Setup and Configuration of a Windows Server Core Hyper-V host using PowerShell and EqualLogic Storage and Tools

Dell Engineering
May 2014

A Dell Deployment and Configuration Guide

Revisions

Date	Description
May 2014	Initial release

Table of contents

Executive summary

This document focuses on building the first part of the virtualized Windows environment: the Hyper-V Host Server. It examines the requirements and options, and provides step-by-step guidance for efficiently accomplishing this task.

The Microsoft recommended OS platform for the Hyper-V host server is Server Core because it installs the minimum amount of services required for the Windows OS. This provides more host resources for the Hyper-V role and results in higher security. The traditional Windows GUI and configuration wizards provide a methodology for an administrator to perform a specific task. In Server Core, there is no GUI or wizard to provide this methodology. Server Core configuration and management has to be done mostly with PowerShell; therefore, initial Server Core experiences can be intimidating for administrators who do not have Windows command line or PowerShell experience.

The step-by-step methodologies described in this document will help eliminate this concern. The intent of the methodology shown in this document is to create a repeatable process that an administrator can reuse for hyper-v installations and configurations.

Audience

This is paper is a technical solution guide intended for Information Technology administrators in the Small Medium Business space (SMB). This paper focuses on how an administrator can virtualize a server environment using Microsoft Server 2012 / Server 2012 R2 Hyper-V with Dell EqualLogic PS Series Storage and Dell EqualLogic Storage Software. This paper assumes that the reader has:

- Previous Windows Server administration experience which has included installation and configuration
- Understanding of virtualization technologies
- Understanding of Storage Area Networks and Basic LAN Networking
- 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 should 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 excellent resources:

- Dell Technical Reports: Windows Command-line Automation Techniques for Dell EqualLogic PS Series Arrays,
- Ed Wilson (2013), Windows PowerShell 3 Step by Step, O'Rielly Media, Inc,
- Don Jones and Jeffery D. Hicks (2013), Learn Windows PowerShell 3 in a Month of Lunches, Manning Publications

Although Microsoft's System Center is a key component of their virtualization strategy, especially in larger environments, providing a detailed discussion on System Center is also beyond the scope of this paper. This paper will focus on small to medium Hyper-V deployments in which System Center functionality is not yet required. For more information on Microsoft System Center and its integration with Dell EqualLogic Storage, consult the following Dell Technical Whitepaper:

Dell Technical Reports: Automation and Integration with Microsoft System Center Virtual Machine Manager 2012 SP1 and Dell EqualLogic Storage

Software versions supported by this document

The following table lists the software versions that this document will support

Software	Notes
Microsoft Windows Server 2012	
Microsoft Windows Server 2012 R2	
Dell EqualLogic Host Integration Toolkit Microsoft Edition (HIT/ME) V4.7	Support for both Windows Server 2012 and Server 2012 R2
Dell EqualLogic Host Integration Toolkit Microsoft Edition (HIT/ME) V4.6	Support for Windows Server 2012
Dell EqualLogic PS Series Array Firmware V7.0.1	

Overview

At a high level, the Hyper-V host installation and configuration process breaks down into seven required and three optional steps. These steps are detailed in Sections 1 through 8.

1. Install Windows Server Core (Data Center) as the OS.
2. Configure the Network for the Hyper-V host.
3. Complete the OS installation with the **sconfig** utility.
4. Install Hyper-V and reboot.
5. Setup and configure the Hyper-V virtual switches.
6. Connect Hyper-V Host to an EqualLogic PS Series iSCSI SAN.
7. Verify that Remote PowerShell (PSRemoting) is enabled.
8. Optionally:
   a. Setup Converged Networking and QOS support.
   b. Disable unused services such as IPV6.
   c. Convert from Server Core to Minimal Server Interface or the full GUI.

This flow chart illustrates the configuration and setup process of the Hyper-V host. Each process of the flow chart is detailed in Sections 1 through 8.

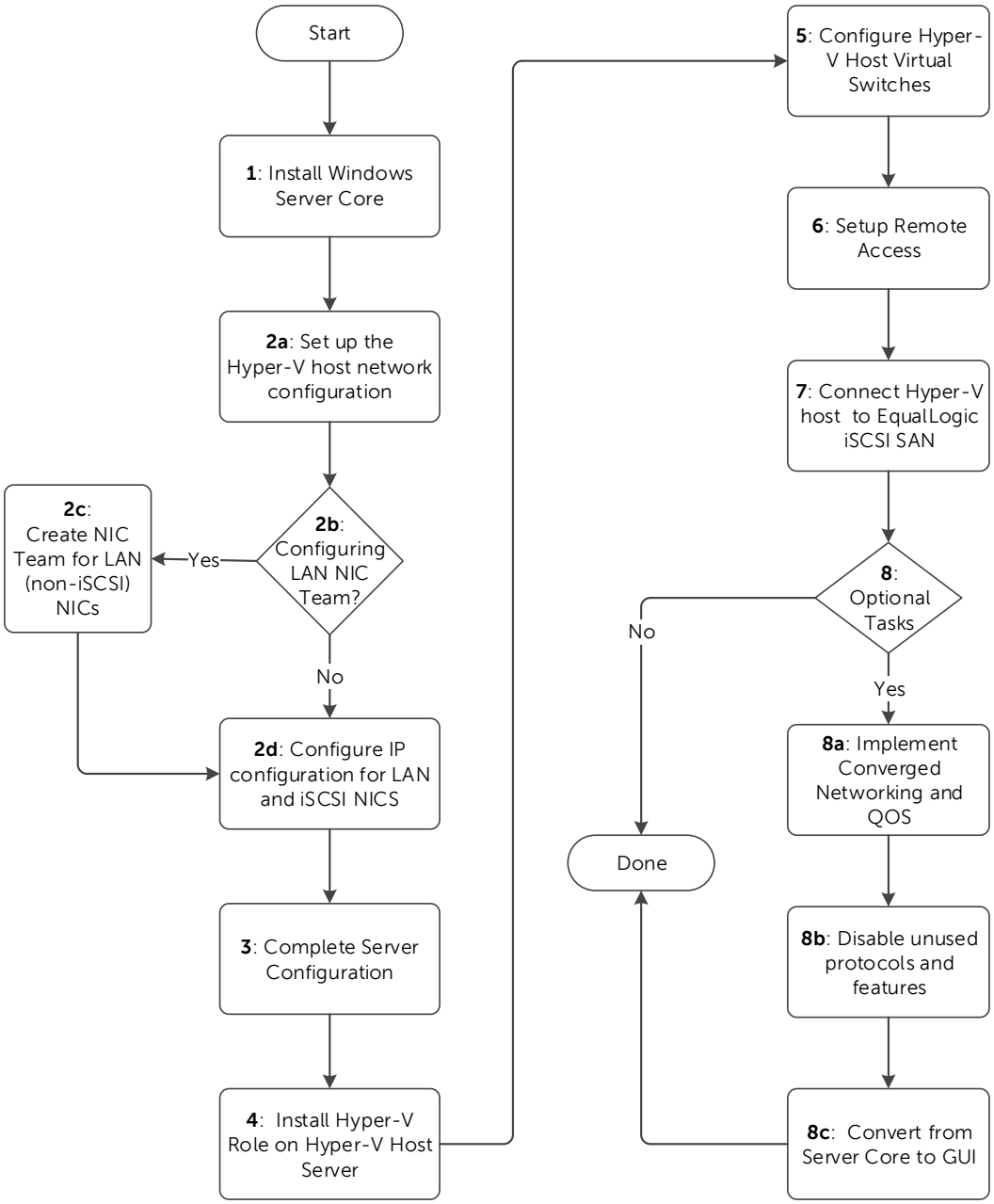


Figure 1 Configuration and setup process of the Hyper-V host

1 Step 1: Install Windows Server Core

Installing Windows Server Core is not any different than installing the Windows Server version with a GUI. The installer requires access to a Windows Server ISO when is starts. This can be a DVD, USB drive, or boot from network setup in the BIOS that allows direct physical access or access through a KVM switch.

Note: The installation can be fully automated with the use of an answer file using the Windows ADK. Setting up an answer file using the SDK is discussed in detail at the following blog: http://www.derekseaman.com/2012/07/windows-server-2012-unattended.html

Begin by selecting the installation language and type of Windows operating system to install (GUI or Core).

In the screen below, the default option (Server Core) is selected. Starting with Server 2012, Microsoft recommends the Server Core installation mode, especially for Hyper-V hosts.

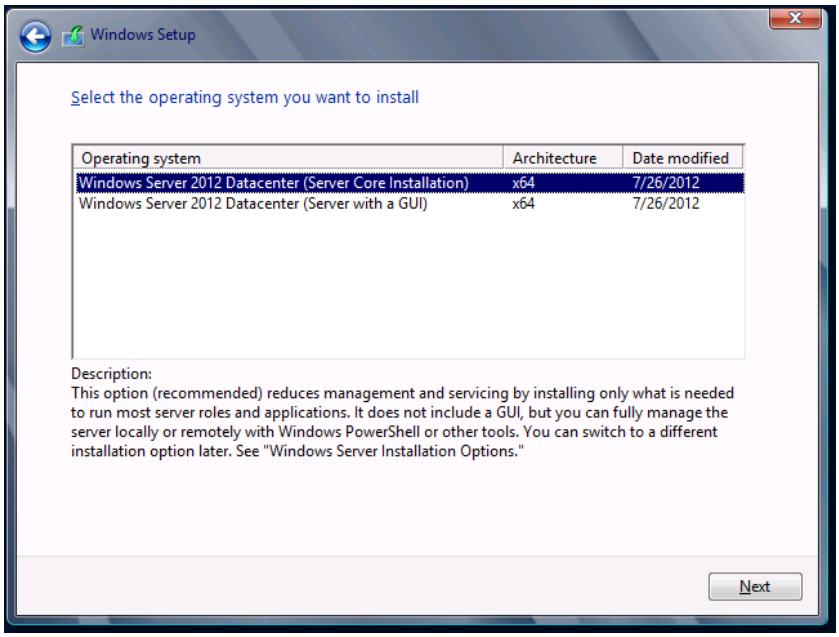


Figure 2 Server Core as the default selection

The remaining Windows OS installation steps are the same for both the GUI and Core versions. For Windows Server Core installation details, refer to the "Install and Deploy" article on the Microsoft TechNet library at http://technet.microsoft.com/en-us/library/hh831620.aspx

After completing the installation and logging into the host for the first time, the user is greeted with a very different experience on server core than with the traditional GUI. Server Core presents the user with a blank screen and DOS prompt.

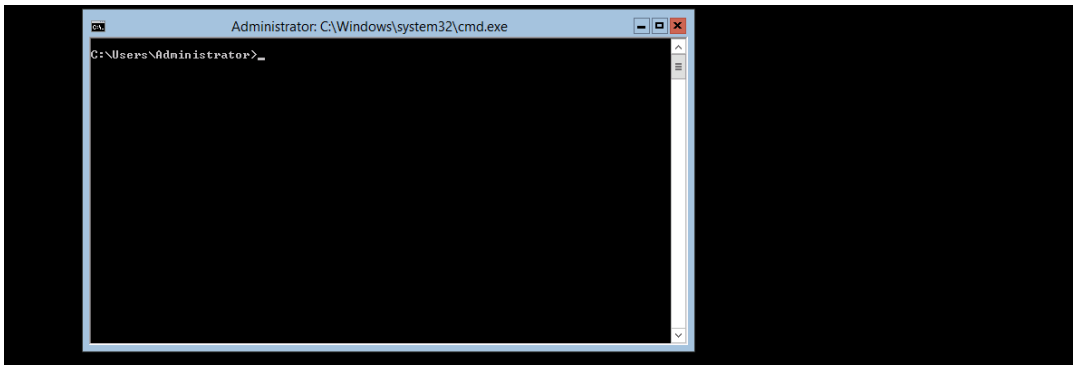


Figure 3 The Default Windows Server Core Desktop

Upon installation and first login, launch two utilities with Server Core:

- A task manager to monitor system tasks and application performance, and
- A shell window where commands and scripts can be run.

1.1 Launch Task Manager

1. Press Ctrl+Alt+Delete or use the equivalent key strokes in the KVM screen to display the lock screen menu. In some remote consoles, Ctrl+Alt+End also performs this function.
2. From the lock screen menu, select **Task Manager** to open the task manager on the desktop.

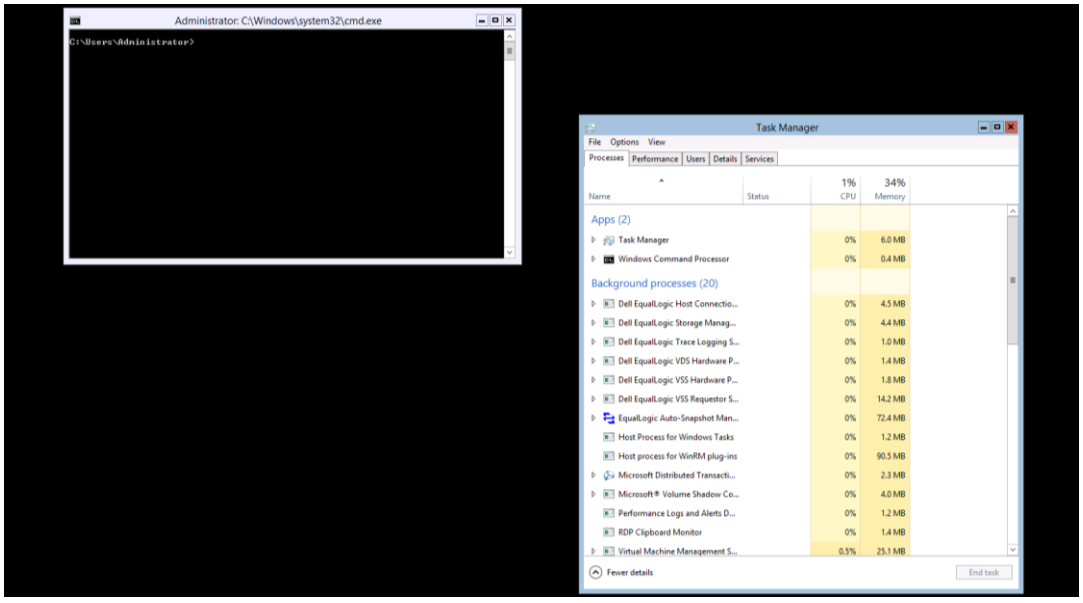


Figure 4 Launching Task Manager

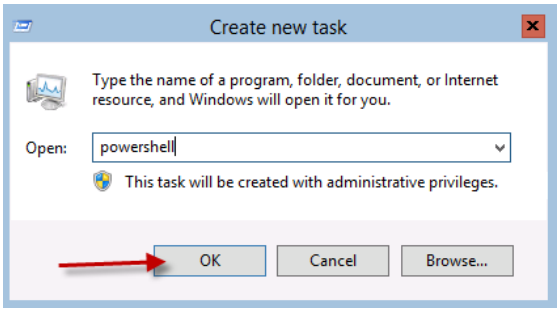
1.2 Launch a PowerShell window

The primary shell for Windows Server Core is PowerShell. To use server core, and to be an effective Windows Server Core administrator, some understanding of PowerShell is required. According to Microsoft, having the understanding and the ability to use PowerShell will be a key component to being a successful Windows Server administrator going forward.

Note: Often, PowerShell documentation uses the terms *command* and *cmdlets* interchangeably. For all practical purposes, the two terms are equivalent, but the specific definitions are:
• A cmdlet is a native PowerShell command-line utility which uses the .NET Framework language like C#. It is lightweight; can utilize a series of parameters; and returns an object or list of objects which can be passed on as input to other cmdlets.
• A command can consist of a single cmdlet and its parameters, or a series of cmdlets and parameters executed from a command line utility.

The PowerShell examples in this paper consist of a single cmdlet and as well as multiple cmdlets strung together. For simplicity, all PowerShell examples in this paper will be referred to as commands even if they contain a single cmdlet.

1. Go to the Task Manager and select **File** > **Run new task**.
2. When the **Create new task** window opens, type *PowerShell* and click **OK**.



a. Alternatively, PowerShell can be launched from the console window by typing "PowerShell" at the command prompt.

```
C:\Users\Administrator> PowerShell
```

b. And then opening a separate PowerShell window.

```
PS C:\Users\Administrator> start-process PowerShell
```

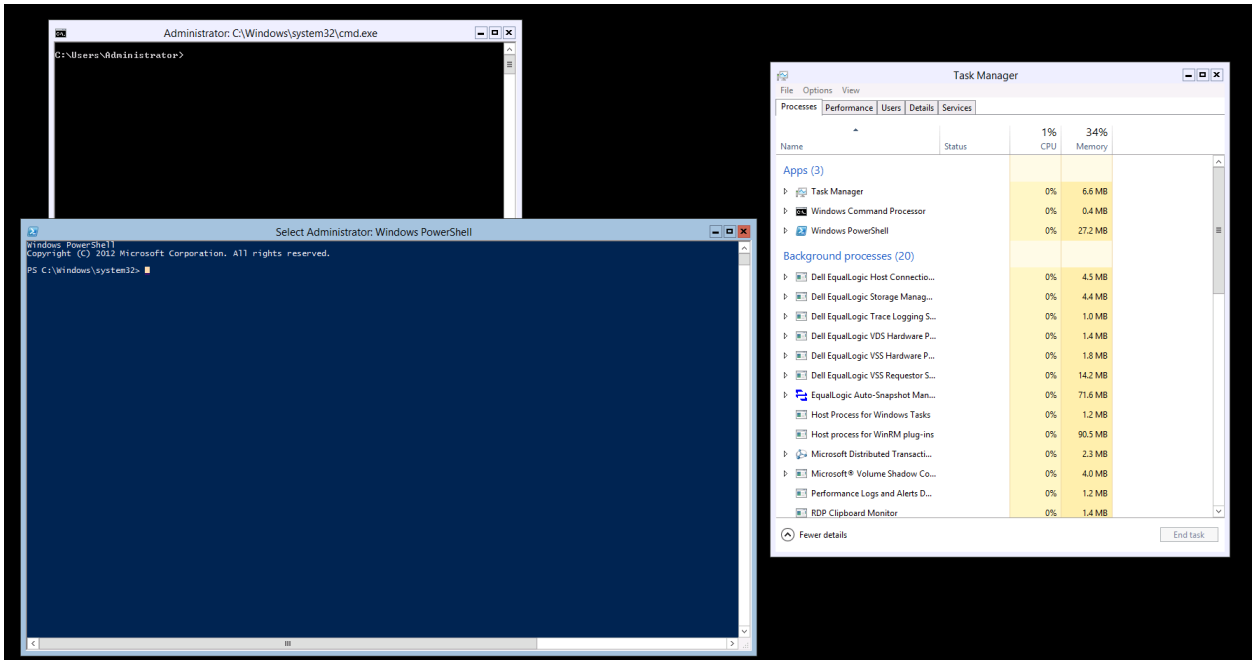


Figure 5 Windows Server Core desktop with a PowerShell window open

1.3 Feature and performance review

After the installation is complete and the task manager and PowerShell window have been opened on the desktop, review the installed features and running services. Also, check the event logs and some baseline performance metrics.

