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

Dell Engineering
May 2014
Table of contents

Revisions ................................................................................................................................. 2
Executive summary .................................................................................................................. 5
Audience .................................................................................................................................. 5
Software versions supported by this document ........................................................................ 6
Overview ................................................................................................................................. 6
1  Step 1: Install Windows Server Core ...................................................................................... 8
   1.1  Launch Task Manager ..................................................................................................... 9
   1.2  Launch a PowerShell window .......................................................................................... 10
   1.3  Feature and performance review .................................................................................... 11
2  Step 2: Setup the Hyper-V Host network configuration .......................................................... 16
   2.1  View network adapter information for the server .............................................................. 16
   2.2  Modify physical NIC parameters on the Hyper-V host ....................................................... 18
   2.2.1  Rename LAN and iSCSI NICs ..................................................................................... 18
   2.2.2  Enable iSCSI NIC parameters for Jumbo Frames and Flow Control ............................... 19
   2.3  Optional: Create NIC team using LAN NICs as team members ....................................... 21
   2.4  Configure LAN NIC or LAN team .................................................................................... 24
   2.5  Set DNS server attributes for LAN NIC or LAN NIC team .............................................. 25
   2.6  Configure iSCSI NIC IP addresses .................................................................................. 26
3  Step 3: Complete the server configuration ............................................................................ 29
4  Step 4: Install Hyper-V .......................................................................................................... 31
   4.1  Install Hyper-V with Management Tools and Reboot ....................................................... 31
   4.2  Verify the Hyper-V installation ....................................................................................... 32
5  Step 5: Configure Hyper-V host virtual switches ................................................................. 35
   5.1  Configure Hyper-V virtual switches for LAN NICs or NIC team .................................. 35
   5.2  Configure Hyper-V virtual switches iSCSI NICs ............................................................. 36
   5.3  Verify successful Hyper-V virtual switch creation ......................................................... 37
   5.4  Enable jumbo frames on the management OS iSCSI vNICs ........................................... 39
6  Step 6: Setup remote access to the Hyper-V server ............................................................... 42
   6.1  Remote desktop ............................................................................................................... 42
   6.2  Remote PowerShell (PSRemoting) .................................................................................. 43
7  Step 7: Connect the Hyper-V Host to an EqualLogic iSCSI SAN ........................................ 47
7.1 Install the EQL Host Integration Tools for Microsoft (HIT / ME) onto the Hyper-V Host ........................................... 47
7.2 Locate and run HIT setup executable and Reboot ........................................................................................................... 49
7.3 Post-HIT installation tasks on the Hyper-V host ........................................................................................................... 51
7.4 Examining the HIT log files (optional) ....................................................................................................................... 55
7.5 Register and connect an EqualLogic PS Series array to the Hyper-V host ............................................................. 57
7.6 Provision an EqualLogic iSCSI Volume to the Hyper-V host ..................................................................................... 61
7.6.1 Identify key parameter values and store into variables ......................................................................................... 62
7.6.2 Create an EqualLogic volume using HIT PowerShell commands ........................................................................ 63
7.6.3 Refresh iSCSI Targets on Hyper-V Host .................................................................................................................. 64
7.6.4 Connect and register new iSCSI target and session ............................................................................................ 64
7.6.5 Rescan Hyper-V host storage ............................................................................................................................... 66
7.6.6 Get a physical disk number for the new EqualLogic iSCSI target ..................................................................... 67
7.6.7 Initialize, partition, assign and format the new EqualLogic volume ............................................................... 68
7.7 Change the default virtual machine files and virtual hard disks storage locations .............................................. 70
8  Step 8: Optional steps ...................................................................................................................................................... 72
8.1 Setting up a Converged Network for the Management OS ...................................................................................... 72
8.2 Disabling protocols and features such as IPV6 ......................................................................................................... 76
8.3 Converting from Server Core to GUI ......................................................................................................................... 78
9  Summary ....................................................................................................................................................................... 79
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 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:

- 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

<table>
<thead>
<tr>
<th>Software</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows Server 2012</td>
<td></td>
</tr>
<tr>
<td>Microsoft Windows Server 2012 R2</td>
<td></td>
</tr>
<tr>
<td>Dell EqualLogic Host Integration Toolkit Microsoft Edition (HIT/ME) V4.7</td>
<td>Support for both Windows Server 2012 and Server 2012 R2</td>
</tr>
<tr>
<td>Dell EqualLogic PS Series Array Firmware V7.0.1</td>
<td></td>
</tr>
</tbody>
</table>

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
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](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.

![Server Core as the default selection](image)

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](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.
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.
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**.
   
   ![PowerShell Window](image)

   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`
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.

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

An example of features list is displayed below.

