Upgrading Your Skills to MCSA: Windows Server 2016

Exam Ref 70-743

Charles Pluta

Microsoft

Exam Ref 70-743 Upgrading Your Skills to MCSA: Windows Server 2016

Charles Pluta

Exam Ref 70-743 Upgrading Your Skills to MCSA: Windows Server 2016

Published with the authorization of Microsoft Corporation by:
Pearson Education, Inc.

ISBN-13: 978-0-7356-9743-0

ISBN-10: 0-7356-9743-4

Library of Congress Control Number: 2016959957

First Printing December 2016

Editor-in-Chief	Greg Wiegand
Acquisitions Editor	Trina MacDonald
Development Editor	Rick Kughen
Managing Editor	Sandra Schroeder
Senior Project Editor	Tracey Croom
Editorial Production	Backstop Media, Troy Mott
Copy Editor	Jordan Severn
Indexer	Julie Grady
Proofreader	Christina Rudloff
Technical Editor	Ron Handlon
Cover Designer	Twist Creative, Seattle

Contents at a glance

This page intentionally left blank

Contents

This page intentionally left blank

Introduction

With each release of Windows Server, more and more features are added or modified that makes knowing the product inside and out more and more difficult. The 70-743 exam "Upgrading your skills to Windows Server 2016" is for administrators that have previously achieved the MCSA certification for Windows Server 2008, or Windows Server 2012, and plan to achieve the latest certification offering.

Understanding that the exam is geared specifically towards administrators with existing knowledge, this Exam Ref book assumes you remember and know the knowledge that is necessary to pass the previous versions of the exam. Therefore, we focus solely on the skills that are measured in the 70-743 exam, sometimes skipping the basics of the skill. A lot of these skills build on the knowledge you've retained from Windows Server 2008 or Windows Server 2012. However, some of the skills are brand new to Windows Server 2016, and are expected to be highlighted on the exam.

The goal of this book is to act as a reference to give you the tools and knowledge that you need to succeed in passing the exam. While we cover every skill that the exam measures and focus on real-world examples of how to use the technologies that are listed, there is no way of guaranteeing that you will pass the exam simply by using this book. As you are well aware as an existing MCSA credential holder, nothing is better than getting hands-on experience with each of the roles and features in Windows Server 2016 before taking the exam. It is recommended that you use the information in this book, combined with a hands-on approach of trying each role or feature discussed by using both graphical and Windows PowerShell (or command-line) tools. This will ensure that you have the best opportunity to succeed when taking the exam.

This book covers every major topic area found on the exam, but it does not cover every exam question. Only the Microsoft exam team has access to the exam questions, and Microsoft regularly adds new questions to the exam, making it impossible to cover specific questions. You should consider this book a supplement to your relevant real-world experience and other study materials. If you encounter a topic in this book that you do not feel completely comfortable with, use the "Need more review?" links you'll find in the text to find more information and take the time to research and study the topic. Great information is available on MSDN, TechNet, MVA, and in blogs and forums.

Organization of this book

This book is organized by the "Skills measured" list published for the exam. The "Skills measured" list is available for each exam on the Microsoft Learning website: *https://aka.ms/examlist*. Each chapter in this book corresponds to a major topic area in the list, and the technical tasks in each topic area determine a chapter's organization. If an exam covers six major topic areas, for example, the book will contain six chapters.

Microsoft certifications

Microsoft certifications distinguish you by proving your command of a broad set of skills and experience with current Microsoft products and technologies. The exams and corresponding certifications are developed to validate your mastery of critical competencies as you design and develop, or implement and support, solutions with Microsoft products and technologies both on-premises and in the cloud. Certification brings a variety of benefits to the individual and to employers and organizations.

MORE INFO **ALL MICROSOFT CERTIFICATIONS**

For information about Microsoft certifications, including a full list of available certifications, go to *https://www.microsoft.com/learning*.

Acknowledgments

Charles Pluta I would like to thank my wife Jen for her love and support throughout all of my projects. I would also like to thank Greg Baker for giving me the opportunity to succeed from the beginning of my career. I would also like to thank Brian Svidergol, Elias Mereb, and Mike Corkery, who have provided their continued friendship and technical expertise throughout the years. Finally, I would like to thank Trina, Troy, and all of the editors and reviewers behind the scenes that dedicated their time to making this book happen.

Free ebooks from Microsoft Press

From technical overviews to in-depth information on special topics, the free ebooks from Microsoft Press cover a wide range of topics. These ebooks are available in PDF, EPUB, and Mobi for Kindle formats, ready for you to download at:

https://aka.ms/mspressfree

Check back often to see what is new!