1. To review the installed features, go to the PowerShell window and run the following command.

```
PS C:\ get-windowsfeature | where-object {$_.installstate -eq "installed"}
```

An example of features list is displayed below.

```
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] User Interfaces and Infrastructure                  User-Interfaces-Infra      Installed
[X] Windows PowerShell                                  PowerShellRoot             Installed
    [X] Windows PowerShell 3.0                          PowerShell                 Installed
[X] WoW64 Support                                       WoW64-Support              Installed
```

2. To review the installed services, go to the PowerShell window and run the following command.

```
PS C:\ get-service | where-object {$_.status -eq "running"}
```

An example of the services list is displayed below.

```
Status           Name                 DisplayName
------           ----                 -----------
Running          BFE                  Base Filtering Engine
Running          BITS                 Background Intelligent Transfer Ser...
Running          CertPropSvc          Certificate Propagation
Running          CryptSvc             Cryptographic Services
...
Running          WinRM                Windows Remote Management (WS-Manag...
Running          wuauserv             Windows Update
```

3. To display the startup mode and process ID for each installed service, use the following PowerShell command.

```
PS C:\get-wmiobject win32_service | ft –AutoSize
```

An example of this list is displayed below.

```
ExitCode   Name            ProcessId       StartMode            State       Status
--------   ----            ---------       ---------            -----       ------
0          AeLookupSvc     0               Manual               Stopped     OK
1077       AppMgmt         0               Manual               Stopped     OK
0          BFE             1352            Auto                 Running     OK
0          BITS            1036            Manual               Running     OK
...
0          WinRM           1208            Auto                 Running     OK
1077       wmiApSrv        0               Manual               Stopped     OK
0          wuauserv        0               Manual               Stopped     OK
1077       wudfsvc         0               Manual               Stopped     OK
```

4. To display the processes that are running, use the following PowerShell command.

```
PS C:\get-process
```

An example of the list of running processes is displayed below.

```
Handles     NPM(K)      PM(K)       WS(K)       VM(M)       CPU(s)      Id      ProcessName
-------     ------      -----       -----       -----       ------      --      -----------
22          3           1436        1892        12          0.02        1156    cmd
50          7           2272        6836        67          3.79        1408    conhost
47          6           1048        4676        64          0.16        2404    Conhost
133         10          1364        3356        42          1.76        776     csrss
...
800         41          16160       24212       870         77.11       1208    svchost
343         31          11112       13112       55          25.97       1352    svchost
650         29          36988       44556       159         12.29       1464    Svchost
256         20          12548       10952       1101        11.06       1516    svchost
1030        0           112         304         3           7,060.73    4       System
142         11          3564        6072        93          0.09        1364    taskhostex
285         21          11740       20892       191         6,238.34    1380    Taskmgr
656         35          44928       31544       167         11.58       1568    Vmms
84          8           876         3472        21          0.06        836     wininit
96          7           1156        4800        48          0.09        888     winlogon
132         8           1224        5448        49          0.28        2136    winlogon
```

If these command outputs were compared to a Windows Server Core GUI installation, it would be evident that many of the traditional features installed with the GUI version (such as DNS, Active Directory) are not present with Server Core. This results in an installation which takes less disk space, CPU cycles, less memory, and presents a smaller attack surface; all are key requirements for a secure and efficient Hyper-V host server.

5. Another important post installation task is to examine the event logs as they are invaluable when troubleshooting host issues. To examine the event logs (system, application, security, etc..) use the **get-eventlog** command.

```
PS C:\ get-eventlog -LogName System -Newest 25 | ft –AutoSize
```

This command returns a response such as:

```
Index   Time        EntryType     Source                 InstanceID    Message
-----   ----        ---------     ------                 ----------    -------
101063  Nov 07      Information   Service Control        1073748860    The WMI Performance
        10:00                     Manager                              Adapter…
101062  Nov 07      Information   Service Control        1073748860    The WMI Performance
        10:00                     Manager                              Adapter…
...
```

Note: The **get-eventlog** command has been superseded by the **get-winevent** command. The above **get-eventlog** command equivalent using the **get-winevent** command is:

```
PS C:\ get-winevent -LogName System -Maxevents 25 | ft –AutoSize
```

Note: For details about viewing Windows events using PowerShell go to: http://blogs.msdn.com/b/powershell/archive/2009/05/21/processing-event-logs-in-powershell.aspx

6. Finally, to examine performance counters (CPU, Memory, Network, etc.), use either task manager or the **get-counter** command.

```
PS C:\ Get-Counter -Counter "\\tmer4r805s30\memory\% committed bytes in use"
-SampleInterval 2 -MaxSamples 3
```

Performance counters will be displayed as shown in the following example.

```
Timestamp              CounterSamples
---------              --------------
11/7/2013 10:44:29 AM  \\tmer4r805s30\memory\% committed bytes in use :6.2952370668016
11/7/2013 10:44:31 AM  \\tmer4r805s30\memory\% committed bytes in use :6.2957212138282
11/7/2013 10:44:33 AM  \\tmer4r805s30\memory\% committed bytes in use :6.2960685366951
```

Notes about this section:

- Help can be found on all PowerShell commands by typing **man** *commandname*
  Example: `PS C:\ man get-counter`

```
NAME
    Get-Counter
Description
...
```

- PowerShell allows users to use short hand notation. For example: **man** is equivalent for **get-help** in PowerShell (old school UNIX administrators take note), and the **?** can be substituted for **where-object** when filtering.
  For example:

```
get-service | where-object {$_.status -eq "running"}
```

  is equivalent to…

```
get-service | ? {$_.status -eq "running"}
```

- The **get-wmiobject** command, Windows Management Instrumentation (WMI) and the Common Information Model (CIM). Each Windows server contains thousands of pieces of management information. WMI is a hierarchical namespace in which the management information of a Windows server is stored and accessed by external utilities such as PowerShell. WMI is extremely powerful, but historically it has been poorly documented. CIM is new in PowerShell V3.0 and it creates a new and less

confusing programming interface (API) for interfacing with the WMI. As personal knowledge of PowerShell increases, it is important to get an understanding of what WMI and CIM can offer, particularly in Hyper-V. One of the modules used later in this document uses a module to call WMI classes that provide detailed information about the virtual machines on the Hyper-V host. This script uses the information and allows the IP addresses to be set for the virtual machine from the Hyper-V host at the time the virtual machine is created. The EqualLogic HIT kit includes an EqualLogic Storage Management Provider (SMP) component that uses the WMI API under the covers to interface with Windows Server Manager.

2 Step 2: Setup the Hyper-V Host network configuration

Once the Windows Server Core installation is complete on the Hyper-V host, configure the network adapters to access and manage the server remotely. Until the network configuration is complete, the Hyper-V host can only be accessed through a direct physical console or KVM switch as shown in this document. The diagram below shows the initial configuration for this example immediately after installing Server Core.

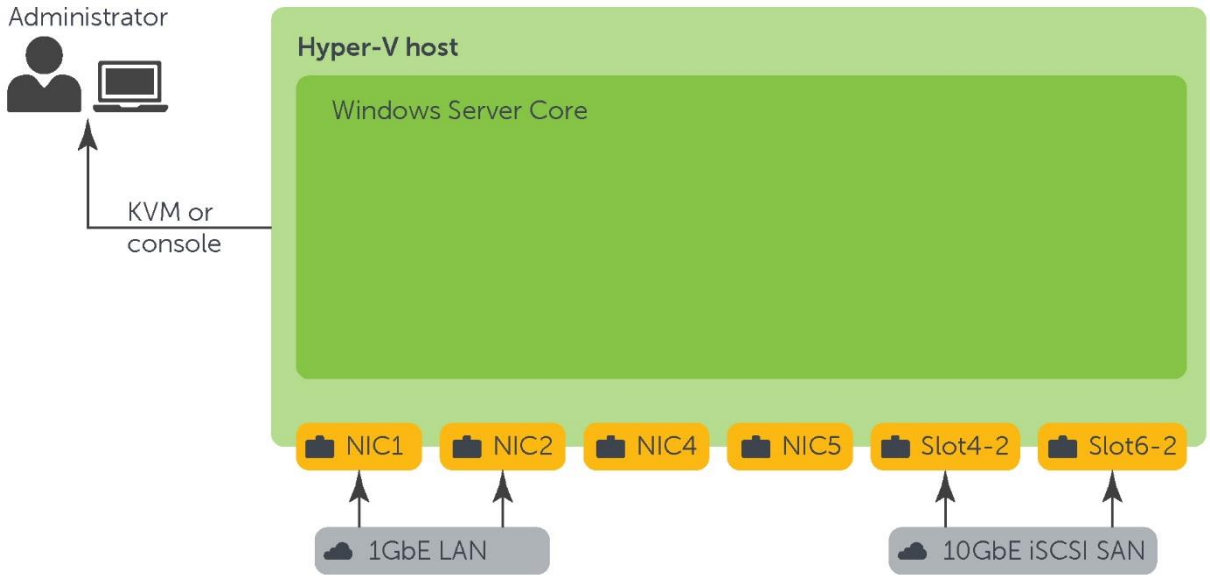


Figure 6 Initial configuration after installing Server Core

2.1 View network adapter information for the server

The first step in setting up the network for the Hyper-V host server is to identify the physical NICs on the server. This can be accomplished by using the **get-netadapter** command.

```
PS C:\ get-netadapter | ft –autosize
```

Sample output:

```
Name   InterfaceDescription                ifIndex   Status      MacAddress      LinkSpeed
----   --------------------                -------   ------      ----------      ---------
NIC5   Broadcom NetXtreme Gigabit Ethernet 14        Not Present 90-B1-1C-56-23- 1 Gbps
       #5                                                        B3
NIC4   Broadcom NetXtreme Gigabit Ethernet 13        Not Present 90-B1-1C-56-23- 1 Gbps
       #4                                                        B2
NIC2   Broadcom NetXtreme Gigabit Ethernet 11        Up          90-B1-1C-56-21- 1 Gbps
       #2                                                        B2
NIC1   Broadcom NetXtreme Gigabit Ethernet 10        Up          90-B1-1C-56-21- 1 Gbps
                                                                 B1
SLOT 4 Broadcom BCM57810 NetXtreme II      12        Up          00-10-18-ED-C1- 10 Gbps
Port 2 1...#134                                                  32
SLOT 6 Broadcom BCM57810 NetXtreme II      13        Up          00-10-18-ED-A0- 10 Gbps
```

```
Port 2 1...#131                                                    82
```

When the NIC is in the **Up** state, it means that the NIC is cabled and has access to a network. However, the **Up** state does not mean the NIC has network connectivity.

Since most Hyper-V servers have multiple NIC cards installed, it is a good idea to filter the output of the **get-netadapter** command so that only the NICs in the **Up** state are shown.

```
PS C:\ get-netadapter | where-object {$_.status -eq "up"} | ft –autosize
```

Sample output:

```
Name     InterfaceDescription                ifInde   Status       MacAddress      LinkSpeed
                                             x
----     --------------------                ------   ------       ----------      ---------
                                             -
NIC2     Broadcom NetXtreme Gigabit          11       Up           90-B1-1C-56-    1 Gbps
         Ethernet #2                                               21-B2
NIC1     Broadcom NetXtreme Gigabit          10       Up           90-B1-1C-56-    1 Gbps
         Ethernet                                                  21-B1
SLOT 4   Broadcom BCM57810 NetXtreme II      12       Up           00-10-18-ED-    10 Gbps
Port 2   1...#134                                                  C1-32
SLOT 6   Broadcom BCM57810 NetXtreme II      13       Up           00-10-18-ED-    10 Gbps
Port 2   1...#131                                                  A0-82
```

Another best practice is to disable or disconnect any NICs that are not in use with the **disable-netadapter** command:

```
PS C:\ disable-netadapter -name "NIC4"
PS C:\ disable-netadapter -name "NIC5"
```

Disabling unused NICs saves CPU cycles and power consumption. If needed, the NICs can be enabled again with the **enable-netadapter** command.

2.2 Modify physical NIC parameters on the Hyper-V host

2.2.1 Rename LAN and iSCSI NICs

On Hyper-V host servers that have multiple NICs installed, rename the NICs for their intended purpose or network connection. This helps to organize and document the configuration. Renaming NICs in PowerShell is simple with the **rename-netadapter** command as shown in the examples below.

```
PS C:\ rename-netadapter -name "NIC1" LAN1
PS C:\ rename-netadapter -name "NIC2" LAN2
PS C:\ rename-netadapter -name "SLOT 4 Port 2" SAN1
PS C:\ rename-netadapter -name "SLOT 6 Port 2" SAN2
```

After using the **rename-netadapter** command, verify that LAN and iSCSI SAN NICs have been renamed successfully using **get-netadapter** PowerShell command.

```
PS C:\ get-netadapter | where-object {$_.status -eq "up"} | ft –autosize
```

Sample output:

```
Name   InterfaceDescription                ifIndex   Status   MacAddress         LinkSpeed
----   --------------------                -------   ------   ----------         ---------
LAN2   Broadcom NetXtreme Gigabit          11        Up       90-B1-1C-56-21-B2  1 Gbps
       Ethernet #2
LAN1   Broadcom NetXtreme Gigabit          10        Up       90-B1-1C-56-21-B1  1 Gbps
       Ethernet
SAN1   Broadcom BCM57810 NetXtreme II      12        Up       00-10-18-ED-C1-32  10 Gbps
       1...#134
SAN2   Broadcom BCM57810 NetXtreme II      13        Up       00-10-18-ED-A0-82  10 Gbps
       1...#131
```

After identifying and disabling the unused NICs, and then renaming the NICs in the **Up** state, the configuration for the Hyper-V host in this implementation example is illustrated in Figure 7.

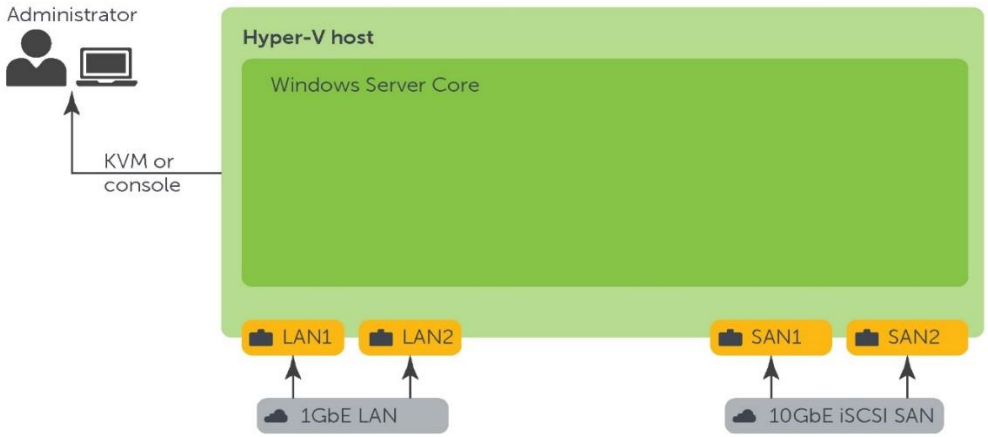


Figure 7 Configuration with renamed NICs

2.2.2 Enable iSCSI NIC parameters for Jumbo Frames and Flow Control

To help ensure iSCSI performance to the Hyper-V host, enable Jumbo Frames **(**9000 byte frames) and Flow Control for the physical NICs attached to the iSCSI SAN. In this example, these are the SAN1 and SAN2 NICs .

1. Use the **get-netadapteradvancedproperty** PowerShell command to what parameters have been enabled for the SAN1 and SAN2 NIC parameters. This command returns a lot of valuable information. To fully understand the command capabilities and output filtering, run man get-netadapteradvancedproperty –full.

   The following command string is lengthy, but it provides the Jumbo Frame and Flow Control information for both the SAN1 and SAN2 NICs. It formats the output to display the Name of the NIC, the property display name, the current value for the property, and other valid values for the property.

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

   Sample output:

   ```
   Name  Displayname    Displayvalue     validdisplayvalues
   ----  -----------    ------------     ------------------
   SAN1  Flow Control   Rx & Tx Enabled  {Disabled, Tx Enabled, Rx Enabled, Rx & Tx
                                         Enabled}
   SAN2  Flow Control   Rx & Tx Enabled  {Disabled, Tx Enabled, Rx Enabled, Rx & Tx
                                         Enabled}
   SAN1  Jumbo Packet   1514             {1514, 4088, 9014}
   SAN2  Jumbo Packet   1514             {1514, 4088, 9014}
   ```

The above output shows that flow control is enabled for both the SAN1 and SAN2 NICs. However, the 1514 bytes Jumbo Packet value reveals that this feature is not enabled.

Note: The valid display values shown in the output of the get-netadapteradvancedproperty command will vary depending upon the device driver provided by the NIC manufacturer. For example, Broadcom NICs use a jumbo frame value of "9014" while Intel NICs will use a value of "9014 Bytes".

2. To enable Jumbo Packets for the SAN1 and SAN2 NICs, set the display value to 9014 bytes (9000 MTU) using the **set-netadapteradvancedproperty** PowerShell command as seen in the next example.

```
PS C:\> set-netadapteradvancedproperty -name SAN* -displayname "Jumbo Packet"
-displayvalue 9014
```

3. Once the **set-netadapteradvancedproperty** command has been run, re-examine the Jumbo Packet settings for the SAN1 and SAN2 NICs.

```
PS C:\> get-netadapteradvancedproperty -name SAN* -displayname "Jumbo Packet"
| ft -property Name, Displayname, Displayvalue, validdisplayvalues –AutoSize
```

Sample output:

```
Name Displayname  Displayvalue validdisplayvalues
---- -----------  ------------ ------------------
SAN1 Jumbo Packet 9014         {1514, 4088, 9014}
SAN2 Jumbo Packet 9014         {1514, 4088, 9014}
```

The **set-netadapteradvancedproperty** can be used to modify other NIC properties in the same manner. For full details on the **set-netadapteradvancedproperty** command use **man set-netadapteradvancedproperty –full**.

Another excellent resource is a Technet blog on examining and setting NIC properties using at http://blogs.technet.com/b/wincat/archive/2012/08/27/using-powershell-for-nic-configuration-tasks.aspx

Note: Always configure Jumbo Frames according to the NIC manufacturer instructions. Configure Jumbo Frames end-to-end on the network, with the smallest setting governing the end-to-end packet size. Most modern iSCSI storage arrays (including EqualLogic) have Jumbo Frames enabled by default. Enabling Jumbo Frames and Flow Control on the LAN network NICs / NIC teams depends on the LAN network configuration. If these features are not enabled on the LAN network, then enabling them for the LAN NICs / NIC team for the Hyper-V host will provide no benefit. Before enabling the LAN NICs / NIC team, consult with the local network administrators and see if Jumbo Frames and Flow Control have been enabled.

