



# Installation, Storage and Compute with Windows Server 2016

Exam Ref

70-740

Craig Zacker

FREE SAMPLE CHAPTER

SHARE WITH OTHERS



# **Exam Ref 70-740 Installation, Storage and Compute with Windows Server 2016**

Craig Zacker

## **Exam Ref 70-740 Installation, Storage, and Compute with Windows Server 2016**

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

**Copyright © 2017 by Craig Zacker**

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, request forms, and the appropriate contacts within the Pearson Education Global Rights & Permissions Department, please visit [www.pearsoned.com/permissions/](http://www.pearsoned.com/permissions/). No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

ISBN-13: 978-0-7356-9882-6

ISBN-10: 0-7356-9882-1

Library of Congress Control Number: 2016962646

First Printing January 2017

### **Trademarks**

Microsoft and the trademarks listed at <https://www.microsoft.com> on the “Trademarks” webpage are trademarks of the Microsoft group of companies. All other marks are property of their respective owners.

### **Warning and Disclaimer**

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an “as is” basis. The authors, the publisher, and Microsoft Corporation shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or programs accompanying it.

### **Special Sales**

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at [corpsales@pearsoned.com](mailto:corpsales@pearsoned.com) or (800) 382-3419.

For government sales inquiries, please contact [governmentsales@pearsoned.com](mailto:governmentsales@pearsoned.com).

For questions about sales outside the U.S., please contact [intlcs@pearson.com](mailto:intlcs@pearson.com).

<b>Editor-in-Chief</b>	Greg Wiegand
<b>Acquisitions Editor</b>	Trina MacDonald
<b>Development Editor</b>	Rick Kughen
<b>Managing Editor</b>	Sandra Schroeder
<b>Senior Project Editor</b>	Tracey Croom
<b>Editorial Production</b>	Backstop Media
<b>Copy Editor</b>	Christina Rudloff
<b>Indexer</b>	Julie Grady
<b>Proofreader</b>	Christina Rudloff
<b>Technical Editor</b>	Ajay Kakkar
<b>Cover Designer</b>	Twist Creative, Seattle

# Contents at a glance

	<i>Introduction</i>	<i>xv</i>
	<i>Preparing for the exam</i>	<i>xix</i>
<b>CHAPTER 1</b>	<b>Install Windows Servers in host and compute environments</b>	<b>1</b>
<b>CHAPTER 2</b>	<b>Implement storage solutions</b>	<b>81</b>
<b>CHAPTER 3</b>	<b>Implement Hyper-V</b>	<b>165</b>
<b>CHAPTER 4</b>	<b>Implement Windows containers</b>	<b>259</b>
<b>CHAPTER 5</b>	<b>Implement high availability</b>	<b>297</b>
<b>CHAPTER 6</b>	<b>Maintain and monitor server environments</b>	<b>387</b>
	<i>Index</i>	<i>445</i>

*This page intentionally left blank*

# Contents

<b>Introduction</b>	<b>xv</b>
Organization of this book	.xvi
Microsoft certifications	.xvi
Free ebooks from Microsoft Press	.xvi
Microsoft Virtual Academy	.xvi
Quick access to online references	xvii
Errata, updates, & book support	xvii
We want to hear from you	xvii
Stay in touch	xvii
<i>Preparing for the exam</i>	<i>xix</i>

<b>Chapter 1</b>	<b>Install Windows Servers in host and compute environments</b>	<b>1</b>
	Skill 1.1: Install, upgrade, and migrate servers and workloads	1
	Determine Windows Server 2016 installation requirements	2
	Determine appropriate Windows Server 2016 editions per workloads	4
	Install Windows Server 2016	6
	Install Windows Server 2016 features and roles	11
	Install and configure Windows Server Core	17
	Manage Windows Server Core installations using Windows PowerShell, command line, and remote management capabilities	21

Implement Windows PowerShell Desired State Configuration (DSC) to install and maintain integrity of installed environments	26
Perform upgrades and migrations of servers and core workloads from Windows Server 2008 and Windows Server 2012 to Windows Server 2016	27
Determine the appropriate activation model for server installation	35
Skill 1.2: Install and configure Nano Server	42
Determine appropriate usage scenarios and requirements for Nano Server	43
Install Nano Server	44
Implement Roles and Features on Nano Server	48
Manage and configure Nano Server	50
Managing Nano Server remotely using PowerShell	55
Skill 1.3: Create, manage, and maintain images for deployment	58
Plan for Windows Server virtualization	58
Plan for Linux and FreeBSD deployments	61
Assess virtualization workloads using the Microsoft Assessment and Planning (MAP) Toolkit	61
Determine considerations for deploying workloads into virtualized environments	69
Update images with patches, hotfixes, and drivers	70
Install Roles and Features in offline images	75
Manage and maintain Windows Server Core, Nano Server images, and VHDs using Windows PowerShell	76
Chapter summary	79
Thought experiment	80
Thought experiment answer	80

## **Chapter 2 Implement storage solutions 81**

Skill 2.1: Configure disks and volumes	81
Configure sector sizes appropriate for various workloads	82
Configure GUID partition table (GPT) disks	84
Create VHD and VHDX files using Server Manager or Windows PowerShell	88
Mount Virtual Hard Disks (VHDs)	91

Determine when to use NTFS and ReFS File Systems	93
Configure NFS and SMB shares using Server Manager	95
Configure SMB share and session settings using Windows PowerShell	106
Configure SMB server and SMB client configuration settings using Windows PowerShell	108
Configure file and folder permissions	112
Skill 2.2: Implement server storage . . . . .	123
Configure storage pools	123
Implement simple, mirror, and parity storage layout options for disks or enclosures	125
Configure tiered storage	131
Configure iSCSI target and initiator	133
Configure iSNS	140
Configure Datacenter Bridging (DCB)	142
Configure Multipath I/O (MPIO)	145
Determine usage scenarios for Storage Replica	148
Implement Storage Replica for server-to-server, cluster-to-cluster, and stretch cluster scenarios	151
Skill 2.3: Implement data deduplication . . . . .	155
Implement and configure deduplication	155
Determine appropriate usage scenarios for deduplication	158
Monitor deduplication	160
Implement a backup and restore solution with deduplication	162
Chapter summary . . . . .	162
Thought experiment. . . . .	164
Thought experiment answer. . . . .	164

**Chapter 3 Implement Hyper-V 165**

Skill 3.1: Install and configure Hyper-V . . . . .	165
Determine hardware and compatibility requirements for installing Hyper-V	166
Install Hyper-V	170
Install management tools	172

Upgrade from existing versions of Hyper-V	173
Delegate virtual machine management	174
Perform remote management of Hyper-V hosts	174
Configure virtual machines using Windows PowerShell Direct	180
Implement nested virtualization	181
Skill 3.2: Configure virtual machine (VM) settings . . . . .	182
Creating a virtual machine	182
Add or remove memory in running a VM	185
Configure dynamic memory	186
Configure Non-Uniform Memory Access (NUMA) support	189
Configure smart paging	192
Configure resource metering	193
Manage Integration Services	195
Create and configure Generation 1 and 2 VMs and determine appropriate usage scenarios	197
Implement enhanced session mode	199
Create Linux and FreeBSD VMs	201
Install and configure Linux Integration Services (LIS)	204
Install and configure FreeBSD Integration Services (BIS)	205
Implement Secure Boot for Windows and Linux environments	205
Move and convert VMs from previous versions of Hyper-V to Windows Server 2016 Hyper-V	208
Export and import VMs	209
Implement Discrete Device Assignment (DDA)	212
Skill 3.3: Configure Hyper-V storage . . . . .	213
Create VHDs and VHDX files using Hyper-V Manager	214
Create shared VHDX files	220
Configure differencing disks	222
Modify virtual hard disks	223
Configure pass-through disks	225
Resize a virtual hard disk	226
Manage checkpoints	228
Implement production checkpoints	230

Implement a virtual fibre channel adapter	231
Configure Storage Quality of Service (QoS)	233
Skill 3.4: Configure Hyper-V networking . . . . .	235
Add and remove virtual network interface cards (vNICs)	236
Configure Hyper-V virtual switches	238
Optimize network performance	243
Configure MAC addresses	244
Configure network isolation	246
Configure synthetic and legacy virtual network adapters	247
Configure NIC teaming in VMs	249
Configure virtual machine queue (VMQ)	251
Enable Remote Direct Memory Access (RDMA) on network adapters bound to a Hyper-V virtual switch using Switch Embedded Teaming (SET)	253
Configure bandwidth management	254
Chapter summary . . . . .	256
Thought experiment . . . . .	258
Thought experiment answer . . . . .	258

**Chapter 4 Implement Windows containers 259**

Skill 4.1: Deploy Windows containers . . . . .	259
Determine installation requirements and appropriate scenarios for Windows containers	260
Install and configure Windows Server Container Host in physical or virtualized environments	261
Install and configure Windows Server container host to Windows Server Core or Nano Server in a physical or virtualized environment	264
Install Docker on Windows Server and Nano Server	266
Configure Docker Daemon start-up options	269
Configure Windows PowerShell for use with containers	270
Install a base operating system	271
Tag an image	272
Uninstall an operating system image	273

Create Windows Server containers	274
Create Hyper-V containers	275
Skill 4.2: Manage Windows containers	277
Manage Windows or Linux containers using the Docker daemon	277
Manage Windows or Linux containers using Windows PowerShell	279
Manage container networking	281
Manage container data volumes	286
Manage resource control	287
Create new container images using Dockerfile	289
Manage container images using DockerHub	
Repository for public and private scenarios	291
Manage container images using Microsoft Azure	293
Chapter summary	293
Thought experiment	295
Thought experiment answer	295

## **Chapter 5 Implement high availability 297**

Skill 5.1: Implement high availability and disaster recovery options in Hyper-V	297
Implement Hyper-V Replica	298
Implement live migration	303
Implement shared nothing live migration	307
Configure CredSSP or Kerberos authentication protocol for Live Migration	308
Implement storage migration	309
Skill 5.2: Implement failover clustering	311
Implement workgroup, single, and multi domain clusters	314
Configure quorum	317
Configure cluster networking	321
Restore single node or cluster configuration	324
Configure cluster storage	326
Implement cluster-aware updating	328
Implement cluster operating system rolling upgrade	332

Configure and optimize clustered shared volumes (CSVs)	333
Configure clusters without network names	337
Implement Scale-Out File Server (SoFS)	337
Determine different scenarios for the use of SoFS vs. clustered file server	341
Determine usage scenarios for implementing guest clustering	341
Implement a clustered Storage Spaces solution using shared SAS storage enclosures	342
Implement Storage Replica	345
Implement cloud witness	345
Implement VM resiliency	348
Implement shared VHDX as a storage solution for guest clusters	349
Skill 5.3: Implement Storage Spaces Direct . . . . .	352
Determine scenario requirements for implementing Storage Spaces Direct	352
Enable Storage Spaces direct using Windows PowerShell	354
Implement a disaggregated Storage Spaces Direct scenario in a cluster	355
Implement a hyper-converged Storage Spaces Direct scenario in a cluster	357
Skill 5.4: Manage failover clustering . . . . .	359
Configure role-specific settings, including continuously available shares	359
Configure VM monitoring	361
Configure failover and preference settings	364
Implement stretch and site-aware failover clusters	365
Enable and configure node fairness	367
Skill 5.5: Manage VM movement in clustered nodes . . . . .	369
Perform a live migration	369
Perform a quick migration	370
Perform a storage migration	371
Import, export, and copy VMs	372
Configure VM network health protection	373
Configure drain on shutdown	374

Skill 5.6: Implement Network Load Balancing (NLB) . . . . .	375
Configure NLB prerequisites	375
Install NLB nodes	377
Configure affinity	381
Configure port rules	382
Configure cluster operation mode	384
Upgrade an NLB cluster	384
Chapter summary . . . . .	385
Thought experiment . . . . .	386
Thought experiment answer . . . . .	386

## **Chapter 6 Maintain and monitor server environments 387**

Skill 6.1: Maintain server installations . . . . .	387
Implement Windows Server Update Services (WSUS) solutions	388
Configure WSUS groups	398
Manage patch management in mixed environments	401
Implement an antimalware solution with Windows Defender	405
Integrate Windows Defender with WSUS and Windows Update	409
Perform backup and restore operations using Windows Server Backup	411
Determine backup strategies for different Windows Server roles and workloads, including Hyper-V Host, Hyper-V Guests, Active Directory, File Servers, and Web Servers using Windows Server 2016 native tools and solutions	421
Skill 6.2: Monitor server installations . . . . .	425
Monitor workloads using Performance Monitor	425
Configure data collector sets	431
Determine appropriate CPU, memory, disk, and networking counters for storage and compute workloads	433
Configure alerts	438
Monitor workloads using Resource Monitor	440

Chapter summary .....	442
Thought experiment .....	443
Thought experiment answer .....	443
<i>Index</i>	445

*This page intentionally left blank*

# Introduction

---

Many Windows Server books take the approach of teaching you every detail about the product. Such books end up being huge and tough to read. Not to mention that remembering everything you read is incredibly challenging. That's why those books aren't the best choice for preparing for a certification exam such as the Microsoft Exam 70-740, "Installation, Storage, and Compute with Windows Server 2016." For this book, we focus on your review of the Windows Server skills that you need to maximize your chances of passing the exam. Our goal is to cover all of the skills measured on the exam, while bringing a real-world focus to the information. This book shouldn't be your only resource for exam preparation, but it can be your primary resource. We recommend combining the information in this book with some hands-on work in a lab environment (or as part of your job in a real-world environment).

The 70-740 exam is geared toward IT professionals who have a minimum of 3 years of experience working with Windows Server. That doesn't mean you can't take and pass the exam with less experience, but it probably means that it will be harder. Of course, everyone is different. It is possible to get the knowledge and skills required to pass the 70-740 exam in fewer than 3 years. But whether you are a senior-level Windows Server administrator or just a couple of years into your Windows Server journey, we think you'll find the information in this book valuable as your primary exam prep resource.

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, 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>.

## 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/examref740/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/examref740/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](mailto: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 <http://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>.

*This page intentionally left blank*

## **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.

# Implement Windows containers

Containers are a means of rapidly deploying virtualized, isolated operating system environments, for application deployment and execution. Windows Server 2016 includes support for containers, in cooperation with an open source container engine called Docker.

## Skills in this chapter:

- Deploy Windows containers
- Manage Windows containers

## Skill 4.1: Deploy Windows containers

---

Virtualization has been an important watchword since the early days of Windows. Virtual memory has been around for decades; Windows can use disk space to make the system seem like it has more memory than it has. Hyper-V virtualizes hardware, creating computers within a computer that seem to have their own processors, memory, and disks, when in fact they are sharing the resources of the host server. *Containers* is a new feature in Windows Server 2016 that virtualizes operating systems.

### 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
- Configure Windows PowerShell for use with containers
- 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

Just as virtual machines provide what appear to be separate computers, containers provide what appear to be separate instances of the operating system, each with its own memory and file system, and running a clean, new copy of the operating system. Unlike virtual machines, however, which run separate copies of the operating system, containers share the operating system of the host system. There is no need to install a separate instance of the operating system for each container, nor does the container perform a boot sequence, load libraries, or devote memory to the operating system files. Containers start in seconds, and you can create more containers on a host system than you can virtual machines.

To users working with containers, what they appear to see at first is a clean operating system installation, ready for applications. The environment is completely separated from the host, and from other containers, using namespace isolation and resource governance.

*Namespace isolation* means that each container only has access to the resources that are available to it. Files, ports, and running processes all appear to be dedicated to the container, even when they are being shared with the host and with other containers. The working environment appears like that of a virtual machine, but unlike a virtual machine, which maintains separate copies of all the operating system files, a container is sharing these files with the host, not copying them. It is only when a user or application in a container modifies a file that a copy is made in the container's file system.

*Resource governance* means that a container has access only to a specified amount of processor cycles, system memory, network bandwidth, and other resources, and no more. An ap-

plication running in a container has a clean sandbox environment, with no access to resources allocated to other containers or to the host.

## Container images

The ability to create new containers in seconds, and the isolated nature of each container, make them an ideal platform for application development and software testing. However, there is more to them than that.

Containers are based on images. To create a new container, you download an image from a repository and run it. If you run an image of Windows Server 2016 Server Core, you get a container with a clean instance of the operating system running in it. Alternatively, you can download Windows Server images with roles or applications, such as Internet Information Services (IIS) or Microsoft SQL Server, already installed and ready to run.

The base operating system image never changes. If you install an application in the container and then create a new image, the resulting image contains only the files and settings needed to run the application. Naturally, the new image you created is relatively small, because it does not contain the entire operating system. To share the application with other people, you only have to send them the new, smaller image, as long as they already have the base operating system image.

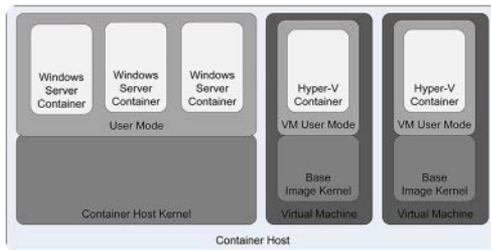
This process can continue through as many iterations as you need, with layer upon layer of images building on that original base. This can result in an extremely efficient software development environment. Instead of transferring huge VHD files, or constantly creating and installing new virtual machines, you can transfer small container images that run without hardware compatibility issues.

