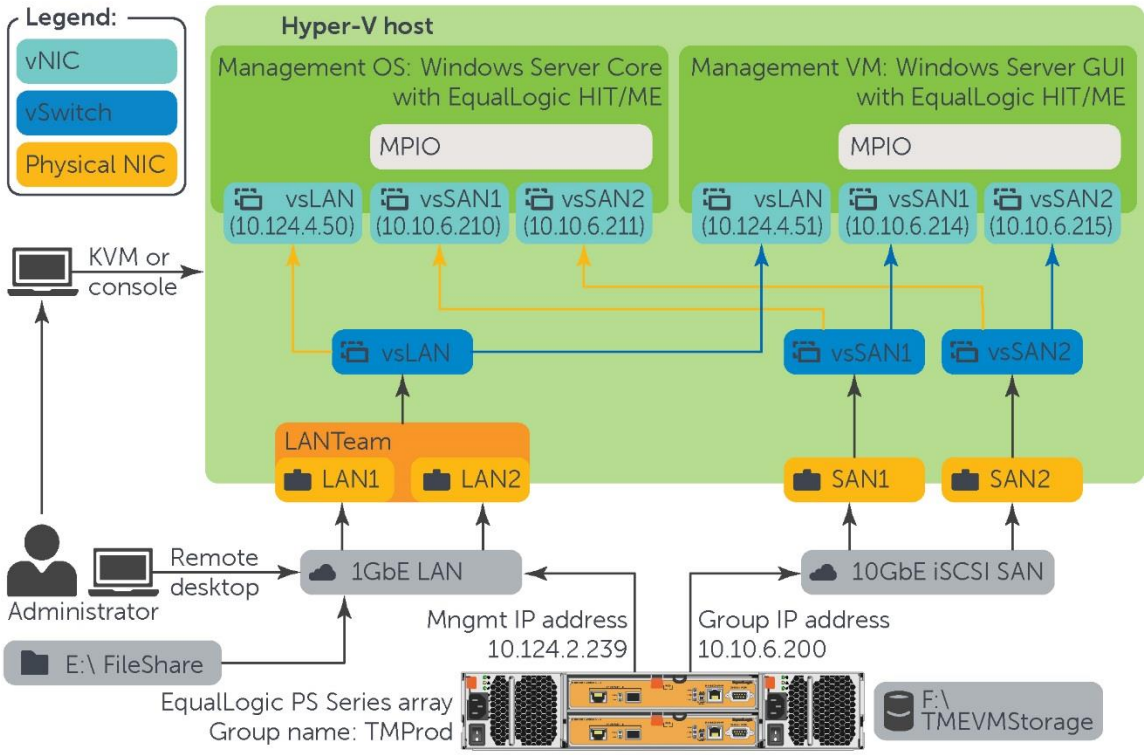


Figure 6 Final Hyper-V configuration

Table 5 Final Hyper-V component list

Hyper-V Host	Component	vNIC Name	Attached Virtual Switch	IP Address	Gateway	Subnet Mask (Prefix – address)	DNS Servers
TMER4R805S30	Management OS	vsLAN	vsLAN	10.124.4.50	10.124.4.1	22 – 255.255.252.0	10.124.6.245, 8.8.8.8
		vsSAN1	vsSAN1	10.10.6.210	NA	16 – 255.255.0.0	NA
		vsSAN2	vsSAN2	10.10.6.211	NA	16 – 255.255.0.0	NA
	ManagementVM	vsLAN	vsLAN	10.124.4.51	10.124.4.1	22 – 255.255.252.0	10.124.6.245, 8.8.8.8
		vsSAN1	vsSAN1	10.10.6.214	NA	16 – 255.255.0.0	NA
		vsSAN2	vsSAN2	10.10.6.215	NA	16 – 255.255.0.0	NA

9.8 Chapter Summary

In this document, the steps involved in creating the first virtual machine on the Hyper-V host were examined in detail. One important concept revealed is that when creating new virtual machines with Server Core, automation through PowerShell scripting and other tools allows for an accurate and fast virtual machine deployment. The abilities are similar to those provided by the GUI and wizards.

Early on in the life cycle of a Server Core based Hyper-V environment, an administrator needs to decide the best way to manage the environment. As a suggested best practice, create or turn an existing virtual machine into a dedicated Management Virtual Machine that is used to manage the Hyper-V host server. A dedicated Hyper-V management virtual machine provides the following benefits to the administrator.

- It enables a centralized management and administration of the Hyper-V host from a separate and independent interface.
- The Management VM can be installed with a full Windows Server GUI installation so that all of the graphical management tools such as Server Manager, Hyper-V Manager, PowerShell ISE, iSCSI Initiator, etc. are available to an administrator, without adding this additional overhead to the Management OS.
- A dedicated management virtual machine increases security if specific profiles and users are created with sole administrative privileges from the management virtual machine. This reduces the attack footprint.
- By installing the EqualLogic HIT Kit on a management virtual machine, an administrator who is a EqualLogic customer has the ability to easily protect the Hyper-V environment by creating a HIT group in the ASM/ME application that includes the Hyper-V host server. The administrator can then use the ASM/ME from the management virtual machine to create Smart Copy Snapshots or Clones of all the virtual machines associated with the Hyper-V server.