Microsoft Virtual Academy

Build your knowledge of Microsoft technologies with free expert-led online training from Microsoft Virtual Academy (MVA). MVA offers a comprehensive library of videos, live events, and more to help you learn the latest technologies and prepare for certification exams. You'll find what you need here:

https://www.microsoftvirtualacademy.com

Quick access to online references

Throughout this book are addresses to webpages that the author has recommended you visit for more information. Some of these addresses (also known as URLs) can be painstaking to type into a web browser, so we've compiled all of them into a single list that readers of the print edition can refer to while they read.

Download the list at *https://aka.ms/examref743/downloads*.

The URLs are organized by chapter and heading. Every time you come across a URL in the book, find the hyperlink in the list to go directly to the webpage.

Errata, updates, & book support

We've made every effort to ensure the accuracy of this book and its companion content. You can access updates to this book—in the form of a list of submitted errata and their related corrections—at:

https://aka.ms/examref743/errata

If you discover an error that is not already listed, please submit it to us at the same page.

If you need additional support, email Microsoft Press Book Support at *mspinput@microsoft.com*.

Please note that product support for Microsoft software and hardware is not offered through the previous addresses. For help with Microsoft software or hardware, go to *https://support.microsoft.com*.

We want to hear from you

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

https://aka.ms/tellpress

We know you're busy, so we've kept it short with just a few questions. Your answers go directly to the editors at Microsoft Press. (No personal information will be requested.) Thanks in advance for your input!

Stay in touch

Let's keep the conversation going! We're on Twitter: *http://twitter.com/MicrosoftPress*.

Important: How to use this book to study for the exam

Certification exams validate your on-the-job experience and product knowledge. To gauge your readiness to take an exam, use this Exam Ref to help you check your understanding of the skills tested by the exam. Determine the topics you know well and the areas in which you need more experience. To help you refresh your skills in specific areas, we have also provided "Need more review?" pointers, which direct you to more in-depth information outside the book.

The Exam Ref is not a substitute for hands-on experience. This book is not designed to teach you new skills.

We recommend that you round out your exam preparation by using a combination of available study materials and courses. Learn more about available classroom training at *https://www.microsoft.com/learning*. Microsoft Official Practice Tests are available for many exams at *https://aka.ms/practicetests*. You can also find free online courses and live events from Microsoft Virtual Academy at *https://www.microsoftvirtualacademy.com*.

This book is organized by the "Skills measured" list published for the exam. The "Skills measured" list for each exam is available on the Microsoft Learning website: *https://aka.ms/examlist*.

Note that this Exam Ref is based on this publicly available information and the author's experience. To safeguard the integrity of the exam, authors do not have access to the exam questions.

This page intentionally left blank

CHAPTER 4

Implement Windows Containers

In this chapter we cover how to use containers to host virtualized images on a server. Containers are supported on both Windows Server and Hyper-V, however, the way they act and respond are slightly different. Containers can be isolated to ensure they operate independently of any other container or host that they are running on. In the first section of this chapter, we cover the basic process to deploy containers and go through the basic management aspects for containers.

Skills in this chapter:

- Deploy Windows Containers
- Manage Windows Containers

Skill 4.1: Deploy Windows Containers

In this section we outline the basics for deploying containers on either Windows Server, Nano Server, or Hyper-V. We also detail how to change the Docker daemon configuration for startup, and detail specifics for images, such as tagging.

This section covers how to:

- Determine installation requirements and appropriate scenarios for Windows Containers
- Install and configure Windows Server container host in physical or virtualized environments
- Install and configure Windows Server container host to Windows Server Core or Nano Server in a physical or virtualized environment
- Install Docker on Windows Server and Nano Server
- Configure Docker daemon start-up options
- Install a base operating system
- Tag an image
- Uninstall an operating system image
- Create Windows Server containers
- Create Hyper-V containers

Determine installation requirements and appropriate scenarios for Windows Containers

Windows Containers is a new feature that is only available on Windows Server 2016, Nano Server, and Windows 10 Professional and Enterprise Anniversary Update editions. If you plan on using Hyper-V containers, then the Hyper-V role must also be installed on the computer or server. To use Windows Containers, the operating system must be installed as the C drive. If you plan to only use Hyper-V containers, then the operating system can be installed on any drive.