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Install State</th>
</tr>
</thead>
<tbody>
<tr>
<td>[X] File and Storage Services</td>
<td>FileAndStorage-Services</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] Storage Services</td>
<td>Storage-Services</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] .NET Framework 4.5 Features</td>
<td>NET-Framework-45-Fea...</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] .NET Framework 4.5</td>
<td>NET-Framework-45-Core</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] WCF Services</td>
<td>NET-WCF-Services45</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] TCP Port Sharing</td>
<td>NET-WCF-TCP-PortShar...</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] User Interfaces and Infrastructure</td>
<td>User-Interfaces-Infra</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] Windows PowerShell</td>
<td>PowerShellRoot</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] Windows PowerShell 3.0</td>
<td>PowerShell</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] WoW64 Support</td>
<td>WoW64-Support</td>
<td>Installed</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Status</th>
<th>Name</th>
<th>DisplayName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running</td>
<td>BFE</td>
<td>Base Filtering Engine</td>
</tr>
<tr>
<td>Running</td>
<td>BITS</td>
<td>Background Intelligent Transfer Ser...</td>
</tr>
<tr>
<td>Running</td>
<td>CertPropSvc</td>
<td>Certificate Propagation</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>Cryptographic Services</td>
</tr>
<tr>
<td>Running</td>
<td>WinRM</td>
<td>Windows Remote Management (WS-...</td>
</tr>
<tr>
<td>Running</td>
<td>wuauserv</td>
<td>Windows Update</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>ExitCode</th>
<th>Name</th>
<th>ProcessId</th>
<th>StartMode</th>
<th>State</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>AeLookupSvc</td>
<td>0</td>
<td>Manual</td>
<td>Stopped</td>
<td>OK</td>
</tr>
<tr>
<td>1077</td>
<td>AppMgmt</td>
<td>0</td>
<td>Manual</td>
<td>Stopped</td>
<td>OK</td>
</tr>
<tr>
<td>0</td>
<td>BFE</td>
<td>1352</td>
<td>Auto</td>
<td>Running</td>
<td>OK</td>
</tr>
<tr>
<td>0</td>
<td>BITS</td>
<td>1036</td>
<td>Manual</td>
<td>Running</td>
<td>OK</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>WinRM</td>
<td>1208</td>
<td>Auto</td>
<td>Running</td>
<td>OK</td>
</tr>
<tr>
<td>1077</td>
<td>wmiApSrv</td>
<td>0</td>
<td>Manual</td>
<td>Stopped</td>
<td>OK</td>
</tr>
<tr>
<td>0</td>
<td>wuauserv</td>
<td>0</td>
<td>Manual</td>
<td>Stopped</td>
<td>OK</td>
</tr>
<tr>
<td>1077</td>
<td>wudfsvc</td>
<td>0</td>
<td>Manual</td>
<td>Stopped</td>
<td>OK</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Handles</th>
<th>NPM(K)</th>
<th>PM(K)</th>
<th>WS(K)</th>
<th>VM(M)</th>
<th>CPU(s)</th>
<th>Id</th>
<th>ProcessName</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>3</td>
<td>1436</td>
<td>1892</td>
<td>12</td>
<td>0.02</td>
<td>1156</td>
<td>cmd</td>
</tr>
<tr>
<td>50</td>
<td>7</td>
<td>2272</td>
<td>6836</td>
<td>67</td>
<td>3.79</td>
<td>1408</td>
<td>conhost</td>
</tr>
<tr>
<td>47</td>
<td>6</td>
<td>1048</td>
<td>4676</td>
<td>64</td>
<td>0.16</td>
<td>2044</td>
<td>Conhost</td>
</tr>
<tr>
<td>133</td>
<td>10</td>
<td>1364</td>
<td>3356</td>
<td>42</td>
<td>1.76</td>
<td>776</td>
<td>csrss</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>800</td>
<td>41</td>
<td>16160</td>
<td>24212</td>
<td>870</td>
<td>77.11</td>
<td>1208</td>
<td>svchost</td>
</tr>
<tr>
<td>343</td>
<td>31</td>
<td>11112</td>
<td>13112</td>
<td>55</td>
<td>25.97</td>
<td>1352</td>
<td>svchost</td>
</tr>
<tr>
<td>650</td>
<td>29</td>
<td>36988</td>
<td>44556</td>
<td>159</td>
<td>12.29</td>
<td>1464</td>
<td>Svchost</td>
</tr>
<tr>
<td>256</td>
<td>20</td>
<td>12548</td>
<td>10952</td>
<td>1101</td>
<td>11.06</td>
<td>1516</td>
<td>svchost</td>
</tr>
<tr>
<td>1030</td>
<td>0</td>
<td>112</td>
<td>304</td>
<td>3</td>
<td>7,060.73</td>
<td>4</td>
<td>System</td>
</tr>
<tr>
<td>142</td>
<td>11</td>
<td>3564</td>
<td>6072</td>
<td>93</td>
<td>0.09</td>
<td>1364</td>
<td>taskhostex</td>
</tr>
<tr>
<td>285</td>
<td>21</td>
<td>11740</td>
<td>20892</td>
<td>191</td>
<td>6,238.34</td>
<td>1380</td>
<td>Taskmgr</td>
</tr>
<tr>
<td>656</td>
<td>35</td>
<td>44928</td>
<td>31544</td>
<td>167</td>
<td>11.58</td>
<td>1568</td>
<td>Vmms</td>
</tr>
<tr>
<td>84</td>
<td>8</td>
<td>876</td>
<td>3472</td>
<td>21</td>
<td>0.06</td>
<td>836</td>
<td>wininit</td>
</tr>
<tr>
<td>96</td>
<td>7</td>
<td>1156</td>
<td>4800</td>
<td>48</td>
<td>0.09</td>
<td>888</td>
<td>winlogon</td>
</tr>
<tr>
<td>132</td>
<td>8</td>
<td>1224</td>
<td>5448</td>
<td>49</td>
<td>0.28</td>
<td>2136</td>
<td>winlogon</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Index</th>
<th>Time</th>
<th>EntryType</th>
<th>Source</th>
<th>InstanceID</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>101063</td>
<td>Nov 07 10:00</td>
<td>Information</td>
<td>Service Control Manager</td>
<td>1073748860</td>
<td>The WMI Performance Adapter…</td>
</tr>
<tr>
<td>101062</td>
<td>Nov 07 10:00</td>
<td>Information</td>
<td>Service Control Manager</td>
<td>1073748860</td>
<td>The WMI Performance Adapter…</td>
</tr>
</tbody>
</table>

**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: 

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.