2.3 Optional: Create NIC team using LAN NICs as team members

Windows Server Core has built in native NIC teaming functionality which does not require third party software. NIC teaming (also known as load balancing and Failover (LBFO)) enables multiple NICs to be placed into a team interface. This provides bandwidth aggregation and traffic failover, which prevents loss of connectivity in the event of a NIC failure on the host. The primary benefits of NIC teaming are:

- Protection against NIC failures by automatically moving the traffic to remaining operational members of the team (as in a Failover)
- Increased throughput by combining the bandwidth of the team members as though they were a single larger bandwidth interface (as in bandwidth aggregation)
- Easier management because multiple NICs can be teamed together and utilized a single IP address

In Windows 2012, a NIC team can be made up of one to 32 physical NICs. These teamed NICs are known as team members. Team members can be from different manufactures and even support different network speeds (although this is not recommended). The only requirement for NIC teaming is that a team member NIC be on the Windows Server Core Hardware compatibility list (HCL), which can be found at www.windowsservercatalog.com.

Note: iSCSI network traffic should use two or more dedicated NICs, each with their own associated virtual switch. NIC teaming can be used for individual NICs processing regular network traffic but should not be used for dedicated NICs which are processing iSCSI traffic. Load Balancing and Failover (LBFO) of iSCSI traffic should be accomplished using Multipath I/O. See the forum discussion "NIC Teaming and iSCSI MPIO" on the Microsoft TechNet at http://social.technet.microsoft.com/Forums/en-US/d2fa0a9d-a3f0-4f39-8816-bebcc598417a/nic-teaming-and-iscsi-mpio?forum=windowsserverpreview

When creating a Windows NIC team, two primary configurable variables are:

- Teaming Mode
  - Switch-independent teaming: Used when members of the NIC team are connected to separate switches.
  - Switch-dependent teaming: All the members of the NIC team are connected to the same physical switch.
- Load Balancing Algorithm
  - Hyper-V Port: Traffic is balanced by VM with as each VM is assigned to a particular NIC team member. The downside is that a failure of the NIC the VM is utilizing disrupts communication between the VM and the network. This setting is typically used when the number of virtual NICs will greatly exceed the number of team members, or if there will be teaming of virtual NICs.
  - Address Hash: Traffic is balanced between physical NICs in the team. This setting is typically adequate for most environments. The components that can be specified as inputs to the hashing function include MAC addresses, IP addresses and TCP ports (or TransportPorts) for both the source and the destination.

Using Address Hashing as a load-balancing algorithm is ample for most SMB environments since the virtualized servers in this environment typically house less than ten virtual machines per server. Having fewer virtual machines in the Hyper-V server means there are fewer virtual NICs. It is important that the virtual NICs not overwhelm the physical NIC team members.

Out of the available address hashing inputs, TransportPorts create the most granular distribution of traffic streams. This results in smaller streams that can be independently moved between members. TransportPorts cannot be used for traffic that is not TCP or UDP-based or where the TCP and UDP ports are hidden from the stack (such as IPsec-protected traffic). In these cases, the hash automatically falls back to the IP address hash or, if the traffic is not IP traffic, to the MAC address hash.

Windows Server 2012 NIC Teaming (LBFO) Deployment and Management is a great resource for this topic on the Microsoft Download Center at http://www.microsoft.com/en-us/download/details.aspx?id=30160.

1. The following PowerShell command creates a NIC team using the LAN1 and LAN2 NICs as members. In this case, each NIC is connected to a different switch and the Hyper-V host houses no more than 5 virtual machines so the teaming mode will be switch independent and the load balancing algorithm used is address hashing (TransportPorts).

   ```
   PS C:\ New-NetLbfoTeam -Name LANTeam -TeamMembers LAN1,LAN2 -TeamingMode
   SwitchIndependent -LoadBalancingAlgorithm TransportPorts
   ```

   Sample output:

   ```
   Name                   : LANTeam
   Members                : {LAN1, LAN2}
   TeamNics               : LANTeam
   TeamingMode            : SwitchIndependent
   LoadBalancingAlgorithm : TransportPorts
   Status                 : Down
   ```

2. After creating the NIC team, use the **get-netadapter** command to verify that the NIC team has been created. In the output, notice that the NIC team **LANTeam** has been given its own interface index – 27, and that the link speed is showing as 2 Gbps (an aggregate of LAN1 and LAN2 link speeds).

   ```
   PS C:\ get-netadapter | where-object {$_.status -eq "up"} | ft –autosize
   ```

   Sample output:

```
Name    InterfaceDescription                        ifIndex Status MacAddress         LinkSpeed
----    --------------------                        ------- ------ ----------         ---------
LANTeam Microsoft Network Adapter Multiplexor Driver 27      Up     90-B1-1C-56-21-B2 2 Gbps
LAN2    Broadcom NetXtreme Gigabit Ethernet #2       11      Up     90-B1-1C-56-21-B2 1 Gbps
LAN1    Broadcom NetXtreme Gigabit Ethernet          10      Up     90-B1-1C-56-21-B1 1 Gbps
SAN1    Broadcom BCM57810 NetXtreme II 1...#134      12      Up     00-10-18-ED-C1-32 10 Gbps
SAN2    Broadcom BCM57810 NetXtreme II 1...#131      13      Up     00-10-18-ED-A0-82 10 Gbps
```

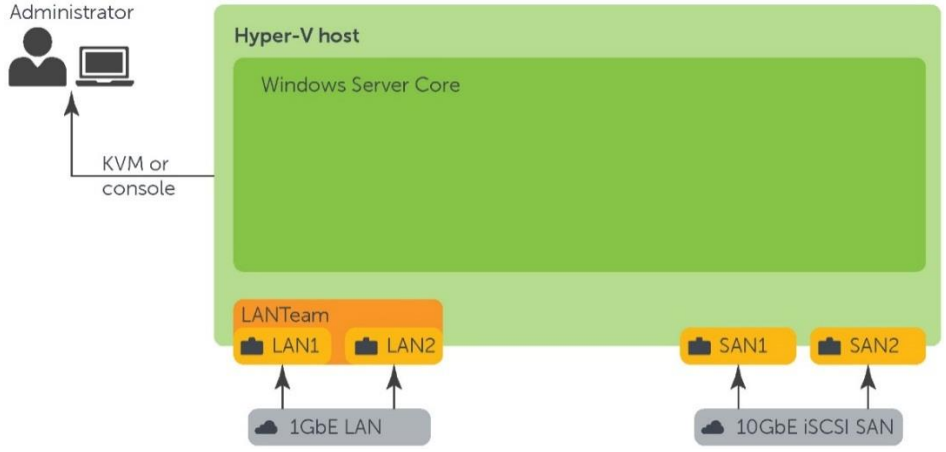


Figure 8 Updated network configuration for the Hyper-V host

2.4 Configure LAN NIC or LAN team

One of the primary benefits of NIC teaming is that it results in easier management because multiple NIC team members can utilize a single IP address. The **New-Netipaddress** command sets the IP address for any NIC or NIC team on the server by specifying the interface index. The **new-netipaddress** command can also specify the default gateway and set the subnet mask. The subnet mask is set using the **–prefixlength** flag where the prefix of the subnet mask is used.

Table 1 Common subnet masks and associated prefix lengths

Common subnet masks	Prefix length
255.255.255.252	30
255.255.255.192	26
255.255.255.0	24
255.255.252.0	22
255.255.0.0	16

The following command sets the IP address of LANTeam to 10.124.4.50, default gateway to 10.124.4.1, and subnet mask to 255.255.0.0

```
PS C:\ New-NetIPAddress -IPAddress 10.124.4.50 -InterfaceIndex 27 -
DefaultGateway 10.124.4.1 -PrefixLength 22
```

Sample output:

```
IPAddress         : 10.124.4.50
InterfaceIndex    : 27
InterfaceAlias    : LANTeam
AddressFamily     : IPv4
Type              : Unicast
PrefixLength      : 22
    ...
```

2.5 Set DNS server attributes for LAN NIC or LAN NIC team

1. After setting the IP address, subnets, and default gateway, specify the DNS server attributes using the **set-dnsclientserveraddress** command.

   ```
   PS C:\ Set-DnsClientServerAddress -InterfaceIndex 27 -ServerAddresses
   10.124.6.245, 8.8.8.8
   ```

2. After setting the DNS attributes, reassess the IP configuration for the LAN NIC or LAN NIC team.

   ```
   PS C:\ get-netipconfiguration -InterfaceIndex 27
   InterfaceAlias       : LANTeam
   InterfaceIndex       : 27
   InterfaceDescription : Microsoft Network Adapter Multiplexor Driver
   NetProfile.Name      : SKYNET.lab.local
   IPv4Address          : 10.124.4.50
   IPv4DefaultGateway   : 10.124.4.1
   DNSServer            : 10.124.6.245
                          8.8.8.8
   ```

3. To see the full network configuration details of the NIC or NIC team, use the **get-wmiobject** command and specify the **win32-networkadapterconfiguration** class and the target IP address.

   ```
   PS C:\ get-wmiobject -computer . -class "win32_networkadapterconfiguration" |
   Where-Object {$_.ipaddress -eq "10.124.4.50"} | fl *
   ```

   Sample output:

   ```
   ...
   DNSServerSearchOrder         : {10.124.6.245, 8.8.8.8}
   ...
   IPAddress                    : {10.124.4.50}
   ...
   DefaultIPGateway             : {10.124.4.1}
   ...
   IPSubnet                     : {255.255.252.0}
   ...
   ```

Note: the **get-wmiobject** command is very handy for identifying the full IP configuration for an adapter or NIC team. It is the only PowerShell command that actually returns the full subnet mask as well as all of the other relevant IP configuration information. The other command that returns the full IP configuration is the older Windows **ipconfig** command.

```
PS c:\> ipconfig
```

Sample output:

```
Windows IP Configuration
Ethernet adapter vEthernet (SAN2):
   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 10.10.6.211
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . :
Ethernet adapter vEthernet (SAN1):
   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 10.10.6.210
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . :
Ethernet adapter vEthernet (LANTeam):
   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 10.124.4.50
   Subnet Mask . . . . . . . . . . . : 255.255.252.0
   Default Gateway . . . . . . . . . : 10.124.4.1
```

Note: Install multiple network interface cards
For networking, be sure to have more than one physical NIC installed on the Hyper-V server and dedicate one NIC to administration and management. If virtual machines share a physical NIC as well, take steps so that they do not oversubscribe. Use the Reliability and Performance Monitor to establish a performance baseline for the load and then adjust NIC configurations and loads accordingly.

2.6 Configure iSCSI NIC IP addresses

Once the LAN IP configuration is set for the Hyper-V Host server, the iSCSI IP configuration can be set. This is done with the **new-netipaddress** command.

1. Reexamine the NICs to get iSCSI SAN NIC interface indexes.

```
PS C:\get-netadapter | where-object {$_.status -eq "up"} | ft -autosize
```

Sample output:

```
Name     InterfaceDescription                    ifIndex  Status  MacAddress         LinkSpeed
----     --------------------                    -------  ------  ----------         ---------
LANTeam  Microsoft Network Adapter Multiplexor   27       Up      90-B1-1C-56-21-    2 Gbps
         Driver                                                  B2
LAN2     Broadcom NetXtreme Gigabit Ethernet #2  11       Up      90-B1-1C-56-21-    1 Gbps
                                                                 B2
LAN1     Broadcom NetXtreme Gigabit Ethernet     10       Up      90-B1-1C-56-21-    1 Gbps
                                                                 B1
SAN1     Broadcom BCM57810 NetXtreme II 1...#134 12       Up      00-10-18-ED-C1-    10 Gbps
                                                                 32
SAN2     Broadcom BCM57810 NetXtreme II 1...#131 13       Up      00-10-18-ED-A0-    10 Gbps
                                                                 82
```

Assign IP addresses and the appropriate subnet mask prefixes to the appropriate iSCSI interface index.

```
PS C:\ new-netipaddress -interfaceindex 12 -ipaddress 10.10.6.210
-prefixlength 16
PS C:\ new-netipaddress -interfaceindex 13 -ipaddress 10.10.6.211
-prefixlength 16
```

Typically, the iSCSI network is a private non-routable network with no access to the outside LAN network. These commands assigned a subnet mask of 255.255.0.0 and no default gateway to the iSCSI NICs.

2. Once the IP addresses are assigned to the iSCSI NICs, reexamine the full server IP configuration using following PowerShell command.

```
PS C:\ get-netadapter | where-object {$_.status -eq "up"} | get-
netipconfiguration
```

Sample output:

```
InterfaceAlias       : SAN1
InterfaceIndex       : 12
InterfaceDescription : Broadcom BCM57810 NetXtreme II 10 GigE (NDIS VBD
Client) #133
NetProfile.Name      : Unidentified network
IPv4Address          : 10.10.6.210
IPv6DefaultGateway   :
IPv4DefaultGateway   :
DNSServer            : fec0:0:0:ffff::1
                       fec0:0:0:ffff::2
                       fec0:0:0:ffff::3

InterfaceAlias       : LANTeam
InterfaceIndex       : 27
InterfaceDescription : Microsoft Network Adapter Multiplexor Driver
NetProfile.Name      : SKYNET.lab.local
IPv6Address          : fc00:495::5d87:2eb2:1868:1260
IPv4Address          : 10.124.4.50
IPv6DefaultGateway   : fe80::21b:edff:fe10:1600
IPv4DefaultGateway   : 10.124.4.1
DNSServer            : 10.124.6.245
                       8.8.8.8

InterfaceAlias       : SAN2
```

```
InterfaceIndex       : 13
InterfaceDescription : Broadcom BCM57810 NetXtreme II 10 GigE (NDIS VBD
Client) #130
NetProfile.Name      : Unidentified network
IPv4Address          : 10.10.6.211
IPv6DefaultGateway   :
IPv4DefaultGateway   :
DNSServer            : fec0:0:0:ffff::1
                       fec0:0:0:ffff::2
                       fec0:0:0:fff
```

After completing the above steps, the network configuration for the Hyper-V host looks like the diagram in Figure 9.

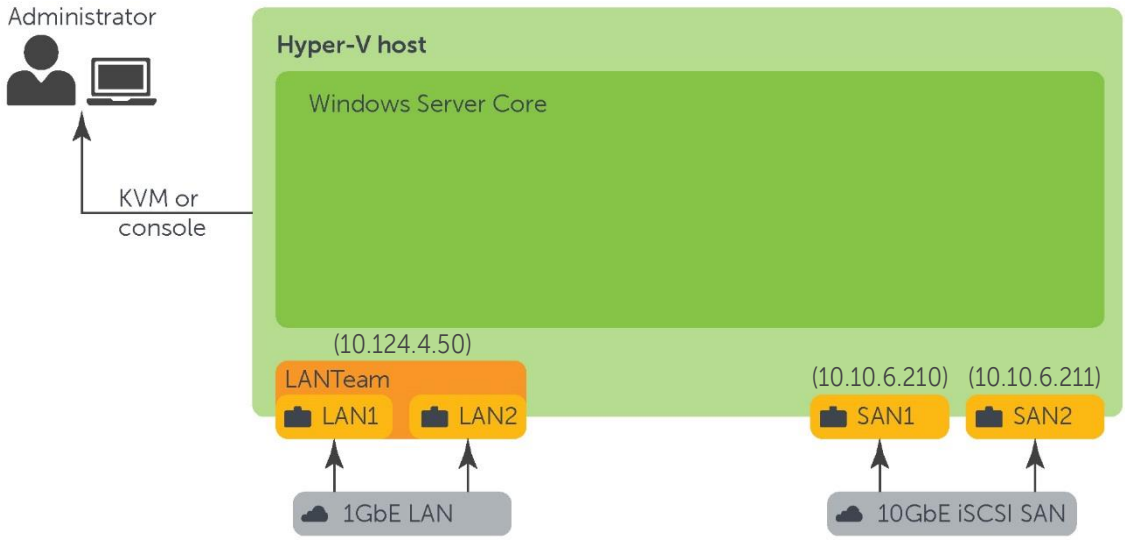


Figure 9 iSCSI NIC IP addresses configured

3 Step 3: Complete the server configuration

Server Core does not provide administrator access to the System Manager GUI for performing server configuration. Windows Server Core has a utility called **sconfig**. This utility is similar to the AIX SMIT tool that enables an administrator to configure the server from a relatively simple menu driven utility. The **sconfig** utility:

- Adds servers to the appropriate domain
- Renames servers
- Enables Remote Desktop
- Sets the data and time
- Configures automatic Windows updates
- Configures or Modifies physical NIC network settings

Note: The **sconfig** utility cannot be used to configure the network settings for NIC teams.

Type **sconfig** at the command prompt to launch the utility, and then select the **Domain/Workgroup** option.

```
PS C:\ >sconfig
```

Sample output:

```
Inspecting system...
================================================================================
                         Server Configuration
================================================================================
1) Domain/Workgroup:                    Workgroup:  WORKGROUP
2) Computer Name:                       WIN-RG0E9PS3UK9
3) Add Local Administrator
4) Configure Remote Management          Enabled
5) Windows Update Settings:             Manual
6) Download and Install Updates
7) Remote Desktop:                      Disabled
8) Network Settings
9) Date and Time
10) Help improve the product with CEIP  Not participating
11) Windows Activation
12) Log Off User
13) Restart Server
14) Shut Down Server
15) Exit to Command Line

Enter number to select an option: 1
```

The **sconfig** utility in this example is used to rename the server, add it to the domain, and enable remote desktop. Set the server up for automatic Windows updates in the initial configuration so that the latest patches can be downloaded from Microsoft. This can also be turned off with **sconfig**. An example of a properly configured server using the sconfig utility is shown below.

```
===============================================================================
                         Server Configuration
===============================================================================
1) Domain/Workgroup:                    Domain:  SKYNET.lab.local
2) Computer Name:                       TMER4R805S30
3) Add Local Administrator
4) Configure Remote Management          Enabled
5) Windows Update Settings:             Automatic
6) Download and Install Updates
7) Remote Desktop:                      Enabled (more secure clients only)
8) Network Settings
9) Date and Time
10) Help improve the product with CEIP  Not participating
11) Windows Activation
12) Log Off User
13) Restart Server
14) Shut Down Server
15) Exit to Command Line
```

Instructions for utilizing the individual steps in the sconfig menu are available on the Microsoft TechNet site at http://technet.microsoft.com/en-us/library/jj647766.aspx.

4 Step 4: Install Hyper-V

4.1 Install Hyper-V with Management Tools and Reboot

Now that the Hyper-V host has been configured with the basic tools needed to support it, install Hyper-V. This simple process is another server role and is completed with a PowerShell command.

```
PS C:\> Install-windowsfeature Hyper-V -IncludeManagementTools –restart
```

This command installs the Hyper-V role that installs the Hyper-V hypervisor, as well as the Hyper-V management tools.

Note: In Server 2012 installs with a GUI, the management tools include the Virtual Machine Manager (VMM) as well as the Hyper-V PowerShell command set. In Server Core installations, only the PowerShell commands are installed.

The hypervisor is the core component of Hyper-V. In Windows Server Core, the hypervisor is created when the Hyper-V role is installed and enabled. Installing the Hyper-V role requires the host to reboot twice. After the Hyper-V role is installed on the host, the Windows Server Core operating system runs exclusively in the Management OS partition as shown in Figure 10.