From a physical aspect, Windows Containers with Hyper-V requires nested virtualization. Nested virtualization has the following requirements:

- At least 4 GB of RAM for the Hyper-V host
- A processes that uses Intel VT-x

Also, the container host VM must have at least two virtual processors and dynamic memory must be disabled. As of this writing, Windows Server 2016 offers two container images: Server Core and Nano Server. If the host operating system is a Nano Server, then only the Nano Server image is available.

Install and configure containers

For the purpose of preparing for the exam, we've combined two of the listed skills:

- Install and configure Windows Server container host in physical or virtualized environments
- Install and configure Windows Server container host to Windows Server Core or Nano Server in a physical or virtualized environment

For either host's operating system, whether it is physical or virtual, containers is listed as a Windows Feature. For servers with a GUI, it can be installed from the Add Roles and Features wizard. Containers can also be installed by using Windows PowerShell by using the Install-WindowsFeature cmdlet. For example:

Figure 4-1 shows installing the Containers feature by using the Install-WindowsFeature cmdlet.

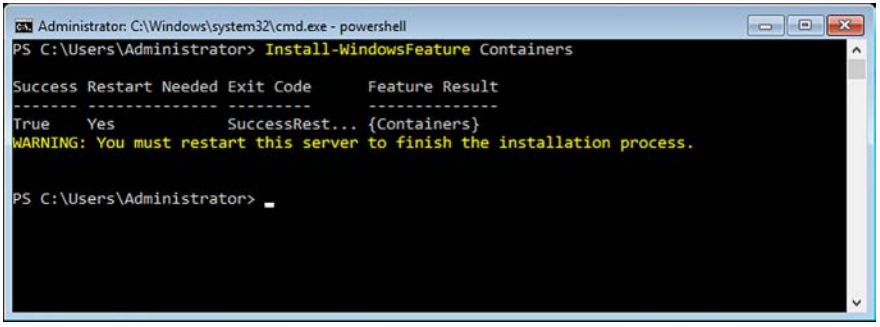

FIGURE 4-1 Install-WindowsFeature

```
Install-WindowsFeature Containers
```

If you're using Nano Server, you must first install the Nano Server Package, and then install the Container Feature. For example:

```
Install-PackageProvider NanoServerPacakage

Install-NanoServerPackage -Name Microsoft-NanoServer-Containers-Package
```

Install Docker on Windows Server and Nano Server

To manage containers on either Windows Server 2016 or Nano Server, you must also install the Docker service. Most all Docker installation and configuration options have both a PowerShell cmdlet or a Docker command line option. To install Docker on Windows Server 2016, it must be downloaded from the Docker website. You can do this manually, or by using PowerShell. For example:

Figure 4-2 shows downloading and configuring the environment for the docker service to run.

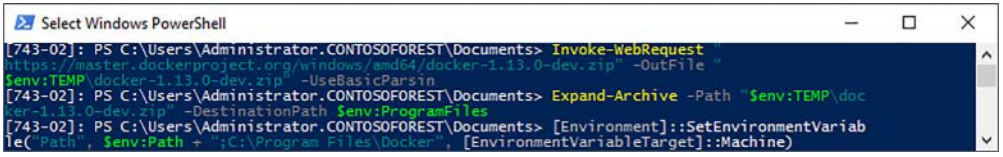

FIGURE 4-2 Obtaining docker

```
Invoke-WebRequest "https://aka.ms/tp5/b/dockerd" -OutFile "$env:TEMP\docker-1.12.0.zip"
-UseBasicParsing

Expand-Archive -Path "$env:TEMP\docker-1.12.0.zip" -DestinationPath $env:ProgramFiles

[Environment]::SetEnvironmentVariable("Path", $env:Path + ";C:\Program Files\Docker",
[EnvironmentVariableTarget]::Machine)

& $env:ProgramFiles\docker\dockerd.exe --register-service

Start-Service Docker

docker tag windowsservercore:10.0.14300.1000 windowsservercore:latest
```

> ***NOTE***
>
> **The Invoke-WebRequest command in this example specifically uses Technical Preview 5, which was available at the time of writing. Locate the latest version that is available by using the Docker website before using this command in a lab environment.**

After the installation is complete, run the **docker info** command. A portion of the output is shown in Figure 4-3.

FIGURE 4-3 Docker info

The above example is broken down like this:

1. First, the Docker engine and client is downloaded from the Docker website.
2. Then, the code extracts the compressed folder into the Program Files directory.
3. The path is set as a system variable, and the service is created and started.
4. Finally, the Docker image must be tagged with the version "latest."