## Install and configure Windows Server Container Host in physical or virtualized environments

Windows Server 2016 supports two types of containers: Windows Server Containers and Hyper-V containers. The difference between the two is in the degree of container isolation they provide. Windows Server Containers operate user mode and share everything with the host computer, including the operating system kernel and the system memory.

Because of this, it is conceivable that an application, whether accidentally or deliberately, might be able to escape from the confines of its container and affect other processes running on the host or in other containers. This option is therefore presumed to be preferable when the applications running in different containers are basically trustworthy.

Hyper-V containers provide an additional level of isolation by using the hypervisor to create a separate copy of the operating system kernel for each container. Although they are not visible or exposed to manual management, Hyper-V creates virtual machines with Windows containers inside them, using the base container images, as shown in Figure 4-1. The container implementation is essentially the same; the difference is in the environments where the two types of containers exist.



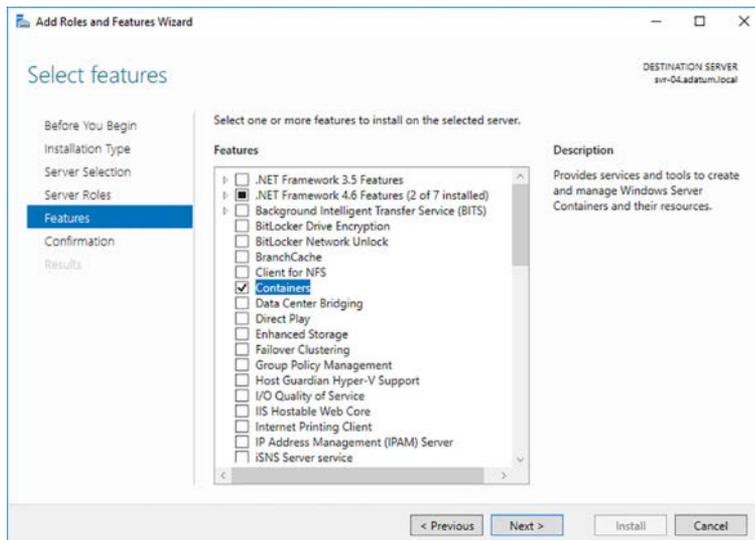
**FIGURE 4-1** Windows container architecture

Because they exist inside a VM, Hyper-V containers have their own memory assigned to them, as well as isolated storage and network I/O. This provides a container environment that is suitable for what Microsoft calls “hostile multi-tenant” applications, such as a situation in which a business provides containers to clients for running their own code, which might not be trustworthy. Thus, with the addition of Hyper-V containers, Windows Server 2016 provides three levels of isolation, ranging from the separate operating system installation of Hyper-V virtual machines, to the separate kernel and memory of Hyper-V containers, to the shared kernel and other resources of Windows Server Containers.

## Installing a container host

Windows Server 2016 includes a feature called Containers, which you must install to provide container support, but to create and manage containers you must download and install Docker, the application that supports the feature.

To install the Containers feature, you can use the Add Roles And Features Wizard in Hyper-V Manager, selecting Containers on the Select Features page, as shown in Figure 4-2.



**FIGURE 4-2** Installing the Containers feature in Hyper-V Manager

#### **NOTE WINDOWS SERVER INSTALLATION**

To create Windows Server containers, the host operating system must be installed on the computer's C drive, which is the installation default. This is to facilitate the sharing the operating system kernel. This is not a requirement for creating Hyper-V containers, as the hypervisor is responsible for providing a copy of the kernel to each container.

To create Hyper-V containers, you must install both the Containers feature and the Hyper-V role. Even though you will not be creating virtual machines for the containers, the Hyper-V role installs the hypervisor that will be needed to create the separate copy of the Windows kernel for each Hyper-V container.

The Hyper-V role has general hardware requirements that exceed those of the Windows Server 2016 operating system itself. Before you can install the Hyper-V role on a server running Windows Server 2016, you must have the following hardware:

- A 64-bit processor that includes hardware-assisted virtualization and second-level address translation (SLAT). This type of virtualization is available in processors that include a virtualization option, such as Intel Virtualization Technology (Intel VT) or AMD Virtualization (AMD-V) technology.
- Hardware-enforced Data Execution Prevention (DEP), which Intel describes as eXecuted Disable (XD) and AMD describes as No eXecute (NS). CPUs use this technology to segregate areas of memory for either storage of processor instructions or for storage of data. Specifically, you must enable Intel XD bit (execute disable bit) or AMD NX bit (no execute bit).
- VM Monitor Mode extensions, found on Intel processors as VT-c.
- A system BIOS or UEFI that supports the virtualization hardware and on which the virtualization feature has been enabled.

When you install the Hyper-V role using Hyper-V Manager, the Add Roles And Features Wizard prompts to install the Hyper-V Management tools as well. If you are creating Hyper-V containers but not Hyper-V virtual machines, there is no need to install the management tools.

## **Virtualizing containers**

Windows Server 2016 supports the use of containers within Hyper-V virtual machines. You can install the Containers feature and the Docker files in any virtual machine. However, to create Hyper-V containers on a virtual machine, the system must meet the requirements for nested virtualization.

To create a nested Hyper-V host server, the physical host and the virtual machine on which you create the Hyper-V containers must both be running Windows Server 2016. The VM can run the full Desktop Experience, Server Core, or Nano Server installation option. In addition, the physical host must have an Intel processor with VT-x and Extended Page Tables (EPT) virtualization support.

Before you install Hyper-V on the virtual machine, you must provide its virtual processor with access to the virtualization technology on the physical computer. To do this, you must shut down the virtual machine and run a command like the following on the physical host, in a PowerShell session with administrator privileges:

```
set-vmprocessor -vmname server1 -exposevirtualizationextensions $true
```

In addition, you must make the following configuration changes on the VM that functions as a Hyper-V host. Each is given first as the location in the VM Settings dialog box in Hyper-V Manager, and then as a PowerShell command:

- On the Memory page, provide the VM with at least 4 gigabytes (GB) of RAM and disable Dynamic Memory.

```
set-vmmemory -vmname server1 -startupbytes 4gb -dynamicmemoryenabled $false
```

- On the Processor page, set Number of Virtual Processors to 2.

```
set-vmprocessor -vmname server1 -count 2
```

- On the Network Adapter/Advanced Features page, turn on MAC Address Spoofing.

```
set-vmnetworkadapter -vmname server1 -name "network adapter" -macaddressspoofing on
```

Once you have made these changes, you can start the VM, install the Hyper-V role, and proceed to use Docker to create Hyper-V containers.

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

A computer installed using the Server Core option can function as a container host. The requirements are the same as for a server installed with the full Desktop Experience, except that you must either use the command line to install the required features or manage the system remotely.

After switching to a PowerShell session, you can install the Containers feature and the Hyper-V role using the following command:

```
install-windowsfeature -name containers, hyper-v
```

### Configuring Nano Server as a container host

Nano Server, included with Windows Server 2016, supports both Windows Server containers and Hyper-V containers. The Nano Server implementation includes packages supporting both the Containers feature and the Hyper-V role, which you can add when you create a Nano Server image with the New-NanoServerImage cmdlet in Windows PowerShell, as in the following example:

```
new-nanoserverimage -deploymenttype guest -edition datacenter -mediapath d:\ -targetpath c:\nano\nano1.vhdx -computername nano1 -domainname contoso -containers
```

This command creates a Nano Server image with the following characteristics:

- **deploymenttype guest** Creates an image for use on a Hyper-V virtual machine
- **edition datacenter** Creates an image using the Datacenter edition of Windows Server
- **mediapath d:\** Accesses the Nano Server source files from the D drive
- **targetpath c:\nano\nano1.vhdx** Creates an VHDX image file in the C:\nano folder with the name Nano1.vhdx
- **computername nano1** Assigns the Nano Server the computer name Nano1
- **domainname contoso** Joins the computer to the Contoso domain
- **containers** Installs the Containers feature as part of the image
- **compute** Installs the Hyper-V role as part of the image

If you plan on creating Hyper-V containers on the guest Nano Server, you must provide it with access to the virtualization capabilities of the Hyper-V server, using the following procedure.

1. Create a new virtual machine, using the Nano Server image file you created, but do not start it.
2. On the Hyper-V host server, grant the virtual machine with access to the virtualization capabilities of the Hyper-V server's physical processor, using a command like the following:  

```
set-vmprocessor -vmname nano1 -exposevirtualizationextensions $true
```
3. Start the Nano Server virtual machine.

Once the Nano Server virtual machine is running, you must establish a remote PowerShell session from another computer, so you can manage it. To do this, run a command like the following on the computer you use to manage Nano Server:

```
enter-psession -computername nano1 -credential
```

#### **NOTE REMOTE NANO SERVER MANAGEMENT**

This section assumes that the Nano Server is located on a network with a DHCP server that assigns its TCP/IP settings and that it has successfully joined an Active Directory Domain Services domain. If those are not the case, you must configure the TCP/IP settings for the Nano Server manually, from its console, and then add the Nano Server to the Trusted Hosts list on the computer you use to manage it.

## Install Docker on Windows Server and Nano Server

Docker is an open source tool that has been providing container capabilities to the Linux community for years. Now that it has been ported, you can implement those same capabilities in Windows. Docker consists of two files:

- **Dockerd.exe** The Docker engine, also referred to as a service or daemon, which runs in the background on the Windows computer
- **Docker.exe** The Docker client, a command shell that you use to create and manage containers

In addition to these two files, which you must download and install to create containers, Docker also includes the following resources:

- **Dockerfiles** Script files containing instructions for the creation of container images
- **Docker Hub** A cloud-based registry that enables Docker users to link to image and code repositories, as well as build and store their own images
- **Docker Cloud** A cloud-based service you can use to deploy your containerized applications

### Installing Docker on Windows Server

Because Docker is an open source product, it is not included with Windows Server 2016. On a Windows Server 2016 Desktop Experience or Server Core computer, you must download Docker and install it before you can create containers. To download Docker, you use OneGet, a cloud-based package manager for Windows.

To access OneGet, you must install the DockerMsftProvider module, using the following command. If you are prompted to install a NuGet provider, answer Yes.

```
install-module -name dockersmftprovider -repository psgallery -force
```

The Install-Module cmdlet downloads the requested module and installs it to the C:\Program Files\Windows PowerShell\Modules folder, where it is accessible from any PowerShell prompt. Next, to download and install Docker, run the following Install-Package command. If the command prompts you to confirm that you want to install an untrusted package, answer Yes.

```
install-package -name docker -providername dockersmftprovider
```

This command, after downloading the Docker files, registers Dockerd.exe as a Windows service and adds the Docker.exe client to the path, so that it is executable from any location in the file system.

Once the installation is completed, restart the computer with the following command:

```
restart-computer -force
```

## Installing Docker on Nano Server

Once you have entered a remote PowerShell session with a Nano Server computer, you can install Docker using the same commands as for a Desktop Experience or Server Core system. However, Microsoft recommends that, once the Dockerd service is installed on the Nano Server, you run the Docker client from the remote system.

To do this, you must complete the following tasks:

1. Create a firewall rule. For the Nano Server to allow Docker client traffic into the system, you must create a new firewall rule opening port 2375 to TCP traffic. To do this, run the following command in the Nano Server session:

```
netsh advfirewall firewall add rule name="docker daemon" dir=in action=allow  
protocol=tcp localport=2375
```

2. Configure the Dockerd engine to accept network traffic. Docker has its origins in Linux, and like most Linux applications, it uses text files for configuration. To enable the Dockerd engine to accept client traffic over the network, you must create a text file called `daemon.json` in the `C:\ProgramData\Docker` directory on the Nano Server that contains the following line:

```
{ "hosts": ["tcp://0.0.0.0:2375", "npipe://"] }
```

The following two PowerShell commands create the new file and insert the required text:

```
new-item -type file c:\programdata\docker\config\daemon.json
```

```
add-content 'c:\programdata\docker\config\daemon.json' '{ "hosts":  
["tcp://0.0.0.0:2375", "npipe://"] }'
```

3. Restart the Dockerd engine. Once you have created the `daemon.json` file, you must restart the Dockerd engine, using the following command:

```
restart-service docker
```

4. Download the Docker client. To manage the Dockerd engine remotely, you must download and install the `Docker.exe` client on the remote system (not within the Nano Server session). To do this, you can open a browser and type in the following URL to download the Docker package:

```
https://download.docker.com/components/engine/windows-server/cs-1.12/docker.zip
```

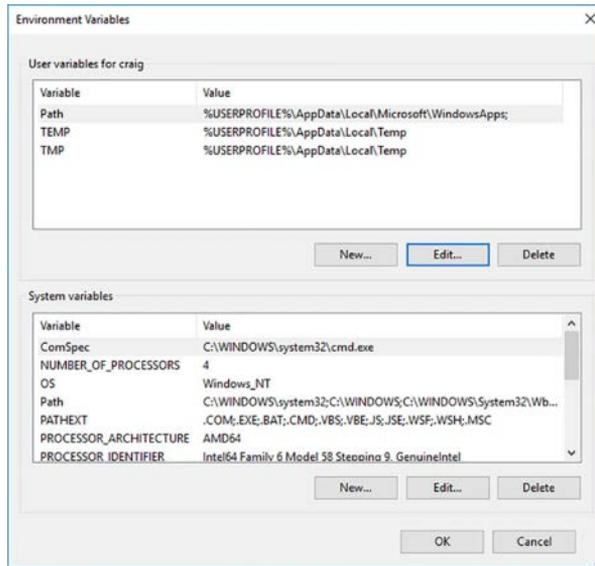
5. To do this in PowerShell, use the following command:

```
invoke-webrequest "https://download.docker.com/components/engine/windows-server/  
cs-1.12/docker.zip" -outfile "$env:temp\docker.zip" -usebasicparsing
```

6. Install `Docker.exe`. If you downloaded the `Docker.zip` file through a browser, you install the application by extracting the `Docker.exe` file from the zip archive and copying it to a folder you must create called `C:\ProgramData\Docker`. To do this using PowerShell, run the following command:

```
expand-archive -path "$env:temp\docker.zip" -destinationpath $env:programfiles
```

7. Set the PATH environment variable. To run the Docker client from any location on the management system, you must add the C:\ProgramData\Docker folder to the system's PATH environment variable. To do this graphically, open the System Properties sheet from the Control Panel and, on the Advanced tab, click Environment Variables to display the dialog box shown in Figure 4-3.



**FIGURE 4-3** The Environment Variables dialog box

8. To do this in PowerShell, run the following command:

```
[environment]::setenvironmentvariable("path", $env:path + ";c:\program files\  
docker", [environmentvariabletarget]::machine)
```

Once you have completed these steps, you can run the Docker.exe client outside of the Nano Server session, but you must include the following parameter in every command, where the `ipaddress` variable is replaced by the address of the Nano Server you want to manage:

```
-h tcp://ipaddress:2375
```

For example, to create a new container with the `microsoft/nanoserver` image, you would use a command like the following:

```
docker -h tcp://172.21.96.1:2375 run -it microsoft/nanoserver cmd
```

To avoid having to add the `-h` parameter to every command, you can create a new environment variable as follows:

```
docker_host = "tcp://ipaddress:2375"
```

To do this in PowerShell, use a command like the following:

```
$env:docker_host = "tcp://172.21.96.1:2375"
```

## Configure Docker Daemon start-up options

As mentioned in the previous section, the configuration file for the Dockerd engine is a plain text file called `daemon.json`, which you place in the same folder as the `Dockerd.exe` file. In addition to the one you used earlier to permit client traffic over the network, there are many other configuration settings you can include in the file. All of the settings you include in a single `daemon.json` file should be enclosed in a single set of curly braces, as in the following example:

```
{
  "graph": "d:\\docker"
  "bridge" : "none"
  "group" : "docker"
  {"dns": 192.168.9.2, 192.168.9.6 }
}
```



---

### **EXAM TIP**

Be aware that while the Windows port of Docker supports many of the Linux Dockerd configuration settings, it does not support all of them. If you are studying Docker documentation, be sure to look for the Windows version of the documents.

---

## Redirecting images and containers

To configure the Dockerd engine to store image files and containers in an alternate location, you include the following command in the `daemon.json` file, where `d:\\docker` is replaced by the location you want to use:

```
{ "graph": "d:\\docker" }
```

## Suppressing NAT

By default, the Dockerd engine creates a network address translation (NAT) environment for containers, enabling them to communicate with each other and with the outside network. To modify this default behavior and prevent the engine from using NAT, you include the following command in the `daemon.json` file:

```
{ "bridge" : "none" }
```

## Creating an administrative group

By default, only members of the local Administrators group can use the Docker client to control the Dockerd engine when working on the local system. In some cases, you can grant users this ability without giving them Administrators membership. You can configure Dockerd to recognize another group—in this case, the group is called “docker”—by including the following setting in the `daemon.json` file.

```
{ "group" : "docker" }
```

## Setting DNS server addresses

To specify alternative DNS server addresses for the operating systems in containers, you can add the following setting to the daemon.json file, where address1 and address2 are the IP addresses of DNS servers:

```
{"dns": "address1" , "address2" }
```

## Configure Windows PowerShell for use with containers

The Dockerd engine is supplied with a Docker.exe client shell, but it is not dependent on it. You can also use Windows PowerShell cmdlets to perform the same functions. The Docker PowerShell module, like Docker itself, is in a constant state of cooperative development, and it is therefore not included with Windows Server 2016.