A set-vmnetworkconfiguration.psm1

```
<#
This function was created by Ravikanth Chaganti. It allows you to set up the
Hyper-V guest VM network configuration from the Hyper-V host. This function
simulates VMWare ESX's PowerCLI command "set-vmguestnetworkinface". Currently in
Hyper-V there is no equivalent command in native powershell vm tools (without
using SCVMM) so special thanks goes to Ravikanth for creating this. Being able
to #set the guest VM network from the Hyper-V host is extremely useful
when performing automated guest OS installations or rapid VM deployments.

For more information on this function, go
http://www.ravichaganti.com/blog/?p=2766

Typical Usage:
First -Embed this function in a script or use it as a module then import it by:
 $ModulePath = "C:\Users\Administrator\Documents\WindowsPowerShell\Modules"
 import-module $ModulePath\set-vmnetworkconfiguration.psm1

Second - pipeline get-vmnetwork adapter output into the function and set the
network information for the adapter:

Get-VMNetworkAdapter -VMName Demo-VM-1 -Name iSCSINet |
Set-VMNetworkConfiguration -IPAddress 192.168.100.1 -Subnet 255.255.0.0
-DNSServer 192.168.100.101 -DefaultGateway 192.168.100.1
#>

  Function Set-VMNetworkConfiguration {
      [CmdletBinding()]
      Param (
          [Parameter(Mandatory=$true,
                     Position=1,
                     ParameterSetName='DHCP',
                     ValueFromPipeline=$true)]
          [Parameter(Mandatory=$true,
                     Position=0,
                     ParameterSetName='Static',
                     ValueFromPipeline=$true)]
          [Microsoft.HyperV.PowerShell.VMNetworkAdapter]$NetworkAdapter,

          [Parameter(Mandatory=$true,
                     Position=1,
                     ParameterSetName='Static')]
          [String[]]$IPAddress=@(),

          [Parameter(Mandatory=$false,
```

```
                    Position=2,
                    ParameterSetName='Static')]
            [String[]]$Subnet=@(),

            [Parameter(Mandatory=$false,
                    Position=3,
                    ParameterSetName='Static')]
            [String[]]$DefaultGateway = @(),

            [Parameter(Mandatory=$false,
                    Position=4,
                    ParameterSetName='Static')]
            [String[]]$DNSServer = @(),

            [Parameter(Mandatory=$false,
                    Position=0,
                    ParameterSetName='DHCP')]
            [Switch]$Dhcp
        )

        $VM = Get-WmiObject -Namespace 'root\virtualization\v2' -Class
'Msvm_ComputerSystem' |
Where-Object { $_.ElementName -eq $NetworkAdapter.VMName }
        $VMSettings = $vm.GetRelated('Msvm_VirtualSystemSettingData') |
Where-Object { $_.VirtualSystemType -eq 'Microsoft:Hyper-V:System:Realized' }
        $VMNetAdapters =
$VMSettings.GetRelated('Msvm_SyntheticEthernetPortSettingData')

        $NetworkSettings = @()
        foreach ($NetAdapter in $VMNetAdapters) {
            if ($NetAdapter.Address -eq $NetworkAdapter.MacAddress) {
$NetworkSettings = $NetworkSettings + `
$NetAdapter.GetRelated("Msvm_GuestNetworkAdapterConfiguration")
            }
        }

        $NetworkSettings[0].IPAddresses = $IPAddress
        $NetworkSettings[0].Subnets = $Subnet
        $NetworkSettings[0].DefaultGateways = $DefaultGateway
        $NetworkSettings[0].DNSServers = $DNSServer
        $NetworkSettings[0].ProtocolIFType = 4096

        if ($dhcp) {
            $NetworkSettings[0].DHCPEnabled = $true
        } else {
            $NetworkSettings[0].DHCPEnabled = $false
        }
```

```
      $Service = Get-WmiObject -Class "Msvm_VirtualSystemManagementService" `
-Namespace "root\virtualization\v2"

      $setIP = $Service.SetGuestNetworkAdapterConfiguration($VM,
$NetworkSettings[0].GetText(1))

      if ($setip.ReturnValue -eq 4096) {
          $job=[WMI]$setip.job

          while ($job.JobState -eq 3 -or $job.JobState -eq 4) {
              start-sleep 1
              $job=[WMI]$setip.job
          }

          if ($job.JobState -eq 7) {
              write-host "Success"
          }
          else {
              $job.GetError()
          }
      } elseif($setip.ReturnValue -eq 0) {
          Write-Host "Success"
      }
  }
```