For installing Docker on Nano Server, the same overall process must be followed. However, Nano Server does not currently support the Invoke-WebRequest cmdlet. Therefore, you must manually download the Docker files and copy them to the Nano Server operating system. From there you can set the environment variable, create the service, and then start the service. For Nano Server, you must also enable the FPS-SMB-In-TCP firewall rule. For example:

```
Set-NetFirewallRule -Name FPS-SMB-In-TCP -Enabled True
```

Configure Docker daemon start-up options

Docker is configured by using a daemon.json file, which is located in the installation path of the directory. When using Docker on Windows Server 2016, only a subset of the configuration options is available. When creating the JSON file, only the necessary configuration changes need to be included in the file. For example, to configure the Docker Engine to accept connections on port 2375, add the following to the daemon.json file:

```
{

    "hosts": ["tcp://0.0.0.0:2375"]

}
```

You can also configure Docker by using the `sc config` command. When using `sc config`, you are modifying the Docker Engine configuration flags directly on the Docker service. For example:

Figure 4-4 shows running the sc command to modify the docker service.

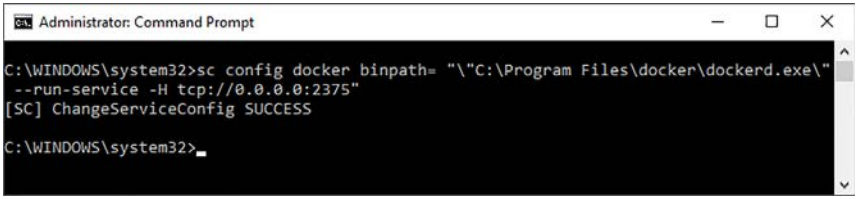

FIGURE 4-4 Service configuration

```
sc config docker binpath= "\"C:\Program Files\docker\dockerd.exe\" --run-service -H
tcp://0.0.0.0:2375"
```

> ***NEED MORE REVIEW?*** **DOCKER DAEMON**
>
> For more information on configuring the Docker daemon, visit *https://msdn.microsoft.com/en-us/virtualization/windowscontainers/docker/configure_docker_daemon*.

Install a base operating system

Before you can deploy a container, you must download a base operating system image. The procedure is the same whether you plan to manage Server Core or Nano Server base images. Obtaining the image is accomplished by running two PowerShell cmdlets: Install-PackageProvider, and Install-ContainerImage. For example:

```
Install-PackageProvider ContainerImage -Force

Install ContainerImage -Name WindowsServerCore
```

This process might take a few minutes because it downloads the Server Core container image. After installing the image, you need to restart the Docker service. For example:

```
Restart-Service Docker
```

You can also use the `docker` command to download the base image. For example:

```
docker pull microsoft/windowsservercore
```

After downloading the images, you can also view the downloaded images with the `docker` command. For example:

```
docker images
```

Figure 4-5 shows the results of downloading the images and how they are displayed after being downloaded.

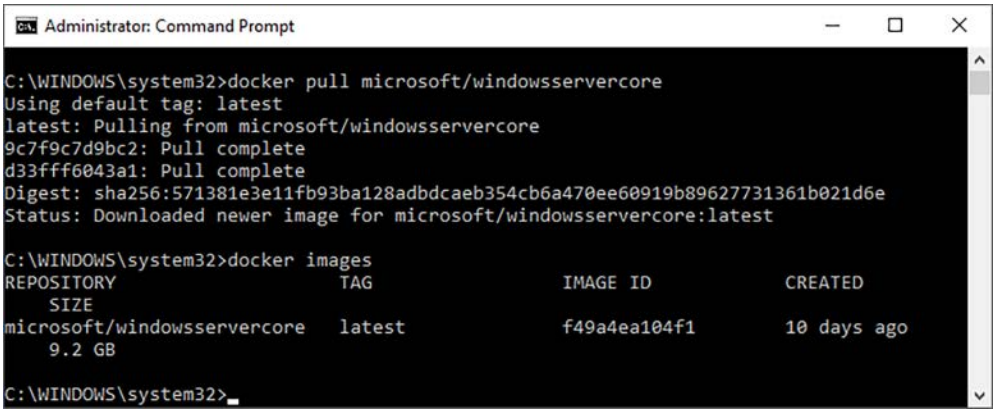

FIGURE 4-5 Obtaining images

Tag an image

When you download an image into the repository, you must also assign a tag to the image. Tagging an image enables you to set a version on the image, which is useful if you plan to have multiple versions. Microsoft suggests after downloading an image, to tag it at the "latest." For example:

```
docker tag windowsservercore:10.0.14300.1000 windowsservercore:latest
```

The Docker tag can contain upper and lowercase characters, digits, underscores, periods, and dashes. However, the tag cannot start with a period or dash, and can be up to 128 characters.