You can download and install the current version of the PowerShell module from a repository called DockerPS-Dev, using the following commands:

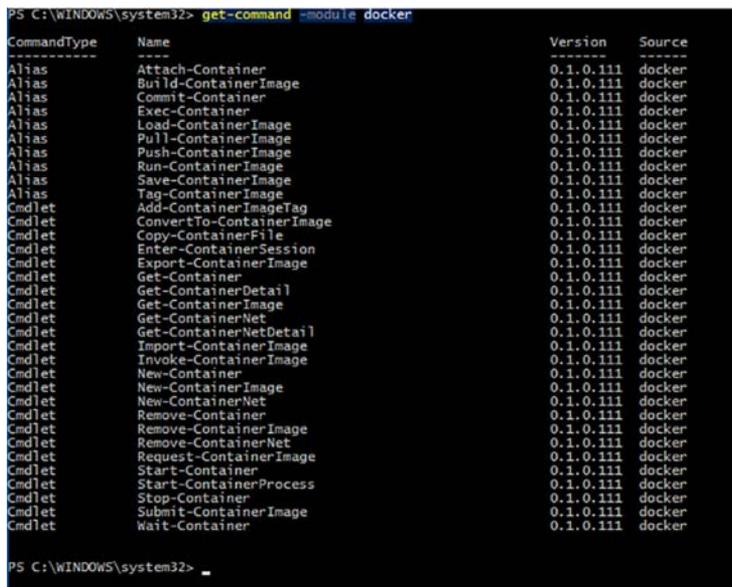
```
register-psrepository -name dockerps-dev -sourcelocation https://ci.appveyor.com/nuget/docker-powershell-dev
```

```
install-module docker -repository dockerps-dev -scope currentuser
```

Once the download is completed, you can view a list of the Docker cmdlets by running the following command:

```
get-command -module docker
```

The current resulting output is shown in Figure 4-4.



```
PS C:\WINDOWS\system32> get-command -module docker
CommandType      Name                                     Version      Source
-----
Alias            Attach-Container                       0.1.0.111   docker
Alias            Build-ContainerImage                  0.1.0.111   docker
Alias            Commit-Container                      0.1.0.111   docker
Alias            Exec-Container                        0.1.0.111   docker
Alias            Load-ContainerImage                   0.1.0.111   docker
Alias            Pull-ContainerImage                   0.1.0.111   docker
Alias            Push-ContainerImage                   0.1.0.111   docker
Alias            Run-ContainerImage                    0.1.0.111   docker
Alias            Save-ContainerImage                   0.1.0.111   docker
Alias            Tag-ContainerImage                    0.1.0.111   docker
Cmdlet           Add-ContainerImageTag                  0.1.0.111   docker
Cmdlet           ConvertTo-ContainerImage               0.1.0.111   docker
Cmdlet           Copy-ContainerFile                     0.1.0.111   docker
Cmdlet           Enter-ContainerSession                 0.1.0.111   docker
Cmdlet           Export-ContainerImage                  0.1.0.111   docker
Cmdlet           Get-Container                          0.1.0.111   docker
Cmdlet           Get-ContainerDetail                    0.1.0.111   docker
Cmdlet           Get-ContainerImage                     0.1.0.111   docker
Cmdlet           Get-ContainerNet                       0.1.0.111   docker
Cmdlet           Get-ContainerNetDetail                 0.1.0.111   docker
Cmdlet           Import-ContainerImage                  0.1.0.111   docker
Cmdlet           Invoke-ContainerImage                  0.1.0.111   docker
Cmdlet           New-Container                          0.1.0.111   docker
Cmdlet           New-ContainerImage                     0.1.0.111   docker
Cmdlet           New-ContainerNet                       0.1.0.111   docker
Cmdlet           Remove-Container                       0.1.0.111   docker
Cmdlet           Remove-ContainerImage                  0.1.0.111   docker
Cmdlet           Remove-ContainerNet                    0.1.0.111   docker
Cmdlet           Request-ContainerImage                 0.1.0.111   docker
Cmdlet           Start-Container                        0.1.0.111   docker
Cmdlet           Start-ContainerProcess                 0.1.0.111   docker
Cmdlet           Stop-Container                         0.1.0.111   docker
Cmdlet           Submit-ContainerImage                  0.1.0.111   docker
Cmdlet           Wait-Container                          0.1.0.111   docker
PS C:\WINDOWS\system32>
```

FIGURE 4-4 Cmdlets in the Docker module for Windows PowerShell

Once you have registered the repository and imported the Docker module, you do not have to run those commands again. You can always obtain the latest version of the module by running the following command:

```
update-module docker
```

## Install a base operating system

With the Dockerd engine and the Docker client installed and operational, you can take the first step toward creating containers, which is to download a base operating system image from the Docker Hub repository. Microsoft has provided the repository with Windows Server 2016 Server Core and Nano Server images, which you can download and use to create containers and then build your own container images.

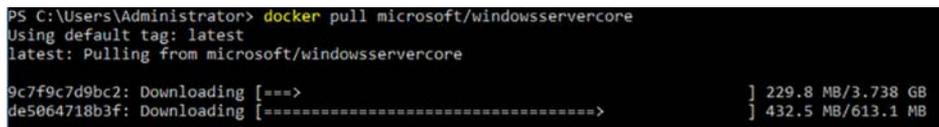
To use the Docker client, you execute the Docker.exe file with a command and sometimes additional options and parameters. To download an image, you run Docker with the Pull command and the name of the image. For example, the following command downloads the Server Core image from the repository.

```
docker pull microsoft/windowsservercore
```

The PowerShell equivalent is as follows:

```
request-containerimage -repository microsoft/windowsservercore
```

The output of the command (which can take some time, depending on the speed of your Internet connection) is shown in Figure 4-5.



```
PS C:\Users\Administrator> docker pull microsoft/windowsservercore
Using default tag: latest
latest: Pulling from microsoft/windowsservercore
9c7f9c7d9bc2: Downloading [===>] 229.8 MB/3.738 GB
de5064718b3f: Downloading [=====] 432.5 MB/613.1 MB
```

**FIGURE 4-5** Output of the Docker Pull command

By default, the Docker Pull command downloads the latest version of the specified image, which is identified by the tag: “latest.” When there are multiple versions of the same image available, as in an application development project, for example, you can specify any one of the previous images to download, by specifying its tag. If you run the Docker Pull command with the -a parameter, you get all versions of the image. If the image you are pulling consists of multiple layers, the command automatically downloads all of the layers needed to deploy the image in a container.

If you know that the repository has a Nano Server image, but you are not sure of its name, you can use the Docker Search command to locate it, and then use Docker Pull to download it, as shown in Figure 4-6

```

PS C:\Users\Administrator> docker search microsoft
NAME                DESCRIPTION                                STARS    OFFICIAL    AUTOM
ATED
microsoft/aspnet    ASP.NET is an open source server-side Web ... 498      [OK]
microsoft/dotnet    Official images for .NET Core for Linux an... 327      [OK]
mono                Mono is an open source implementation of M... 195      [OK]
microsoft/windowsservercore
Windows Server 2016 Server Core base OS im... 69
microsoft/nanoserver
Windows Server 2016 Nano Server base OS im... 66
microsoft/azure-cli
Docker image for Microsoft Azure Command L... 66
microsoft/iis       Internet Information Services (IIS) instal... 50
microsoft/mssql-server-2014-express-windows
Microsoft SQL Server 2014 Express installe... 41
microsoft/aspnetcore
Official images for running compiled ASP.N... 28
microsoft/mssql-server-2016-express-windows
Microsoft SQL Server 2016 Express installe... 28
microsoft/powershell
Official PowerShell Core releases from htt... 8
microsoft/oms       Monitor your containers using the Operatio... 7
microsoft/aspnetcore-build
Official images for building ASP.NET Core ... 6
microsoft/dotnet35
The .NET Framework 3.5 image has moved to ... 4
microsoft/vsts-agent
Official images for the Visual Studio Team... 4
microsoft/applicationinsights
Application Insights for Docker helps you ... 4
microsoft/nginx     Nginx installed in Windows Server Core and... 4
microsoft/dotnet-nightly
Preview bits of the .NET Core CLI          2
microsoft/powershell-nightly
Nightly builds of PowerShell Core for CI    2
microsoft/sample-dotnet
.NET Core running in a Nano Server container 1
microsoft/cntk      CNTK                                         0
dreher/microsoft    Microsoft Test Repo                         0
microsoft/aspnetcore-build-nightly
Images to build preview versions of ASP.NE... 0
microsoft/dotnet-samples
.NET Core Docker Samples                    0
berliius/microsoft-malmo
Microsoft-Malmo - artificial intelligence ... 0
PS C:\Users\Administrator> docker pull microsoft/nanoserver
Using default tag: latest
latest: Pulling from microsoft/nanoserver
5496abde368a: Pull complete
94b4ce/ac4c7: Pull complete
Digest: sha256:86cfd90ee6f711086d9cd637b7d8f250270c46cfe4e08f7527aea7968b9c8ff
Status: Downloaded newer image for microsoft/nanoserver:latest
PS C:\Users\Administrator>

```

FIGURE 4-6 Output of the Docker Search command

## Tag an image

Tagging, in a container repository, is a version control mechanism. When you create multiple versions of the same image, such as the successive builds of an application, Docker enables you to assign tags to them that identify the versions. Tags are typically numbers indicating the relative ages of the image iterations, such as 1.1, 1.2, 2.0, and so forth.

There are two ways to assign a tag to an image. One is to run Docker with the Tag command, and the other is to run Docker Build with the -t parameter. In both cases, the format of the image identifier is the same.

To tag an image on your local container host, you use the following syntax:

```
docker tag imagename:tag
```

If you are going to be uploading the image to the Docker Hub, you must prefix the image name with your Docker Hub user name and a slash, as follows:

```
docker tag username/imagename:tag
```

For example, a user called Holly Holt might tag the latest build of her new application as follows:

```
docker tag hholt/killerapp:1.5
```

To do the same thing in Windows PowerShell, you would use the Add-ContainerImageTag cmdlet, as follows:

```
add-containerimagetag -imageidorname c452b8c6ee1a -repository hholt/killerapp -tag 1.5
```

If you omit the tag value from the command, Docker automatically assigns the image a tag value of the word “latest,” which can lead to some confusion. When you pull an image from a repository without specifying a tag, the repository gives you the image with the “latest” tag. However, this does not necessarily mean that the image you are getting is the newest.

The “latest” tag is supposed to indicate that the image possessing it is the most recent version. However, whether that is true or not depends on the people managing the tags for that repository. Some people think that the “latest” tag is automatically reassigned to the most recent version of an image, but this is not the case. You can assign the “latest” tag to any version of an image, the oldest or the newest. It is solely up to the managers of the repository to maintain the tag values properly. When someone tells you to get the latest build of an image, is the person referring to the most recent build or the build with the “latest” tag? They are not always the same thing.

## Uninstall an operating system image

Running Docker with the Images command displays all of the images on the container host, as shown in Figure 4-7.

```
PS C:\WINDOWS\system32> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
microsoft/sample-dotnet  latest            c14528829a37      9 days ago        911 MB
microsoft/iis        latest            b6a44de60ef9      3 weeks ago       8.96 GB
microsoft/windowsservercore latest            93a9c37b36d0      6 weeks ago       8.68 GB
microsoft/nanoserver 10.0.14393.206     853f9db844af      6 weeks ago       652 MB
microsoft/nanoserver latest            e14bc0ceea12      6 weeks ago       810 MB
microsoft/nanoserver 10.0.14393.206_de-de a896e5590871     6 weeks ago       658 MB
microsoft/nanoserver 10.0.14393.206_cs-cz ef42b616e27e     6 weeks ago       653 MB
microsoft/nanoserver 10.0.14300.1030    3a703c6e97a2     4 months ago     970 MB
PS C:\WINDOWS\system32> _
```

FIGURE 4-7 Output of the Docker Images command

In some instances, you might examine the list of images and find yourself with images that you do not need. In this example, there are two non-English versions of Nano Server that were downloaded accidentally.

To remove images that you do not need and free up the storage space they’re consuming, you run Docker with the Rmi command and specify either the repository and tag of the specific image to delete, or the Image ID value, as in the following examples:

```
docker rmi -f microsoft/nanoserver:10.0.14393.206_de-de
```

```
docker rmi -f a896e5590871
```

The PowerShell equivalent is the Remove-ContainerImage cmdlet, as in the following:

```
remove-containerimage microsoft/nanoserver:10.0.14393.206_de-de
```

```
remove-containerimage a896e5590871
```

It is possible for the same image to be listed with multiple tags. You can tell this by the matching Image ID values. If you attempt to remove one of the images using the tag, an error

appears, because the image is in use with other tags, Adding the `-f` parameter forces the command to delete all the tagged references to the same image.

## Create Windows Server containers

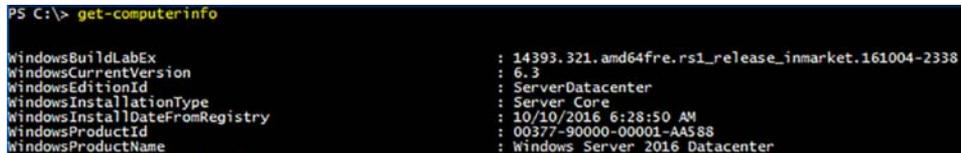
With the Containers feature in place and Docker installed, you are ready to create a Windows Server container. To do this, you use the Docker Run command and specify the image that you want to run in the container. For example, the following command creates a new container with the Server Core image downloaded from Docker Hub:

```
docker run -it microsoft/windowsservercore powershell
```

In addition to loading the image into the container, the parameters in this command do the following:

- **i** Creates an interactive session with the container
- **t** Opens a terminal window into the container
- **powershell** Executes the PowerShell command in the container session

The result is that after the container loads, a PowerShell session appears, enabling you to work inside the container. If you run the `Get-ComputerInfo` cmdlet in this session, you can see at the top of the output, shown in Figure 4-8, that Server Core is running in the container, when the full Desktop Experience edition is running on the container host.



```
PS C:\> get-computerinfo
WindowsBuildLabEx           : 14393.321.amd64fre.rs1_release_inmarket.161004-2338
WindowsCurrentVersion       : 6.3
WindowsEditionId            : ServerDatacenter
WindowsInstallationType     : Server Core
WindowsInstallDateFromRegistry : 10/10/2016 6:28:50 AM
WindowsProductId            : 00377-90000-00001-AA588
WindowsProductName          : Windows Server 2016 Datacenter
```

**FIGURE 4-8** Output of the `Get-ComputerInfo` cmdlet

You can combine Docker Run switches, so the `-l` and `-t` appear as `-it`. After the name of the image, you can specify any command to run in the container. For example, specifying `cmd` would open the standard Windows command shell instead of PowerShell.

### **NOTE** OBTAINING IMAGES

Pulling an image from the Docker Hub is not a required step before you can run it. If you execute a Docker Run command, and you don't have the required image on your container host, Docker initiates a pull automatically and then creates the container. For large images, however, pulling them beforehand can save time when creating new containers.

The Docker Run command supports many command line parameters and switches, which you can use to tune the environment of the container you are creating. To display them, you can run the following command:

```
docker run --help
```

## NOTE EXECUTING DOCKER COMMANDS

Note that this, and other, Docker commands sometimes use double hyphens to process command line parameters.

