



Inside **OUT**

The ultimate in-depth reference  
Hundreds of timesaving solutions  
Supremely organized, packed  
with expert advice

# Windows Server 2019

**Orin Thomas**

*Microsoft Cloud Operations Advocate, Cloud and Datacenter expert, and leading Windows author*

# Windows Server 2019 Inside Out

**Orin Thomas**

## **Windows Server 2019 Inside Out**

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

Copyright © 2020 by Orin Thomas

All rights reserved. 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.pearson.com/permissions](http://www.pearson.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: 978-0-13-549227-7

ISBN-10: 0-13-549227-0

Library of Congress Control Number: 2020935953

ScoutAutomatedPrintCode

### **Trademarks**

Microsoft and the trademarks listed at <http://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 author, 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 from the use of the 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).

**Editor-in-Chief:** Brett Bartow

**Executive Editor:** Loretta Yates

**Sponsoring Editor:** Charvi Arora

**Development Editor:** Rick Kughen

**Managing Editor:** Sandra Schroeder

**Senior Project Editor:** Tracey Croom

**Copy Editor:** Charlotte Kughen

**Indexer:** Valerie Haynes Perry

**Proofreader:** Dan Foster

**Technical Editor:** Vince Averello

**Editorial Assistant:** Cindy Teeters

**Cover Designer:** Twist Creative, Seattle

**Composer:** Danielle Foster

**Graphics:** Vived Graphics



# Contents at a glance

Chapter 1		Chapter 13	
<b>Administration tools</b> .....	<b>1</b>	<b>Active Directory Federation Services</b> .....	<b>437</b>
Chapter 2		Chapter 14	
<b>Installation options</b> .....	<b>31</b>	<b>Dynamic Access Control and Active</b>	
Chapter 3		<b>Directory Rights Management Services</b> .....	<b>455</b>
<b>Deployment and configuration</b> .....	<b>47</b>	Chapter 15	
Chapter 4		<b>Routing and Remote Access</b> .....	<b>475</b>
<b>Active Directory</b> .....	<b>109</b>	Chapter 16	
Chapter 5		<b>Remote Desktop Services</b> .....	<b>499</b>
<b>DNS, DHCP, and IPAM</b> .....	<b>159</b>	Chapter 17	
Chapter 6		<b>Azure IaaS and hybrid services</b> .....	<b>519</b>
<b>Hyper-V</b> .....	<b>191</b>	Chapter 18	
Chapter 7		<b>Windows Subsystem for Linux</b> .....	<b>575</b>
<b>Storage</b> .....	<b>229</b>	Chapter 19	
Chapter 8		<b>Hardening Windows Server and</b>	
<b>File servers</b> .....	<b>273</b>	<b>Active Directory</b> .....	<b>581</b>
Chapter 9		Chapter 20	
<b>Internet Information Services</b> .....	<b>309</b>	<b>Security systems and services</b> .....	<b>635</b>
Chapter 10		Chapter 21	
<b>Containers</b> .....	<b>337</b>	<b>Maintenance and monitoring</b> .....	<b>653</b>
Chapter 11		Chapter 22	
<b>Clustering and high availability</b> .....	<b>369</b>	<b>Upgrade and migration</b> .....	<b>685</b>
Chapter 12		Chapter 23	
<b>Active Directory Certificate Services</b> .....	<b>391</b>	<b>Troubleshooting</b> .....	<b>723</b>
		<b>Index</b> .....	<b>755</b>