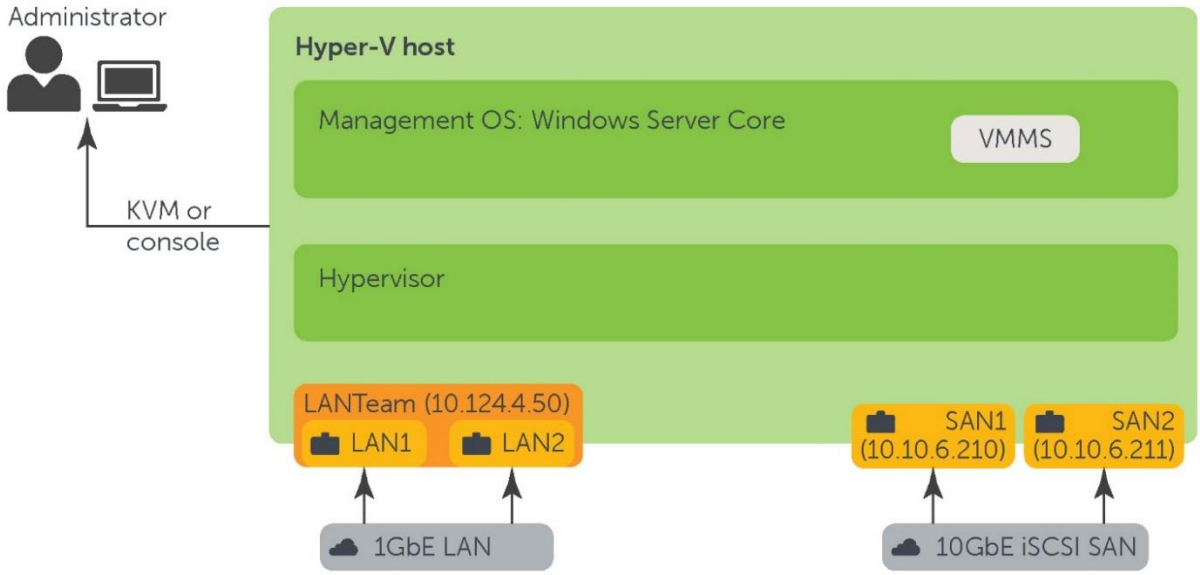


Figure 10 The Windows Server Core Management OS partition

The VMMS service that is running on the Management OS partition is the core of the Hyper-V host. It manages the state of the virtual machines running in the guest partitions (such as active, offline or stopped) and controls the tasks that can be performed on a VM based on the current state (such as the addition and removal of devices). When a VM is started, the VMMS process is responsible for creating a corresponding Virtual Machine Worker Process. This VMMS can also be identified as VMMS.exe in the **get-process** command output.

4.2 Verify the Hyper-V installation

1. To verify that Hyper-V has installed properly on the Hyper-V host, verify that the Virtual Machine Manager Service (VMMS) has started and is running by using the **get-service** command.

   ```
   PS c:\> get-service *vmms*
   ```

   Sample output:

   ```
   Status   Name               DisplayName
   ------   ----               -----------
   Running  vmms               Hyper-V Virtual Machine Management
   ```

2. In addition, verify that the VMMS service startup mode is set to **auto**.

   ```
   PS C:\> Get-WmiObject win32_service | ? {$_.name -like "*vmms*"} |ft –AutoSize
   ```

   Sample output:

   ```
   ExitCode Name ProcessId StartMode State   Status
   -------- ---- --------- --------- -----   ------
          0 vmms      1616 Auto      Running OK
   ```

3. If the startup mode for the VMMS service is not set to **auto,** use the following command.

   ```
   PS C:\> set-service -Name vmms -StartupType Automatic
   ```

4. The Hyper-V-VMMS logs in the event viewer are a great place to start troubleshooting the Hyper-V host. To examine the VMMS event log, use the following command.

   ```
   PS C:\> get-winevent -LogName Microsoft-Windows-Hyper-V-VMMS-Admin -MaxEvents 25 | ft -AutoSize
   ```

   Sample output:

   ```
   ProviderName: Microsoft-Windows-Hyper-V-VMMS

   TimeCreated           Id      LevelDisplayName   Message
   -----------           --      ----------------   -------
   11/13/2013 12:25:31   19500   Information        The Integration Services Setup Disk image
   PM                                               was …
   11/13/2013 12:25:31   19020   Information        The WMI provider 'VmmsWmiEventProvider' has
   PM                                               started.
   11/13/2013 12:25:31   19020   Information        The WMI provider
   PM                                               'VmmsWmiInstanceAndMethodProvider'
   11/13/2013 12:25:30   19020   Information        The WMI provider 'VmmsWmiEventProvider' has
   PM                                               started…
   11/13/2013 12:25:30   19020   Information        The WMI provider
   PM                                               'VmmsWmiInstanceAndMethodProvider'…
   ```

```
11/13/2013 12:25:30 PM   14052   Information   The Virtual Machine Management service successfully…
11/13/2013 12:25:30 PM   14052   Information   The Virtual Machine Management service successfully…
11/13/2013 12:25:30 PM   14052   Information   The Virtual Machine Management service successfully
11/13/2013 12:25:30 PM   14094   Information   Virtual Machine Management service is started…
```

5. To see all of the event logs associated with Hyper-V on the Hyper-V host, run the following command.

```
PS C:\> get-winevent -listlog *hyper*
```

Sample output:

```
LogMode   MaximumSizeInBytes  RecordCount  LogName
-------   ------------------  -----------  -------
Circular  1052672             66           Microsoft-Windows-Hyper-V-Config-Admin
Circular  1052672             0            Microsoft-Windows-Hyper-V-Config-
                                           Operational
Circular  1052672             0            Microsoft-Windows-Hyper-V-Hypervisor-
                                           Admin
Circular  1052672             215          Microsoft-Windows-Hyper-V-Hypervisor-
                                           Operational
Circular  1052672             624          Microsoft-Windows-Hyper-V-Integration-
                                           Admin
Circular  1052672             0            Microsoft-Windows-Hyper-V-SynthFc-Admin
Circular  1052672             867          Microsoft-Windows-Hyper-V-SynthNic-Admin
Circular  1052672             0            Microsoft-Windows-Hyper-V-SynthStor-
                                           Admin
Circular  1052672             0            Microsoft-Windows-Hyper-V-SynthStor-
                                           Operational
Circular  1052672             0            Microsoft-Windows-Hyper-V-VID-Admin
Circular  1052672             189          Microsoft-Windows-Hyper-V-VMMS-Admin
Circular  1052672             21           Microsoft-Windows-Hyper-V-VMMS-
                                           Networking
Circular  1052672             32           Microsoft-Windows-Hyper-V-VMMS-
                                           Operational
Circular  1052672             1319         Microsoft-Windows-Hyper-V-VMMS-Storage
Circular  1052672             322          Microsoft-Windows-Hyper-V-Worker-Admin
```

6. To get a listing of the features installed with the Hyper-V role and management tools, run the **get-windowsfeature** command.

```
PS C:\> get-windowsfeature | where-object installed
```

Sample output:

```
Display Name                                            Name                       Install State
------------                                            ----                       -------------
[X] File And Storage Services                           FileAndStorage-Services    Installed
    [X] Storage Services                                Storage-Services           Installed
[X] Hyper-V                                             Hyper-V                    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 Module for Windows PowerShell   Hyper-V-PowerShell         Installed
[X] User Interfaces and Infrastructure                  User-Interfaces-Infra      Installed
[X] Windows PowerShell                                  PowerShellRoot             Installed
    [X] Windows PowerShell 3.0                          PowerShell                 Installed
        [X] WoW64 Support                               WoW64-Support              Installed
```

Notes:
1: The primary reason for using Server 2012 Core on the Hyper-V host is to minimize the installation footprint and conserve system resources. **Dedicate the Hyper-V host to only run the Hyper-V role.** Do not load any other roles (with the exception of failover cluster) or any other Microsoft software applications such as SQL Server on the Hyper-V host. Doing so can negatively affect performance of the server and the hyper-v guests. Minimize the roles and shut down any unused and unnecessary services.
2: System Center management agents (but not the application components) and EqualLogic services can be installed on the Hyper-V host without negative performance impacts.

5 Step 5: Configure Hyper-V host virtual switches

Once Hyper-V is installed, the next step is to setup the virtual switches. The Hyper-V virtual switch is one of the most important parts of the Hyper-V host because the virtual machines require a virtual switch for communication. External virtual switches were installed for this example. When creating an external virtual switch, consider the following points:

- The physical NIC connected to the virtual switch must be specified. Retrieve the physical NIC name and information in PowerShell with the **get-netadapter** command.
- The Allow management OS option is set to yes by default, but it can be changed. In larger environments with many virtual machines per host, it is best to keep VM traffic separate from the management network. The **–allowmanagementOS** flag can be set to $false or 0 to keep the management and VM traffic separate
- If the virtual switch is using quality of service (QOS), the bandwidth reservation mode must be specified. This can be either weight based (where each VM gets a share of available bandwidth) or absolute (by assigning bits per second).

For more information on external virtual switches, see the *Hyper-V Architecture and Networking Considerations for Small to Medium Businesses* document in this series at http://en.community.dell.com/techcenter/extras/m/white_papers/20439155/download.aspx.

5.1 Configure Hyper-V virtual switches for LAN NICs or NIC team

The following **new-vmswitch** command creates a new virtual switch called vsLAN that links to the physical NIC team "LANTeam" (LAN network); allows for the mixing of Management OS traffic and virtual machine traffic; and sets the QOS bandwidth reservation to weight based.

```
PS C:\> New-VMSwitch -name "vsLAN" -netadaptername "LANTeam" -allowmangementOS
$true -minimumbandwidthmode weight
```

Sample output:

```
Name  SwitchType NetAdapterInterfaceDescription
----  ---------- ------------------------------
vsLAN External   Microsoft Network Adapter Multiplexor Driver
```

Note: When an external virtual switch is created and shared by the management OS, an associated vNIC is created and assumes the IP configuration of the physical NIC. This vNIC is attached to a port on the associated virtual switch. Users connected to the Hyper-V host via remote desktop at the time the external virtual switch is created are disconnected. As a best practice, configure the Hyper-V host through a KVM, or on a physical console, by creating virtual switches. After the virtual switches are successfully created, use a remote desktop connection to finish the remaining configuration tasks. It is strongly recommended to keep the KVM access, or some other access method, as a backdoor to

connect to the Hyper-V server console incase network connectivity is lost or a mistake is made during network configuration.

5.2 Configure Hyper-V virtual switches iSCSI NICs

The **new-VMSwitch** command in section 5.1 can be used to create the external virtual switches for the physical NICs associated with the iSCSI SAN network.

```
PS C:\> New-VMSwitch -name "vsSAN1" -netadaptername "SAN1" -allowmangementOS
$true -minimumbandwidthmode weight
```

Sample output:

```
Name    SwitchType NetAdapterInterfaceDescription
----    ---------- ------------------------------
vsSAN1 External   Broadcom BCM57810 NetXtreme II 10 GigE (NDIS VBD Client) #134
```

Re-use the new-VMSwitch command for any other iSCSI NICs in the Hyper-V Server

```
PS C:\> New-VMSwitch -name "vsSAN2" -netadaptername "SAN2" -allowmangementOS
$true -minimumbandwidthmode weight
```

Sample output:

```
Name    SwitchType NetAdapterInterfaceDescription
----    ---------- ------------------------------
vsSAN2 External   Broadcom BCM57810 NetXtreme II 10 GigE (NDIS VBD Client) #131
```

Note: Allowing management OS connectivity for the iSCSI virtual switches means that iSCSI storage can be provisioned to the Management OS partition on the Hyper-V host. It is very important to enable this so the Hyper-V host can store virtual machines, or other data, on iSCSI SAN volumes.

5.3 Verify successful Hyper-V virtual switch creation

1. Verify that virtual switches have been created successfully by running the **get-vmswitch** command

```
PS C:\> Get-VMSwitch
```

Sample output:

```
Name      SwitchType    NetAdapterInterfaceDescription
----      ----------    ------------------------------
vsSAN2    External      Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client) #134
vsSAN1    External      Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client) #131
vsLAN     External      Microsoft Network Adapter Multiplexor Driver
```

2. Run the **get-netadapter** command to see all of the network adapters and virtual switches for the Hyper-V host.

```
PS C:\> get-netadapter | where-object {$_.status -eq "up"} | ft -autosize
```

Sample output:

```
Name         InterfaceDescription          ifIndex   Status       MacAddress         LinkSpeed
----         --------------------          -------   ------       ----------         ---------
vEthernet    Hyper-V Virtual Ethernet      44        Up           00-10-18-ED-A0-    10 Gbps
(vsSAN2)     Adapter #4                                           82
vEthernet    Hyper-V Virtual Ethernet      51        Up           00-10-18-ED-C1-    10 Gbps
(vsSAN1)     Adapter #3                                           32
vEthernet    Hyper-V Virtual Ethernet      54        Up           90-B1-1C-56-21-    10 Gbps
(vsLAN)      Adapter #2                                           B1
LANTeam      Microsoft Network Adapter     27        Up           90-B1-1C-56-21-    2 Gbps
             Multi...                                             B2
LAN2         Broadcom NetXtreme Gigabit    11        Up           90-B1-1C-56-21-    1 Gbps
             Ethernet #2                                          B2
LAN1         Broadcom NetXtreme Gigabit    10        Up           90-B1-1C-56-21-    1 Gbps
             Ethernet                                             B1
SAN1         Broadcom BCM57810 NetXtreme   12        Up           00-10-18-ED-C1-    10 Gbps
             II 1...#134                                          32
SAN2         Broadcom BCM57810 NetXtreme   13        Up           00-10-18-ED-A0-    10 Gbps
             II 1...#131                                          82
```

3. Examine the IP configuration for the vNIC adapters using the **get-netipconfiguration** command. Notice that the vNICs have assumed the configuration of the associated physical NICs.

```
PS C:\> Get-NetIPConfiguration
```

Sample output:

```
InterfaceAlias       : vEthernet (vsSAN2)
InterfaceIndex       : 44
InterfaceDescription : Hyper-V Virtual Ethernet Adapter #4
NetProfile.Name      : Unidentified network
IPv4Address          : 10.10.6.211
IPv4DefaultGateway   :
DNSServer            : fec0:0:0:ffff::1
                       fec0:0:0:ffff::2
                       fec0:0:0:fff

InterfaceAlias       : vEthernet (vsSAN1)
InterfaceIndex       : 51
InterfaceDescription : Hyper-V Virtual Ethernet Adapter #3
NetProfile.Name      : Unidentified network
IPv4Address          : 10.10.6.210
IPv4DefaultGateway   :
DNSServer            : fec0:0:0:ffff::1
                       fec0:0:0:ffff::2
                       fec0:0:0:ffff::3

InterfaceAlias       : vEthernet (vsLAN)
InterfaceIndex       : 54
InterfaceDescription : Hyper-V Virtual Ethernet Adapter #2
NetProfile.Name      : SKYNET.lab.local
IPv6Address          : fc00:495::5d87:2eb2:1868:1260
IPv4Address          : 10.124.4.50
IPv6DefaultGateway   : fe80::21b:edff:fe10:1600
IPv4DefaultGateway   : 10.124.4.1
DNSServer            : 10.124.6.245
                       8.8.8.8
```

5.4 Enable jumbo frames on the management OS iSCSI vNICs

Enable jumbo frames across the network including both physical and virtual components as well as management OS vNICs and future vNICs on the virtual machines that are attached to the iSCSI virtual switches.. The procedure to do this on the iSCSI Management OS vNICs is the same as the one shown in section 2.2 for the physical iSCSI NIC. First, verity that jumbo frames is enabled for the Management OS iSCSI vNICs using the **get-netadapteradvancedproperty** command.

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

Sample output:

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

The above output shows that jumbo frames are not enabled by default on the Management OS iSCSI vNICs, even though it has been enabled on iSCSI physical NICs. Enabling jumbo frames on the iSCI vNICs is performed the same way as shown in section 2.2.2 with the **set-netadapteradvancedproperty** command

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

One difference to note in setting jumbo frames on the Management OS iSCSI vNICs is that the jumbo frame **displayvalue** for the vNIC is 9014 Bytes compared to 9014 for the physical NIC.

Verify that the new Jumbo Frame value has successfully been changed by running the **get-netadapteradvancedproperty** command again.

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

Sample output:

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

Once the Management OS iSCSI vNICs have been set up to use jumbo frames, test the Hyper-V host and verify that it can transmit a jumbo packet without fragmentation via the iSCSI network. To do this, ping an iSCSI target destination with a jumbo frame using the following command.

```
ping -f –l <size of packet> <iSCSI destination>
```

where the –f flag will prevent fragmentation and the –l flag specifies the packet size.

Note: When using the 9000 Byte test packet with the above command (jumbo frame size), the packet is fragmented.

Example of a fragmented packet:

```
PS C:\> ping -f -l 9000 10.10.6.200

Pinging 10.10.6.200 with 9000 bytes of data:
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
```

The fragmentation is caused by the Windows ping command that does not take into account the 28 Bytes of required overhead transmission. Verify that the jumbo frames can be transmitted using a ping on Windows. The 28 Bytes of overhead must be subtracted from the 9000 MTU value, therefore the size of the test packet needs to be 8972 Bytes.