> ***NEED MORE REVIEW?*** **DOCKER TAG**
>
> **For more information on using the Docker tag, visit *https://docs.docker.com/engine/reference/commandline/tag/*.**

Uninstall an operating system image

As we have mentioned, most actions when using Docker can be completed by using PowerShell or the Docker daemon. To uninstall a container image from the repository, use the Uninstall-ContainerOSImage cmdlet. For example:

```
Uninstall-ContainerOSImage -FullName CN=Microsoft_NanoServer_10.0.14304.1003
```

Create Windows Server containers

You can deploy a container by using the Docker daemon. One of the first tasks you might need to do is view a list of the available container images. For example, the following command returns a list of available Microsoft images:

```
docker search Microsoft
```

A portion of the output is included for reference:

```
NAME                                    DESCRIPTION

microsoft/aspnet                        ASP.NET is an open source server-side Web ...

microsoft/dotnet                        Official images for working with .NET Core...

mono                                    Mono is an open source implementation of M...

microsoft/azure-cli                     Docker image for Microsoft Azure Command L...

microsoft/iis                           Internet Information Services (IIS) instal...
```

Therefore, if you want to use the ASP.NET image, use the Docker daemon to pull the image:

```
docker pull microsoft/aspnet
```

Create Hyper-V containers

Windows Server containers and Hyper-V containers are created and managed, and are functionally identical. Both types of containers also use the same container images. The difference between a Windows Server container and a Hyper-V container is the level of isolation that is present to the host, or other containers on that host. The first difference is that when creating the container, specify the `--isolation=hyperv` parameter.

```
docker run -it --isolation=hyperv nanoserver cmd
```

To demonstrate the isolation of a Hyper-V container, assume that a Windows Server container has been deployed. You start a running ping on the container.

```
docker run -d windowsservercore ping localhost -t
```

If you use the docker daemon, you can view the task thread that is running the ping.

```
docker top windowservercore

4369 ping
```

In this example, the process ID within the container is 4369. Within the container, you can also view the thread.

```
get-process -Name ping
```

The following output is returned:

```
Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id  SI ProcessName

-------  ------    -----      ----- -----   ------     --  -- -----------

     67       5      820       3836 ...71     0.03   4369   3 PING
```

If you follow the same process when using a Hyper-V container, you receive a different end result. You can create and view the process from the host, using the Docker daemon.

```
docker run -d --isolation=hyperv nanoserver ping -t localhost

docker top nanoserver

2371 ping
```

However, the difference is when trying to view the process on the container host.

```
Get-process -Name ping

Get-Process : Cannot find a process with the name "ping". Verify the process name and
call the cmdlet again.

At line:1 char:1

+ Get-Process -Name ping

+ ~~~~~~~~~~~~~~~~~~~~

    + CategoryInfo          : ObjectNotFound: (ping:String) [Get-Process],
ProcessCommandException

    + FullyQualifiedErrorId : NoProcessFoundForGivenName,Microsoft.PowerShell.Commands.
GetProcessCommand
```

The difference is in the process name. By using a Hyper-V container, the process is run by the vmwp process. The vmwp process is the virtual machine process on the host, and is protecting the process from the host operating system.

```
Get-Process -Name vmwp

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id  SI ProcessName

-------  ------    -----      ----- -----   ------     --  -- -----------

   1737      15    39452      19620 ...61     5.55   2376   0 vmwp
```

Skill 4.2: Manage Windows Containers

In this section, we outline how to manage containers after they have been deployed. This includes using the Docker daemon to manage images, as well as using Windows PowerShell. We also cover configuring port mapping and networking options for use with Windows Containers.

This section covers how to:

- Manage Windows or Linux containers using the Docker daemon
- Manage Windows or Linux containers using Windows PowerShell
- Manage container networking
- Manage container data volumes
- Manage Resource Control
- Create new container images using Dockerfile
- Manage container images using DockerHub repository for public and private scenarios
- Manage container images using Microsoft Azure

Manage Windows or Linux containers using the Docker daemon

After you have downloaded the image type that you plan to use, you can use the daemon to identify the images that have been downloaded.

```
docker images
```

The following output is returned:

```
REPOSITORY            TAG             IMAGE ID          CREATED             SIZE

microsoft/aspnet      latest          accd044753c1      11 days ago         7.907 GB
```

You can also deploy a container by using the Docker daemon.