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>CounterSamples</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/7/2013 10:44:29 AM</td>
<td>\tmer4r805s30\memory% committed bytes in use: 6.2952370668016</td>
</tr>
<tr>
<td>11/7/2013 10:44:31 AM</td>
<td>\tmer4r805s30\memory% committed bytes in use: 6.2957212138282</td>
</tr>
<tr>
<td>11/7/2013 10:44:33 AM</td>
<td>\tmer4r805s30\memory% committed bytes in use: 6.2960685366951</td>
</tr>
</tbody>
</table>

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.

![Hyper-V host diagram](image)

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:

<table>
<thead>
<tr>
<th>Name</th>
<th>InterfaceDescription</th>
<th>ifIndex</th>
<th>Status</th>
<th>MacAddress</th>
<th>LinkSpeed</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIC5</td>
<td>Broadcom NetXtreme Gigabit Ethernet #5</td>
<td>14</td>
<td>Not Present</td>
<td>90-B1-1C-56-23-1 Gbps</td>
<td></td>
</tr>
<tr>
<td>NIC4</td>
<td>Broadcom NetXtreme Gigabit Ethernet #4</td>
<td>13</td>
<td>Not Present</td>
<td>90-B1-1C-56-23-1 Gbps</td>
<td></td>
</tr>
<tr>
<td>NIC2</td>
<td>Broadcom NetXtreme Gigabit Ethernet #2</td>
<td>11</td>
<td>Up</td>
<td>90-B1-1C-56-21-1 Gbps</td>
<td></td>
</tr>
<tr>
<td>NIC1</td>
<td>Broadcom NetXtreme Gigabit Ethernet #10</td>
<td>10</td>
<td>Up</td>
<td>90-B1-1C-56-21-1 Gbps</td>
<td></td>
</tr>
<tr>
<td>SLOT 4</td>
<td>Broadcom BCM57810 NetXtreme II Port 2</td>
<td>12</td>
<td>Up</td>
<td>00-10-18-ED-C1-10 Gbps</td>
<td></td>
</tr>
<tr>
<td>SLOT 6</td>
<td>Broadcom BCM57810 NetXtreme II Port 1...#134</td>
<td>13</td>
<td>Up</td>
<td>00-10-18-ED-A0-10 Gbps</td>
<td></td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>InterfaceDescription</th>
<th>ifInd</th>
<th>Status</th>
<th>MacAddress</th>
<th>LinkSpeed</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIC2</td>
<td>Broadcom NetXtreme Gigabit Ethernet #2</td>
<td>11</td>
<td>Up</td>
<td>90-B1-1C-56-</td>
<td>1 Gbps</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21-B2</td>
<td></td>
</tr>
<tr>
<td>NIC1</td>
<td>Broadcom NetXtreme Gigabit Ethernet</td>
<td>10</td>
<td>Up</td>
<td>90-B1-1C-56-</td>
<td>1 Gbps</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21-B1</td>
<td></td>
</tr>
<tr>
<td>SLOT 4</td>
<td>Broadcom BCM57810 NetXtreme II Port 2</td>
<td>12</td>
<td>Up</td>
<td>00-10-18-ED-</td>
<td>10 Gbps</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C1-32</td>
<td></td>
</tr>
<tr>
<td>SLOT 6</td>
<td>Broadcom BCM57810 NetXtreme II Port 2</td>
<td>13</td>
<td>Up</td>
<td>00-10-18-ED-</td>
<td>10 Gbps</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A0-82</td>
<td></td>
</tr>
</tbody>
</table>

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
```

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

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.
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.

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

**Sample output:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Displayname</th>
<th>Displayvalue</th>
<th>validdisplayvalues</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAN1</td>
<td>Flow Control</td>
<td>Rx &amp; Tx Enabled</td>
<td>{Disabled, Tx Enabled, Rx Enabled, Rx &amp; Tx Enabled}</td>
</tr>
<tr>
<td>SAN2</td>
<td>Flow Control</td>
<td>Rx &amp; Tx Enabled</td>
<td>{Disabled, Tx Enabled, Rx Enabled, Rx &amp; Tx Enabled}</td>
</tr>
<tr>
<td>SAN1</td>
<td>Jumbo Packet</td>
<td>1514</td>
<td>{1514, 4088, 9014}</td>
</tr>
<tr>
<td>SAN2</td>
<td>Jumbo Packet</td>
<td>1514</td>
<td>{1514, 4088, 9014}</td>
</tr>
</tbody>
</table>

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.
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.

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

Sample output:

<table>
<thead>
<tr>
<th>Name</th>
<th>Displayname</th>
<th>Displayvalue</th>
<th>validdisplayvalues</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAN1</td>
<td>Jumbo Packet</td>
<td>9014</td>
<td>{1514, 4088, 9014}</td>
</tr>
<tr>
<td>SAN2</td>
<td>Jumbo Packet</td>
<td>9014</td>
<td>{1514, 4088, 9014}</td>
</tr>
</tbody>
</table>

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.


**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.

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).

```powershell
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).

```powershell
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

<table>
<thead>
<tr>
<th>Common subnet masks</th>
<th>Prefix length</th>
</tr>
</thead>
<tbody>
<tr>
<td>255.255.255.252</td>
<td>30</td>
</tr>
<tr>
<td>255.255.255.192</td>
<td>26</td>
</tr>
<tr>
<td>255.255.255.0</td>
<td>24</td>
</tr>
<tr>
<td>255.255.252.0</td>
<td>22</td>
</tr>
<tr>
<td>255.255.0.0</td>
<td>16</td>
</tr>
</tbody>
</table>

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.

   ```powershell
   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.

   ```powershell
   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.

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

   Sample output:

   ```powershell
   ... 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.

   ```powershell
   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.

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

Sample output:

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

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](image)

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.

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

   Sample output:

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

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

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

   Sample output:

   ```text
   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.