Size of packet = 9000 Byte MTU – 28 Bytes overhead = 8972

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

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
```

At this point, the network connectivity to the Management OS partition on the Hyper-V host is complete. The virtual switches have been created and the vNICs have successfully assumed the IP configuration of the physical NICSs. Figure 11 illustrates the Hyper-V host virtual switch and Management OS network configuration used in this example.

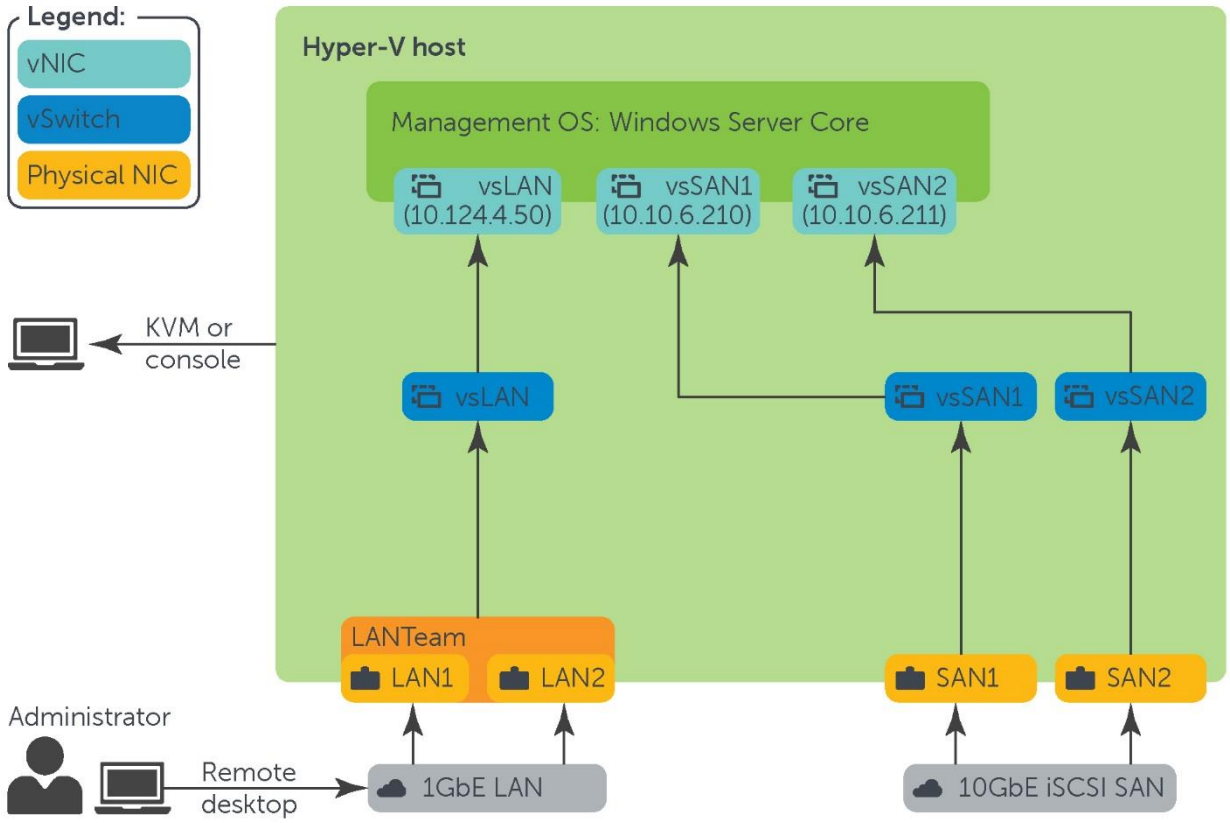


Figure 11 Hyper-V host virtual switch and Management OS network configuration

6 Step 6: Setup remote access to the Hyper-V server

Once the network connectivity to the management OS partition on the Hyper-V host is complete, it can be accessed by remote clients through remote desktop or remote PowerShell sessions. Section 6 demonstrates accessing the Hyper-V host through these methods.

6.1 Remote desktop

Accessing any Windows Server through remote desktop is a simple process. However, there are two requirements that need to be met in order to access the Hyper-V host through remote desktop.

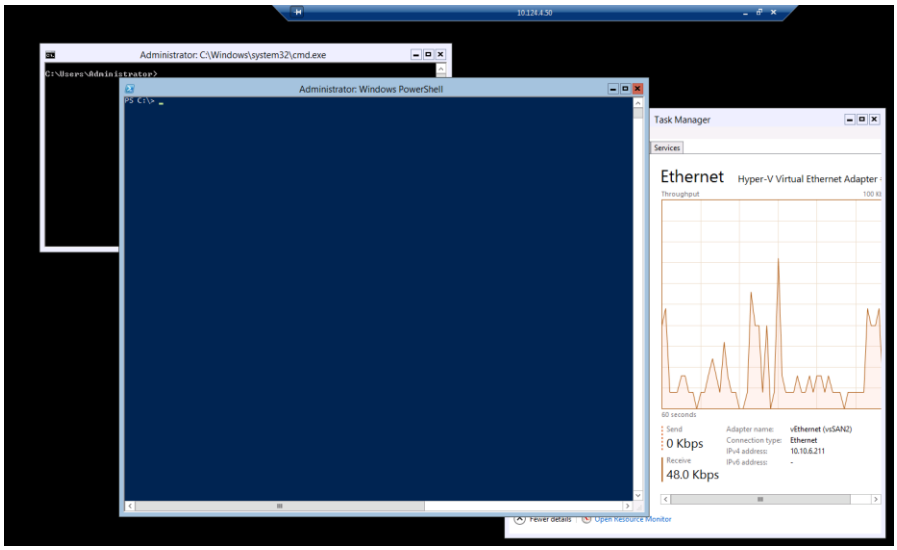
1. The virtual switches have been created and the vNICs have successfully assumed the IP configuration of the physical NICs.
2. Remote Desktop has been enabled on the Hyper-V host (see section 3).

To access the Hyper-V server through a remote desktop session on a remote client, launch the remote desktop utility or use the **mstsc** command at the command prompt on the client PC.

```
C:\ > mstsc /v 10.124.4.50 /f
```

This **mstsc** command uses a **/v** flag with the IP address of the vsLAN vNIC on the Hyper-V host and the **/f** flag which opens the session in full screen mode. The default user will be the Hyper-V host Administrator. Once entered at the command line, the above **mstsc** command opens a credential window where the administrator password is entered.

Upon selecting **OK**, the remote desktop session to the Server Core Hyper-V host opens in full screen mode.



More information on accessing Windows Servers through a remote desktop is on the Microsoft website at http://windows.microsoft.com/en-us/windows/command-line-parameters-remote-desktop-connection#1TC=windows-7

As the Hyper-V environment grows with the addition of virtual machines and other Hyper-V servers, an administrator often has multiple remote desktop windows open resulting in clutter on the client desktop. A great tool to manage multiple Windows Server remote desktop connections is the free Microsoft **RDCman** utility that can be found at http://www.microsoft.com/en-us/download/details.aspx?id=21101. For Windows 8 clients, Microsoft recommends using RSAT tools such as Server Manager for managing multiple remote servers.

6.2 Remote PowerShell (PSRemoting)

Once network connectivity is established, much of the configuration and management of the Hyper-V host can be performed using remote PowerShell sessions (PSRemoting). When Windows Server Core is installed, the Windows Remote Management service (WinRM) enables PSRemoting by default. To verify that the startup mode is set to **auto**, run the following command on the Hyper-V host.

```
PS C:\> Get-WmiObject win32_service | ? {$_.name -like "*WinRM*"} |ft -AutoSize
```

Sample output:

```
ExitCode Name   ProcessId StartMode State   Status
-------- ----   --------- --------- -----   ------
       0 WinRM      1144 Auto      Running OK
```

If the WinRM service is not running, check the WinRM event log to see if there is an error or condition by using the **get-winevent** command.

```
PS C:\> get-winevent -LogName Microsoft-Windows-WinRM/Operational -MaxEvents 25
| ft -AutoSize
```

If WinRM is not running and there are no error conditions, run the following command on the Hyper-V host.

```
PS C:\> enable-PSRemoting
```

The **enable-PSRemoting** command configures the host computer to receive Windows PowerShell remote commands. This command only needs to be run once on each computer receiving commands (the Hyper-V host servers). There is no need to run it on computers that only send commands (clients). The Enable-**PSRemoting** cmdlet runs the **Set-WSManQuickConfig** cmdlet, which performs the following tasks:

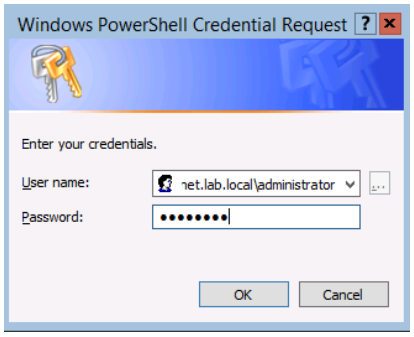
- Starts the WinRM service
- Sets the startup type on the WinRM service to Automatic
- Creates a listener to accept requests on any IP address
- Enables a firewall exception for WS-Management communications
- Enables all session configurations
- Changes the security descriptor of all session configurations to allow remote access
- Restarts the WinRM service to make the preceding changes effective

Connect as Domain Administrator

To make a remote PowerShell connection to the Hyper-V host, launch a PowerShell window on a remote client machine and run the following command as the Domain Administrator.

```
[SomeClient]: PS C:\> New-PSSession -ComputerName TMER4R805S30 -Credential
skynet.lab.local\administrator | Enter-PSSession
```

This command opens a credential window where the domain administrator password is entered.



The correct credentials open a remote PowerShell session to the Hyper-V host from the client and the session information is passed as an object into the **enter-session** command.

Note: Use a fully resolvable name for the Hyper-V server or the IP address as input for the **new-pssession** command **-computername** parameter.

Once the session is entered, the command prompt changes to display a name that designates the session is live on the Hyper-V host.

```
[Hyper-V Host Name]: PS C:\<user home directory>
```

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

To exit the remote PowerShell session, run the **exit-pssession** command (or type **exit**).

```
[TMER4R805S30]: PS C:\Users\Administrator.LAB\Documents> exit-PSsession
```

Connect as Hyper-V Host Administrator

The remote session above used the domain administrator credential, which is the highest level of network privileges. Using the domain administrator account is not always practical. However, as a trusted account it will always be able to open a remote PowerShell session on the Hyper-V server from any client in the domain. Because of the heightened security and permissions associated with the domain administrator account, most users will want to use the Hyper-V server administrator account to open a remote PowerShell session from a client. There are a few extra steps needed on the client side for this to work properly. For example, to use the Hyper-V server administrator account to create the session from the client machine with the previous method:

```
[SomeClient]: PS C:\> New-PSSession -ComputerName TMER4R805S30 -Credential
TMER4R805S30\administrator | Enter-PSSession
```

The command string will fail with the following message:

```
New-PSSession : [TMER4R805S30] Connecting to remote server TMER4R805S30 failed
with the following error message : WinRM cannot process the request. The
following error with
errorcode 0x80090311 occurred while using Kerberos authentication: There are
currently no logon servers available to service the logon request.
 Possible causes are:
...
  -Change the authentication method; add the destination computer to the WinRM
TrustedHosts configuration setting or use HTTPS transport.
 Note that computers in the TrustedHosts list might not be authenticated.
   -For more information about WinRM configuration, run the following command:
winrm help config. For more information, see the about_Remote_Troubleshooting
Help topic.
...
```

This occurs because the Hyper-V host server is not in the trusted hosts list on the client machine. In order to fix this, add the Hyper-V host server to the trusted host list using the following command:

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

Select **Yes** when prompted and then run the following **get-item** command to verify that the Hyper-V host was added to the list.

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

   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:
`[SomeClient]: PS C:\>` **`set-item wsman:localhost\client\trustedhosts 10.124.4.50`**

Once verified, re-attempt to open a remote PowerShell session to the Hyper-V host

```
[SomeClient]: 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 admin password is entered.

With the correct credentials, the command string opens the remote PowerShell session to the Hyper-V server from the client machine 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>
```

In both of the proceeding remote PowerShell connection examples, the connecting client is in the same domain as the Hyper-V server (in the case of the example, the skynet.local.lab domain). Opening a PowerShell connection when the client and server are in the same domain is straightforward. However, when the client and server are in different domains, establishing a remote PowerShell connection is more involved. Going over all of the methods for establishing a remote PowerShell connection between a client and a remote Hyper-V server is beyond the scope of this document. Only establishing a remote PowerShell connection from within the domain will be discussed. For more information on PSRemoting, consult the TechNet blog at
http://blogs.technet.com/b/heyscriptingguy/archive/2010/11/16/enable-powershell-remoting-to-enable-running-commands.aspx

For remote PowerShell troubleshooting help, run **get-help** for **about_remote_troubleshooting**:

```
PS C:\Users\Administrator> man about_remote_troubleshooting
```

7 Step 7: Connect the Hyper-V Host to an EqualLogic iSCSI SAN

iSCSI is a key technology that delivers a scalable, cost-effective; high-performance data storage solution for Windows virtualized environments. With its peer storage architecture, the Dell EqualLogic iSCSI PS Series storage array was designed with virtualization in its "DNA" which makes it an ideal storage solution for Windows virtualized environments for SMB customers. The PS Series array is comprised of redundant controllers with battery backed cache; redundant power supplies; disks (SSD,SAS,SATA compatible); multiple high performance network interfaces; and is accompanied by a full set of software tools for managing, monitoring, and recovery at no additional cost to the customer

7.1 Install the EQL Host Integration Tools for Microsoft (HIT / ME) onto the Hyper-V Host

Dell™ EqualLogic Host Integration Tools for Microsoft (HIT/ME) provides an administrator the ability to manage and configure PS Series storage arrays from the servers that use them. The HIT/ME is essentially a toolkit which allows administrators to perform a wide variety of tasks such as initializing new arrays; to creating application consistent snapshots; and to scripting management operations, The EqualLogic Host Tools are available to all EqualLogic customers at no additional cost.

The first step in installing the EqualLogic Host Integration tools on the Hyper-V host is to copy the HIT setup executable to a local drive or run the setup executable from a network share. This section will show how to install the EqualLogic Host Integration Tools from a mapped network share

1. Examine the currently mapped logical drives and network shares using the **get-psdrive** command as show below.

   ```
   PS C:\> get-psdrive -psprovider FileSystem | ft -autosize
   ```

   Sample output:

   ```
   Name Used (GB) Free (GB) Provider   Root  CurrentLocation
   ---- --------- --------- --------   ----  ---------------
   C        13.62     53.78 FileSystem C:\  Windows\system32
   D                        FileSystem D:\
   ```

2. Map the network share that houses the EqualLogic HIT setup file to the Hyper-V host using the **new-psdrive** command.

Note: Mapped network drives are specific to a user account. Mapped network drives that are created in PowerShell sessions which are started with the **Run as administrator** option (or with the credential of another user) are not visible in sessions that started without the explicit credentials of the current user.

To map a network drive using the explicit credential of the Hyper-V server administrator user, use the following command sequence. Capture the credentials of the administrator user in a variable.

```
PS C:\> $credential = Get-Credential
```

This will open a credential window where the administrator user name and password is entered. Enter the username and password and then click OK to save the credential information in the variable.

3. Mount the network share with the **new-psdrive** command calling the specific credential of the administrator user as follows:

```
PS C:\> New-PSDrive -Persist -name E -PSProvider FileSystem -Root
\\10.124.4.66\FileShare1 -credential $credential
```

Sample output:

```
Name  Used (GB)  Free (GB)       Provider         Root
----  ---------  ---------       --------         ----
E     10.64      489.24          FileSystem       \\10.124.4.66\FileShare1
```

Note: Using the **–persist** flag with the **new-psdrive** command makes the mapping persistent and not session specific. This means that other PowerShell sessions can see the mapping as well as file explorer and other tools. If the network share mapping is not persistent, when the PowerShell session that mapped the network share is closed, the network share would be unmapped.

4. Re-run the get-psdrive command to verify that the network share has been mapped and is visible.

```
PS C:\> get-psdrive -psprovider Filesystem | ft -autosize
```

Sample output:

```
Name  Used (GB)  Free (GB)       Provider         Root
----  ---------  ---------       --------         ----
C     13.62      53.78           FileSystem       C:\
D                                FileSystem       D:\
E     10.64      489.24          FileSystem       \\10.124.4.66\Fileshare1
```

7.2 Locate and run HIT setup executable and Reboot

Locate the HIT Setup64.exe file on the network share

```
PS C:\> ls -r E:\EQLSoftware\HIT_47_GA | ? {$_.name -like "Setup64.exe"}
```

Sample output:

```
    Directory: E:\EQLSoftware\HIT_47_GA

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         10/7/2013  10:12 AM  127082320 Setup64.exe
```

Run the HIT setup executable by using the ” command.

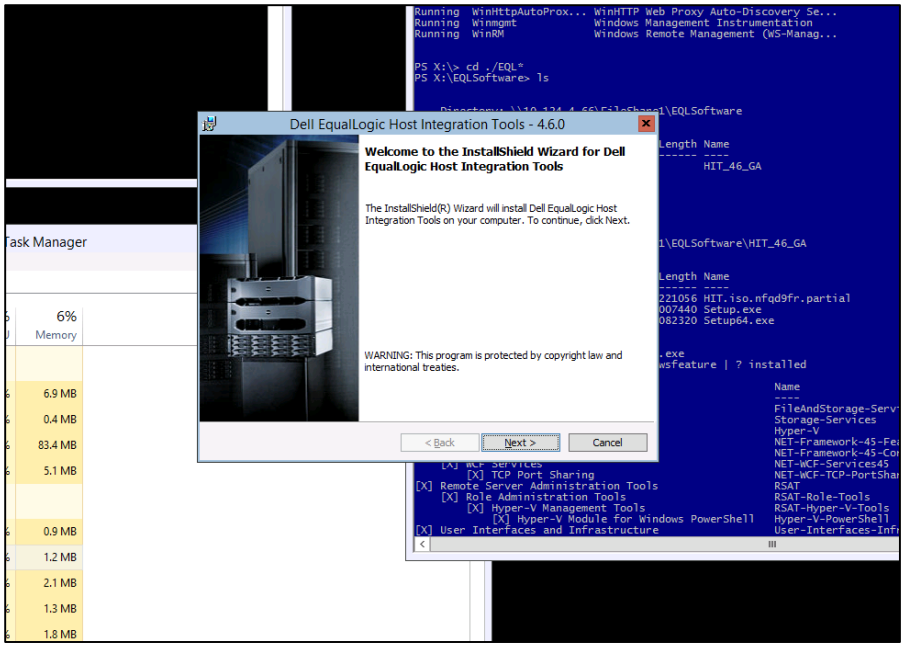
```
PS C:\> Start-process E:\EQLSoftware\HIT_47_GA\Setup64.exe
```

Alternatively, it is possible to combine the above two commands into a single (and more powerful) PowerShell command by specifying the **fullname** parameter from the **ls** command.

```
PS C:\> start-process (ls -r E:\EQLSoftware\HIT_47_GA | ? {$_.name -like
"Setup64.exe"}).fullname
```

Note: If there are multiple Setup64.exe files on the network share, the above command will return an error. As a best practice, always confirm the path to an executable prior to running it on a host.

The HIT installer utilizes a Window InstallShield that runs on native Server 2012 Core. When the setup64.exe is run on the Server Core Hyper-V host, it will open an install window on the desktop that is accessed through a remote desktop connection.



During the installation, most of the defaults can be selected. Typical steps are:

- Accept license agreement
- Select default installation folder (c:\Program Files\EqualLogic)
- Select default complete installation
- Click install to start installation

Note: For full instructions on installing the EqualLogic HIT Kit, consult the Dell EqualLogic Host Integration Tools for Microsoft, Installation and User's Guide on dell.com.

As the installation progresses, the installer will display a list of features that are being installed as well as a status for each feature.

Once the installation finishes, the wizard will prompt to launch the remote setup wizard. Leave the Launch Remote Setup Wizard checkbox empty and click **Finish**.

Because the HIT Kit installation installed the Windows features (MPIO) and adjusted service parameter configurations, the Hyper-V host needs to reboot at this point. When the wizard closes, the installer prompts for a reboot. Select **Yes** and reboot the server to complete the HIT installation process.

7.3 Post-HIT installation tasks on the Hyper-V host

After the HIT installs and the Hyper-V host has rebooted, examine and verify the configuration changes it made on the Hyper-V host.

1. Examine the installed HIT directories. The Setup64.exe installs the HIT directories at c:\Program Files\EqualLogic.

```
PS C:\Program Files\EqualLogic> ls
```

Sample output:

```
    Directory: C:\Program Files\EqualLogic

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----         11/20/2013   4:54 PM            bin
d----         11/20/2013   4:54 PM            Doc
```

As expected, the bin directory will contain all of the HIT executable files while the doc directory will contain the HIT version release notes, installation guides, and user guide documentation in pdf format.