# Table of contents

	<b>Introduction</b> .....	<b>xxi</b>
	Changes since <i>Windows Server 2016 Inside Out</i> .....	xxi
	Acknowledgments .....	xxii
	Errata, updates, and book support .....	xxii
<b>Chapter 1</b>	<b>Administration tools</b> .....	<b>1</b>
	Remote not local .....	1
	Privileged Access Workstations .....	2
	Windows Admin Center .....	4
	Installing Windows Admin Center .....	6
	Windows Admin Center extensions .....	9
	Show script .....	10
	Remote Server Administration Tools .....	11
	RSAT consoles .....	11
	Server Manager console .....	14
	PowerShell .....	17
	Modules .....	18
	PowerShell Gallery .....	19
	Remoting .....	19
	One-to-many remoting .....	21
	PowerShell ISE .....	21
	PowerShell Direct .....	24
	Remote Desktop .....	25
	SSH .....	27
<b>Chapter 2</b>	<b>Installation options</b> .....	<b>31</b>
	Windows Server 2019 editions .....	31
	Windows Server servicing branches .....	33
	Long Term Servicing Channel .....	33
	Semi Annual Channel .....	33
	Insider Preview Builds .....	34
	Server Core .....	35
	Server Core interface .....	36
	Server Core roles .....	37
	Server Core App Compatibility Features on Demand .....	42
	When to deploy Server Core .....	43

	Server with Desktop Experience . . . . .	44
	Roles and features . . . . .	45
<b>Chapter 3</b>	<b>Deployment and configuration . . . . .</b>	<b>47</b>
	Bare metal versus virtualized . . . . .	47
	Windows images . . . . .	48
	Modifying Windows images. . . . .	49
	Servicing Windows images. . . . .	50
	Mounting images. . . . .	50
	Adding drivers and updates to images . . . . .	53
	Adding roles and features. . . . .	54
	Committing an image. . . . .	56
	Build and capture. . . . .	56
	Answer files . . . . .	57
	Windows Deployment Services. . . . .	59
	WDS requirements . . . . .	60
	Managing images . . . . .	62
	Configuring WDS. . . . .	62
	Configuring transmissions. . . . .	67
	Driver groups and packages. . . . .	68
	Virtual Machine Manager . . . . .	68
	Virtual machine templates . . . . .	68
	VMM storage . . . . .	69
	VMM networking. . . . .	71
	Adding a WDS to VMM . . . . .	75
	VMM host groups . . . . .	76
	Infrastructure configuration as code. . . . .	79
	Desired State Configuration. . . . .	81
	DSC configuration files. . . . .	82
	Local Configuration Manager . . . . .	83
	DSC resources . . . . .	83
	DSC push model. . . . .	84
	DSC pull server . . . . .	84
	Chef Infra Server. . . . .	85
	Chef servers . . . . .	85
	Chef Development Kit . . . . .	89
	Deploying Chef agents. . . . .	94
	Deploying Chef cookbooks and recipes. . . . .	95
	Puppet. . . . .	96
	Puppet Master Server. . . . .	96
	Deploying Puppet agent to Windows Server . . . . .	99
	Managing Windows Server configuration. . . . .	101
	Puppet Windows Module Pack . . . . .	102
	Package-management utilities . . . . .	103
	PowerShell Gallery. . . . .	104
	Chocolatey. . . . .	105

<b>Chapter 4</b>	<b>Active Directory</b> .....	<b>109</b>
	Managing Active Directory .....	109
	Remote rather than local administration .....	110
	Active Directory Administrative Center .....	110
	Active Directory Users and Computers console .....	113
	Active Directory Sites and Services console .....	114
	Active Directory Domains and Trusts console .....	117
	Domain controllers .....	118
	Deployment .....	118
	Server Core .....	120
	Global catalog servers .....	121
	Read only domain controllers .....	121
	Virtual domain controller cloning .....	124
	AD DS structure .....	125
	Domains .....	125
	Domain functional levels .....	125
	Forests .....	126
	Account and resource forests .....	127
	Organizational units .....	127
	Flexible Single Master Operations roles .....	128
	Accounts .....	130
	User accounts .....	130
	Computer accounts .....	132
	Group accounts .....	133
	Default groups .....	134
	Service accounts .....	136
	Group policy .....	138
	GPO management .....	139
	Policy processing .....	141
	Group Policy preferences .....	143
	Administrative templates .....	145
	Restoring deleted items .....	146
	Active Directory Recycle Bin .....	147
	Authoritative restore .....	149
	Active Directory snapshots .....	151
	Managing AD DS with PowerShell .....	152
	Active Directory module .....	152
	Group Policy module .....	156
	ADDSDeployment module .....	157
<b>Chapter 5</b>	<b>DNS, DHCP, and IPAM</b> .....	<b>159</b>
	DNS .....	159
	DNS zone types .....	159
	Zone delegation .....	162
	Forwarders and conditional forwarders .....	163
	Stub zones .....	164