```
docker run -d -p 80:80 microsoft/iis ping -t localhost
```

Creating a new image can be performed by using the Docker daemon with the `commit` parameter.

```
docker commit 475059caef8f windowsservercoreiis
```

Removing an image is performed by using the Docker daemon with the `rmi` parameter. However, if any other container depends on the image that you are trying to remove, the command fails. The `rmi` parameter accepts either the image name or the ID of the image.

```
docker rmi windowsservercoreiis
```

To view the list of dependencies with Docker, use the `history` parameter.

```
docker history windowsservercoreiis
```

The following output is returned:

```
IMAGE               CREATED             CREATED BY          SIZE                COMMENT

2236b49aaaef        3 minutes ago       cmd                 171.2 MB

6801d964fda5        2 weeks ago                             0 B
```

Manage Windows or Linux containers using Windows PowerShell

As of this writing, the PowerShell for Docker module is in development. The team writing the module has adopted the Microsoft Open Source Code of Conduct, and welcomes contributions to the project in the form of bugs, suggestions, proposals, and pull requests through the Github repository. The project is available on Github here: *https://github.com/Microsoft/Docker-PowerShell/*.

The PowerShell module for Docker is simply an alternative to the Docker daemon. You can use the module as a replacement for, or in conjunction with, the Docker daemon. The PowerShell module can target any operating system that is running the Docker engine on both Windows and Linux.

To compile the project, you need to obtain the .NET Core SDK, and the .NET SDKs for versions 4.5 and 4.6. The Docker endpoint that you are planning to connect to must support the API version 1.24.

The latest release version of Docker can also be downloaded from GitHub here: *https://github.com/Microsoft/Docker-PowerShell/releases*. Download and extract the compressed folder, and then use the Import-Module cmdlet, pointing to the extracted folder. This makes the Docker cmdlets available on the computer.

Manage container networking

Container networks are similar to virtual networks through Hyper-V. Each container has a virtual network adapter that is connected to a virtual switch. To force isolation between containers that are running on the same host, compartments are created for each container. A Windows Server host uses Host vNICs to attach to the virtual, while Hyper-V containers use a synthetic VM NIC to attach to the virtual switch.

Containers support four different networking modes:

- **Network Address Translation (NAT)** Each container receives an IP address from a private address pool. Port forwarding or mapping can be configured to transmit data from the host to the container.
- **Transparent** Each container endpoint has a direct connection to the physical network that the host is using. The IP address range that is being used on the physical network can be used on the container either as a static address or dynamically assigned.
- **L2 Bridge** Each container endpoint is in the same subnet as the host that is running it. The container IP address is assigned statically from the same prefix as the host. All container endpoints on the host use the same MAC address.
- **L2 Tunnel** This mode should only be used in a Microsoft Cloud Stack.

By default, the Docker engine creates an NAT network when the Docker service runs for the first time. The default network that is used is 172.16.0.0/12. You can customize the network prefix used by modifying the daemon.json configuration file. The endpoints in the container are attached to this network and assigned an IP address from the private network. Table 4-1 outlines connections for a single-host environment.

TABLE 4-1 Single host connection types

Single host	Container to container	Container to external
NAT	Connects using Hyper-V Virtual Switch	Routed through WinNAT with address translation
Transparent	Connects using Hyper-V Virtual Switch	Direct access to physical network
L2 Bridge	Connects through Hyper-V Virtual Switch	Access to physical network by using MAC address translation

Additionally, Table 4-2 outlines the connections for a multi-host environment.

TABLE 4-2 Multi-host connection types

Multi-host	container to Container	Container to external
NAT	References external container host IP and port, routed through WinNAT with translations	References external container host IP and port, routed through WinNAT with translations
Transparent	Directly references container IP endpoint	Direct access to physical network
L2 Bridge	Directly references container IP endpoint	Access to physical network by using MAC address translation

NAT networks

By default, when an endpoint is created, it connects to the NAT network. To specify the network that a container should attach to, use the `--network` parameter.

```
docker run -it --network=NatNetwork <image>
```

To access any applications that run within a container, you need to map the ports from the host to the endpoint.

```
docker run -it -p 80:80 <image>

docker run -it -p 8082:80 windowsservercore cmd
```

The first command creates a port map between TCP port 80 on the host to TCP port 80 of the container endpoint. The second command uses port 8082 on the host, and forwards it to port 80 of the endpoint.

EXAM TIP

Port mapping must either be configured when the endpoint is created, or when the endpoint is in a STOPPED state. You cannot modify container port mapping while the endpoint is running.

Transparent networks

To use a transparent network, you must first create the network.