```
PS C:\Program Files\EqualLogic\doc> ls
```

Sample output:

```
    Directory: C:\Program Files\EqualLogic\doc

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---          6/4/2013   4:05 PM    4223970 asm-user-guide.pdf
-a---          6/4/2013   4:05 PM     552920 Equallogic_PowerShellTools_Quick_Reference.pdf
-a---         7/31/2013   2:00 PM     465463 hit-release-notes.pdf
-a---          6/4/2013   4:05 PM    1856816 hit-user-guide.pdf
-a---          6/4/2013   4:05 PM    1308196 PowerShellModule_UserGuide.pdf
...
```

2. Verify that MPIO has been installed using the **get-windowsfeature** command.

```
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] Hyper-V                                             Hyper-V                    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] 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 Module for Windows              Hyper-V-PowerShell         Installed
PowerShell
[X] User Interfaces and Infrastructure                  User-Interfaces-Infra      Installed
[X] Windows PowerShell                                  PowerShellRoot             Installed
    [X] Windows PowerShell 3.0                          PowerShell                 Installed
[X] WoW64 Support                                       WoW64-Support              Installed
```

3. Examine the MSiSCSI service and make sure that it is running and that the **startmode** is set to **auto** using the following **get-wmiobject** command.

```
PS C:\> get-wmiobject -class win32_service | ? {$_.name -eq "msiscsi"}
```

Sample output:

```
ExitCode  : 0
Name      : MSiSCSI
ProcessId : 1028
StartMode : Auto
State     : Running
Status    : OK
```

4. If for some reason, the MSiSCI service is not started and/or set to automatic startup, run the following command.

```
PS C:\ > set-service -name msiscsi -status running -startuptype auto
```

5. Examine the firewall rules for the MSiSCSI service and make sure that these rules have been enabled using the following command.

```
PS C:\> get-netfirewallservicefilter -Service msiscsi | get-netfirewallrule
```

Sample output:

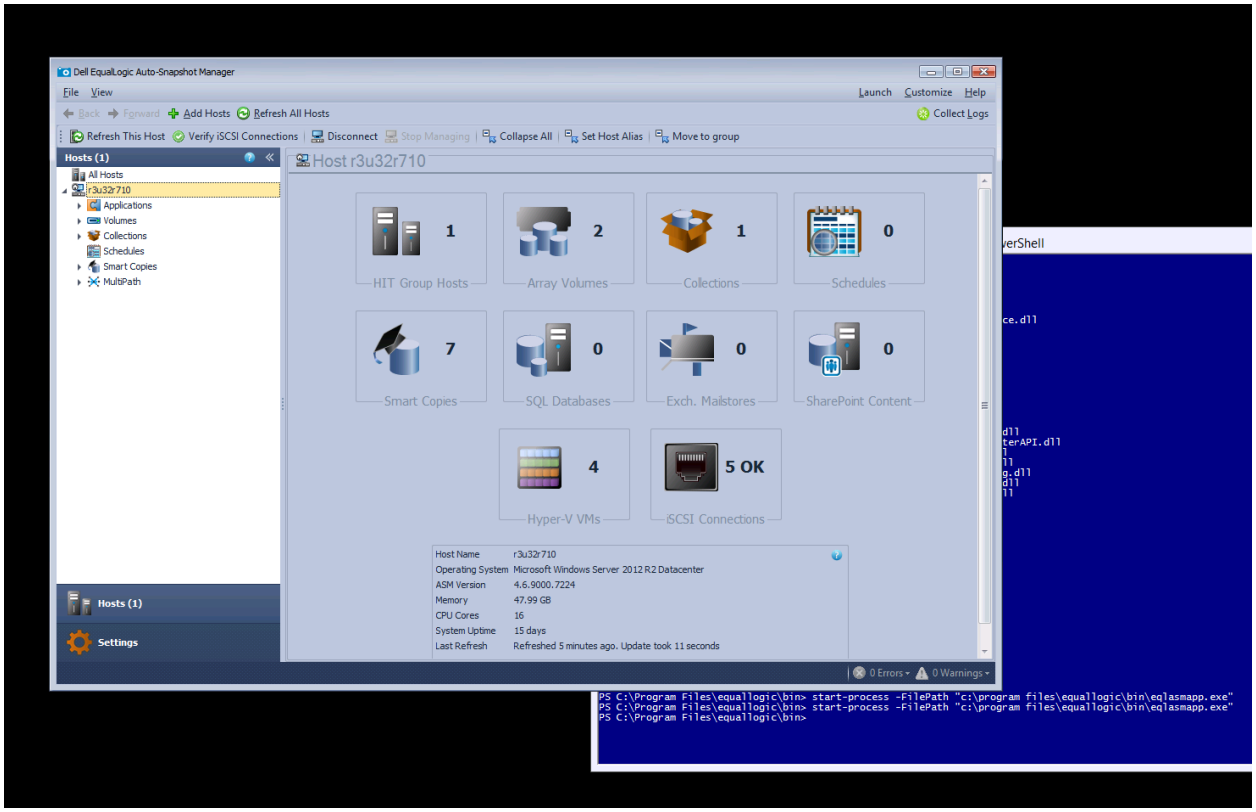
```
Name                  : MsiScsi-Out-TCP
DisplayName           : iSCSI Service (TCP-Out)
Description           : Outbound rule for the iSCSI Service to allow
communications with an iSCSI server or device. [TCP]
...
Enabled               : True
...
Direction             : Outbound
...

Name                  : MsiScsi-In-TCP
DisplayName           : iSCSI Service (TCP-In)
Description           : Inbound rule for the iSCSI Service to allow
communications with an iSCSI server or device. [TCP]
...
Enabled               : True
...
Direction             : Inbound
```

6. If for some reason, the MSiSCI rules have not been enabled, enable them with the following commands.

```
PS C:\> set-NetFirewallRule -Name MsiScsi-in-TCP -Enabled True
PS C:\> set-NetFirewallRule -Name MsiScsi-out-TCP -Enabled True
```

7. Verify that the Auto-Snapshot Manager application can start on the Server Core host by typing:

```
PS C:\> start-process -FilePath "c:\program files\equallogic\bin\eqlasmapp.exe"
```



8. Finally, examine the EqualLogic HIT services and verify that they are running using the **get-service** command.

```
PS C:\> get-service | ? {$_.displayname -like "*equal*"}
```

Sample output:

```
Status   Name               DisplayName
------   ----               -----------
Running  EHCMService        EqualLogic Host Connection Manageme...
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
```

Note: The EqlVdsHwPrv service is typically not started after the HIT installation.

7.4 Examining the HIT log files (optional)

The EqualLogic HIT services write to the Windows Application event log. When diagnosing or troubleshooting it is always a good to idea to examine the application event with the **get-winevent** command, filtering for EqualLogic as a provider name.

```
PS C:\ > get-winevent -LogName application | ? {$_.providername -like "*equal*"}
```

Sample output:

```
   ProviderName: EqualLogic

TimeCreated              Id LevelDisplayName Message
-----------              -- ---------------- -------
11/20/2013 4:59:34 PM  1000 Information      The service EqlVss has been …
11/20/2013 4:59:27 PM  1000 Information      The service EHCMService has been …
11/20/2013 4:57:23 PM  1000 Information      The service EqlLogd has been …
11/20/2013 4:54:12 PM  1000 Information      The service EqlVss has been …
11/20/2013 4:54:08 PM  1000 Information      The service EqlLogd has been …
...
```

Another critical log is written by the EqualLogic Trace Logging Service, EqlLogd. This service is used by the EqlASMAgent to write to a log called **eqltrace.log** which is found in the hidden ProgramData folder under the C:\ parent directory.

Note: When using the standard **ls** command on the c:\ parent directory, none of the hidden files and folders are displayed.

```
PS C:\> ls
```

Sample output:

```
    Directory: C:\

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----         12/19/2012   3:44 PM            Dell
d----          7/26/2012   3:44 AM            PerfLogs
d-r--          10/9/2013  10:31 AM            Program Files
d----          10/9/2013  10:31 AM            Program Files (x86)
d-r--          10/4/2013   4:37 PM            Users
d----          10/2/2013   4:01 PM            Windows
d----          10/2/2013   2:37 PM            Windows.old
```

In PowerShell, hidden folders and files can be exposed using the **–force** flag with the **ls** command.

```
PS C:\> ls –force
```

Sample output:

```
    Directory: C:\

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d--hs          7/26/2012   4:24 AM            $Recycle.Bin
d----         12/19/2012   3:44 PM            Dell
d--hs          7/26/2012   3:14 AM            Documents and Settings
d----          7/26/2012   3:44 AM            PerfLogs
d-r--          10/9/2013  10:31 AM            Program Files
d----          10/9/2013  10:31 AM            Program Files (x86)
d--h-         11/20/2013   4:57 PM            ProgramData
d--hs          10/2/2013   2:28 PM            Recovery
d--hs          11/5/2013   2:45 PM            System Volume Information
d-r--          10/4/2013   4:37 PM            Users
d----          10/2/2013   4:01 PM            Windows
d----          10/2/2013   2:37 PM            Windows.old
-arhs          7/25/2012  11:44 PM     398156 bootmgr
-a-hs           6/2/2012  10:30 AM          1 BOOTNXT
-a-hs         11/20/2013   4:57 PM 4563402752 pagefile.sys
```

The **eqltrace.log** file is found in the c:\programdata\equallogic\log directory.

```
PS C:\> cd ProgramData\Equallogic\log
PS C:\ProgramData\Equallogic\log> ls
```

Sample output:

```
    Directory: C:\ProgramData\Equallogic\log

Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         11/20/2013   4:42 AM   52428933 eqltrace.0.log
-a---         11/20/2013   5:40 PM   18274611 eqltrace.log
```

The eqltrace.log file can get very large. The output that is returned from a **more** command could take several minutes to display. A more efficient way of examining this file is to use the **get-content** command and pipe the output to **select-object**. The following command line gets the latest 25 entries written to the log.

```
PS C:\ProgramData\Equallogic\log> get-content -path ./eqltrace.log | select-
object -last 25
```

Sample output:

```
...
20-Nov-13
17:51:14.323|EqlASMAgent|2336|12||INFO|||EqlSimpleScheduler.GetEqlTaskStatus >
starting
20-Nov-13
17:51:14.323|EqlASMAgent|2336|12||INFO|||EqlSimpleScheduler.GetEqlTaskStatus >
ending
```

When calling Dell Customer Support, chances are the support engineer will want to see the **application** log and the **eqltrace.log** files attached to any opened support case. Make a note of these logs and their locations for future reference.

7.5 Register and connect an EqualLogic PS Series array to the Hyper-V host

Installing the EqualLogic HIT loaded both the EqualLogic Storage Management Provider (SMP) as well as the EqualLogic PowerShell cmdlets on the Hyper-V host. The HIT PowerShell cmdlets and SMP complement provide industry leading storage integration with Windows Server Core. The SMP component enables an administrator to use the Dell EqualLogic PowerShell cmdlets to manage Dell EqualLogic storage directly. If the GUI was installed on the Hyper-V host, the SMP Component would allow an administrator to use Server Manager to manage the PS Series array.

Note: For this example, it is assumed that an EqualLogic PS Series array has already been installed and configured with a Group Name and Group IP address associated to the iSCSI SAN network. A management IP address for the PS Series array has also been configured and is associated to the corporate management LAN network. With PS4110, PS6110, and later arrays, it is an EqualLogic best practice to enable and configure the management IP address for the PS Series array so array management and iSCSI traffic utilize separate networks to minimize any possibility of contention. This paper will not go into detail on set up and initial configuration of a PS Series array. For information on this go to Dell EqualLogic Customer Support and consult the *PS Series Array Setup and Installation Guide*.

For this example, the Hyper-V host connection and PS Series Array network connections are illustrated in Figure 12.

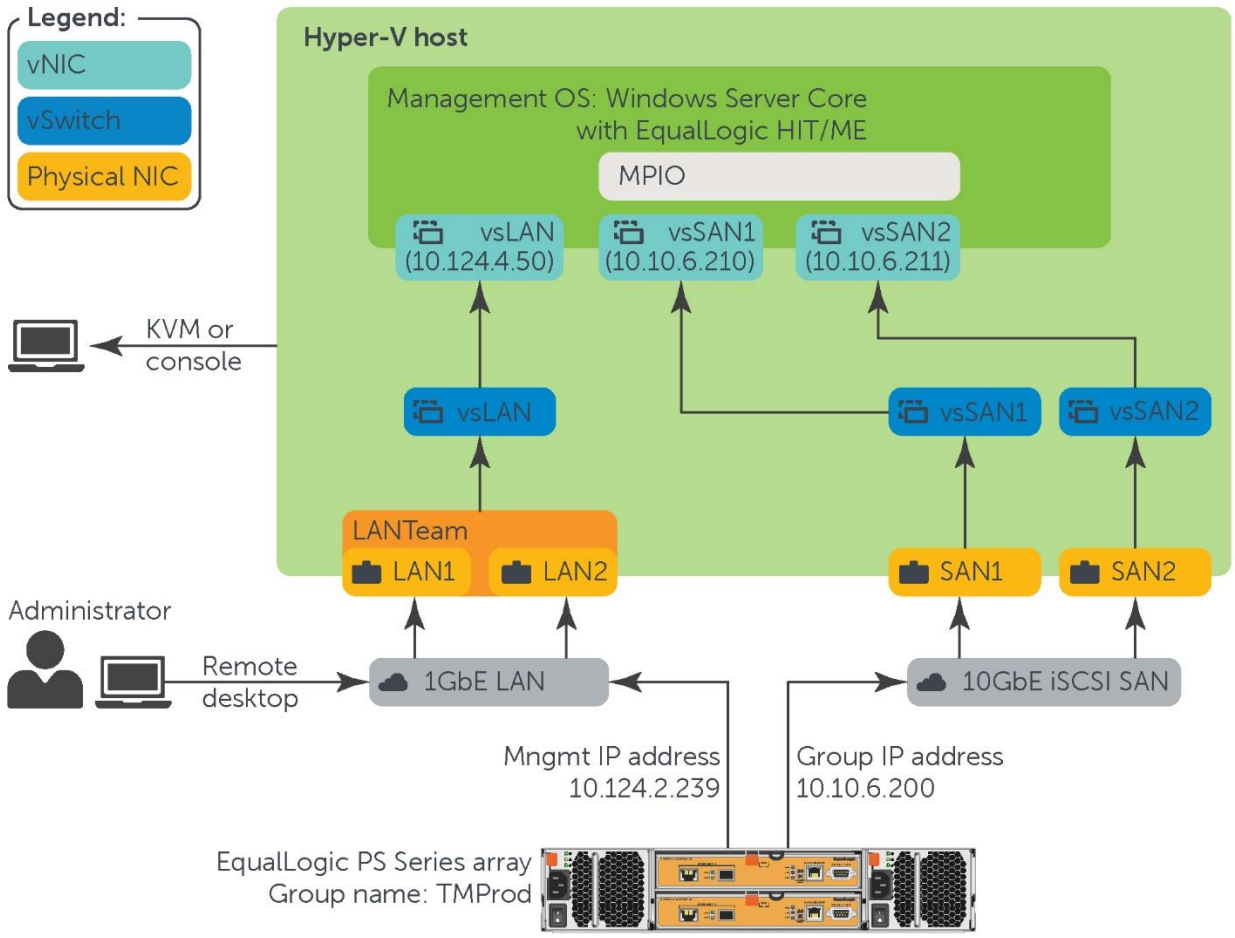


Figure 12 Hyper-V host and PS Series array network connections

Figure 12 shows that the Group IP address of the EqualLogic PS Series array has been configured to 10.10.6.200 and is associated to the iSCSI SAN Network. In addition, the Management IP address for the array has been configured with an IP address of 10.124.2.239 and is associated to the corporate management LAN network. The group name associated with the PS Series array is TMProd.

1. The first step after the HIT installation is complete is to use the EqualLogic PowerShell cmdlet tools to register and make a management session connection to an EqualLogic PS Series array. These tools are found in .dll in the EqualLogic\Bin directory. To make these cmdlets tools accessible to the current PowerShell session, use the **import-module** command

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

Note: For detailed information about working with the EqualLogic PowerShell tools in PowerShell Sessions consult: Dell Technical Reports TR1089 V1.0 (2013): Windows Command-line Automation Techniques for Dell EqualLogic PS Series Arrays,

2. Once the module is imported into the PowerShell runspace, register the PS group name by using the **new-eqlgroupaccess** command.

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

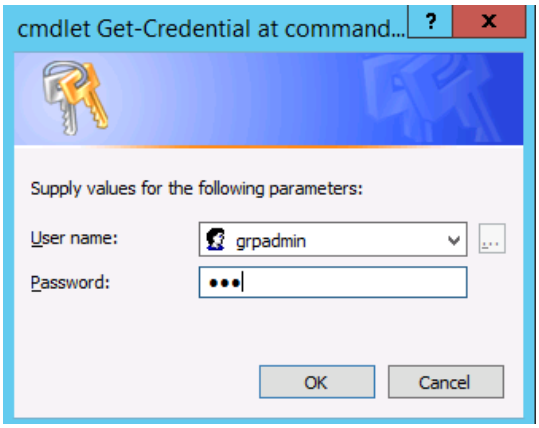
The **new-eqlgroupaccess** command registers a PS Series group on the Hyper-V host so that it is automatically connected whenever the EQL PowerShell tools are called. This command only needs to be run once per group as this information is stored locally on the Hyper-V host. The primary parameters the command takes are the group name (**-groupname**), which in the example is TMProd and the group IP address (**-groupwkaddress**), which in the example is 10.10.6.200.

Note: The "wk" in the –groupwkaddress parameter stands for "well known". Always use the EqualLogic group IP address for this parameter.

3. Once the group is registered, a management session can be opened in the current PowerShell runspace by using the **connect-eqlgroup** command.

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

The above command will establish a connection to the registered Dell EqualLogic PS Series group at the given ip address with the credentials returned from the Get-Credential popup. In the popup window enter in the PS Group administrator username and password.



Since the management IP address for the group is active in the example, the Management IP address must be specified with the **–groupaddress** parameter. Using the Group IP address with this parameter will not work when the array is configured to use the Management IP address for management communications.

The other parameter used by the above **connect-eqlgroup** command is **-ignoreSavedCredentials**. This parameter allows for the entered credentials to override of the stored credential information of the registered PS Group. Use this parameter when the after registering the PS Group. If the connect-eqlgroup is run prior to registering the EQL Group, then this parameter can be omitted.

4. After the EqualLogic group has been registered and connected, there are several group administration commands that provide more information about the group and the management session connection. For example, run the **get-eqlgroupaccess** command to get information about the management session connection.

   ```
   PS C:\ > get-eqlgroupaccess
   ```

   Sample output:

   ```
   SessionId           : 6019CBF194DA71B4F7221504000020C3
   GroupAddress        : 10.10.6.200
   GroupWKAddress      : 10.10.6.200
   GroupMKAddress      :
   GroupName           : TMProd
   UseSSO              : no
   Username            :
   VSSUserName         :
   SnapshotUserName    :
   UseCHAPForDiscovery : no
   UseHostBusAdapters  : no
   ```

5. Another useful command is **get-eqlgroupconfiguration**. This command retrieves the current configuration settings for a PS Series group where a connection exists in the current PowerShell runspace.

   ```
   PS C:\> get-eqlgroupconfiguration
   ```

   Sample output:

   ```
   SessionId                          : 6019CBF194DA71B4F7221504000020C3
   groupId                            : 6019CBF194DA71B4F7221504000020C3
   GroupName                          : TMProd
   GroupAddress                       : 10.10.6.200
   GroupAddressIPv4                   : 10.10.6.200
   GroupAddressIPv6                   : 0000:0000:0000:0000:0000:0000:0000:0000
   ClientAccessSSH                    : enabled
   ClientAccessTelnet                 : enabled
   WebAccess                          : enabled
   WebAccessNoEncrypt                 : enabled
   ConnectionBalancing                : enabled
   PerformanceBalancing               : enabled
   GroupDate                          : 11/22/13
   GroupTime                          : 17:14:04
   ...
   GroupTimeZone                      : america_New_York
   OptimizedMPIOSessions              : Enabled
   UnlimitedMPIOSessions              : Disabled
   MinimalMPIOSessions                : Disabled
   ```

6. Once the PS group has been registered and connected to the Hyper-V host, the PS group address is displayed in the native **get-iscsitargetportal** command output.