	GlobalNames zones .....	164
	Peer Name Resolution Protocol .....	165
	Resource records .....	166
	Zone aging and scavenging .....	167
	DNSSEC .....	168
	DNS event logs .....	169
	DNS options .....	170
	Delegated administration .....	174
	Managing DNS with PowerShell .....	174
	DHCP .....	177
	Scopes .....	177
	Server and scope options .....	178
	Reservations .....	178
	DHCP filtering .....	179
	Superscopes .....	179
	Multicast scopes .....	180
	Split scopes .....	180
	Name protection .....	180
	DHCP failover .....	181
	Administration .....	182
	IPAM .....	185
	Deploy IPAM .....	185
	Configure server discovery .....	185
	IPAM Administration .....	186
	Managing IPAM with PowerShell .....	188
<b>Chapter 6</b>	<b>Hyper-V .....</b>	<b>191</b>
	Dynamic memory .....	191
	Smart paging .....	192
	Resource metering .....	193
	Guest integration services .....	193
	Generation 2 VMs .....	194
	Enhanced Session Mode .....	195
	Discrete Device Assignment .....	195
	Nested virtualization .....	197
	Nested virtualization dynamic memory .....	197
	Nested virtualization networking .....	197
	PowerShell Direct .....	198
	HVC for Linux .....	198
	Virtual hard disks .....	199
	Fixed-sized disks .....	199
	Dynamically expanding disks .....	199
	Differencing disks .....	200
	Modifying virtual hard disks .....	200
	Pass-through disks .....	201
	Managing checkpoints .....	202
	Virtual Fibre Channel adapters .....	203
	Storage QoS .....	204

Hyper-V storage optimization.....	204
Deduplication .....	204
Storage tiering .....	204
Hyper-V virtual switches .....	205
External switches .....	205
Internal switches.....	205
Private switches .....	206
Virtual machine network adapters.....	206
Optimizing network performance .....	206
Bandwidth management.....	207
SR-IOV.....	207
Dynamic virtual machine queue .....	207
Virtual machine NIC teaming.....	208
Virtual machine MAC addresses .....	208
Network isolation .....	209
Hyper-V replica .....	209
Configuring Hyper-V replica servers.....	210
Configuring VM replicas .....	210
Replica failover .....	211
Hyper-V replica broker .....	211
Hyper-V failover clusters .....	212
Hyper-V host cluster storage .....	212
Cluster quorum .....	213
Cluster networking .....	214
Force Quorum Resiliency .....	215
Cluster Shared Volumes .....	215
Active Directory detached clusters .....	216
Preferred owner and failover settings .....	216
Hyper-V guest clusters.....	217
Hyper-V guest cluster storage.....	218
Shared virtual hard disk .....	218
Hyper-V VHD Sets .....	219
Live migration .....	219
Storage migration .....	221
Exporting, importing, and copying VMs .....	221
VM Network Health Detection .....	222
VM drain on shutdown .....	222
Domain controller cloning .....	223
Shielded virtual machines.....	223
Managing Hyper-V using PowerShell.....	224
<b>Chapter 7 Storage .....</b>	<b>229</b>
Storage spaces and storage pools .....	229
Storage pools .....	230
Storage space resiliency.....	234
Storage space tiering .....	235
Thin provisioning and trim .....	237
Creating virtual disks .....	239
Storage Spaces Direct.....	240

- Storage Replica .....247
  - Supported configurations .....248
  - Configuring replication .....249
- SMB 3.1.1 ..... 251
- iSCSI .....252
- iSNS server .....256
- Scale-Out File Servers .....258
- Server for NFS .....259
- Deduplication .....260
- Storage Quality of Service .....263
- ReFS .....264
- Storage-related PowerShell cmdlets .....266
  - Deduplication .....266
  - iSCSI .....266
  - iSCSITarget .....267
  - NFS .....267
  - Storage .....268
  - Storage Replica .....271