```
docker network create -d transparent TransparentNetwork
```

If the container host is virtualized, and you need to use DHCP for the IP address assignment, then you must also use MAC address spoofing on the VM network adapter. Without MAC address spoofing, the Hyper-V host blocks the network traffic from the containers in the VM with identical MAC addresses.

```
Get-VMNetworkAdapter -VMName ContainerHost | Set-VMNetworkAdapter -MacAddressSpoofing On
```

L2 Bridge networks

To use a L2 Bridge network, you must create a container network that uses the driver named l2bridge. The subnet and gateway for the network must also be specified when creating the object.

```
docker network create -d l2bridge --subnet=10.10.0.0/16 --gateway=10.10.0.1
BridgeNetwork
```

EXAM TIP

When using an L2 Bridge network type, only static IP addresses are supported.

Options for all network types

You can use the Docker daemon to list the available networks.

```
docker network ls
```

The following output is returned:

```
NETWORK ID          NAME                DRIVER              SCOPE

0a297065f06a        nat                 nat                 local

d42516aa0250        none                null                local
```

To remove a network, use the network rm parameter.

```
docker network rm "nat"
```

Figure 4-6 displays the networks on a docker host.

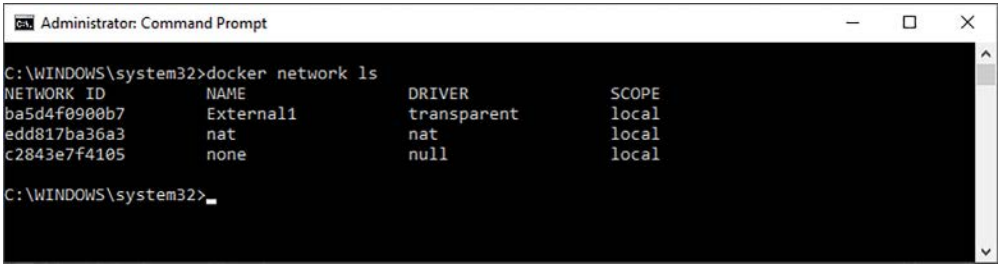

FIGURE 4-6 Listing networks

Manage container data volumes

Data volumes are storage locations that are visible to both the container host and the container endpoint. The data that is in the volume can be shared between the two systems, as well as with other containers on the same host. Creating a new volume is part of the run parameter with the Docker daemon.

```
docker run -it -v c:\volume1 windowsservercore cmd
```

By default, new data volumes are created in C:\ProgramData\Docker\Volumes on the container host. In the command, the C:\Volume1 indicates that the volume is be accessible within the container endpoint at that path.

After you have created a volume, to mount it to a different container, specify the source and destination paths using the same parameters:

```
docker run -it -v c:\source:c:\destination windowsservercore cmd
```

You can also pass-through a single file from the container host to the endpoint. The syntax is basically the same as specifying an existing volume.

```
docker run -it -v c:\container-share\config.ini windowsservercore cmd
```

Similarly, you can also mount a full drive from the container host to the endpoint. Note that when mounting a full drive, a backslash is not included with the drive letter.

```
docker run -it -v d: windowsservercore cmd
```

Finally, data volumes can be inherited from other endpoints using the **--**volumes-from switch in the run parameter. This is useful if the applications in multiple containers are sharing the same data.

```
docker run -it --volumes-from Volume1 windowsservercore cmd
```

Manage resource control

Docker includes the ability to manage the CPU, disk IO, network, and memory consumption that an endpoint consumes. This ensures that you are able to manage the container host resources efficiently, as well as ensuring that you maximize the performance of all services running on a host.

By default, the CPU is divided equally among all endpoints running on a container host. To change the share that an endpoint has, use the **--**cpu-shares switch with the run parameter. The **--**cpushares parameter accepts a value between 1 and 10000. The default weight of all endpoints is 5,000.

```
docker run -it --cpu-shares 2 --name dockerdemo windowsservercore cmd
```

NEED MORE REVIEW? **CPU RESOURCES**

For more information on managing CPU resources for an endpoint, visit *https://docs.docker.com/engine/reference/run/#cpu-share-constraint*.

Create new container images using Dockerfile

You can use Docker to automatically build images by reading the instructions that are placed in a Dockerfile. A Dockerfile is a text document that lists the commands that you would use in the CLI to create an image manually. After creating the Dockerfile, use the `build` parameter with the Docker daemon to automatically create the image.