Figure 4-9 displays roughly half of the available parameters. For example, including the `-h` parameter enables you to specify a host name for the container, other than the hexadecimal string that the command assigns by default.

```
PS C:\WINDOWS\system32> docker run --help
Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
Run a command in a new container
Options:
--add-host value          Add a custom host-to-IP mapping (host:ip) (default [])
-a, --attach value       Attach to STDIN, STDOUT or STDERR (default [])
--blkio-weight value     Block IO (relative weight), between 10 and 1000
--blkio-weight-device value Block IO weight (relative device weight) (default [])
--cap-add value         Add Linux capabilities (default [])
--cap-drop value        Drop Linux capabilities (default [])
--cgroup-parent string   Optional parent cgroup for the container
--cidfile string        Write the container ID to the file
--cpu-percent int       CPU percent (Windows only)
--cpu-period int        Limit CPU CFS (Completely Fair Scheduler) period
--cpu-quota int         Limit CPU CFS (Completely Fair Scheduler) quota
-c, --cpu-shares int     CPU shares (relative weight)
--cpuset-cpus string    CPUs in which to allow execution (0-3, 0,1)
--cpuset-mems string    MEMS in which to allow execution (0-3, 0,1)
--credential-spec string Credential spec for managed service account (Windows only)
-d, --detach            Run container in background and print container ID
--detach-keys string    Override the key sequence for detaching a container
--device value         Add a host device to the container (default [])
--device-read-bps value Limit read rate (bytes per second) from a device (default [])
--device-read-iops value Limit read rate (IO per second) from a device (default [])
--device-write-bps value Limit write rate (bytes per second) to a device (default [])
--device-write-iops value Limit write rate (IO per second) to a device (default [])
--disable-content-trust Skip image verification (default true)
--dns value            Set custom DNS servers (default [])
--dns-opt value        Set DNS options (default [])
--dns-search value     Set custom DNS search domains (default [])
--entrypoint string    Overwrite the default ENTRYPOINT of the image
-e, --env value        Set environment variables (default [])
--env-file value       Read in a file of environment variables (default [])
--expose value         Expose a port or a range of ports (default [])
--group-add value      Add additional groups to join (default [])
--health-cmd string    Command to run to check health
--health-interval duration Time between running the check (default 0s)
--health-retries int   Consecutive failures needed to report unhealthy
--health-timeout duration Maximum time to allow one check to run (default 0s)
--help                Print usage
-h, --hostname string  Container hostname
--init                Run an init inside the container that forwards signals and reaps processes
-i, --interactive      Keep STDIN open even if not attached
--io-maxbandwidth string Maximum IO bandwidth limit for the system drive (Windows only)
--io-maxiops uint     Maximum IOps limit for the system drive (Windows only)
--ip string            Container IPv4 address (e.g. 172.30.100.104)
--ip6 string           Container IPv6 address (e.g. 2001:db8::33)
```

FIGURE 4-9 Output of the Docker Run `--help` command

The PowerShell equivalent of the Docker Run command uses the `New-Container` cmdlet, as in the following example:

```
new-container -imageidorname microsoft/windowsservercore -input -terminal -command powershell
```

## Create Hyper-V containers

The process of creating a Hyper-V container is almost identical to that of creating a Windows Server container. You use the same Docker Run command, except that you add the `--isolation=hyperv` parameter, as shown in the following example:

```
docker run -it --isolation=hyperv microsoft/windowsservercore powershell
```

Once you create a Hyper-V container, it is all but indistinguishable from a Windows Server container. One of the few ways to tell the types of containers apart is to examine how they handle processes. For example, you can create two containers and execute a command in each one that starts them pinging themselves continuously, as shown in the following commands:

```
docker run -it microsoft/windowsservercore ping -t localhost
```

```
docker run -it --isolation=hyperv microsoft/windowsservercore ping -t localhost
```

The Windows Server container created by the first command has a PING process running in the container, as shown by the Docker Top command in Figure 4-10. The process ID (PID) number, in this case, is 404. Then, when you run the Get-Process cmdlet, to display the processes (starting with P) running on the container host, you see the same PING process with the 404 ID. This is because the container is sharing the kernel of the container host.

```
PS C:\WINDOWS\system32> docker ps
CONTAINER ID        IMAGE                                 COMMAND                  CREATED            STATUS
0e38bdac48ca      microsoft/windowsservercore        "powershell"           5 hours ago       Up 5 hours
PS C:\WINDOWS\system32> docker top 0e38bdac48ca
Name                PID                CPU                Private Working Set
smss.exe            2420              00:00:00.125      229.4 kB
csrss.exe           8444              00:00:00.390      946.2 kB
wininit.exe         3220              00:00:00.187      806.9 kB
services.exe        7636              00:00:00.453      1.794 MB
lsass.exe           8584              00:00:01.156      3.633 MB
svchost.exe         5860              00:00:00.406      2.208 MB
svchost.exe         8360              00:00:00.265      1.745 MB
svchost.exe         7296              00:00:00.281      2.06 MB
svchost.exe         6916              00:00:00.578      3.912 MB
svchost.exe         7888              00:00:09.015      10.95 MB
svchost.exe         2460              00:00:00.640      3.219 MB
svchost.exe         4340              00:00:04.140      8.409 MB
svchost.exe         4880              00:00:00.062      839.7 kB
CEExecSvc.exe       3528              00:00:00.796      815.1 kB
svchost.exe         4288              00:00:02.218      4.235 MB
powershell.exe     2016              00:00:10.781      33.62 MB
msdtc.exe           7816              00:00:00.078      1.876 MB
powershell.exe     6768              00:00:07.515      30.3 MB
PING.EXE            404               00:00:00.031      589.8 kB
PS C:\WINDOWS\system32> get-process p*
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
71	5	788	3752	0.03	404	7	PING
549	27	56560	64640	1.02	1752	2	powershell
542	43	60420	76424	10.78	2016	7	powershell
517	40	56816	70660	7.52	6768	7	powershell
664	40	77556	98004	5.48	8224	2	powershell
537	27	54696	57236	3.75	8864	2	powershell

```
PS C:\WINDOWS\system32>
```

FIGURE 4-10 Output of Docker Top and Get-Process commands for a Windows Server container

On the other hand, when you run the Docker Top command on the Hyper-V container, you again see the PING process, this time with a PID of 1852, as shown in Figure 4-11. However, the Get-Process cmdlet shows no PING process, because this container has its own kernel provided by the hypervisor.

```

PS C:\Users\Administrator> docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
8d67f1679c68      microsoft/windowsservercore   "ping -t localhost"   6 minutes ago      Up 5 minutes
y_lovelace
PS C:\Users\Administrator>
PS C:\Users\Administrator> docker top 8d67f1679c68
Name                PID                CPU                Private Working Set
smss.exe            248                00:00:01.156      233.5 kB
csrss.exe           312                00:00:00.750      921.6 kB
wininit.exe         644                00:00:00.453      737.3 kB
services.exe        920                00:00:01.296      1.532 MB
lsass.exe           828                00:00:00.515      2.22 MB
svchost.exe         1076               00:00:00.359      1.958 MB
svchost.exe         1124               00:00:00.234      1.401 MB
svchost.exe         1212               00:00:00.421      2.023 MB
svchost.exe         1228               00:00:00.953      4.919 MB
svchost.exe         1268               00:00:00.359      2.605 MB
svchost.exe         1280               00:00:09.000      13.43 MB
svchost.exe         1368               00:00:08.500      3.174 MB
svchost.exe         1528               00:00:00.703      3.146 MB
svchost.exe         1540               00:00:00.093      819.2 kB
CEXecSvc.exe        1592               00:00:00.046      688.1 kB
PING.EXE            1852               00:00:00.031      589.8 kB
msdtc.exe           872                00:00:00.203      1.901 MB
WmiPrvSE.exe        2004               00:00:02.078      5.526 MB
PS C:\Users\Administrator>
PS C:\Users\Administrator> get-process p*

Handles  NPM(K)  PM(K)  WS(K)  CPU(s)  Id  SI ProcessName
-----  -
561      27     53540  61920  1.42    1096  2 powershell
696      27     53540  61628  3.20    4564  2 powershell

```

FIGURE 4-11 Output of the Docker Top and Get-Process commands for a Hyper-V container

## Skill 4.2: Manage Windows containers

- 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

When you use the Docker Run command to create a new container, you can include the `-it` switches to work with it interactively, or you can omit them and let the container run in the background. Either way, you can continue to use the Docker client to manage container, either Windows or Linux.

## Listing containers

To leave a PowerShell or CMD session you started in a container, you can just type the following:

```
exit
```

However, this not only closes the session, it also stops the container. A stopped container still exists on the host; it is just functionally turned off. To exit a session without stopping the container, press Ctrl+P, then Ctrl+Q.

You can display a list of all the running containers on the host by using the Docker PS command. If you add the `-a` (for all) switch, as in the following example, the command displays all of the containers on the host, whether running or not, as shown in Figure 4-12.

```
docker ps -a
```

```
PS C:\WINDOWS\system32> docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS          NAMES
dbf9674d13b9   microsoft/windowsservercore        "powershell"           4 hours ago   Up 22 minutes                focused_gplick
0e38bdac48ca   microsoft/windowsservercore        "powershell"           9 hours ago   Up 9 hours                drunk_jones
2270ee954537   microsoft/iis                       "C:\ServiceMonitor..." 41 hours ago   Exited (0) 41 hours ago      admiring_ferhat
38105f3fd40e   microsoft/sample-dotnet            "dotnet dotnetbot.dll"   2 days ago    Exited (0) 2 days ago      prickly_engelbart
PS C:\WINDOWS\system32>
```

FIGURE 4-12 Output of the Docker ps a command

## Starting and stopping containers

To start a stopped container, you use the Docker Start command, as in the following example:

```
docker start dbf9674d13b9
```

You can also forcibly stop a container by using the Docker Stop command, as follows:

```
docker stop dbf9674d13b9
```

The six-byte hexadecimal string in these commands is the Container ID that Docker assigns to the container when creating it. You use this value in Docker commands to identify the container that you want to manage. This value also becomes the container's computer name, as you can see if you run `Get-ComputerInfo` from within a container session.

If you run Docker PS with the `--no-trunc` (for no truncation) parameter, as shown in Figure 4-13, you can see that the Container ID is a 32-byte hexadecimal string, although it is far more convenient to use just the first six bytes on the command line.

```
PS C:\WINDOWS\system32> docker ps -a --no-trunc
CONTAINER ID   PORTS          NAMES                                     IMAGE                                COMMAND                  CREATED        STATUS
dbf9674d13b9   3991f5e911ed6c037017f07398431af4fe7ff89e248ae8e9107   focused_gplick   microsoft/windowsservercore        "powershell"           3 hours ago   Exited (0) 19 se
cndle_ago
0e38bdac48ca   0120eff6491a7b9d1908e65180213b2c1707b924991ae8d1504f   drunk_jones     microsoft/windowsservercore        "powershell"           9 hours ago   Up 9 hours
2270ee954537   fca809c176d126221e832a9e6750c865e17b3088adf3e9c3f     admiring_ferhat microsoft/iis                       "C:\ServiceMonitor.exe w3svc cmd" 42 hours ago   Exited (0) 42 ho
urs_ago
38105f3fd40e   780150ff3b0509c64a6316a17bc6461698304299e3488147c0b   prickly_engelbart microsoft/sample-dotnet            "dotnet dotnetbot.dll"   2 days ago    Exited (0) 2 day
s_ago
PS C:\WINDOWS\system32>
```

FIGURE 4-13 Output of the Docker ps -a --no-trunc command

## Attaching to containers

To connect to a session on a running container, use the Docker Attach command, as in the following example:

```
docker attach dbf9674d13b9
```

Running the command in multiple windows opens additional sessions, enabling you to work in multiple windows at once.

## Creating images

If you have modified a container in any way, you can save the modifications to a new image by running the Docker Commit command, as in the following example:

```
docker commit dbf9674d13b9 hholt/killerapp:1.5
```

This command creates a new image called hholt/killerapp with a tag value of 1.5. The Docker Commit command does not create a duplicate of the base image with the changes you have made; it only saves the changes. If, for example, you use the Microsoft/windowsservercore base image to create the container, and then you install your application, running Docker Commit will only save the application. If you provide the new image to a colleague, she must have (or obtain) the base image, in order to run the container.

## Removing containers

To remove a container completely, use the Docker RM command, as shown in the following example:

```
docker rm dbf9674d13b9
```

Containers must be in a stopped state before you can remove them this way. However, adding the `-f` (for force) switch will cause the Docker RM command to remove any container, even one that is running.

## Manage Windows or Linux containers using Windows PowerShell

As mentioned earlier, the Dockerd engine does not require the use of the Docker.exe client program. Because Docker is an open source project, it is possible to create an alternative client implementation that you can use with Dockerd, and Microsoft, in cooperation with the Docker community, is doing just that in creating a PowerShell module that you can use to create and manage Docker containers.

Because the Docker module for PowerShell is under development, it does not necessarily support all of the functions possible with the Docker.exe client. However, the primary functions are there, as shown in the following sections.

## Listing containers

You can display a list of all the containers on the host by running the Get-Container cmdlet in Windows PowerShell, as shown in Figure 4-14. Unlike the Docker PS command, the Get-Container cmdlet displays all of the containers on the host, whether they are running or stopped.

```
PS C:\WINDOWS\system32> get-container
ID                Image                Command                Created                Status                Name
--                -
080096dce22901167... microsoft/wi... powershell            11/5/2016 9:14:09 AM  Up 5 minutes         infallible_mccarthy
d8d297343e8a1c27c... microsoft/wi... powershell            11/5/2016 6:26:51 AM  Exited (1067) 54 ... small_brown
dbf9674d13b91f3e9... microsoft/wi... powershell            11/4/2016 10:39:56 PM  Up 4 hours           focused_golick
0e38bdac48ca0120e... microsoft/wi... powershell            11/4/2016 6:09:47 PM  Up 16 hours          drunk_jones
2270ee954537765fc... microsoft/iis  C:\ServiceMonitor... 11/3/2016 9:43:16 AM  Exited (0) 2 days... admiring_fermat
38105f3fda0e78015... microsoft/sa... dotnet dotnetbot.dll 11/2/2016 5:41:28 AM  Exited (0) 3 days... prickly_engelbart
PS C:\WINDOWS\system32>
```

FIGURE 4-14 Output of the Get-Container cmdlet

## Starting and stopping containers

When you create a container using the New-Container cmdlet, the container is not started by default. You must explicitly start it. To start a stopped container, you use the Start-Container cmdlet, as in the following example:

```
start-container dbf9674d13b9
```

You can also stop a container by simply changing the verb to the Stop-Container cmdlet, as follows:

```
stop-container dbf9674d13b9
```

## Attaching to containers

To connect to a session on a running container, use the Enter-ContainerSession cmdlet, as in the following example:

```
Enter-containersession dbf9674d13b9
```

This cmdlet is also aliased as Attach-Container, enabling to reuse another command with just a verb change.

## Creating images

If you have modified a container in any way, you can save the modifications to a new image by running the ConvertTo-ContainerImage cmdlet, as in the following example:

```
convertto-containerimage -containeridorname dbf9674d13b9 -repository hho1t/killerapp -tag 1.5
```

This cmdlet is also aliased as Commit-Container.

## Removing containers

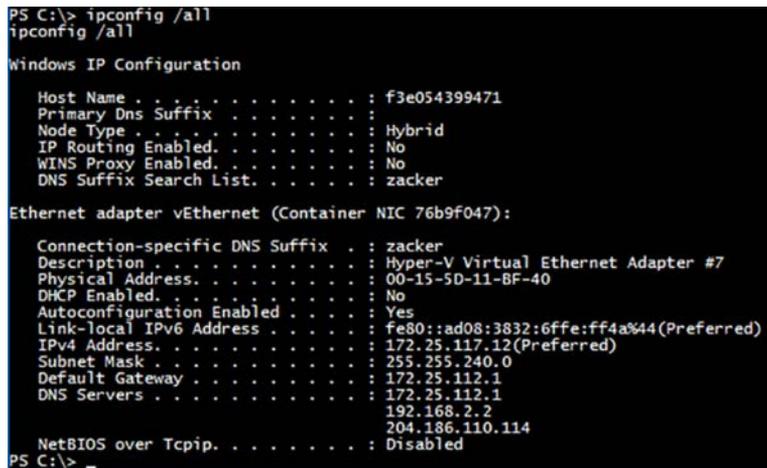
To remove a container, use the Remove-Container cmdlet, as shown in the following example:

```
remove-container dbf9674d13b9
```

As with the Docker RM command, containers must be in a stopped state before you can remove them. However, adding the Force switch will cause the cmdlet command to remove any container, even one that is running.

## Manage container networking

Containers can access the outside network. This is easy to prove, by pinging a server on the local network or the Internet. However, if you run the Ipconfig /all command in a container session, as shown in Figure 4-15, you might be surprised at what you see.



```
PS C:\> ipconfig /all
ipconfig /all

Windows IP Configuration

Host Name . . . . . : f3e054399471
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : zacker

Ethernet adapter vEthernet (Container NIC 76b9f047):

Connection-specific DNS Suffix . . : zacker
Description . . . . . : Hyper-V Virtual Ethernet Adapter #7
Physical Address. . . . . : 00-15-5D-11-BF-40
Dhcp Enabled. . . . . : No
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::ad08:3832:6ffe:ff4a%44(Preferred)
IPv4 Address. . . . . : 172.25.117.12(Preferred)
Subnet Mask . . . . . : 255.255.240.0
Default Gateway . . . . . : 172.25.112.1
DNS Servers . . . . . : 172.25.112.1
                          192.168.2.2
                          204.186.110.114
NetBIOS over Tcpip. . . . . : Disabled

PS C:\>
```

FIGURE 4-15 Output of Ipconfig /all command on a container

In this example, the IP address of the network adapter in the container is 172.25.117.12/12, which is nothing like the address of the network on which the container host is located. However, if you run the Ipconfig /all command on the container host, as shown in Figure 4-16, the situation becomes clearer.