```
PS C:\> get-iscsitargetportal

InitiatorInstanceName  :
InitiatorPortalAddress :
IsDataDigest           : False
IsHeaderDigest         : False
TargetPortalAddress    : 10.10.6.200
TargetPortalPortNumber : 3260
PSComputerName         :
```

In summary, the HIT PowerShell tools offer 78 array management commands for an administrator to choose from when working with EqualLogic PS Storage arrays. For a full listing of these commands, use **get-command** and specify **EQLPSTools** with the module parameter.

```
PS C:\ > get-command -module EQLPStools
```

Complete information is provided about all of these commands in the *Dell EqualLogic PowerShell Tools User Guide* found in the **c:\program files\equallogic\doc** directory on the Hyper-V host or in the resources section of the Dell EqualLogic Customer Support site. Also, consult Dell Technical Reports: Windows Command-line Automation Techniques for Dell EqualLogic PS Series Arrays, for more information on the usage of the EqualLogic HIT PowerShell tools and commands

7.6 Provision an EqualLogic iSCSI Volume to the Hyper-V host

Once the EqualLogic array is registered and connected, an administrator can use the EqualLogic PowerShell tools to create a volume and provision it to the Hyper-V host. In this example, the iSCSI volume will be used as a data store to house virtual machine, virtual hard disks and configuration files.

Use the following steps to provision an EqualLogic volume to the Hyper-V host.

- Identify key parameter values and store into variables.
- Create an EqualLogic volume using HIT/ME powershell cmds.
- Refresh the iSCSI targets on the Hyper-V host.
- Connect and register the iSCSI target and iSCSI session.
- Rescan the Hyper-V host storage.
- Get Physical Disk Number for New EqualLogic iSCSI Target.
- Initialize, create a partition, assign a drive letter, and format new EqualLogic volume on Hyper-V host.

7.6.1 Identify key parameter values and store into variables

7. Identify the attached PS Group(s) name and IP Address. To identify the attached PS Group for the current PowerShell runspace run the "get-eqlgroupconfiguration" command and filter the output to display the group name and group IP address

```
PS C:\> get-eqlgroupconfiguration | ft -property GroupName, GroupAddress -
autosize
```

Sample output:

```
GroupName GroupAddress
--------- ------------
TMProd    10.10.6.200
```

As shown earlier in the example, the Hyper-V host is connected to a single PS Group called "TMProd".

Note: The **get-eqlgroupconfiguraion** and **get-eqlgroup** commands return similar output. The difference is that the **get-eqlgroupconfiguration** command will only show the current configuration settings for the PS Group where a connection exists in the current PowerShell runspace. The **get-psgroup** command returns configuration information for all PS Groups connected to the Hyper-V host.

PowerShell Scripting Note: Using the **–property** flag with the format-table cmdlet (ft) allows customization of the object properties that appear in the output. Some commands return a more verbose output than others. To consistently see a full list of an object properties, run the desired command and pipe to format-list (fl) with an asteric (*) as a wild card.

```
PS C:\>(PowerShell command)| fl *
```

8. Once the group has been identified, get the associated storage pools with the get-eqlpool command. Filter the output to get the storage pool name, total space in MB, used Space in MB, free space in MB as shown below:

```
PS C:\> get-eqlpool | ft -property StoragePoolName, TotalSpaceMB,
UsedSpaceMB, FreeSpaceMB -autosize

StoragePoolName TotalSpaceMB UsedSpaceMB FreeSpaceMB
--------------- ------------ ----------- -----------
Gold                15614805      285570    15120270
```

From the above output, it can be seen that there is a single pool named "Gold" associated with PS Group TMProd which has over 15TB of free space.The Hyper-V host iSCSI Qualified Name for the Hyper-V host.

9. Get the IQN of the Hyper-V host. The IQN is used to set the access controls for the new iSCSI volume to the Hyper-V host. The IQN of the Hyper-V host is acquired by using the following PowerShell command.

```
PS C:\> (get-initiatorport | where {$_.portaddress -like
'*ISCSI*'}).nodeaddress
```

Sample output:

```
iqn.1991-05.com.microsoft:tmer4r805s30.skynet.lab.local
```

10. Place some of the above information as well as the name of the new volume into some variables thatcan then be reused in the PowerShell runspace.

```
PS C:\> $iqn = (get-initiatorport | where {$_.portaddress -like
"*ISCSI*"}).nodeaddress

PS C:\> $groupname = (get-eqlgroupconfiguration).groupname

PS C:\> $poolname =(get-eqlpool -groupname $groupname).storagepoolname

PS C:\> $groupaddress = (get-eqlgroupconfiguration).groupaddress

PS C:\> $volumename = "TMEVMStorage"
```

7.6.2 Create an EqualLogic volume using HIT PowerShell commands

The above variables store information and are passed directly into the EqualLogic PowerShell commands to create and then assign access to the new volume. This action is done with a single command line using both the **new-eqlvolume** command and **new-eqlvolumeacl** command together.

```
PS C:\>New-EqlVolume -VolumeName $volumename -StoragePoolName $poolname -
VolumeSizeMB 512000 -ThinProvision Yes -loadbalance true -AccessType read_write
-PassThru | New-EqlVolumeAcl -InitiatorName $iqn -AclTargetType
volume_and_snapshot
```

The above command string is a powerful "one liner command" that consists of two parts. The first part of the command string (**new-eqlvolume**) creates the new 500 GB volume named TMEVMStorage on the EqualLogic PS Group using storage from the Gold pool. The volume is set up to use thin provisioning, will utilize load balancing; and is set to read-write access. By using the **–passthru** parameter, the **new-eqlvolume** command returns a volume object that is passed into the second part of the command (**new-**

eqlvolumeaql). The **new-eqlvolumeaql** command creates a new access control list, grants access to the Hyper-V host for the new volume (TMEVMStorage) and its snapshots.

After the volume is created, confirm that it was created successfully on the EqualLogic array and examine its various parameters using the **get-eqlvolume** command.

```
PS C:\> get-eqlvolume | where-object {$_.volumename -eq "$volumename"} | ft –
property Volumename, volumesizeMB, Groupname, storagepoolname -AutoSize
```

Sample output:

```
VolumeName   VolumeSizeMB GroupName StoragePoolName
----------   ------------ --------- ---------------
TMEVMStorage       512000 TMProd    Gold
```

7.6.3 Refresh iSCSI Targets on Hyper-V Host

After confirmation that the volume has been successfully created, refresh the cached iSCSI target portal information on the Hyper-V host for the PS Group using the **update-iscsitargetportal** command specifying the PS group address stored in the **$groupaddress** variable.

```
PS C:\> update-iscsitargetportal -targetportaladdress $groupaddress
```

Note: The above command is equivalent to performing a refresh in the iSCSI initiator tool GUI.

Once the iSCSI targets have been refreshed, verify that the new EQL volume appears in the iSCSI targets visible to the Hyper-V host.

```
PS C:\> get-iscsitarget | ft -AutoSize
```

Sample output:

```
IsConnected     NodeAddress
-----------     -----------
False           iqn.2001-05.com.equallogic:0-1cb196-b481da94f-d8600000042522f7-
                vss-control
False           iqn.2001-05.com.equallogic:0-1cb196-dcc1da94f-b06000000af525d7-
                tmevmstorage
```

7.6.4 Connect and register new iSCSI target and session

Initially after the refresh, the new volume will be a visible iSCSI target but not connected (**IsConnected = False**). Follow the next steps is to connect and register the new iSCSI target.

1. Place the iSCSI node address for the new volume into a variable.

```
PS C:\> $iscsitarget = (Get-IscsiTarget | Where-Object -Filter
{$_.nodeaddress -like "*$volumename"}).nodeaddress
```

2. Make a new iSCSI connection to the target using the **connect-iscsitarget** command.

```
PS C:\> Connect-IscsiTarget -NodeAddress $iscsitarget
```

3. Verify that the target has connected successfully by re-running the **get-iscsitarget** command.

```
PS C:\> get-iscsitarget | ft -AutoSize
```

Sample output:

```
IsConnected   NodeAddress
-----------   -----------
False         iqn.2001-05.com.equallogic:0-1cb196-b481da94f-d8600000042522f7-vss-
              control
True          iqn.2001-05.com.equallogic:0-1cb196-dcc1da94f-b06000000af525d7-
              tmevmstorage
```

4. After connecting it, register the iSCSI session for the new volume so that the connection becomes persistent and will reconnect across Hyper-V host reboots. This task is done with a command string that first obtains the iSCSI session ID for the new volume iSCSI target connection, and then passes it into the **register-iscsisession** command.

```
PS C:\> get-iscsitarget | where-object {$_.nodeaddress -eq $iscsitarget} |
Get-IscsiSession | register-iscsisession
```

5. Verify that the session is registered by running the following command string and check that the **IsPersistent** property has been set to **True**.

```
PS C:\> get-iscsitarget | where-object {$_.nodeaddress -eq $iscsitarget} |
Get-IscsiSession
```

Sample output:

```
...
IsDiscovered            : True
IsHeaderDigest          : False
IsPersistent            : True
NumberOfConnections     : 1
SessionIdentifier       : fffffa801bba1430-400001370000001e
...
```

7.6.5 Rescan Hyper-V host storage

Now that the new volume has been connected and registered, perform a host storage rescan since there have been changes to the storage configuration on the Hyper-V host. There are two ways this can be done; the traditional method, or the native PowerShell command. The traditional **"rescan" | diskpart** command (Example A) is found in the diskpart utility. The native PowerShell command (Example B) is **update-hoststoragecache**. Both of these commands are equivalent to the **Rescan Storage** action in the GUI Server Manager utility.

Example A

```
PS C:\> "rescan" | diskpart
```

Sample output:

```
Microsoft DiskPart version 6.2.9200

Copyright (C) 1999-2012 Microsoft Corporation.
On computer: TMER4R805S30

DISKPART>
Please wait while DiskPart scans your configuration...

DiskPart has finished scanning your configuration.
```

Example B (the native PowerShell command)

```
PS C:\> update-hoststoragecache
```

The **update-hoststoragecache** command returns no output and depending on the number of attached devices, can take a considerable amount of time to complete. This command can be made to run in the back ground by including the **–asjob** parameter.

```
PS C:\> update-hoststoragecache –asjob | ft –autosize
```

Sample output:

```
Id Name    PSJobTypeName State   HasMoreData Location     Command
-- ----    ------------- -----   ----------- --------     -------
12 CimJob6 CimJob        Running True        TMER4R805S30 update-
hoststoragecache -asjob | …
```

When the **–asjob** parameter is used, the **update-hoststoragecache** command returns information about the job created from the command, including the job id. The progress of the job can be monitored with the **get-job** command for the specific job id of the **update-hoststoragecache** command.

```
PS C:\> get-job -id 12 | ft –AutoSize
```

Sample output:

```
Id Name    PSJobTypeName State     HasMoreData Location     Command
-- ----    ------------- -----     ----------- --------     -------
12 CimJob6 CimJob        Completed True        TMER4R805S30 update-
hoststoragecache -asjob …
```

Scripting tip: By grabbing the job id for a potentially long running command like **update-hoststoragecache**, a simple script can be written to monitor the progress of the job and then perform an action when the state of the job turns completed. An example of this script is:

```
$jobid = (update-hoststoragecache –asjob).id
Write-host “Updating host storage cache. Please wait…`n”
While ((get-job –id $jobid).state –ne “Completed”) {
  Write-host “>” –nonewline
  Start-sleep –seconds 10
}
Write-host “Host storage cache updated”
```

7.6.6 Get a physical disk number for the new EqualLogic iSCSI target

Once the host storage cache is updated, the new EqualLogic volume is visible to the operating system as an offline physical disk allowing the final steps of the provisioning process to occur.

First, identify the physical disk number associated with the new EqualLogic volume. This can be accomplished by using the **get-disk** command.

```
PS C:\> get-disk | ft -AutoSize
```

Sample output:

```
Number Friendly Name                          OperationalStatus Total Size
Partition Style
------ -------------                          ----------------- ---------- ----
-----------
0      Dell VIRTUAL DISK SCSI Disk Device      Online              67.75  GB MBR
2      EQLOGIC 100E-00  Multi-Path Disk Device Offline            500.01 GB RAW
```

Unfortunately, the output of the **get-disk** command does not readily identify the new EqualLogic volume by name. On a host with a single EqualLogic volume, identification is not be an issue, but on a host

provisioned with multiple EqualLogic volumes of the same size, this identification becomes more complicated. Fortunately, there is a one-line PowerShell command that provides a solution. Use the **get-iscsitarget** > **get-iscsisession** > **get-disk** command string to identify the disk derived from the specific iSCSI target associated with the EqualLogic volume.

```
PS C:\> get-iscsitarget -NodeAddress $iscsitarget | get-iscsisession |
get-disk | ft –autosize
```

Sample output:

```
Number   Friendly Name     OperationalStatus                Total Size   Partition Style
------   -------------     -----------------                ----------   ---------------
2        EQLOGIC 100E-00   Multi-Path Disk Device Offline   500.01 GB    RAW
```

First, the above command string gets the iSCSI session information for the iSCSI node address associated with the EqualLogic volume (stored in the $iscsitarget variable). The iSCSI session information is then passed into the **get-disk** command. This action focuses the get-disk on the target EqualLogic volume and displays the output when the command completes.

The returned disk number of the new EqualLogic volume into a variable as follows:

```
PS C:\> $disknumber = (get-iscsitarget -NodeAddress $iscsitarget |
get-iscsisession | get-disk).number
```

7.6.7 Initialize, partition, assign and format the new EqualLogic volume

The next step is to initialize, partition, and format the new EqualLogic volume using another powerful one-line command string.

```
PS C:\> initialize-disk -number $disknumber -partitionstyle GPT -passthru |
% {
   $_ | set-disk -isreadonly 0
   $_ | set-disk -isoffline 0
   $_ | new-partition -assigndriveletter -usemaximumsize |
   format-volume -filesystem NTFS -newfilesystemlabel $volumename
   -confirm:$false
}
```

First, the above command initializes the disk by specifying the disk number of the EqualLogic volume stored in the **$disknumber** variable. The object returned by this command can be passed down the pipeline to other commands by using the **–passthru** parameter. The object from the **initialize-disk** command is passed into the **set-disk** commands to set the disk to write enables and online.

The **new-partition** command runs next and creates a maximum size partition on the disk (**-usemaximumsize**). It also assigns the next available drive letter automatically (**-assigndriveletter**). The resulting object of the **new-partion** command is then passed to the final command **format-volume.** This places a NTFS file system (**-filesystem NTFS**) on the partition and assigns the name of the EqualLogic

volume stored earlier in the **$volumename** variable to the file system label (**-newfilesystemlabel $volumename**). The **–confirm:$false** tells the command to complete the tasks without prompting the user for confirmation.

Note: The **%** symbol in the above command string is short hand for the **foreach-object** cmdlet. This feature that performs an operation on a collection of input objects became available in Windows PowerShell 3.0. Within the script block, the **$_** variable is used to represent the current object. Use the PowerShell **get-help** command for more information on how to use the **foreach-object** cmdlet.

The newly provisioned volume is displayed in output of the previous command. It can also be seen by running the **get-volume** command.

```
PS C:\> get-volume | ft –AutoSize
```

Sample output:

```
DriveLetter  FileSystemLabel FileSystem   DriveType   HealthStatus  SizeRemaining  Size
-----------  --------------- ----------   ---------   ------------  -------------  ----
             System Reserved NTFS         Fixed       Healthy       108.69 MB      350 MB
F            TMEVMStorage    NTFS         Fixed       Healthy       499.8 GB       499.9 GB
C                            NTFS         Fixed       Healthy       54.2 GB        67.41 GB
D                                         CD-ROM      Healthy       0 B            0 B
```

To verify that the new volume is accessible, change the directory to it, create a file and then read from that file.

```
PS C:\ > cd F:\ ; get-process | out-file ./gp.txt ; ls
```

Sample output:

```
    Directory: F:\
Mode                 LastWriteTime          Length   Name
----                 -------------          ------   ----
-a---                11/26/2013   5:05 PM   18908    gp.txt
```

Read from the created file:

```
PS F:\> get-content -first 10 ./gp.txt
```

Sample output:

```
Handles   NPM(K)   PM(K)   WS(K)   VM(M)   CPU(s)   Id     ProcessName
-------   ------   -----   -----   -----   ------   --     -----------
21        3        1448    2036    15      0.00     2248   cmd
43        6        2404    6208    68      31.78    1576   conhost
...
175       15       3192    10228   73      5.19     1728   EHCMService
```

The newly provisioned EqualLogic volume is now connected, registered, initialized, and formatted with a NTFS file system on the Hyper-V host

7.7 Change the default virtual machine files and virtual hard disks storage locations

In Hyper-V, the default storage location for virtual machine files is **C:\programData\microsoft\windows\Hyper-V**, and the default location for the virtual machine virtual hard disks is **C:\users\public\documents\Hyper-V**.

Storing these critical components at the default locations on the internal c:\ drive of the Hyper-V host is unsuitable for long term. Change these defaults to the EqualLogic volume on the Hyper-V host as soon as possible after the initial setup of the Hyper-V server is complete using the following process.

1. Examine the default settings for the Hyper-V host using the **get-vmhost** command with the format-list (**fl**) command. This formats the output of a command as a list of properties where each one is displayed on a separate line. You can use format-list to format and display all or selected properties of an object as a list (**format-list ***).

   ```
   PS C:\> get-vmhost | fl *
   ```

   The following sample output shows that the virtual hard disk path and virtual machine path are using the default locations on the C:\ drive.

   ```
   ComputerName                                : TMER4R805S30
   VirtualHardDiskPath                         : C:\Users\Public\Documents\Hyper-
   V\Virtual Hard Disks
   VirtualMachinePath                          :
   C:\ProgramData\Microsoft\Windows\Hyper-V
   FullyQualifiedDomainName                    : Skynet.lab.local
   Name                                        : TMER4R805S30
   ...
   MaximumStorageMigrations                    : 2
   MaximumVirtualMachineMigrations             : 2
   VirtualMachineMigrationEnabled              : False
   VirtualMachineMigrationAuthenticationType : CredSSP
   UseAnyNetworkForMigration                   : False
   ...
   LogicalProcessorCount                       : 24
   MemoryCapacity                              : 274830245888
   ...
   InternalNetworkAdapters                     : {vsSAN2, vsSAN1, vsLAN}
   ...
   ```

2. Change these locations to the newly provisioned EqualLogic volume using the **set-vmhost** command.