   ```powershell
   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.

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

   Sample output:

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

<table>
<thead>
<tr>
<th>LogMode</th>
<th>MaximumSizeInBytes</th>
<th>RecordCount</th>
<th>LogName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circular</td>
<td>1052672</td>
<td>66</td>
<td>Microsoft-Windows-Hyper-V-Config-Admin</td>
</tr>
<tr>
<td>Circular</td>
<td>1052672</td>
<td>0</td>
<td>Microsoft-Windows-Hyper-V-Config-Operational</td>
</tr>
<tr>
<td>Circular</td>
<td>1052672</td>
<td>0</td>
<td>Microsoft-Windows-Hyper-V-Hypervisor-Admin</td>
</tr>
<tr>
<td>Circular</td>
<td>1052672</td>
<td>215</td>
<td>Microsoft-Windows-Hyper-V-Hypervisor-Admin</td>
</tr>
<tr>
<td>Circular</td>
<td>1052672</td>
<td>624</td>
<td>Microsoft-Windows-Hyper-V-Integration-Admin</td>
</tr>
<tr>
<td>Circular</td>
<td>1052672</td>
<td>0</td>
<td>Microsoft-Windows-Hyper-V-SynthFc-Admin</td>
</tr>
<tr>
<td>Circular</td>
<td>1052672</td>
<td>867</td>
<td>Microsoft-Windows-Hyper-V-SynthNic-Admin</td>
</tr>
<tr>
<td>Circular</td>
<td>1052672</td>
<td>0</td>
<td>Microsoft-Windows-Hyper-V-SynthStor-Admin</td>
</tr>
<tr>
<td>Circular</td>
<td>1052672</td>
<td>0</td>
<td>Microsoft-Windows-Hyper-V-SynthStor-Admin</td>
</tr>
<tr>
<td>Circular</td>
<td>1052672</td>
<td>0</td>
<td>Microsoft-Windows-Hyper-V-VID-Admin</td>
</tr>
<tr>
<td>Circular</td>
<td>1052672</td>
<td>189</td>
<td>Microsoft-Windows-Hyper-V-VMMS-Admin</td>
</tr>
<tr>
<td>Circular</td>
<td>1052672</td>
<td>21</td>
<td>Microsoft-Windows-Hyper-V-VMMS-Networking</td>
</tr>
<tr>
<td>Circular</td>
<td>1052672</td>
<td>32</td>
<td>Microsoft-Windows-Hyper-V-VMMS-Operational</td>
</tr>
<tr>
<td>Circular</td>
<td>1052672</td>
<td>1319</td>
<td>Microsoft-Windows-Hyper-V-VMMS-Storage</td>
</tr>
<tr>
<td>Circular</td>
<td>1052672</td>
<td>322</td>
<td>Microsoft-Windows-Hyper-V-Worker-Admin</td>
</tr>
</tbody>
</table>
6. To get a listing of the features installed with the Hyper-V role and management tools, run the `get-windwindowsfeature` command.

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

   Sample output:

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Install State</th>
</tr>
</thead>
<tbody>
<tr>
<td>[X] File And Storage Services</td>
<td>FileAndStorage-Services</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] Storage Services</td>
<td>Storage-Services</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] Hyper-V</td>
<td>Hyper-V</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] .NET Framework 4.5 Features</td>
<td>NET-Framework-45-Fea...</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] .NET Framework 4.5</td>
<td>NET-Framework-45-Core</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] WCF Services</td>
<td>NET-WCF-Services45</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] TCP Port Sharing</td>
<td>NET-WCF-TCP-PortShar...</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] Multipath I/O</td>
<td>Multipath-IO</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] Remote Server Administration Tools</td>
<td>RSAT</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] Role Administration Tools</td>
<td>RSAT-Role-Tools</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] Hyper-V Management Tools</td>
<td>RSAT-Hyper-V-Tools</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] Hyper-V Module for Windows PowerShell</td>
<td>Hyper-V-PowerShell</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] User Interfaces and Infrastructure</td>
<td>User-Interfaces-Infra</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] Windows PowerShell</td>
<td>PowerShellRoot</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] Windows PowerShell 3.0</td>
<td>PowerShell</td>
<td>Installed</td>
</tr>
<tr>
<td>[X] WoW64 Support</td>
<td>WoW64-Support</td>
<td>Installed</td>
</tr>
</tbody>
</table>

**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](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" -allowmanagementOS $true -minimumbandwidthmode weight
```

Sample output:

<table>
<thead>
<tr>
<th>Name</th>
<th>SwitchType</th>
<th>NetAdapterInterfaceDescription</th>
</tr>
</thead>
<tbody>
<tr>
<td>vsLAN</td>
<td>External</td>
<td>Microsoft Network Adapter Multiplexor Driver</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Name</th>
<th>SwitchType</th>
<th>NetAdapterInterfaceDescription</th>
</tr>
</thead>
<tbody>
<tr>
<td>vsSAN2</td>
<td>External</td>
<td>Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client) #134</td>
</tr>
<tr>
<td>vsSAN1</td>
<td>External</td>
<td>Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client) #131</td>
</tr>
<tr>
<td>vsLAN</td>
<td>External</td>
<td>Microsoft Network Adapter Multiplexor Driver</td>
</tr>
</tbody>
</table>

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:

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

<table>
<thead>
<tr>
<th>InterfaceAlias</th>
<th>vEthernet (vsSAN2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>InterfaceIndex</td>
<td>44</td>
</tr>
<tr>
<td>InterfaceDescription</td>
<td>Hyper-V Virtual Ethernet Adapter #4</td>
</tr>
<tr>
<td>NetProfile.Name</td>
<td>Unidentified network</td>
</tr>
<tr>
<td>IPv4Address</td>
<td>10.10.6.211</td>
</tr>
<tr>
<td>IPv4DefaultGateway</td>
<td></td>
</tr>
<tr>
<td>DNSServer</td>
<td>fec0:0:0:ffff::1</td>
</tr>
<tr>
<td></td>
<td>fec0:0:0:ffff::2</td>
</tr>
<tr>
<td></td>
<td>fec0:0:0:ffff</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>InterfaceAlias</th>
<th>vEthernet (vsSAN1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>InterfaceIndex</td>
<td>51</td>
</tr>
<tr>
<td>InterfaceDescription</td>
<td>Hyper-V Virtual Ethernet Adapter #3</td>
</tr>
<tr>
<td>NetProfile.Name</td>
<td>Unidentified network</td>
</tr>
<tr>
<td>IPv4Address</td>
<td>10.10.6.210</td>
</tr>
<tr>
<td>IPv4DefaultGateway</td>
<td></td>
</tr>
<tr>
<td>DNSServer</td>
<td>fec0:0:0:ffff::1</td>
</tr>
<tr>
<td></td>
<td>fec0:0:0:ffff::2</td>
</tr>
<tr>
<td></td>
<td>fec0:0:0:ffff::3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>InterfaceAlias</th>
<th>vEthernet (vsLAN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>InterfaceIndex</td>
<td>54</td>
</tr>
<tr>
<td>InterfaceDescription</td>
<td>Hyper-V Virtual Ethernet Adapter #2</td>
</tr>
<tr>
<td>NetProfile.Name</td>
<td>SKYNET.lab.local</td>
</tr>
<tr>
<td>IPv6Address</td>
<td>fc00:495::5d87:2eb2:1868:1260</td>
</tr>
<tr>
<td>IPv4Address</td>
<td>10.124.4.50</td>
</tr>
<tr>
<td>IPv4DefaultGateway</td>
<td>fe80::21b:edff:fe10:1600</td>
</tr>
<tr>
<td>DNSServer</td>
<td>10.124.6.245</td>
</tr>
<tr>
<td></td>
<td>8.8.8.8</td>
</tr>
</tbody>
</table>
```
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, verify that jumbo frames is enabled for the Management OS iSCSI vNICs using the `get-netadapteradvancedproperty` command.

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

Sample output:

<table>
<thead>
<tr>
<th>Name</th>
<th>Displayname</th>
<th>Displayvalue</th>
<th>validdisplayvalues</th>
</tr>
</thead>
<tbody>
<tr>
<td>vEthernet (vsSAN2)</td>
<td>Jumbo Packet</td>
<td>Disabled</td>
<td>{Disabled, 4088 Bytes, 9014 Bytes}</td>
</tr>
<tr>
<td>vEthernet (vsSAN1)</td>
<td>Jumbo Packet</td>
<td>Disabled</td>
<td>{Disabled, 4088 Bytes, 9014 Bytes}</td>
</tr>
</tbody>
</table>

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.

```powershell
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.

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

Sample output:

<table>
<thead>
<tr>
<th>Name</th>
<th>Displayname</th>
<th>Displayvalue</th>
<th>validdisplayvalues</th>
</tr>
</thead>
<tbody>
<tr>
<td>vEthernet (vsSAN2)</td>
<td>Jumbo Packet</td>
<td>9014 Bytes</td>
<td>{Disabled, 4088 Bytes, 9014 Bytes}</td>
</tr>
<tr>
<td>vEthernet (vsSAN1)</td>
<td>Jumbo Packet</td>
<td>9014 Bytes</td>
<td>{Disabled, 4088 Bytes, 9014 Bytes}</td>
</tr>
</tbody>
</table>
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.

\[
\text{Size of packet = 9000 Byte MTU – 28 Bytes overhead = 8972}
\]

```
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 NICs. 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
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.

```
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.

```
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.

```
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.

![Windows PowerShell Credential Request](image)

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

```
[TR1098] 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 : [TR1098] 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.
WSManConfig: Microsoft.WSMan.Management\WSMan::localhost\Client

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>SourceOfValue</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>System.String</td>
<td>TrustedHosts</td>
<td></td>
<td>TMER4R805S30</td>
</tr>
</tbody>
</table>

**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-PSession**

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](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.

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

   **Sample output:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Used (GB)</th>
<th>Free (GB)</th>
<th>Provider</th>
<th>Root</th>
<th>CurrentLocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>13.62</td>
<td>53.78</td>
<td>FileSystem</td>
<td>C:</td>
<td>Windows\system32</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td>FileSystem</td>
<td>D:</td>
<td></td>
</tr>
</tbody>
</table>

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.

```bash
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:

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

Sample output:

<table>
<thead>
<tr>
<th>Name</th>
<th>Used (GB)</th>
<th>Free (GB)</th>
<th>Provider</th>
<th>Root</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>10.64</td>
<td>489.24</td>
<td>FileSystem</td>
<td>\10.124.4.66\FileShare</td>
</tr>
</tbody>
</table>

**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.

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

Sample output:

<table>
<thead>
<tr>
<th>Name</th>
<th>Used (GB)</th>
<th>Free (GB)</th>
<th>Provider</th>
<th>Root</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>13.62</td>
<td>53.78</td>
<td>FileSystem</td>
<td>C:\</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td>FileSystem</td>
<td>D:\</td>
</tr>
<tr>
<td>E</td>
<td>10.64</td>
<td>489.24</td>
<td>FileSystem</td>
<td>\10.124.4.66\FileShare</td>
</tr>
</tbody>
</table>
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-- 7/31/2013 2:00 PM   465463 hit-release-notes.pdf
...
```
2. Verify that MPIO has been installed using the `get-windowsfeature` command.

PS C:\> `get-windowsfeature` | ? installed | `ft -autosize`

Sample output:

<table>
<thead>
<tr>
<th>Display Name</th>
<th>Name</th>
<th>Install State</th>
</tr>
</thead>
<tbody>
<tr>
<td>File And Storage Services</td>
<td>FileAndStorage-Services</td>
<td>Installed</td>
</tr>
<tr>
<td>Storage Services</td>
<td>Storage-Services</td>
<td>Installed</td>
</tr>
<tr>
<td>Hyper-V</td>
<td>Hyper-V</td>
<td>Installed</td>
</tr>
<tr>
<td>.NET Framework 4.5 Features</td>
<td>NET-Framework-45-Features</td>
<td>Installed</td>
</tr>
<tr>
<td>.NET Framework 4.5</td>
<td>NET-Framework-45-Core</td>
<td>Installed</td>
</tr>
<tr>
<td>WCF Services</td>
<td>NET-WCF-Services45</td>
<td>Installed</td>
</tr>
<tr>
<td>TCP Port Sharing</td>
<td>NET-WCF-TCP-PortSharing45</td>
<td>Installed</td>
</tr>
<tr>
<td>Multipath I/O</td>
<td>Multipath-I0</td>
<td>Installed</td>
</tr>
<tr>
<td>Remote Server Administration Tools</td>
<td>RSAT</td>
<td>Installed</td>
</tr>
<tr>
<td>Role Administration Tools</td>
<td>RSAT-Role-Tools</td>
<td>Installed</td>
</tr>
<tr>
<td>Hyper-V Management Tools</td>
<td>RSAT-Hyper-V-Tools</td>
<td>Installed</td>
</tr>
<tr>
<td>Hyper-V Module for Windows</td>
<td>Hyper-V-PowerShell</td>
<td>Installed</td>
</tr>
<tr>
<td>User Interfaces and Infrastructure</td>
<td>User-Interfaces-Infra</td>
<td>Installed</td>
</tr>
<tr>
<td>Windows PowerShell</td>
<td>PowerShellRoot</td>
<td>Installed</td>
</tr>
<tr>
<td>Windows PowerShell 3.0</td>
<td>PowerShell</td>
<td>Installed</td>
</tr>
<tr>
<td>WoW64 Support</td>
<td>WoW64-Support</td>
<td>Installed</td>
</tr>
</tbody>
</table>

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 MSiSCI 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   Eq1Logd           EqualLogic Trace Logging Service
Running   Eq1ReqService      EqualLogic VSS Requestor
Running   Eq1SMFHost         EqualLogic SMP Host Service
Stopped   Eq1VdsHWPrv        EqualLogic VDS Hardware Provider
Running   Eq1Vss            EqualLogic VSS Service
```

**Note:** The `Eq1VdsHWPrv` 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 idea to examine the application event with the `get-winevent` command, filtering for EqualLogic as a provider name.

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

Sample output:

```
ProviderName: EqualLogic

<table>
<thead>
<tr>
<th>TimeCreated</th>
<th>Id</th>
<th>LevelDisplayName</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/20/2013 4:59:34 PM</td>
<td>1000</td>
<td>Information</td>
<td>The service EqlVss has been</td>
</tr>
<tr>
<td>11/20/2013 4:59:27 PM</td>
<td>1000</td>
<td>Information</td>
<td>The service EHCMService has been</td>
</tr>
<tr>
<td>11/20/2013 4:57:23 PM</td>
<td>1000</td>
<td>Information</td>
<td>The service EqlLogd has been</td>
</tr>
<tr>
<td>11/20/2013 4:54:12 PM</td>
<td>1000</td>
<td>Information</td>
<td>The service EqlVss has been</td>
</tr>
<tr>
<td>11/20/2013 4:54:08 PM</td>
<td>1000</td>
<td>Information</td>
<td>The service EqlLogd has been</td>
</tr>
</tbody>
</table>
```

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.

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

Sample output:

```
Directory: C:\

<table>
<thead>
<tr>
<th>Mode</th>
<th>LastWriteTime</th>
<th>Length</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>d----</td>
<td>12/19/2012</td>
<td></td>
<td>Dell</td>
</tr>
<tr>
<td>d----</td>
<td>7/26/2012</td>
<td></td>
<td>PerfLogs</td>
</tr>
<tr>
<td>d-r--</td>
<td>10/9/2013</td>
<td></td>
<td>Program Files</td>
</tr>
<tr>
<td>d----</td>
<td>10/9/2013</td>
<td></td>
<td>Program Files (x86)</td>
</tr>
<tr>
<td>d-r--</td>
<td>10/4/2013</td>
<td></td>
<td>Users</td>
</tr>
<tr>
<td>d----</td>
<td>10/2/2013</td>
<td></td>
<td>Windows</td>
</tr>
<tr>
<td>d----</td>
<td>10/2/2013</td>
<td></td>
<td>Windows.old</td>
</tr>
</tbody>
</table>
```
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
-a-rhs 7/25/2012 11:44 PM 52428933 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.

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

Sample output:

```
...  
```

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](https://www.dell.com/eknowledge/1) and consult the [PS Series Array Setup and Installation Guide](https://www.dell.com/eknowledge/1).

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

   ```powershell
   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 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 : 6019CBF194DA71B4F722150400020C3
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                          : 6019CBF194DA71B4F722150400020C3
groupId                           : 6019CBF194DA71B4F722150400020C3
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
... 
GroupName                          : 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.

```powershell
PS C:\> get-iscsitargetportal

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

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.

```powershell
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.

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

Sample output:

<table>
<thead>
<tr>
<th>GroupName</th>
<th>GroupAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMProd</td>
<td>10.10.6.200</td>
</tr>
</tbody>
</table>

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

**Note:** The `get-eqlgroupconfiguration` 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.