```

PS C:\WINDOWS\system32> ipconfig /all

Windows IP Configuration

Host Name . . . . . : CZ10
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : zacker

Ethernet adapter vEthernet (HNS Internal NIC):

Connection-specific DNS Suffix . . . :
Description . . . . . : Hyper-V Virtual Ethernet Adapter #4
Physical Address. . . . . : 00-15-5D-11-BB-AC
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::49c7:9ebd:f079:2994%29(Preferred)
IPv4 Address. . . . . : 172.25.112.1(Preferred)
Subnet Mask . . . . . : 255.255.240.0
Default Gateway . . . . . :
DHCPv6 IAID . . . . . : 486544733
DHCPv6 Client DUID. . . . . : 00-01-00-01-1F-96-45-81-44-37-E6-C0-9D-DF
DNS Servers . . . . . : fec0:0:0:ffff::1%1
                       fec0:0:0:ffff::2%1
                       fec0:0:0:ffff::3%1
NetBIOS over Tcpip. . . . . : Enabled

Ethernet adapter vEthernet (Intel(R) 82579LM Gigabit Network Connection):

Connection-specific DNS Suffix . . . : zacker
Description . . . . . : Hyper-V Virtual Ethernet Adapter #2
Physical Address. . . . . : 44-37-E6-C0-9D-DF
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::e170:47de:5b5a:d24b%4(Preferred)
IPv4 Address. . . . . : 192.168.2.41(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : Wednesday, November 2, 2016 12:32:22 AM
Lease Expires . . . . . : Monday, November 14, 2016 12:32:22 AM
Default Gateway . . . . . : 192.168.2.99
DHCP Server . . . . . : 192.168.2.2
DHCPv6 IAID . . . . . : 205797350
DHCPv6 Client DUID. . . . . : 00-01-00-01-1F-96-45-81-44-37-E6-C0-9D-DF
DNS Servers . . . . . : 192.168.2.2
                       204.186.110.114
NetBIOS over Tcpip. . . . . : Enabled

```

**FIGURE 4-16** Output of Ipconfig /all command on a container host

There are two Ethernet adapters showing on the container host system. One has an IP address on the 192.168.2.0/24 network, which is the address used for the physical network to which the container host is connected. The other adapter has the address 172.25.112.1/12, which is on the same network as the container's address. In fact, looking back at the container's configuration, the container host's address is listed as the Default Gateway and DNS Server address for the container. The container host is, in essence, functioning as a router between the 172.16.0.0/12 network on which the container is located and 192.168.2.0/24, which is the physical network to which the host is connected. The host is also functioning as the DNS server for the container.

If you look at another container on the same host, it has an IP address on the same network as the first container. The two containers can ping each other's addresses, as well as those of systems outside the 172.16.0.0/12 network.

This is possible because the Containers feature and Docker use network address translation (NAT) by default, to create a networking environment for the containers on the host. NAT is a routing solution in which the network packets generated by and destined for a system have their IP addresses modified, to make them appear as though the system is located on another network.

When you ping a computer on the host network from a container session, the container host modifies the ping packets, substituting its own 192.169.2.43 address for the container's

172.25.117.12 address in each one. When the responses arrive from the system being pinged, the process occurs in reverse.

The Dockerd engine creates a NAT network by default when runs for the first time, and assigns each container an address on that NAT network. The use of the 172.16.0.0/12 network address is also a default coded into Docker. However, you can modify these defaults, by specifying a different NAT address or by not using NAT at all.

The network adapters in the containers are, of course, virtual. You can see in the configuration shown earlier that the adapter for that container is identified as vEthernet (Container NIC 76b9f047). On the container host, there is also a virtual adapter, called vEthernet (HNS Internal NIC). HNS is the Host Network Service, which is the NAT implementation used by Docker. If you run the Get-VMSwitch cmdlet on the container host or look in the Virtual Switch Manager in Hyper-V Manager, as shown in Figure 4-17, you can see that Docker has also created virtual switch called nat. This is the switch to which the adapters in the containers are all connected. Therefore, you can see that containers function much like virtual machines, as far as networking is concerned.

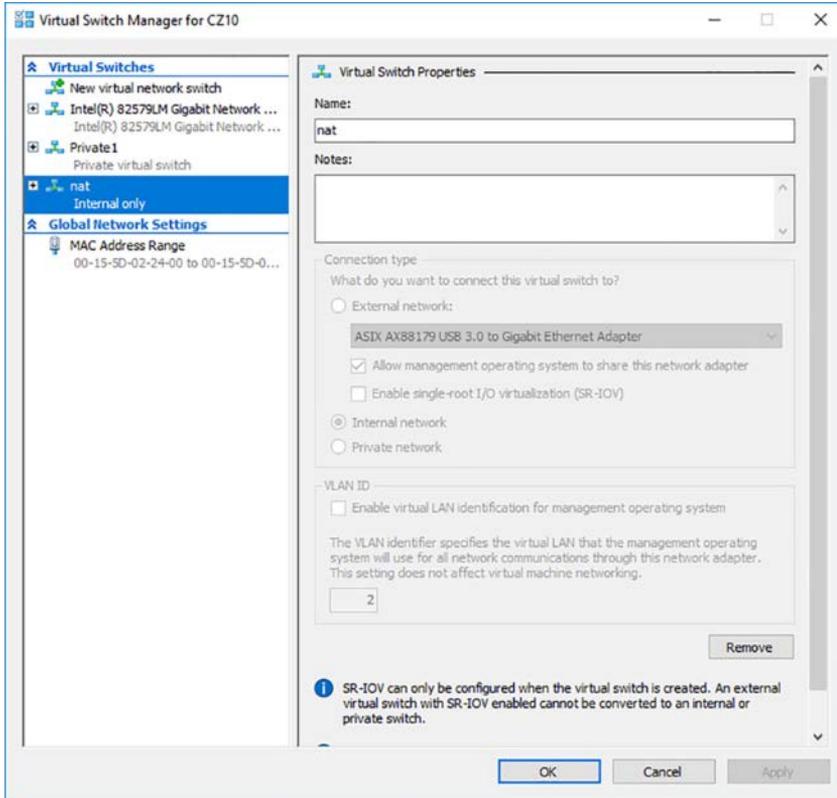


FIGURE 4-17 Nat switch in the Virtual Switch Manager

## Modifying NAT defaults

If you want to use a different network address for Docker's NAT configuration, because you already have a network using that same address, for example, it is possible to do so. To specify an alternate address, you must use the `daemon.json` configuration file, as discussed earlier in the remote Docker client configuration.

`Daemon.json` is a plain text file that you create in the directory where the `Dockerd.exe` program is located. To specify an alternate NAT network address, you include the following text in the file:

```
{ "fixed-cidr": "192.168.10.0/24" }
```

You can use any network address for the NAT implementation, but to prevent address conflicts on the Internet, you should use a network in one of the following reserved private network addresses:

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

To prevent the Dockerd engine from creating any network implementation at all, place the following text in the `daemon.json` file:

```
{ "bridge": "none" }
```

If you do this, you must manually create a container network, if you want your containers to have any network connectivity.

## Port mapping

If you plan to run a server application in a container that must expose ports for incoming client traffic, you must use a technique called *port mapping*. Port mapping enables the container host, which receives the client traffic, to forward the packets to the appropriate port in the container running the application. To use port mapping, you add the `-p` switch to the Docker Run command, along with the port numbers on the container host and the container, respectively, as in the following example:

```
docker run -it -p 8080:80 microsoft\windowsservercore powershell
```

In this example, any traffic arriving through the container host's port 8080 will be forwarded to the container's port 80. Port 80 is the well-known port number for web server traffic, and this arrangement enables the container to use this standard port without monopolizing it on the container host, which might need port 80 for its own web server.

## Creating a transparent network

Instead of using NAT, you can choose to create a transparent network, one in which the containers are connected to the same network as the container host. If the container host is a physical computer, the containers are connected to the physical network. If the container host is a virtual machine, the containers are connected to whatever virtual switch the VM uses.

Docker does not create a transparent network by default, so you must create it, using the Docker Network Create command, as in the following example:

```
docker network create -d transparent trans
```

In this example, the command creates a new network using the transparent driver, signified by the `-d` switch, and assigns it the name `trans`. Running the following command displays a list of all the container networks, which now includes the `trans` network you just created, as shown in Figure 4-18.

```
docker network ls
```



```
PS C:\Users\Administrator> docker network ls
NETWORK ID          NAME      DRIVER      SCOPE
4935e862cb65       nat      nat         local
37d5846ae474       none     null        local
9b62d68c1d58       trans    transparent  local
PS C:\Users\Administrator>
```

**FIGURE 4-18** Output of the Docker Network LS command

Once you have created the transparent network, you can create containers that use it by adding the network parameter to your Docker Run command, as in the following example:

```
docker run -it --network=trans microsoft/windowsservercore powershell
```

When you run the `ipconfig /all` command in this container, you can see that it has an IP address on the `10.0.0.0/24` network, which is the same as the network used by the virtual machine functioning as the container host.

When you create a transparent network and the containers that use it, they all obtain IP addresses from a DHCP on the container host network, if one is available. If there is no DHCP server available, however, you must specify the network address settings when creating the network and manually configure the IP address of each container by specifying it on the Docker Run command line.

To create a transparent network with static IP addresses, you use a command like the following:

```
docker network create -d transparent --subnet=10.0.0.0/24 --gateway=10.0.0.1 trans
```

Then, to create a container with a static IP address on the network you created, you use a Docker Run command like the following:

```
docker run -it --network=trans --ip=10.0.0.16 --dns=10.0.0.10 microsoft/
windowsservercore powershell
```

## Manage container data volumes

In some instances, you might want to preserve data files across containers. Docker enables you to do this by creating data volumes on a container that correspond to a folder on the container host. Once created, the data you place in the data volume on the container is also found in the corresponding folder on the container host. The opposite is also true; you can copy files into the folder on the host and access them in the container.

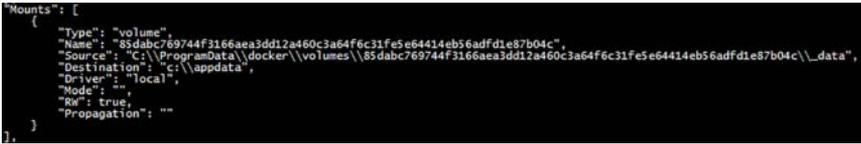
Data volumes persist independent of the container. If you delete the container, the data volume remains on the container host. You can then mount the container host folder in another container, enabling you to retain your data through multiple iterations of an application running in your containers.

To create a data volume, you add the `-v` switch to a Docker Run command, as in the following example:

```
docker run -it -v c:\appdata microsoft/windowsservercore powershell
```

This command creates a folder called `c:\appdata` in the new container and links it to a subfolder in `C:\ProgramData\docker\volumes` on the container host. To learn the exact location, you can run the following command and look in the `Mounts` section, as shown in Figure 4-19.

```
docker inspect dbf9674d13b9
```



```
"Mounts": [
  {
    "Type": "volume",
    "Name": "85dabc769744f3166aea3dd12a460c3a64f6c31fe5e64414eb56adfd1e87b04c",
    "Source": "C:\\ProgramData\\docker\\volumes\\85dabc769744f3166aea3dd12a460c3a64f6c31fe5e64414eb56adfd1e87b04c\\_data",
    "Destination": "c:\\appdata",
    "Driver": "local",
    "Mode": "rw",
    "RW": true,
    "Propagation": ""
  }
],
```

**FIGURE 4-19** Partial output of the Docker Inspect command

The `Mounts` section (which is small part of a long, comprehensive listing of the container's specifications) contains `Source` and `Destination` properties. `Destination` specifies the folder name in the container, and `Source` is the folder on the container host. To reuse a data volume, you can specify both the source and destination folders in the Docker Run command, as in the following example:

```
docker run -it -v c:\sourcedata:c:\appdata microsoft/windowsservercore powershell
```

If you create a data volume, specifying a folder on the container that already contains files, the existing contents are overlaid by the data volume, but are not deleted. Those files are accessible again when the data volume is dismounted.

By default, Docker creates data volumes in read/write mode. To create a read-only data volume, you can add `:ro` to the container folder name, as in the following example:

```
docker run -it -v c:\appdata:ro microsoft/windowsservercore powershell
```

#### **NOTE** ADDING A DATA VOLUME

To add a data volume to an existing container, your only option is to use Docker Commit to save any changes you've made to the existing container to a new image, and then use Docker Run to create a new container from the new image, including the `-v` switch to add the data volume.

## Manage resource control

As noted earlier, the Docker Run command supports many parameters and switches, some of which have already been demonstrated in this chapter. For example, you have seen how the `it` switches create an interactive container that runs a specific shell or other command. To create a container that runs in the background—in what is called detached mode—you use the `-d` switch, as in the following example:

```
docker run -d -p 80:80 microsoft/iis
```

To interact with a detached container, you can use network connections or file system shared. You can also connect to the container using the Docker Attach command.

## Working with container names

By default, when you create a container using the Docker Run command, the Dockerd engine assigns three identifiers to the container, as shown in Figure 4-20:

- **Long UUID** A 32-byte hexadecimal string, represented by 64 digits, as in the following example: `0e38bdac48ca0120eff6491a7b9d1908e65180213b2c1707b924991ae8d1504f`
- **Short UUID** The first six bytes of the long UUID, represented as 12 digits, as in the following example: `0e38bdac48ca`.
- **Name** A randomly chosen name consisting of two words separated by an underscore character, as in the following example: `drunk_jones`



CONTAINER ID	STATUS	PORTS	NAMES	IMAGE	COMMAND	CREATED
0e38bdac48ca0120eff6491a7b9d1908e65180213b2c1707b924991ae8d1504f	Up 32 minutes		drunk_jones	microsoft/windowsservercore	"powershell"	3 days ago

**FIGURE 4-20** Output of the Docker `ps --no-trunc` command

You can use any of the three identifiers when referencing the container on the command line. You can also assign your own name to the container when you create it by adding the `name` parameter to the Docker Run command line, as in the following example:

```
docker run -it microsoft/windowsservercore powershell --name core1
```

## Constraining memory

The Docker Run command supports parameters that enable you to specify how much memory a container is permitted to use. By default, container processes can use as much host memory and swap memory as they need. If you are running multiple containers on the same host or a memory intensive application on the host itself, you might to impose limits on the memory certain containers can use.

The memory parameters you can use in a Docker Run command are as follows:

- **-m (or --memory)** Specifies the amount of memory the container can use. Values consist of an integer and the unit identifier b, k, m, or g (for bytes, kilobytes, megabytes, or gigabytes, respectively).
- **-memory-swap** Specifies the total amount of memory plus virtual memory that the container can use. Values consist of an integer and the unit identifier b, k, m, or g.
- **-memory-reservation** Specifies a soft memory limit that the host retains for the container, even when there is contention for system memory. For example, you might use the -m switch to set a hard limit of 1 GB, and a memory reservation value of 750 MB. When other containers or processes require additional memory, the host might reclaim up to 250 MB of the container's memory, but will leave at least 750 MB intact. Values consist of an integer smaller than that of the m or --memory-swap value and the unit identifier b, k, m, or g.
- **-kernel-memory** Specifies the amount of the memory limit set using the -m switch that can be used for kernel memory. Values consist of an integer and the unit identifier b, k, m, or g.
- **-oom-kill-disable** Prevents the kernel from killing container processes when an out of memory error occurs. Never use this option without the -m switch, to create a memory limit for the container. Otherwise, the kernel could start to kill processes on the host when an OOM error occurs.

## Constraining CPU cycles

You can also specify parameters that limit the CPU cycles allocated to a container. By default, all the containers on a host share the available CPU cycles equally. Using these parameters, you can assign priorities to the containers, which take effect when cpu contention occurs.

The Docker Run parameters that you can use to control container access to CPUs are as follows:

- **-c (or --cpu-shares)** Specifies a value from 0 to 1024 that specifies the weight of the container in contention for the CPU cycles. The actual amount of processor cycles that a container receives depends on the number of containers running on the host and their respective weights.
- **-cpuset-cpus** Specifies which CPUs in a multiprocessor host system that the container can use. Values consist of integers representing the CPUs in the host computer, separated by commas.

- **-cpuset-mems** Specifies which nodes on a NUMA host that the container can use. Values consist of integers representing the CPUs in the host computer, separated by commas.

## Create new container images using Dockerfile

If you have made changes to a container since you first created it with the Docker Run command, you can save those changes by creating a new container image using Docker Commit. However, the recommended method for creating container images is to build them from scratch using a script called a dockerfile.

A *dockerfile* is a plain text file, with the name `dockerfile`, which contains the commands needed to build your new image. Once you have created the `dockerfile`, you use the Docker Build command to execute it and create the new file. The `dockerfile` is just a mechanism that automates the process of executing the steps you used to modify your container manually. When you run the Docker Build command with the `dockerfile`, the Dockerd engine runs each command in the script by creating a container, making the modifications you specify, and executing a Docker Commit command to save the changes as a new image.

A `dockerfile` consists of instructions, such as FROM or RUN, and a statement for each instruction. The accepted format is to capitalize the instruction. You can insert remarks into the script by preceding them with the pound (#) character.

An example of a simple `dockerfile` is as follows:

```
#install DHCP server
FROM microsoft/windowsservercore
RUN powershell -command install-windowsfeature dhcp -includemanagementtools
RUN powershell -configurationname microsoft.powershell -command add-dhcpserverv4scope
-state active -activatepolicies $true -name scopetest -startrange 10.0.0.100 -endrange
10.0.0.200 -subnetmask 255.255.255.0
RUN md boot
COPY ./bootfile.wim c:/boot/
CMD powershell
```

In this example:

- The FROM instruction specifies the base image from which the new image is created. In this case, the new image starts with the `microsoft/windowsservercore` image.
- The first RUN command opens a PowerShell session and uses the `Install-WindowsFeature cmdlet` to install the DHCP role.
- The second RUN command uses the `Add-DhcpServerv4Scope cmdlet` to create a new scope on the DHCP server.
- The third RUN command creates a new directory called `boot`.
- The COPY command copies a file called `bootfile.wim` from the current folder on the container host to the `c:\boot` folder on the container.
- The CMD command opens a PowerShell session when the image is run.

Once you have created the dockerfile script, you use the Docker Build command to create the new image, as in the following example:

```
docker build -t dhcp .
```

This command reads the dockerfile from the current directory and creates an image called dhcp. As the Dockerd engine builds the image, it displays the results of each command and the IDs of the interim containers it creates, as shown in Figure 4-21. Once you have created the image, you can then create a container from it using the Docker Run command in the usual manner.