**Chapter 8 File servers .....273**

- Shared folder permissions .....274
  - Using File Explorer .....275
  - Windows Admin Center .....276
  - Server Manager .....277
- File Server Resource Manager .....280
  - Folder level quotas .....280
  - File screens .....282
  - Storage reports .....286
  - File classification .....288
  - File management tasks .....290
  - Access-Denied Assistance .....292
- Distributed File System .....293
  - DFS namespace .....293
  - DFS replication .....296
- BranchCache .....299
- PowerShell commands .....302
  - Shared Folder cmdlets .....302
  - File Server Resource Manager cmdlets .....303
  - BranchCache Cmdlets .....305
  - DFS Cmdlets .....306

**Chapter 9 Internet Information Services .....309**

- Managing sites .....309
  - Adding websites .....310
  - Virtual directories .....313
  - Modifying site settings .....314
  - Adding web applications .....314
  - Configuring TLS certificates .....315

	Site authentication . . . . .	318
	Modifying custom error response . . . . .	319
	Adding or disabling the default document . . . . .	320
	Directory browsing . . . . .	321
	IP address and domain name filtering . . . . .	322
	URL authorization rules . . . . .	323
	Request filters . . . . .	324
	Application pools . . . . .	326
	Creating application pools . . . . .	327
	Configuring application pool recycling settings . . . . .	328
	IIS users and delegation . . . . .	329
	IIS user accounts . . . . .	330
	Delegating administrative permissions . . . . .	331
	Managing FTP . . . . .	332
	Managing IIS using PowerShell . . . . .	334
<b>Chapter 10</b>	<b>Containers . . . . .</b>	<b>337</b>
	Container concepts . . . . .	337
	Isolation modes . . . . .	339
	Process Isolation mode . . . . .	339
	Hyper-V Isolation mode . . . . .	340
	Managing containers with Docker . . . . .	340
	Installing Docker . . . . .	341
	Demon.json . . . . .	342
	Retrieving container OS image . . . . .	344
	Container registries and images . . . . .	345
	Managing containers . . . . .	347
	Starting a container . . . . .	347
	Modifying a running container . . . . .	350
	Creating a new image from a container . . . . .	351
	Using Dockerfiles . . . . .	351
	Managing container images . . . . .	353
	Service accounts for Windows containers . . . . .	355
	Applying updates . . . . .	356
	Container networking . . . . .	357
	NAT . . . . .	359
	Transparent . . . . .	360
	Overlay . . . . .	362
	Layer 2 Bridge . . . . .	362
	Linux containers on Windows . . . . .	363
	Container orchestration . . . . .	364
	Kubernetes . . . . .	365
	Docker Swarm . . . . .	365
<b>Chapter 11</b>	<b>Clustering and high availability . . . . .</b>	<b>369</b>
	Failover clustering . . . . .	369
	Cluster quorum modes . . . . .	370
	Cluster storage and cluster shared volumes . . . . .	372

- Cluster networks .....372
- MPIO .....373
- Cluster Aware Updating.....373
- Failover and preference settings.....374
- Multisite clusters .....375
- Cloud witness .....376
- Virtual machine failover clustering.....377
- Rolling upgrades .....379
- Workgroup clusters.....381
- Cluster sets.....382
- Managing failover clustering with PowerShell.....383
- Network Load Balancing.....385
  - Network Load Balancing prerequisites.....386
  - NLB cluster operation modes.....386
  - Managing cluster hosts .....387
  - Port rules .....388
  - Filtering and affinity .....388
  - Managing NLB with PowerShell .....389

**Chapter 12 Active Directory Certificate Services ..... 391**

- CA types ..... 391
  - Enterprise CA.....393
  - Standalone CAs .....403
- Certificate revocation lists.....406
  - CRL distribution points.....406
  - Authority Information Access .....407
  - Revoking a certificate.....408
  - Publishing CRLs and delta CRLs.....410
- Certificate Services role services.....412
- Certificate templates.....413
  - Template properties .....414
  - Adding and editing templates.....420
- Certificate autoenrollment and renewal.....420
- CA management .....422
  - Handling certificate requests.....424
  - CA backup and recovery.....425
  - Key archiving and recovery.....427
  - CAPolicy.inf .....432
  - Managing Certificate Services using PowerShell.....433
  - Managing Certificate Services using Certutil.exe and Certreq.exe .....435