```powershell
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:

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

<table>
<thead>
<tr>
<th>StoragePoolName</th>
<th>TotalSpaceMB</th>
<th>UsedSpaceMB</th>
<th>FreeSpaceMB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold</td>
<td>15614805</td>
<td>285570</td>
<td>15120270</td>
</tr>
</tbody>
</table>
From the above output, it can be seen that there is a single pool named “Gold” associated with PS Group TMPProd 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.

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

   Sample output:

   ign.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 that can then be reused in the PowerShell runspace.

   ```powershell
   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.

```
powershell
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-eqlvolumeacl`).
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:

<table>
<thead>
<tr>
<th>VolumeName</th>
<th>VolumeSizeMB</th>
<th>GroupName</th>
<th>StoragePoolName</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMEVMStorage</td>
<td>512000</td>
<td>TM Prod</td>
<td>Gold</td>
</tr>
</tbody>
</table>

### 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:

<table>
<thead>
<tr>
<th>IsConnected</th>
<th>NodeAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>iqn.2001-05.com.equallogic:0-1cb196-b481da94f-d8600000042522f7-vss-control</td>
</tr>
<tr>
<td>False</td>
<td>iqn.2001-05.com.equallogic:0-1cb196-dcc1da94f-b06000000af525d7-tmevmstorage</td>
</tr>
</tbody>
</table>

### 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.
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:

<table>
<thead>
<tr>
<th>IsConnected</th>
<th>NodeAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>iqn.2001-05.com.equallogic:0-1cb196-b481da94f-d8600000042522f7-vss-control</td>
</tr>
<tr>
<td>True</td>
<td>iqn.2001-05.com.equallogic:0-1cb196-dc1da94f-b06000000af525d7-tmevmstorage</td>
</tr>
</tbody>
</table>

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 : fffffffa801bba1430-4000013700000001e
   ...

---

**TR1098-4 | Setup and Configuration of a Windows Server Core Hyper-V host using PowerShell and EqualLogic Storage and Tools**
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:

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>PSJobTypeName</th>
<th>State</th>
<th>HasMoreData</th>
<th>Location</th>
<th>Command</th>
</tr>
</thead>
</table>
| 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:

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>PSJobTypeName</th>
<th>State</th>
<th>HasMoreData</th>
<th>Location</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>CimJob6</td>
<td>CimJob</td>
<td>Completed</td>
<td>True</td>
<td>TMER4R805S30</td>
<td>update-hoststoragecache -asjob</td>
</tr>
</tbody>
</table>

**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:

```powershell
$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:

<table>
<thead>
<tr>
<th>Number</th>
<th>Friendly Name</th>
<th>OperationalStatus</th>
<th>Total Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Dell VIRTUAL DISK SCSI Disk Device</td>
<td>Online</td>
<td>67.75 GB</td>
</tr>
<tr>
<td>2</td>
<td>EQLOGIC 100E-00 Multi-Path Disk Device</td>
<td>Offline</td>
<td>500.01 GB</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Number</th>
<th>Friendly Name</th>
<th>OperationalStatus</th>
<th>Total Size</th>
<th>Partition Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>EQLOGIC 100E-00</td>
<td>Multi-Path Disk Device Offline</td>
<td>500.01 GB</td>
<td>RAW</td>
</tr>
</tbody>
</table>

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 enable 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-partition` 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:

<table>
<thead>
<tr>
<th>DriveLetter</th>
<th>FileSystemLabel</th>
<th>FileSystem</th>
<th>DriveType</th>
<th>HealthStatus</th>
<th>SizeRemaining</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>TMEVMStorage</td>
<td>NTFS</td>
<td>Fixed</td>
<td>Healthy</td>
<td>499.8 GB</td>
<td>499.9 GB</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>NTFS</td>
<td>Fixed</td>
<td>Healthy</td>
<td>54.2 GB</td>
<td>67.41 GB</td>
</tr>
<tr>
<td>D</td>
<td>CD-ROM</td>
<td></td>
<td></td>
<td></td>
<td>0 B</td>
<td>0 B</td>
</tr>
</tbody>
</table>

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 *`).

   ```powershell
   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.

   ```wmi
   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.

```
[TRMER4R805S30]: 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 "[TRMER4R805S30] PS C:\>" prompt.

<table>
<thead>
<tr>
<th>Name</th>
<th>InterfaceDescription</th>
<th>ifIndex</th>
<th>Status</th>
<th>MacAddress</th>
<th>LinkSpeed</th>
</tr>
</thead>
<tbody>
<tr>
<td>vEthernet</td>
<td>Hyper-V Virtual Ethernet Adapter #4</td>
<td>44</td>
<td>Up</td>
<td>00-10-18-...</td>
<td>10 Gbps</td>
</tr>
<tr>
<td>(vsSAN2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vEthernet</td>
<td>Hyper-V Virtual Ethernet Adapter #3</td>
<td>51</td>
<td>Up</td>
<td>00-10-18-...</td>
<td>10 Gbps</td>
</tr>
<tr>
<td>(vsSAN1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vEthernet</td>
<td>Hyper-V Virtual Ethernet Adapter #2</td>
<td>54</td>
<td>Up</td>
<td>90-B1-1C-...</td>
<td>10 Gbps</td>
</tr>
<tr>
<td>(vsLAN)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LANTeam</td>
<td>Microsoft Network Adapter Multiplexor</td>
<td>27</td>
<td>Up</td>
<td>90-B1-1C-...</td>
<td>2 Gbps</td>
</tr>
<tr>
<td>Driver</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAN2</td>
<td>Broadcom NetXtreme Gigabit Ethernet #2</td>
<td>11</td>
<td>Up</td>
<td>90-B1-1C-...</td>
<td>1 Gbps</td>
</tr>
</tbody>
</table>
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.)