```
PS C:\Temp> docker build --no-cache --force-rm -t dhcp .
Sending build context to Docker daemon 8.192 kB
Step 1/6 : FROM microsoft/windows-server
----> 93a9c37b1640
Step 2/6 : RUN powershell install-windowsfeature dhcp -includemanagementtools
----> Running in 71a7f8f39e4f
Success Restart Needed Exit Code      Feature Result
-----
True      No          Success          (OKP_Server)
----> 92a23f3b115c
Removing intermediate container 71a7f8f39e4f
Step 3/6 : RUN powershell -command Add-dhcpserver -scope -state active -activatepolicies $true -name scopetest -starange 172.25.112.10 -end
range 172.25.112.20 -subnetmask 255.255.240.0
----> Running in 41b6925264f7
----> bef09e926548
Removing intermediate container 41b6925264f7
Step 4/6 : RUN md boots
----> Running in 2d6d8cb82ee7
----> 0da72c3a2291
Removing intermediate container 2d6d8cb82ee7
Step 5/6 : COPY ./bootfile.win c:/boots/
----> 5607aa0f2e60
Removing intermediate container 53480275c410
Step 6/6 : CMD powershell
----> Running in 4f23c9326a9d
----> bec3bf056322
Removing intermediate container 4f23c9326a9d
Successfully built bec3bf056322
PS C:\Temp>
```

FIGURE 4-21 Output of the Docker Build command

This is a simple example of a dockerfile, but they can be much longer and more complex.

### ✓ Quick check

Which of the following Docker commands can you use to create new container image files?

1. Docker Run
2. Docker Commit
3. Docker Build
4. Docker Images

### Quick check answer

Answers 2 and 3 are correct. Docker Commit is the command used to create a new image from an existing container. Docker Build is the command used to create a new container image using the instructions in a dockerfile.

# Manage container images using DockerHub Repository for public and private scenarios

DockerHub is a public repository that you can use to store and distribute your container images. When you download container images using the Docker Pull command, they come from DockerHub by default, unless you specify another repository in the command. However, you can upload images as well, using the Docker Push command.

Uploading images to DockerHub enables you to share them with your colleagues, and even with yourself, so you don't have to transfer files manually to deploy a container image on another host.

Before you can upload images to the Docker Hub, you must register at the site at <http://hub.docker.com>. Once you have done this, your user name becomes the name of your repository on the service. For example, the `microsoft/windowsservercore` image you pulled earlier is an image called `windowsservercore` in the Microsoft repository. If your user name on DockerHub is `hholt`, your images will all begin with that repository name, followed by the image name, as in the following example:

```
hholt/nano1
```

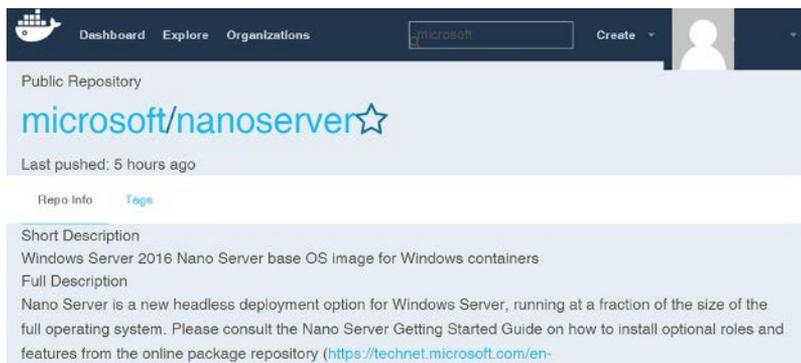
Once you have an account, you must login to the DockerHub service from the command line before you can push images. You do this with the following command:

```
docker login
```

Docker prompts you for your user name and password, and then provides upload access to your repository.

## Searching for images

You can search for images on the DockerHub by using the web site, as shown in Figure 4-22. This interface provides the latest information about the image, as well as comments from other users in the Docker community.



**FIGURE 4-22** Screen capture of a DockerHub web search

You can also search the DockerHub from the command line, using the Docker Search command, as in the following example:

```
docker search microsoft --no-trunc
```

Adding the no-trunc parameter prevents the command from truncating the image descriptions, as shown in Figure 4-23.

```
PS C:\temp> docker search microsoft --no-trunc
NAME                DESCRIPTION                STARS    OFFICIAL    AUTOMATED
microsoft/aspnet    ASP.NET is an open source server-side web application framework    498      [OK]        [OK]
microsoft/dotnet    Official images for .NET Core for Linux and Windows Server 2016 Nano Server    331      [OK]        [OK]
mono                Mono is an open source implementation of Microsoft's .NET Framework    196      [OK]        [OK]
microsoft/windowsservercore    Windows Server 2016 Server Core base OS image for Windows containers    71      [OK]        [OK]
microsoft/nanoserver    Windows Server 2016 Nano Server base OS image for Windows containers    68      [OK]        [OK]
microsoft/azure-cm    Docker image for Microsoft Azure Command Line Interface    66      [OK]        [OK]
microsoft/isis       Internet Information Services (IIS) installed in a Windows Server Core based container    10      [OK]        [OK]
microsoft/mssql-server-2014-express-windows    Microsoft SQL Server 2014 Express installed in Windows Server Core based containers.    42      [OK]        [OK]
microsoft/mssql-server-2016-express-windows    Microsoft SQL Server 2016 Express installed in Windows Server Core based containers.    39      [OK]        [OK]
microsoft/aspnetcore    Official images for running compiled ASP.NET Core applications.    29      [OK]        [OK]
microsoft/dotnet-framework    The Official Docker images for .NET Framework on Windows Server 2016 Server Core.    11      [OK]        [OK]
microsoft/powershell    Official PowerShell Core releases from https://github.com/PowerShell/PowerShell/releases    9        [OK]        [OK]
microsoft/oms        Monitor your containers using the Operations Management Suite (OMS). For minimal OS like CoreOS.    7        [OK]        [OK]
microsoft/aspnetcore-build    Application Insights for Docker helps you monitor your containerized applications.    4        [OK]        [OK]
microsoft/vsts-agent    Official images for the Visual Studio Team Services (VSTS) agent.    4        [OK]        [OK]
microsoft/dotnet35    The .NET Framework 3.5 image has moved to microsoft/dotnet-framework:3.5    2        [OK]        [OK]
microsoft/dotnet-nightly    Preview bits of the .NET Core CLI    2        [OK]        [OK]
microsoft/powershell-nightly    Nightly builds of PowerShell Core for CI    0        [OK]        [OK]
csharpinc/microsoft-prep70533    Images to build preview versions of ASP.NET Core applications.    0        [OK]        [OK]
herliuss/microsoft-malware    Microsoft-malware - artificial intelligence - training agents on KINECRAFT    0        [OK]        [OK]
microsoft/dotnet-samples    .NET Core Docker Samples    0        [OK]        [OK]
microsoft/ons        ONS    0        [OK]        [OK]
dreher/microsoft     Microsoft Test Repo    0        [OK]        [OK]
PS C:\temp>
```

FIGURE 4-23 Output of the Docker Search command

## Pushing images

To upload your own images to the repository, you use the Docker Push command, as in the following example:

```
docker push hho1t/nano1
```

By default, the Docker Push command uploads the specified image to your public repository on the DockerHub, as shown in Figure 4-24. Anyone can access images pushed in this way.

```
PS C:\temp> docker push craigz3/nano1
The push refers to a repository [docker.io/craigz3/nano1]
0bc9597871ad: Pushed
2c195a33d84d: Skipped foreign layer
342d4e407550: Skipped foreign layer
0.9: digest: sha256:1518e997672c384bad70aeb897de64689Fef78Fcb420e9fda86b933e25f1cd6b size: 1155
PS C:\temp>
```

FIGURE 4-24 Output of the Docker Push command

Because Docker is open source software, sharing images and code with the community is a large part of the company's philosophy. However, it is also possible to create private repositories, which you can share with an unlimited number of collaborators you select. This enables you to use DockerHub for secure application development projects or any situation in which you do not want to deploy an image to the public. DockerHub provides a single private repository as part of its free service, but for additional repositories, you must purchase a subscription.

In addition to storing and providing images, DockerHub provides other services as well, such as automated builds. By uploading a dockerfile and any other necessary files to a repository, you can configure DockerHub to automatically execute builds for you, to your exact

specifications. The code files are available to your collaborators, and new builds can occur whenever the code changes.

## Manage container images using Microsoft Azure

In addition to creating containers locally, you can also use them on Microsoft Azure. By creating a Windows Server 2016 virtual machine on Azure, you can create and manage containers just as you would on a local server. Azure also provides the Azure Container Service (ACS), which enables you to create, configure, and manage a cluster of virtual machines, configured to run container-based applications using various open source technologies.

Microsoft Azure is a subscription-based cloud service that enables you to deploy virtual machines and applications and integrate them into your existing enterprise. By paying a monthly fee, you can create a Windows Server 2016 virtual machine, as shown in Figure 4-25. Once you have created the virtual machine, you can install the Containers feature and the Docker engine. Containers and images that you create on an Azure virtual machine are completely compatible with the Docker implementations on your local computers.

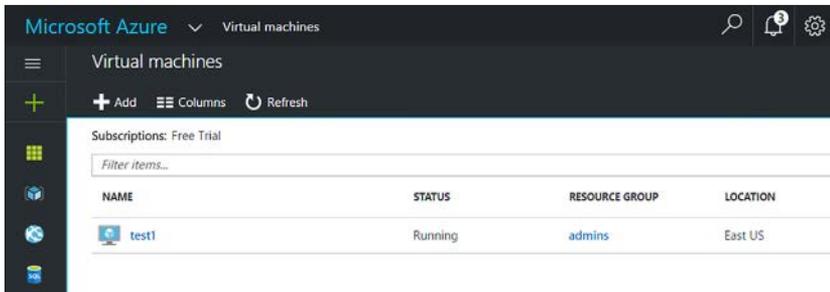


FIGURE 4-25 Microsoft Azure Resource Center

## Chapter summary

- Containers are based on images. You create a container by running an image, and you create an image by saving the contents of a container.
- Windows Server 2016 includes the Containers feature, which provides the support environment for the Docker platform.
- Both the Server Core and Nano Server installation options support the creation of Windows Server and Hyper-V containers. In Nano Server, you can run the Docker.exe client on a remote system.
- Docker is an open source container solution that consists of two files: Dockerd.exe, which is the engine that runs as a service on Windows, and Docker.exe, which is the command line client that controls the Dockerd engine.

- Using a text file called `daemon.json`, you can configure start-up options for the Docker engine.
- The Docker client is one way to control the Docker engine, but it is not the only way. You can also use the Docker module for Windows PowerShell to perform the same tasks.
- To download images from the Docker Hub, you use the Docker Pull command.
- Tags are version indicators that developers can use to track the builds or versions of a container image. To assign tag values, you use the Docker Tag command.
- To uninstall a container image, you use the Docker RMI command.
- To create a Windows Server container, you use the Docker Run command, specifying the name of a container image.
- The procedure for creating a Hyper-V container using Docker differs from a Windows Server container only in the inclusion of the `--isolation` parameter.
- The Docker.exe client enables you to control containers by starting, stopping, saving, and removing them.
- The Docker module for Windows PowerShell provides an alternative to the Docker.exe client that can perform most, if not all, of the same functions.
- By default, Docker uses network address translation to provide containers with network access. However, you can override the default and configure containers to be part of your larger network.
- Docker enables you to create data volumes that exist on the container host and add them to a container. Data volumes remain in place, even if you remove the container itself.
- Using parameters on the Docker Run command line, you can limit the amount of memory and CPU resources a container is permitted to use.
- A `dockerfile` is a script that contains instructions for building a new container image. You use the Docker Build command to execute the script and create the image.
- Docker Hub is a free repository, based in the cloud, on which you can upload your
- Microsoft Azure enables you to create virtual machines that you can use as container hosts.

## Thought experiment

---

In this thought experiment, demonstrate your skills and knowledge of the topics covered in this chapter. You can find answer to this thought experiment in the next section.

Ralph wants to create a virtual machine called Core1 that functions as a container host for both Windows Server and Hyper-V containers. To create the container host, he plans to perform the following tasks:

- Create a virtual machine.
- Configure the virtual machine with 4 GB of memory, two virtual processors, and MAC address spoofing enabled.
- Install Windows Server 2016 on the virtual machine.
- Install the Containers feature.
- Install the Hyper-V role.
- Install the dockermstprovider module.
- Install the Docker package.
- Pull the Server Core image from DockerHub.
- Create containers using the Docker Run command.

What step has Ralph forgotten, that prevents him from creating the containers he needs? What task must he perform to complete his plan, and when should he complete it?

## Thought experiment answer

---

This section contains the solution to the thought experiment.

Ralph has forgotten to expose the virtualization extensions of the physical computer's processor to the VM, so that it can run the Hyper-V role. To do this, he must run the following command in a PowerShell session after creating the virtual machine and before he starts it:

```
set-vmprocessor -vmname server1 -exposevirtualizationextensions $true
```

*This page intentionally left blank*

# Index

## A

- access control entries (ACEs) 112, 115
- access control lists (ACLs) 112
- ACEs. *See* Access Control Entries (ACEs); *See* access control entries (ACEs)
- ACLL. *See* Attempt Copy Last Logs
- ACLs. *See* access control lists (ACLs)
- activation models 35–42
  - Active-Directory based 39–41
  - Automatic Virtual Machine Activation 41–42
  - Key Management Service 36–39
  - multiple activation keys 35–36
- activation threshold 37
- activation validity interval 37
- Active Directory
  - backing up 421–422
- Active Directory-based activation 39–41
- Active Directory-detached clusters 337
- Active Directory Domain Services (AD DS) 337
- Add-ClusterSharedVolume cmdlet 153
- Add-ClusterVirtualMachineRole cmdlet 304
- Add-ClusterVMMonitoredItem cmdlet 362
- Add-Computer cmdlet 20
- Add-ContainerImageTag cmdlet 272
- Add Roles And Features Wizard 11–15
- Add-VMNetworkAdapter cmdlet 237
- administrative access points 337
- Administrators groups 269
- advanced permissions 114–115, 120–121
- allocation unit size 82–84
- antimalware solution 405–410
- asynchronous replication 148
- Attach-Container cmdlet 280
- authentication protocols
  - for Live Migration 308–309

- authorization 113
- Automatic Virtual Machine Activation (AVMA) 41–42
- AVMA. *See* Automatic Virtual Machine Activation
- AVMAkey variable 42
- Azure
  - managing container images using 293
- Azure Access Panel. *See* Access Panel

## B

- backups
  - Active Directory 421–422
  - before upgrading 30
  - data deduplication and 162
  - failover clusters and 324–326
  - group policy objects 423
  - Hyper-V 424–425
  - IIS 424–425
  - incremental 417–418
  - mirrored 416
  - restores from 418–421
  - scheduled 415–417
  - single job creation 412–415
  - strategies for 421–425
  - Windows Server Backup 411–421
- balloon driver 188
- bandwidth management 254–256
- Basic Input/Output System (BIOS) 86
- basic permissions 114, 117–120
- BIOS settings 7
- blob files 45, 46–47
- Block-SmbShareAccess cmdlet 109
- boot
  - Secure Boot 205–208
  - traditional 205
- bottlenecks 433–434

## C

cabinet (CAB) files 48, 74

checkpoints

- applying 229–230
- creating 228–229
- managing 228–230
- production 230–231
- standard 230

child partitions 167

Chkdsk.exe 94

chunks 159

chunk store 158

churn 158, 161

Close-SmbOpenFile cmdlet 108

Close-SmbSession cmdlet 107

cloud-based services 43

cloud deployment 4

cloud witnesses 345–348

Cluster-Aware Updating (CAU) 328–332

Clustered Storage Spaces 342–345

cluster name object (CNO) 313, 337

Cluster Operating System Rolling Upgrade 332–333

cluster shared volume (CSV) 153

cluster shared volumes (CSVs) 333–336, 341

cluster-to-cluster configurations 149, 151–155

CNA. *See* converged network adapter (CNA)

collector technologies 65

Compare-VM cmdlet 212

connectors. *See also* receive connectors;  
*See also* send connectors

containers. *See* Linux containers;  
*See* Windows containers

converged network adapter (CNA) 143

converged networks 143

convergence 376

ConvertTo-ContainerImage cmdlet 280

Convert-VHD cmdlet 228

Copy-Item cmdlet 180

CPU counters 434–435

CPU cycles 288–289

Create New Data Collector Set wizard 438–439

Credential Security Support Provider (CredSSP) 308–309

CSV. *See* cluster shared volume (CSV)

## D

DAC. *See* Datacenter Activation Coordination

daemon.json 269–270

DAGs. *See* Database Availability Groups

DAS. *See* Direct-Attached Storage

databases. *See* mailbox databases

datacenter bridging (DCB) 142–145

Datacenter edition 4, 5, 41

data collector sets 431–433

data deduplication 155–162

- backup and restore solution with 162
- configuration of 155–158
- monitoring 160–161
- optimization rates 159
- usage scenarios for 158–160
- workload evaluation 159–160

Data Deduplication Savings Evaluation Tool 160

Data Protection Manager (DPM) 162

data replication 148–155

data storage. *See also* storage architectures; *See also* storage requirements

data volumes 286–287

DCB. *See* datacenter bridging (DCB)

DCBX Willing bit 143

DDA. *See* Discrete Device Assignment (DDA)

Ddpeval.exe 160

deduplication. *See* data deduplication

Deployment Image Servicing and Management (DISM.exe) 172

- adding drivers to image files using 72–74
- /disable-feature command 76
- /enable-feature command 76–77
- installing roles and features in offline images with 75–77
- umounting image with 74
- updating images with 70–72
- Windows PowerShell equivalents for 77–79

Desired State Configuration (DSC) 26–28

- creating configuration scripts 26
- deploying configurations 27–28

Desktop Experience 2

devices

- detecting 147

Device Specific Module (DSM) 145

- policies 148

DFS. *See* Distributed File Share

- differencing disks 222–223
  - /disable-feature command 76
  - Discrete Device Assignment (DDA) 212–213
  - discretionary access control lists (DACLS) 93
  - disk counters 436–437
  - disk fragmentation 83
  - Disk Management console 84, 139–140
    - creating VHD or VHDX files using 88–90
    - mounting VHD and VHDX files with 91–92
  - disk partitions 9–10
  - disks
    - adding to CSVs 336
    - differencing 222–223
    - GUID partition table 84–88
    - initializing new 84–85
    - MBR 84–85
    - partition style selection 87
    - pass-through 212, 225–226
    - physical 125, 225–226
    - storage layout options 125–131
    - virtual. *See* virtual disks
  - disk sectors
    - size configuration 82–84
  - disk volume
    - allocation unit size 82–84
  - DISM.exe. *See* Deployment Image Servicing and Management
  - Dismount-VHD cmdlet 92
  - Distributed Component Object Model (DCOM) 25
  - Distributed File System (DFS) Replication 150
  - Djoin.exe tool 46
  - DNS round robin 376
  - DNS server addresses 270
  - Docker
    - Attach command 279
    - Build command 290
    - Commit command 279
    - Images command 273
    - installation
      - on Nano Server 267–268
      - on Windows Server 266
    - managing containers with 277–279
    - Network Create command 285
    - PowerShell and 270–271
    - Pull command 271
    - Push command 292
    - RM command 279
    - Run command 274–276, 285, 287, 288
    - Start command 278
    - start-up options 269–270
    - Stop command 278
  - Dockerd.exe 266, 269
  - Docker.exe 266
  - dockerfile 289–290
  - DockerHub 271, 291–293
  - Domain Name System (DNS) 39
  - domains
    - joining, with Nano Server 45–47
  - drive arrays 134
  - drivers
    - adding to image files 72–74
  - DSC. *See* Desired State Configuration (DSC)
  - Dynamic Host Configuration Protocol (DHCP) 51
  - Dynamic Host Configuration Protocol (DHCP) server 11
  - dynamic least queue depth 148
  - dynamic memory
    - allocations 188
    - configuration 186–188
    - limitations 187
    - settings 186–188
  - dynamic quorum management 318
- ## E
- Edit-NanoServerImage cmdlet 49, 51, 77
  - EFS. *See* Encrypting File System
  - emulated adapters 248–249
  - Enable-DedupVolume cmdlet 157–158
  - /enable-feature command 76–77
  - Encrypting File System (EFS) 94
  - enhanced session mode 199–201
  - Enter-ContainerSession cmdlet 280
  - Enter-PsSession cmdlet 22, 56
  - ESRA. *See* EdgeSync replication account (ESRA)
  - Essentials edition 5, 6
  - Ethernet 142–143
  - Exit-PsSession cmdlet 22, 57
  - Exit-PSSession cmdlet 177
  - explicit remoting 176
  - Export-SmigServerSetting cmdlet 33
  - Export-VM cmdlet 210
  - Extended Page Tables (EPT) 263
  - Extensible Firmware Interface (EFI)-based
    - boot partition 87
  - external network switches 239, 241

**F**

- failback policy 148
- failbacks 364
- failover affinity 367
- failover clusters 153, 220–221, 304, 311–351
  - cloud witnesses 345–348
  - Cluster-Aware Updating 328–332
  - cluster configuration 324–326
  - Clustered Storage Spaces 342–345
  - cluster networking 321–324
  - Cluster Operating System Rolling Upgrade 332–333
  - cluster shared volumes 333–336, 341
  - configuring without network names 337
  - guest clustering 341–342, 349–351
  - managing 359–368
  - monitoring VMs in 361–363
  - node fairness 367–368
  - quorum 317–321
  - role-specific settings 359–361
  - Scale-out File Server 337–341
  - shared VHDX files 349–351
  - site-aware 365–367
  - storage configuration 326–328
  - stretch 365–367
  - VM resiliency and 348–349
  - workgroup, single, and multi-domain 314–317
- failover policy 148
- failovers 150
- failover settings 364–365
- fault tolerance 128–131
- features
  - implementation on Nano Server 48–50
  - installation of 13–15
    - in offline images 75–77
  - offline installation 225
- Fiber Channel over Ethernet (FCoE) 142
- Fibre Channel 133, 326
  - adapter 231–233
- file compression 94
- file ownership 122
- file permissions 112–122
- File Server cluster role 360
- File Server Resource Manager (FSRM) 103
- File Server role service 96
- File Sharing dialog box 95
- file systems
  - NTFS 93–95

- ReFS 93–95
- folder ownership 122
- folder permissions 112–122
- folder shares. *See* shares
- Format-List cmdlet 161
- FreeBSD
  - virtual machines 201–203
- FreeBSD deployments 61
- FreeBSD Integration Services (BIS) 61
- FreeBSD Integration Services (FIS) 204, 205
- FSW. *See* File Share Witness

**G**

- garbage collection 158
- Generation 1 VMs 197, 214
- Generation 2 VMs 197–199, 205, 215
- generic volume licensing keys (GVLKs) 39
- Get-Command cmdlet 21
- Get-ComputerInfo cmdlet 274
- Get-Container cmdlet 280
- Get-DedupStatus cmdlet 161
- Get-help cmdlet 21
- Get-NetAdapter cmdlet 19
- Get-NetAdapterVmqQueue cmdlet 252–253
- Get-SmbClientConfiguration cmdlet 111–112
- Get-SmbOpenFile cmdlet 108
- Get-SmbServerConfiguration cmdlet 109–110
- Get-SmbSession cmdlet 107
- Get-SmbShareAccess cmdlet 108
- Get-SmigServerFeature cmdlet 33
- Get-SRGroup cmdlet 154
- Get-VM cmdlet 177
- Get-VMHostSupportedVersion cmdlet 209
- Get-VM PowerShell cmdlet 208
- Get-WindowsFeature cmdlet 15
- globally-unique identifier (GUID) 86
- GPT. *See* GUID partition table (GPT) disks
- Grant-SmbShareAccess cmdlet 109
- Grant-SRAccess cmdlet 153
- Group policy objects (GPOs) 401–403, 409
  - backing up 423
- GRUB boot loader 202
- GUID partition table (GPT) disks
  - advantages of 86
  - booting from 87–88
  - compared with MBR 87
  - configuration of 84–88

## H

- hard disk drives (HDDs) 131
  - hard disks. *See* disks
  - hardware address 244–246
  - hardware requirements 3–4
  - high availability 297–386
    - failover clustering 311–351, 359–368
    - in Hyper-V 297–310
    - Live Migration 303–309, 369–370
    - network load balancing 375–384
    - Storage Migration 309–311
    - Storage Spaces Direct (S2D) 352–358
    - VM movement in clustered nodes 369–375
  - host bus adapter (HBA) 134
  - hotfixes 74–75
  - hot spares 130–131
  - hygiene. *See* message hygiene
  - hyperthreading 4
  - Hyper-V 165–258
    - backing up 424–425
    - checkpoints 228–231
    - containers 261–264, 275–277
    - converting from previous versions 208–209
    - Discrete Device Assignment 212–213
    - enhanced session mode 199–201
    - export and import functions 209–212
    - Fibre Channel adapter 231–233
    - guest operating systems 203–208
    - guests 165, 166
    - hardware limitations 167–169
    - high availability in 297–310
    - hosts 165, 174–179
    - installation 165–173
      - hardware and compatibility requirements 166–170
      - management tools 172–173
      - using PowerShell 171
      - using Server Manager 170–171
    - Integration Services 195–196, 204
    - Nano Server and 43
    - nested virtualization 181
    - networking 235–256
      - bandwidth management 254–256
      - MAC address configuration 244–246
      - network isolation 246–247
      - NIC teaming 249–251
      - performance optimization 243–244
      - Switch Embedded Teaming 253–254
      - synthetic network adapters 247–249
      - virtual machine queue 251–253
      - virtual network interface cards 236–237
      - virtual switches 238–242, 244, 247
    - New Virtual Hard Disk Wizard 88
    - permissions 174
    - PowerShell Direct 180
    - remote management 174–179
    - resource metering 193–195
    - smart paging 192–193
    - storage 213–235
      - differencing disks 222–223
      - quality of service 233–235
      - shared VHDX files 220–222
      - VHDs 214–220, 223–225
      - VHDX files 214–220
    - supported guest VMs 61
    - upgrading from existing versions of 173
    - virtual machine configuration 182–213
  - hypervisor 166–167
  - Hyper-V Manager 172–173
    - conflict handling 212
    - container host installation in 262–263
    - creating VHDs and VHDX files using 214–220
    - creating virtual hard disks in 216–218
    - importing VMs using 210–211
    - remote management using 174–176
    - virtual machine creation in 183–184
  - Hyper-V Replica 298–303
  - Hyper-V Server 168
  - Hyper-V Server edition 5
- ## I
- image files
    - adding drivers to 72–74
    - adding updates to 74–75
    - committing 74
    - container 261, 291–293
    - for deployment 58–79
    - installing roles and features in offline 75–77
    - managing, using Windows PowerShell 76–78
    - mounting 71–72
    - removing 273–274
    - umounting 74
    - updating 70–75
  - implicit remoting 177, 178–179
  - Import-SmigServerSetting cmdlet 33
  - Import-VM cmdlet 211–212
  - inheritance
    - permission 115–116

## initiators

- initiators
  - iSCSI 133–140
- in-place upgrades 28–32
- installation
  - MAP Toolkit 63
  - Nano Server 44–48
  - Server Core 17–19
  - upgrades 28–32
  - Windows Server 2016 1–18
    - activation models 35
    - clean installation 6–9
    - features and roles 11–17
    - mass deployment 11
    - partitions 9–10
    - requirements 2–4
  - Windows Server Migration Tools 33–34
- Install-WindowsFeature cmdlet 15, 171, 377
- Install-WindowsFeature PowerShell cmdlet 225
- Institute of Electrical and Electronics Engineers (IEEE) 143
- Integration Services 195–196, 204
- integrity scrubbing 158–159
- internal network switches 241
- Internet Information Services (IIS)
  - backing up 424–425
- Internet SCSI (iSCSI) 327
- Internet Small Computer System Interface (iSCSI) 133–140
  - creating targets 134–138
  - initiators and targets 133–134
  - using initiators 138–140
- Internet Storage Name Service (iSNS) 140–142
- Inventory And Assessment Wizard 65–67
- Invoke-Command cmdlet 180
- IP addresses
  - configuration
    - Nano Server 51–53
- iSNS Protocol (iSNSP) 141

## J

- just-a-bunch-of-disks (JBOD) arrays 123

## K

- Kerberos 308–309
- Key Management Service (KMS) 36–39
  - client configuration 39
  - host installation 37–39

- limitations 36–37
- KMS. *See* Key Management Service

## L

- legacy network adapters 248–249
- Lightweight Directory Access Protocol (LDAP) 66
- Linux
  - Secure Boot and 206–208
  - virtual machines 201–203
- Linux containers
  - managing
    - using Docker daemon 277–279
    - using PowerShell 279–281
- Linux deployments 61
- Linux Integration Services (LIS) 61, 204–205
- Live Migration
  - CredSSP or Kerberos authentication protocol for 308–309
    - implementing 303–308
    - in cluster 304
    - of VM 369–370
    - Shared Nothing 307–308
    - without a cluster 305–307
- local area network (LAN) 142
- Local Configuration Manager (LCM) 26
- local Hyper-V Administrators 174
- local memory 189
- log files. *See* transaction log files
- logical unit number (LUN) 134
- Lync Online. *See* Skype for Business

## M

- MAK Volume Licensing agreements 35–36
- Management Object Format (MOF) files 27
- MapSetup.exe 63–64
- MAP Toolkit. *See* Microsoft Assessment and Planning (MAP) Toolkit
- master boot record (MBR) 84–85, 87
- maximum hardware configurations 4
- MBR. *See* master boot record (MBR)
- Measure-VM cmdlet 194, 234–235
- Media Access Control (MAC) address
  - configuration of 244–246
- memory
  - adding or removing, in VM 185–186
  - containers 288

- dynamic 186–188
  - local 189
  - Non-Uniform Memory Access 189–192
  - remote 189
  - virtual 259
  - memory counters 435–436
  - Merge-VHD cmdlet 228
  - message transport. *See* transport
  - Microsoft Assessment and Planning (MAP) Toolkit 61–69
    - collection of inventory information 64–68
    - discovery methods 66
    - evaluation of results 68–69
    - functions of 62
    - installation 63–64
  - Microsoft Azure. *See* Azure
  - Microsoft Azure Active Directory. *See* Azure Active Directory (Azure AD)
  - Microsoft Management Console (MMC) snap-ins
    - using remotely 25–26
  - migrations. *See also* Live Migration
    - migration guides 34–35
    - P2V 60
    - Quick Migration 370–371
    - roles 32–33
    - servers 32–35
    - Storage Migration 309–311, 371–372
    - virtual machines 369–372
  - mirror storage layout 128
  - MOF files. *See* Management Object Format (MOF) files
  - Mount-DiskImage cmdlet 92
  - mounting
    - virtual hard disks 91–93, 224–225
  - mounting images
    - images 71–72
  - Mount-VHD cmdlet 92
  - MSU files 74
  - multi-domain clusters 314–317
  - Multipath I/O (MPIO) 145–148
  - multiple activation keys (MAKs) 35–36
  - Multipoint edition 5
- ## N
- namespace isolation 260
  - Nano Server 2, 4, 42–57
    - as container host 264–265
    - authentication screen 50
    - configuration 50–55
      - firewall rules 54–55
      - IP address 51–53
    - Docker installation on 267–268
    - features of 42, 43
    - image creation 44–45
    - installation 44–48
    - joining a domain 45–47
    - managing, using Windows PowerShell 76–78
    - remote management 265
    - remote management of 55–57
    - roles and features implementation on 48–50
    - shortcomings of 44
    - usage scenarios and requirements for 43–44
    - virtual machine creation 47–48
  - Nano Server Recovery Console 50–54