```
PS C:\> set-vmhost -VirtualMachinePath F:\Hyper-V –VirtualHardDiskPath
"F:\Hyper-V\Virtual HardDisks"
```

3. To verify that the changes have taken, re-run the **get-vmhost | fl *** command.

```
PS C:\> get-vmhost | fl *
```

Sample output:

```
ComputerName                                : TMER4R805S30
VirtualHardDiskPath                         : F:\Hyper-V\Virtual Hard Disks
VirtualMachinePath                          : F:\Hyper-V
...
MaximumStorageMigrations                    : 2
MaximumVirtualMachineMigrations             : 2
```

Note: Other Hyper-V parameters can be set with the **set-vmhost** command. For example, the following **set-vmhost** command sets the maximum number of concurrent storage migrations to four.

```
PS C:\> set-vmhost -MaximumStorageMigrations 4

PS C:\> get-vmhost | fl *
...
MaximumStorageMigrations                    : 4
MaximumVirtualMachineMigrations             : 2
```

8 Step 8: Optional steps

At this point, the Hyper-V host installation is essentially complete. However, administrators might want to re-examine the network configuration on the Hyper-V host and shutdown or add features or services to the Windows Server Core installation.

8.1 Setting up a Converged Network for the Management OS

Note: Implementing a converged network and QOS can add a moderate amount of complexity into the virtualized environment. Unless there is a need for a converged network (such as noticeable bandwidth congestion), keep the virtualized network configuration as simple as possible. Another factor to consider is that implementing a converged network and QOS must be done using PowerShell. Earlier in this example, two other physical NIC adapters on the Hyper-V hosts were disabled. For a small to medium business where there are few servers being virtualized, an easier solution could be to re-enable and configure unused physical NIC adapters for clustering and/or migration traffic.

Prior to Windows Server 2012, Hyper-V and other functions (such as failover clustering) required separate networks to perform properly. The Hyper-V host had to have multiple NICs for management each traffic type often resulting in the host having six or more NICs. With Server 2012, this requirement has been removed as different types of network traffic can now share (or be converged over) a single NIC or NIC team. Converging different types of network traffic over a single NIC or NIC team requires a need to assign quality of service (QOS) metrics for each traffic type. For many small to medium businesses virtualizing a few servers, each with virtual machines, it is often not necessary to assign QOS metrics to the converged network traffic. However, in larger environments, it could be required. This optional step provides a method on separating out and assigning QOS metrics to live migration, cluster, and management traffic from the Management OS over a single NIC or NIC team.

1. First, re-verify the current network connections for the Hyper-V host.

```
[TMER4R805S30]: PS C:\> get-netadapter | where-object {$_.status -eq "up"} |
ft -autosize
```

This sample output was run through a remote PowerShell session to the Hyper-V host, designated by the "**[TMER4R805S30] PS C:\>"** prompt.

```
Name        InterfaceDescription                        ifIndex  Status  MacAddress  LinkSpeed
----        --------------------                        -------  ------  ----------  ---------
vEthernet   Hyper-V Virtual Ethernet Adapter #4         44       Up      00-10-18-…  10 Gbps
(vsSAN2)
vEthernet   Hyper-V Virtual Ethernet Adapter #3         51       Up      00-10-18-…  10 Gbps
(vsSAN1)
vEthernet   Hyper-V Virtual Ethernet Adapter #2         54       Up      90-B1-1C-…  10 Gbps
(vsLAN)
LANTeam     Microsoft Network Adapter Multiplexor       27       Up      90-B1-1C-…  2 Gbps
            Driver
LAN2        Broadcom NetXtreme Gigabit Ethernet #2      11       Up      90-B1-1C-…  1 Gbps
```

```
LAN1        Broadcom NetXtreme Gigabit Ethernet           10       Up      90-B1-1C-…  1 Gbps
SAN1        Broadcom BCM57810 NetXtreme II 1...#134       12       Up      00-10-18-…  10 Gbps
SAN2        Broadcom BCM57810 NetXtreme II 1...#131       13       Up      00-10-18-…  10 Gbps
```

2. Rename the vsLAN vNIC that was created automatically for the Management OS in section 5.1, to the Management NIC for easier identification.

```
[TMER4R805S30]: PS C:\> rename-vmnetworkadapter -managementOS -name "vsLAN" -
newname "Management"
```

Note: The **–managementOS** flag specifies that the object being worked on is resident in the Management OS partition.

3. Add two vNICs to the Management OS (one for the cluster traffic and one for migration traffic). Name the vNICs appropriately for easy identification and attach the vNICs to the vsLAN external virtual switch.

```
[TMER4R805S30]: PS C:\> add-vmnetworkadapter -managementOS -name "Cluster" -
switchname "vsLAN"

[TMER4R805S30]: PS C:\> add-vmnetworkadapter -managementOS -name "Migration"
-switchname "vsLAN"
```

4. Examine the updated network layout.

```
[TMER4R805S30]: PS C:\> get-netadapter | where-object {$_.status -eq "up"} |
ft -autosize
```

This sample output shows that the renamed management vNIC and new vNICs for migration and cluster traffic are now present. (This output has been truncated.)

```
Name          InterfaceDescription                    ifIndex   Status   MacAddress   LinkSpeed
----          --------------------                    -------   ------   ----------   ---------
vEthernet     Hyper-V Virtual Ethernet Adapter #6     82        Up       00-15-…      10 Gbps
(Migration)
vEthernet     Hyper-V Virtual Ethernet Adapter #5     78        Up       00-15-…      10 Gbps
(Cluster)
vEthernet     Hyper-V Virtual Ethernet Adapter #4     44        Up       00-10-…      10 Gbps
(vsSAN2)
vEthernet     Hyper-V Virtual Ethernet Adapter #3     51        Up       00-10-…      10 Gbps
(vsSAN1)
vEthernet     Hyper-V Virtual Ethernet Adapter #2     54        Up       90-B1-…      10 Gbps
(Management)
LANTeam       Microsoft Network Adapter Multiplexor…  27        Up       90-B1-…      2 Gbps
LAN2          Broadcom NetXtreme Gigabit Ethernet #2  11        Up       90-B1-…      1 Gbps
LAN1          Broadcom NetXtreme Gigabit Ethernet     10        Up       90-B1-…      1 Gbps
```

```
SAN1            Broadcom BCM57810 NetXtreme II          12        Up       00-10-…     10 Gbps
                1...#134
SAN2            Broadcom BCM57810 NetXtreme II          13        Up       00-10-…     10 Gbps
                1...#131
```

5. Set the bandwidth weight for each vNIC on the vsLAN virtual switch.
   Earlier, the vsLAN virtual switch was created using a minimum band width mode of **weight**. Minimum bandwidth rules are designed to guarantee a particular service level agreement (SLA) for a particular vNIC. When the minimum bandwidth is set to weight, this means that a particular vNIC is guaranteed a minimum bandwidth percentage of the total bandwidth. For example, if a vNIC is assigned a minimum bandwidth weight of 10, this means that the vNIC is guaranteed to have a minimum of 10% of the total bandwidth for a particular switch. If a virtual switch has 10 Gbps of total bandwidth, this particular vNIC will have a minimum of 1 Gbps bandwidth guaranteed.

   Table 2 Minimum bandwidths used for the vNICs on the vsLAN virtual switch

Network traffic	Adapter name	QOS (minimum bandwidth weight)
Management	Management	5
Cluster /CSV	Cluster	20
Live migration	Migration	30

   The following commands set the minimum bandwidth weights for the Management OS vNICs attached to the vsLAN virtual switch as specified in Table 2.

```
[TMER4R805S30]: PS C:\> set-vmnetworkadapter -managementOS -name "Management"
-minimumbandwidthweight "5"

[TMER4R805S30]: PS C:\> set-vmnetworkadapter -managementOS -name "Cluster" -
minimumbandwidthweight "20"

[TMER4R805S30]: PS C:\> set-vmnetworkadapter -managementOS -name "Migration"
-minimumbandwidthweight "30"
```

6. After setting the minimum bandwidth weight, examine each vNIC.

```
[TMER4R805S30]: PS C:\> get-vmnetworkadapter -managementOS
```

Sample output:

```
Name          IsManagementOs   VMName     SwitchName   MacAddress     Status   IPAddresses
----          --------------   ------     ----------   ----------     ------   -----------
Migration     True                        vsLAN        00155D043257   {Ok}
Cluster       True                        vsLAN        00155D043256   {Ok}
vsSAN2        True                        vsSAN2       0010183D8BC6   {Ok}
vsSAN1        True                        vsSAN1       0010183D8BA0   {Ok}
Management    True                        vsLAN        002219D25DC2   {Ok}
```

7. To get individual vNIC details, run the following command.

```
[TMER4R805S30]:PS C:\> get-vmnetworkadapter -managementOS -name "Cluster" |
fl *
```

8. Once the minimum bandwidth weights have been set for the vNICs, the IP addresses can be set for the migration and cluster vNICs using the same method discussed in section 2.4.

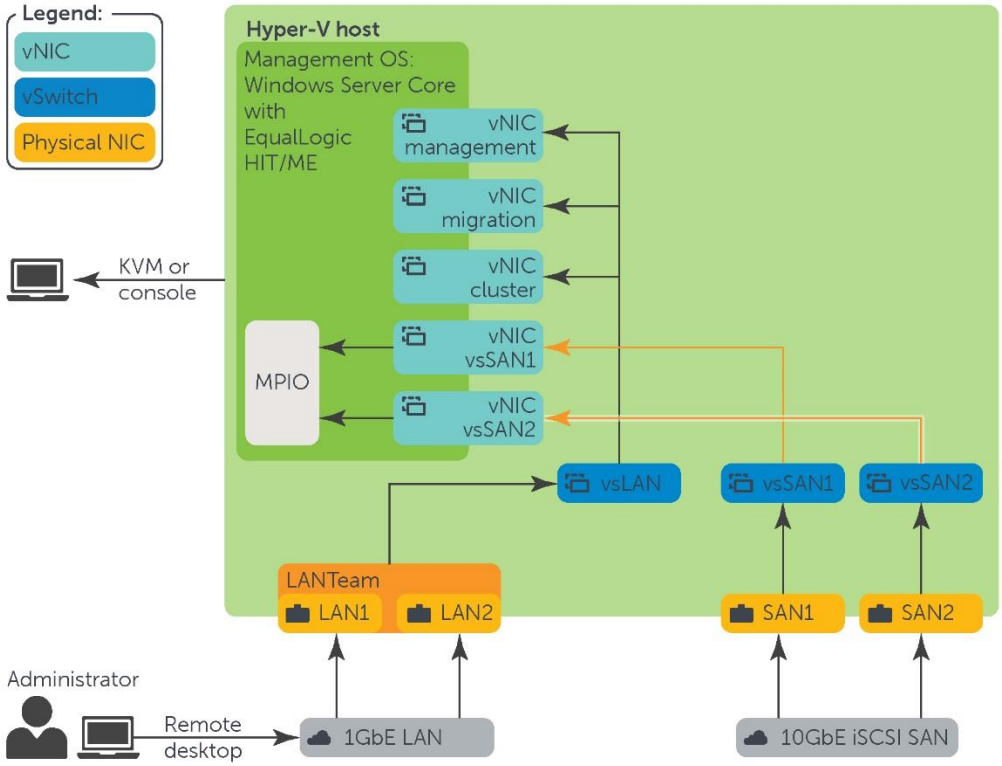


Figure 13 A sample converged network configuration for the Hyper-V server

Note: To keep the example simple, the implementation used in the remainder of this document **does not use** a converged network and QOS for the Management OS. Implementing QOS and a converged network into a virtualized environment increases the complexity of the overall solution. Careful study and bandwidth evaluation should be done prior to implementing QOS and a converged network in the virtualized environment to determine if it is needed. For many SMB environments, the bandwidth loading is often not enough to require these features.

For more information on Hyper-V QOS and converged networks go to:

- http://blogs.technet.com/b/wsnetdoc/archive/2012/10/18/new-quality-of-service-qos-documentation-for-windows-server-2012.aspx
- Aidan Finn (2013), Windows Server 2012 Hyper-V, Installation and Configuration Guide, John Wiley and Sons, Inc.

8.2 Disabling protocols and features such as IPV6

Shutting down unused services and features is part of the final configuration steps for the Hyper-V host. This step helps further reduce the consumed resources and reduces the potential attack footprint. In this example, the Hyper-V host is not using the IPV6 network protocol so this will be disabled on the network adapters.

Note although many companies chose to disable IPV6 since it is not used in their networks, Microsoft urges caution when doing so.
This quote is found at http://technet.microsoft.com/en-us/network/cc987595.aspx
"Microsoft recommends that you leave IPv6 enabled, even if you do not have an IPv6-enabled network, either native or tunneled. By leaving IPv6 enabled, you do not disable IPv6-only applications and services (for example, HomeGroup in Windows 7 and DirectAccess in Windows 7 and Windows Server 2008 R2 are IPv6-only) and your hosts can take advantage of IPv6-enhanced connectivity"

1. The following command string will provide detailed information for the network adapters used by the Hyper-V host (Management OS partition).

```
[TMER4R805S30]: PS C:\> get-netadapter | where-object {$_.status -eq "up"} |
get-netipconfiguration

InterfaceAlias       : vEthernet (vsSAN2)
InterfaceIndex       : 35
InterfaceDescription : Hyper-V Virtual Ethernet Adapter #4
NetProfile.Name      : Unidentified network
IPv4Address          : 10.10.6.211
IPv6DefaultGateway   :
IPv4DefaultGateway   :
DNSServer            : fec0:0:0:ffff::1
                       fec0:0:0:ffff::2
                       fec0:0:0:ffff::3

InterfaceAlias       : vEthernet (vsSAN1)
InterfaceIndex       : 33
```

```
InterfaceDescription : Hyper-V Virtual Ethernet Adapter #3
NetProfile.Name      : Unidentified network
IPv4Address          : 10.10.6.210
IPv6DefaultGateway   :
IPv4DefaultGateway   :
DNSServer            : fec0:0:0:ffff::1
                       fec0:0:0:ffff::2
                       fec0:0:0:ffff::3

InterfaceAlias       : vEthernet (vsLAN)
InterfaceIndex       : 31
InterfaceDescription : Hyper-V Virtual Ethernet Adapter #2
NetProfile.Name      : SKYNET.lab.local
IPv6Address          : fc00:495::35c5:2237:eb8a:8d29
IPv4Address          : 10.124.4.50
IPv6DefaultGateway   : fe80::21b:edff:fe10:1600
IPv4DefaultGateway   : 10.124.4.1
DNSServer            : 10.124.6.245
                       8.8.8.8
```

2. IPV6 can be disabled on the all the network adapters on the Hyper-V host by using the following PowerShell one liner script:

```
[TMER4R805S30]: PS C:\> foreach ($name in (get-netadapter | where-object
{$_.status -eq "up"}).name) {disable-netadapterbinding -name $name -
componentid ms_tcpip6}
```

3. To verify that IPV6 is now disabled on the network adapters run the get-netadapter command string again:

```
[TMER4R805S30]: PS C:\> get-netadapter | where-object {$_.status -eq "up"} |
get-netipconfiguration
```

In the following output example, there is no longer any IPV6 addressing or information for each of the adapters. This shows that IPV6 is disabled. This command line step is equivalent to unchecking the IPV6 protocol in the **Network** tab from the Ethernet Properties Window when an adapter is selected in the Server Manager GUI.

```
InterfaceAlias       : vEthernet (vsSAN2)
InterfaceIndex       : 44
InterfaceDescription : Hyper-V Virtual Ethernet Adapter #4
NetProfile.Name      : Unidentified network
IPv4Address          : 10.10.6.211
IPv4DefaultGateway   :
DNSServer            :

InterfaceAlias       : vEthernet (vsSAN1)
InterfaceIndex       : 51
InterfaceDescription : Hyper-V Virtual Ethernet Adapter #3
NetProfile.Name      : Unidentified network
IPv4Address          : 10.10.6.210
IPv4DefaultGateway   :
```

```
DNSServer             :

InterfaceAlias        : vEthernet (vsLAN)
InterfaceIndex        : 54
InterfaceDescription : Hyper-V Virtual Ethernet Adapter #2
NetProfile.Name       : SKYNET.lab.local
IPv4Address           : 10.124.4.50
IPv4DefaultGateway    : 10.124.4.1
DNSServer             : 10.124.6.245
                        8.8.8.8
```

8.3 Converting from Server Core to GUI

Remember that Server Core runs with the smallest footprint possible to present the most resources possible for the Hyper-V role. The simplicity of Server Core calls for the configuration and management to be done almost exclusively using PowerShell commands. Until familiarity is established with PowerShell, some administrators might want to install and use the Minimal Server Interface (MSI) on the Hyper-V host. The MSI will install the Server Manager utility, the PowerShell ISE, the Microsoft Management Console (MMC), and most other features found in the GUI installation. This does not include the desktop, Windows explorer, Internet Explorer, and the start screen.

The Minimum Server Interface can be installed with a single PowerShell command:

```
[TMER4R805S30]: PS C:\>Install-windowsfeature server-gui-mgmt-infra –restart
```

The full GUI can be installed with the following PowerShell command:

```
[TMER4R805S30]: PS C:\> Install-windowsfeature server-gui-mgmt-infra,
Server-GUI-Shell –restart
```

Notes:
• Installing MSI or the full GUI on a host running Server Core can take 30 minutes or more. If the host does not have access to a Windows Update, the installation process will fail at the very end. Prior to the installation of the MSI or GUI, ensure that the Hyper-V host has access to a Windows Update. This can be done by using the sconfig utility (discussed in section 3).
• After installing the GUI, make sure that all the consoles are working as expected by attempting to launch the Server Manager and then going to **tools** > **Hyper-V Manager**.

9 Summary

Successfully installing the Hyper-V host is the first task that needs to be completed when virtualizing a Windows environment. Installing Hyper-V on a Server Core host is the Microsoft preferred configuration; primarily because Server Core installs with the smallest footprint allowing for the most available resources for the Hyper-V role on the Hyper-V host. With Server Core, configuration and management has to be done almost exclusively with PowerShell and the sconfig utility. The absence of a GUI interface and the methodology provided through wizards for configuration and management can be daunting to an administrator unfamiliar with Server Core. This section provided installation and configuration of Hyper-V on a Server Core host in a logical series of steps. Each step was detailed with the various utilities and PowerShell commands needed to execute the included tasks. With the steps and associated details, the administrator can use this section as a reference to create a repeatable installation process for Hyper-V on Server Core hosts.