<table>
<thead>
<tr>
<th>Name</th>
<th>InterfaceDescription</th>
<th>ifIndex</th>
<th>Status</th>
<th>MacAddress</th>
<th>LinkSpeed</th>
</tr>
</thead>
<tbody>
<tr>
<td>vEthernet (Migration)</td>
<td>Hyper-V Virtual Ethernet Adapter #6</td>
<td>82</td>
<td>Up</td>
<td>00-15-...</td>
<td>10 Gbps</td>
</tr>
<tr>
<td>vEthernet (Cluster)</td>
<td>Hyper-V Virtual Ethernet Adapter #5</td>
<td>78</td>
<td>Up</td>
<td>00-15-...</td>
<td>10 Gbps</td>
</tr>
<tr>
<td>vEthernet (vsSAN2)</td>
<td>Hyper-V Virtual Ethernet Adapter #4</td>
<td>44</td>
<td>Up</td>
<td>00-10-...</td>
<td>10 Gbps</td>
</tr>
<tr>
<td>vEthernet (vsSAN1)</td>
<td>Hyper-V Virtual Ethernet Adapter #3</td>
<td>51</td>
<td>Up</td>
<td>00-10-...</td>
<td>10 Gbps</td>
</tr>
<tr>
<td>vEthernet (Management)</td>
<td>Hyper-V Virtual Ethernet Adapter #2</td>
<td>54</td>
<td>Up</td>
<td>90-B1-...</td>
<td>10 Gbps</td>
</tr>
<tr>
<td>LAN1</td>
<td>Broadcom NetXtreme Gigabit Ethernet #2</td>
<td>11</td>
<td>Up</td>
<td>90-B1-...</td>
<td>1 Gbps</td>
</tr>
<tr>
<td>LAN2</td>
<td>Broadcom NetXtreme Gigabit Ethernet #2</td>
<td>11</td>
<td>Up</td>
<td>90-B1-...</td>
<td>1 Gbps</td>
</tr>
<tr>
<td>LAN1</td>
<td>Broadcom NetXtreme Gigabit Ethernet #2</td>
<td>11</td>
<td>Up</td>
<td>90-B1-...</td>
<td>1 Gbps</td>
</tr>
</tbody>
</table>
5. Set the bandwidth weight for each vNIC on the vsLAN virtual switch.
   Earlier, the vsLAN virtual switch was created using a minimum bandwidth 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>
<thead>
<tr>
<th>Network traffic</th>
<th>Adapter name</th>
<th>QOS (minimum bandwidth weight)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management</td>
<td>Management</td>
<td>5</td>
</tr>
<tr>
<td>Cluster /CSV</td>
<td>Cluster</td>
<td>20</td>
</tr>
<tr>
<td>Live migration</td>
<td>Migration</td>
<td>30</td>
</tr>
</tbody>
</table>

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.

```
[TR4MER805S30]: PS C:\> get-vmnetworkadapter -managementOS
```

Sample output:

<table>
<thead>
<tr>
<th>Name</th>
<th>IsManagementOs</th>
<th>VMName</th>
<th>SwitchName</th>
<th>MacAddress</th>
<th>Status</th>
<th>IPAddresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migration</td>
<td>True</td>
<td></td>
<td>vsLAN</td>
<td>00155D043257</td>
<td>{Ok}</td>
<td></td>
</tr>
<tr>
<td>Cluster</td>
<td>True</td>
<td></td>
<td>vsLAN</td>
<td>00155D043256</td>
<td>{Ok}</td>
<td></td>
</tr>
<tr>
<td>vsSAN2</td>
<td>True</td>
<td></td>
<td>vsSAN2</td>
<td>0010183D8BC6</td>
<td>{Ok}</td>
<td></td>
</tr>
<tr>
<td>vsSAN1</td>
<td>True</td>
<td></td>
<td>vsSAN1</td>
<td>0010183D8BA0</td>
<td>{Ok}</td>
<td></td>
</tr>
<tr>
<td>Management</td>
<td>True</td>
<td></td>
<td>vsLAN</td>
<td>002219D25DC2</td>
<td>{Ok}</td>
<td></td>
</tr>
</tbody>
</table>

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

```
[TR4MER805S30]: 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.

![Diagram](image)

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:


### 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](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).

   ```powershell
   [TRMER4R805S30]: PS C:\> get-netadapter | where-object {$_._status -eq "up"} | get-netipconfiguration
   ```

<table>
<thead>
<tr>
<th>InterfaceAlias</th>
<th>vEthernet (vsSAN2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>InterfaceIndex</td>
<td>35</td>
</tr>
<tr>
<td>InterfaceDescription</td>
<td>Hyper-V Virtual Ethernet Adapter #4</td>
</tr>
<tr>
<td>NetProfile.Name</td>
<td>Unidentified network</td>
</tr>
<tr>
<td>IPv4Address</td>
<td>10.10.6.211</td>
</tr>
<tr>
<td>IPv6DefaultGateway</td>
<td></td>
</tr>
<tr>
<td>IPv4DefaultGateway</td>
<td></td>
</tr>
<tr>
<td>DNSServer</td>
<td>fec0:0:0:ffff::1</td>
</tr>
<tr>
<td></td>
<td>fec0:0:0:ffff::2</td>
</tr>
<tr>
<td></td>
<td>fec0:0:0:ffff::3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>InterfaceAlias</th>
<th>vEthernet (vsSAN1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>InterfaceIndex</td>
<td>33</td>
</tr>
</tbody>
</table>
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.
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.
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.