```
docker build -f C:\Dockerfile .
```

The Docker daemon commits each line of the file one by one before outputting the image ID of for the endpoint that you have created.

NEED MORE REVIEW? **DOCKERFILE DETAILS**

For more information on Dockerfile, visit *https://docs.docker.com/engine/reference/builder/*.

Manage container images using Docker Hub repository for public and private scenarios

The Docker Hub is a repository that contains pre-built images. These images can be downloaded onto a host and used in a development or production environment. These images can also be used as a base for Windows container applications. To retrieve a list of the available images in the Docker Hub, use the search parameter with the Docker daemon:

```
docker search *
```

The following output is returned:

```
NAME                      DESCRIPTION                                     STARS
OFFICIAL   AUTOMATED

microsoft/sample-django   Django installed in a Windows Server Core ...   1
[OK]

microsoft/dotnet35        .NET 3.5 Runtime installed in a Windows Se...   1         [OK]
[OK]

microsoft/sample-golang   Go Programming

…
```

Downloading an image from the Docker Hub is the same as retrieving a base image. Use the `pull` parameter with the Docker daemon:

```
docker pull microsoft/aspnet
```

The following output is returned:

```
Using default tag: latest

latest: Pulling from microsoft/aspnet

f9e8a4cc8f6c: Pull complete



b71a5b8be5a2: Download complete
```

After downloading the image, it is available when viewing the images through the Docker daemon.

```
docker images
```

The following output is returned:

```
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL
SIZE

microsoft/aspnet    latest              b3842ee505e5        5 hours ago         101.7 MB
```

To upload an image to the Docker Hub, use the push parameter with the Docker daemon. First, you must login with your Docker ID to access the Hub.

```
docker login
```

The following output is returned:

```
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a
Docker ID, head over to https://hub.docker.com to create one.

Username: username

Password:
```

```
Login Succeeded

docker push username/containername


The push refers to a repository [docker.io/username/containername]

4341be770beb: Pushed

fed398573696: Pushed

latest: digest: sha256:ae3a2971628c04d5df32c3bbbfc87c477bb814d5e73e2787900da13228676c4f
size: 2410
```

Manage container images using Microsoft Azure

You can use Docker on Microsoft Azure a few different ways:

- Deploy container hosts using the Docker Machine Azure driver
- Use the Docker VM Extension on Azure VMs
- Use the Docker VM Extension with Docker Compose
- Deploy a Docker Swarm cluster on Azure Container Services

The Azure Docker VM Extension installs and configures the Docker daemon, client, and Docker Compose on a Linux VM in Azure. This enables you to define and deploy container applications using Docker Compose and Docker Machine.

Combine the extension with the Azure Resource Manager, and you can create and deploy templates for almost all aspects of your Azure environment.

NEED MORE REVIEW? **DOCKER VM EXTENSION**

For more information on the Docker VM Extension, visit *https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-linux-dockerextension/*.

Chapter summary

- The basics of using containers to run virtualized images.
- How to install Docker on Windows Server and Nano Server
- How to configure the start-up options for the Docker daemon
- Performing a base operating system install
- Tagging an image for use with containers
- Creating containers for both Windows Server and Hyper-V
- Managing containers using the Docker daemon and Windows PowerShell
- Creating NAT, Transparent, and L2 Bridge networks for containers
- Creating and managing data volumes for use by multiple container endpoints
- Managing container host resources using Resource Control
- Automating the build process for an image using Dockerfile
- Using the Azure VM Extension with Docker

Thought Experiment

A company is testing containers and images in their development environment. They have installed the Docker engine on a Windows Server host, and deployed a base image connected to the default network. The company would like the images to connect directly to the physical network. They also plan to automate the creation of future images and store them in the Docker Hub.

Using this information, answer the following questions:

1. What should be modified to configure the Docker daemon startup options?
2. Which network is the image that has been deployed connected to?
3. What type of network must the company create to achieve the goal?
4. What type of file does the Dockerfile need to be?
5. Which Docker daemon command is used to store images in the Docker Hub repository?

Thought Experiment Answers

1. The JSON configuration file should be created or modified to change the startup options of the Docker daemon.
2. By default, images connect to a default NAT network.
3. A transparent network must be created to enable the images to connect directly to the physical network.
4. The Dockerfile script is a plain-text file that contains the actions to create an image.
5. The `docker push` command uploads the specified image to the Docker Hub after logging into the service.

This page intentionally left blank

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W