  - NAS. *See* network attached storage (NAS)
  - NAT. *See* network address translation (NAT)
  - nested virtualization 181
  - Netdom.exe tool 21
  - network adapters 246–247
    - enabling RMDA on 253–254
    - legacy 248–249
    - NIC teaming 249–251
    - synthetic 247–249
    - virtual 251, 283
  - network address translation (NAT) 269, 284
  - network attached storage (NAS) 123
  - network counters 437–438
  - Network File System (NFS) shares 96
    - creation of 101–103
  - network hardware 322
  - network health protection 373–374
  - networking
    - cluster 321–324
    - container 281–285
    - Hyper-V 235–256
      - bandwidth management 254–256
      - MAC address configuration 244–246
      - network isolation 246–247
      - NIC teaming 249–251
      - performance optimization 243–244
      - Switch Embedded Teaming 253–254
      - synthetic network adapters 247–249
      - virtual machine queue 251–253
      - virtual network interface cards 236–237
      - virtual switches 238–242, 244, 247
    - S2D 353–354
    - transparent networks 285
  - network load balancing (NLB) 375–384
    - affinity configuration 381–382

## New-Cluster cmdlet

- cluster operation mode configuration 384
- cluster upgrades 384
- node installation 377–381
- port rules 382–383
- prerequisites 375–377
- New-Cluster cmdlet 337
- New-Container cmdlet 275, 280
- New-NanoServerImage cmdlet 44–49, 51, 52, 264
- New-NetIpAddress cmdlet 19, 20
- New-NetQosPolicy cmdlet 144
- New-NetQosTrafficClass cmdlet 144
- New-PsSession cmdlet 21–22, 55
- New-PSSession cmdlet 177, 180
- New-SmbShare cmdlet 106–107, 340
- New-SRPartnership cmdlet 151
- New-VHD cmdlet 90, 219, 223
- New Virtual Hard Disk Wizard 88
- New-VM cmdlet 47, 184
- New-VM PowerShell cmdlet 197
- NICs. *See* network interface cards (NICs)
- NIC teaming 249–251
- NLB. *See* network load balancing (NLB)
- node fairness 367–368
- nodes 311, 375
- Non-Uniform Memory Access (NUMA) 189–192
  - nodes 189
  - node spanning 189–190
  - ratio 189
  - topology 190–192
- N\_Port ID Virtualization (NPIV) 232
- NTFS file system 93–95
- NTFS permissions 112–114, 117–122

## O

- Office Telemetry. *See* telemetry
- operating system environments (OSEs) 5
- Optimize-VHD cmdlet 228
- organizationally unique identifier (OUI) 244

## P

- P2V migration 60
- packages
  - Nano Server 48–49
- parent partitions 167
- parity storage layout 129
- partitions 9–10, 167

- pass-through disks 212, 225–226
- patches 74–75
- performance counter alerts 438–439
- Performance Metrics Wizard 68
- Performance Monitor
  - bottlenecks and 433–434
  - CPU counters 434–435
  - data collector sets 431–433
  - disk counters 436–437
  - memory counters 435–436
  - monitoring workloads using 425–430
  - network counters 437–438
- permissions
  - advanced 114–115, 120–121
  - allowing 115, 116
  - assigning 117–121
  - basic 114, 117–120
  - configuration 112–122
  - denying 115, 116
  - Hyper-V 174
  - inheritance 115–116
  - NTFS 112–114, 117–122
  - resource ownership and 122
  - share 96, 104–106, 108–109, 112–113
  - understanding effective access 116–117
- physical disks 225–226
  - adding 125
- physical servers
  - migration to virtual 60
- platform-as-a-service. *See* PaaS
- Plug and Play (PnP) 147
- port mapping 284
- power-on self-test (POST) 205
- PowerShell. *See* Windows PowerShell
- PowerShell Core 57
- Preboot Execution Environment (PXE) 198, 249
- Preboot Execution Environment (PXE) feature 11
- Priority-based Flow Control (PFC) 145
- private networks 244
- private network switches 241
- production checkpoints 230–231
- Pull Server 27–28

## Q

- quality of service (QoS) policies 144, 233–235
- Quick Migration 370–371

- quorum 317–321
  - dynamic quorum management 318
  - modifying configuration of 318–320
  - voting 321
  - witnesses 317–318, 320–321
- quotas 94

## R

- Receive-SmigServerData cmdlet 33
- redundancy 128, 145
- ReFS (Resilient File System) 93–95
- Remote Direct Memory Access (RDMA) 253–254
- remote management
  - configuration of 55
  - Hyper-V 174–179
  - Nano Server 265
  - of Nano Server 55–57
  - using MMC snap-ins 25–26
  - using PowerShell 21–22
  - using Server Manager 22–24
- remote memory 189
- Remote Server Administration Tools 174
- Remove-Container cmdlet 281
- Remove-ContainerImage cmdlet 273
- Remove-SmbShare cmdlet 108
- reparse point 158
- replica servers 299–301
- replication
  - asynchronous 148
  - DFS 150
  - Hyper-V Replica 298–303
  - Storage Replica 148–155
  - synchronous 148
- Reset-VMResourceMetering cmdlet 195
- Resize-VHD cmdlet 228
- resource governance 260–261
- resource metering 193–195
- Resource Monitor 440–442
- resource ownership 122
- restores
  - data deduplication and 162
  - from backups 418–421
- Revoke-SmbShareAccess cmdlet 109
- roles
  - implementation on Nano Server 48–50

- installation 11–17
  - in offline images 75–77
- migration of 32–33
- offline installation 225
- round robin policy 148

## S

- SAN. *See* storage area network (SAN)
- saved-state (.vsv) files 182
- Scale-out File Server (SoFS) 337–341
- SCCM. *See* System Center Configuration Manager
- SCSI (Small Computer Systems Interface) controllers
  - 214–215
- sector sizes 82–84
- Secure Boot 205–208
- security identifiers (SIDs) 113
- security principal 112
- self-service deployment. *See* user-driven client deployments
- Send-SmigServerData cmdlet 33
- Serial Attached SCSI (SAS) 327
- Server Core 2, 4, 42
  - configuration 19–20
  - Hyper-V Server and 168
  - installation 17–19
  - management of 21–25
    - using Windows PowerShell 76–78
  - Windows containers and 264
- server folders
  - sharing 95–109
- Server for NFS role service 97
- server installations
  - maintaining 387–425
    - backup strategies 421–425
    - patch management 401–405
    - Windows Defender 405–410
    - Windows Server Backup 411–421
    - Windows Server Update Services 388–405
  - monitoring 425–442
    - performance counter alerts 438–439
    - using Performance Monitor 425–430, 431–438
    - using Resource Monitor 440–442
- Server Manager
  - deduplication configuration using 155–157
  - Hyper-V installation using 170–171
  - installing roles using 11–15
  - managing Server Core using 22–24
  - share configuration using 95–106

## Server Message Blocks (SMB) clients

- Server Message Blocks (SMB) clients
  - configuration settings 111–112
- Server Message Blocks (SMB) server
  - configuration settings 109–111
- Server Message Blocks (SMB) shares 96
  - configuration of 106–108
  - creation of 97–101
- servers. *See also* Windows Server 2016
  - adding, in Server Manager 22–24
  - choosing, to virtualize 59–60
  - configuration of multiple 13
  - DHCP 11, 51
  - fault tolerance 128–131
  - mass deployment of 11
  - migration of 32–35
  - replica 299–301
  - SMB 109–111
  - upgrades 28–32
- server-to-server configurations 148–149, 151–155
- Server Virtualization And Consolidation Wizard 68–69
- Set-Disk cmdlet 226
- Set-DnsClientServerAddress cmdlet 20
- Set-FileStorageTier cmdlet 133
- Set-Item cmdlet 56
- Set-NetAdapterVmq PowerShell cmdlet 253
- Set-NetQoSbcdxSetting cmdlet 143
- Set-SmbPathAcl cmdlet 340
- Set-SmbServerConfiguration cmdlet 109–111
- Set-SRPartnership cmdlet 155
- Set-VM cmdlet 231
- Set-VMFirmware cmdlet 208
- Set-VMMemory cmdlet 185
- Set-VMNetworkAdapter cmdlet 255
- Set-VmReplicationServer cmdlet 300
- Shared Nothing Live Migration 307–308
- shares
  - advanced 103–104
  - configuration
    - using Windows PowerShell 106–108
  - configuration, using Server Manager 95–106
  - continuously available 360–361
  - NFS 96
    - creation of 101–103
  - permissions 96, 104–106, 108–109, 112–113
  - removing 108
  - sessions management 107–108
  - SMB 96
    - creation of 97–101
  - shielded virtual machines 198
  - simple storage layout 128
  - single domain clusters 314–317
- Single Instance Store (SIS) technology 158
- single-root I/O virtualization (SR-IOV) 243
- site-aware failover clusters 365–367
- site-based fault tolerance 376
- slack space 82, 83
- Small Computer System Interface (SCSI) 327
- smart paging 192–193
- SMB 3.0 protocol 360
- SmbShare 106–112
- SMTP. *See* Single Mail Transfer Protocol (SMTP)
- snapshots 228. *See also* checkpoints
- software patches 401–405
- software storage bus 353
- solid state drives (SSDs) 131–132
- SPF. *See* send policy framework (SPF) records
- standard checkpoints 230
- Standard edition 5
- Start-DscConfiguration cmdlet 27
- storage area network (SAN) 123, 133, 142
- storage area networks (SANs) 231
- storage infrastructure 151–152
- Storage Migration 309–311, 371–372
- storage pools 123–125, 342–343, 344
  - expanding 131
  - hot spares 130–131
- Storage Replica (SR)
  - clustering configuration 153
  - event log entries 154
  - implementing 151–155, 345
  - replication partnerships 154–155
  - storage infrastructure for 151–152
  - testing topology 152–153
  - usage scenarios for 148–150
- Storage Server edition 5
- storage solutions 81–164
  - clusters 326–328
  - datacenter bridging 142–144
  - data depulication 155–162
  - fault tolerance and 128–131
  - GUID partition table (GPT) disks 84–88
  - Hyper-V 213–235
  - implementation of 123–155
  - Internet Storage Name Service (iSNS) 140–142
  - iSCSI targets and initiators 133–140
  - NTFS file system 93–95
  - permissions configuration 112–122

- Quality of Service for 233–235
- ReFS file system 93–95
- sector size configuration 82–84
- shared VHDX files 349–351
- shares configuration
  - using Server Manager 95–106
  - using Windows PowerShell 106–108
- storage layout options 125–131
- storage pools 123–125
- Storage Replica 148–155
- tiered storage 131–133
- virtual disks 125–128
- virtual hard disks
  - creating 88–91
  - mounting 91–93
- Storage Spaces 123
  - Clustered 342–345
    - expanding storage pools 131
    - fault tolerance in 128–131
    - tiered storage 131–133
- Storage Spaces Direct 198
- Storage Spaces Direct (S2D) 352–358
  - disaggregated 355–357
  - disk drives 353
    - enabling, using PowerShell 354–355
    - hyper-converged 357–358
    - networking 353–354
  - scenario requirements for 352–354
  - servers 352
- stretch clusters 149–150, 151–155, 345, 365–367
- Suspend-ClusterNode cmdlet 374
- Switch Embedded Teaming (SET) 253–254
- symmetric multiprocessing (SMP) 189
- synchronous replication 148
- synthetic network adapters 247–249
- system boot 205
- System Center Configuration Manager (SCCM) 66, 249
- Systeminfo.exe 169–170

## T

- targets
  - iSCSI 134–138
- Test-SRTopology cmdlet 152–153
- thin provisioning 126
- tiered storage 131–133
- traffic classes 144
- Traffic Control Protocol (TCP) 144
- transparent networks 285

- Type II virtualization 166
- Type I virtualization 167

## U

- Unblock-SmbShareAccess cmdlet 109
- Unified Extensible Firmware Interface (UEFI) 86, 205, 206
- Universal Extensible Firmware Interface (UEFI) 198
- unoptimization 159
- updates
  - patch management 401–405
    - Windows Server Update Services 388–405
- Update-VMVersion cmdlet 209
- upgrades
  - Hyper-V 173
    - in-place 28–32
    - paths 28
      - preparing for 29–30
      - procedure for 30–32
      - virtual machines 208
- user accounts. *See also* identities
- User Datagram Protocol (UDP) 144
- user identities. *See* identities

## V

- VAMT. *See* Volume Activation Management Tool
- VHD Set files 351
- VHD sets 221
- VHDX files 182
  - creating
    - shared 220–222
      - using Hyper-V Manager 214–220
  - creation of 88–91
    - using Disk Management 88–90
      - with Windows PowerShell 90–91
  - mounting 91–93
  - shared 349–351
- virtual disks
  - creating 123, 125–128, 132
- virtual hard disks (VHDs) 44, 60, 182
  - adding to virtual machines 219–220
  - creating
    - in PowerShell 219
      - using Hyper-V Manager 214–220
      - with VMs 215–216
    - creation of 88–91
      - using Disk Management 88–90
        - with Windows PowerShell 90–91

## virtualization

- formats 215
- managing, using Windows PowerShell 76–78
- modifying 223, 223–225
- mounting 91–93, 224–225
- resizing 226–228
- virtualization 259. *See also* Hyper-V
  - advantages of 303
  - architectures 166–167
  - defining scope of 59–60
  - deployment considerations 69–70
  - maximum hardware configurations and 4
  - nested 181
  - N\_Port ID Virtualization (NPIV) 232
  - planning for 58–60
  - single-root I/O virtualization (SR-IOV) 243
  - strategy 3
  - Type I 167
  - Type II 166
  - Windows containers 263–264
  - workload assessment 61–69
- Virtualization Service Client (VSC) 247–248
- Virtualization Service Provider (VSP) 247–248
- Virtualized Backup Server 162
- virtual LANs (VLANs) 247
- virtual machine configuration (.vmc) files 182
- Virtual Machine Connection (VMConnect) 199
- virtual machine monitor (VMM) 166
- virtual machine queue (VMQ) 243, 251–253
- Virtual Machine role 360
- virtual machines (VMs) 166
  - adding or removing memory 185
  - adding virtual disks to 219–220
  - advantages of 58–59
  - Automatic Virtual Machine Activation 41–42
  - configuration
    - dynamic memory 186–188
    - FreeBSD 202–203
    - Integration Services 195–196
    - Linux 202–203
    - resource metering 193–195
    - settings 184–185
    - smart paging 192–193
    - using PowerShell Direct 180
  - configuration of 301–303
  - containers with 263–264
  - converting generations 199
  - creating 47–48, 182–184, 201–202
  - delegating management of 174
  - drain on shutdown configuration 374–375

- enhanced session mode 199–201
- exporting and importing 209–212
- FreeBDS deployment 61
- FreeBSD 201–203
- Generation 1 197, 214
- Generation 2 197–199, 205, 215
- import, export, and copy of 372–373
- installation
  - guest operating system 203
- Linux 201–203
- Linux deployment 61
- Live Migration of 303–309, 369–370
- monitoring 361–363
- movement of, in clustered nodes 369–375
- moving between hosts 297–310
- Nano Server for 43, 47–48
- network health protection 373–374
- Quick Migration 370–371
- resiliency 348–349
- shielded 198
- storage 213–235
  - Storage Migration 309–311, 371–372
  - upgrading to Windows Server 2016 Hyper-V 208–209
- virtual memory 259
- virtual network adapters 283
- virtual network interface cards (vNICs) 236–237
- virtual switches 238–242, 244, 247, 250
- Volume Activation Management Tool (VAMT) 36
- Volume Activation Tools Wizard 38, 41
- volume shadow copies 94
- Volume Shadow Copy Service (VSS) 424

## W

- Wbadm command 325–326
- WDS. *See* Windows Deployment Services
- weighted paths 148
- WIM. *See* Windows Imaging Format (WIM)
- Windows
  - Secure Boot and 205–206
- Windows containers 259–296
  - architecture 262
  - attaching 279, 280
  - container names 287
  - CPU cycles 288–289
  - creating 274–277
  - creating images 279, 280, 289–290
  - deployment of 259–277

- Docker and 266–270
- Hyper-V 261–264, 275–277
- images 261
- image tagging 272–273
- installation
  - base operating system 271–272
  - container host 262–263
  - requirements 260–261
- listing 278, 280
- managing
  - data volumes 286–287
  - networking 281–285
  - resource control 287–289
  - using Docker daemon 277–279
  - using Microsoft Azure 293
  - using PowerShell 279–281
  - with DockerHub 291–293
- memory constraints 288
- Nano Server as container host 264–265
- PowerShell and 270–271
- removing 279, 281
- Server Core and 264
- starting and stopping 278, 280
- uninstalling operating system image 273–274
- use scenarios for 260–261
- virtualizing 263–264
- Windows Server 261, 264–265, 274–275
- Windows Defender 405–410
  - configuration of 405–408
  - integration with WSUS and Windows Update 409–410
- Windows Deployment Services (WDS) 11, 249
- Windows Firewall 300–301
  - configuration 54–55
- Windows PowerShell
  - container management using 279–281
  - creating virtual disks in 219
  - deduplication configuration in 157–158
  - Desired State Configuration 26–28
  - DISM.exe command equivalents 77–79
  - displaying cmdlets 21
  - enabling S2D using 354–355
  - Hyper-V installation using 171
  - importing VMs using 211–212
  - installing roles using 15–16
  - managing Nano Server using 77–79
  - managing Server Core using 21–22, 77–79
  - mounting VHD and VHDX files in 92–93
  - remote management of Nano Server using 55–57
  - remote management using 176–179
  - SMB share configuration using 106–108
  - using containers with 270–271
  - VHD and VHDX file creation in 90–91
  - VM creation in 184
  - Windows Defender configuration using 407–408
- Windows PowerShell Direct
  - VM configuration using 180
- Windows Remote Management (WinRM) 21
  - configuration 55
- Windows Server 2012
  - upgrading 28
- Windows Server 2012 R2
  - upgrading 28
- Windows Server 2016
  - Docker installation on 266
  - editions 2, 4–6
  - images for deployment 58–73
  - installation 1–18
    - activation model for 35–42
    - clean 6–9
    - features and roles 11–17
    - mass deployment 11
    - requirements 2–4
  - migrations 32–35
  - permissions management 112–122
  - upgrades to 28–32
  - virtualization
    - planning for 58–60
    - working with partitions in 9–10
- Windows Server Backup 324–325, 411–421, 424
- Windows Server Migration Tools 32, 33–34
- Windows Server Update Services (WSUS) 388–400
  - architectures 388–391
  - client configuration 401–405
  - configuration of 394–398
  - database 391–392
  - deploying 393
  - groups 398–400
  - storage 392–393
  - Windows Defender integration 409–410
- Windows Setup page 7
- Windows Update Stand-Alone Installer (MSU) files 74
- Winrm.exe tool 56
- witnesses
  - cloud 345–348
  - quorum 317–318, 320–321
- workgroup clusters 314–317
- workload monitoring 425–430, 440–442

## **World Wide Node Names (WWNNs)**

workloads

virtualization considerations for 69–70

World Wide Node Names (WWNNs) 232

World Wide Port Names (WWPNs) 232

WSUS. *See* Windows Server Update Services (WSUS)