**Chapter 13 Active Directory Federation Services.....437**

- AD FS components .....437
- Claims, claim rules, and attribute stores.....438
- Claims provider.....439
- Relying party.....439
- Relying party trust.....439
- Claims provider trust.....440

	Configuring certificate relationship .....	441
	Attribute stores .....	442
	Claim rules .....	443
	Relying party trust claim rules .....	443
	Claims provider trust claim rules .....	444
	Configure Web Application Proxy .....	445
	Workplace Join .....	447
	Multifactor authentication .....	449
	Managing AD FS with PowerShell .....	450
	Managing Web Application Proxy with PowerShell .....	453
<b>Chapter 14</b>	<b>Dynamic Access Control and Active Directory Rights Management Services .....</b>	<b>455</b>
	Dynamic Access Control .....	455
	Configuring Group Policy to support DAC .....	456
	Configuring User and Device Claims .....	456
	Configuring Resource Properties .....	457
	Central access rules .....	459
	Central access policies .....	461
	Staging .....	463
	Access Denied Assistance .....	463
	Installing AD RMS .....	464
	AD RMS certificates and licenses .....	465
	AD RMS Templates .....	466
	AD RMS Administrators and Super Users .....	469
	Trusted User and Publishing Domains .....	470
	Exclusion policies .....	471
	Apply AD RMS templates automatically .....	471
	Managing AD RMS with Windows PowerShell .....	473
	Dynamic Access Control cmdlets .....	474
<b>Chapter 15</b>	<b>Routing and Remote Access .....</b>	<b>475</b>
	Remote Desktop Gateway .....	475
	RD Gateway connection and resource policies .....	476
	Configuring server settings .....	477
	Configuring clients to use RD Gateway .....	477
	Virtual private networks .....	479
	IKEv2 Always On VPN protocol .....	479
	SSTP VPN protocol .....	481
	L2TP/IPsec protocols .....	481
	PPTP VPN protocol .....	481
	VPN authentication .....	482
	Deploying a VPN server .....	482
	Disable VPN protocols .....	483
	Granting access to a VPN server .....	483
	LAN routing .....	487
	Network Address Translation (NAT) .....	487

DirectAccess .....	489
DirectAccess topologies .....	490
DirectAccess server .....	490
Network Location Server .....	492
Configuring DirectAccess .....	493
Managing Remote Access using PowerShell .....	496
<b>Chapter 16 Remote Desktop Services .....</b>	<b>499</b>
Deployment .....	499
Remote Desktop Connection Broker .....	502
Deployment properties .....	502
Remote Desktop Session Host .....	503
Session collection settings .....	504
Personal session desktops .....	506
RemoteApp .....	506
Group Policy configuration .....	507
Remote Desktop Virtualization Host .....	509
Virtual machine preparation .....	510
Virtual desktop collections .....	511
Pooled virtual desktops .....	512
Personal virtual desktops .....	512
DDA and RemoteFX .....	512
Remote Desktop Web Access .....	513
Remote Desktop licensing .....	513
Installing RDS CALs .....	514
Activating a License Server .....	515
Managing Remote Desktop Services using PowerShell .....	515
<b>Chapter 17 Azure IaaS and hybrid services .....</b>	<b>519</b>
Windows Server IaaS VMs .....	519
Creating Azure IaaS VMs .....	520
IaaS VM networking .....	524
IaaS VM administration .....	532
Azure Active Directory .....	538
Azure Active Directory Connect .....	540
Azure AD Connect server requirements .....	541
Installing Azure AD Connect .....	544
Using UPN suffixes and non-routable domains .....	551
Monitor Azure AD Connect Health .....	554
Forcing synchronization .....	555
Configure object filters .....	557
Implement and manage Azure AD self-service password reset .....	560
Azure AD Password Protection .....	562
Azure AD DS .....	563
Azure hybrid cloud services .....	564
Connect Windows Admin Center .....	565
Creating Azure IaaS VMs from Windows Admin Center .....	567
Azure File Sync .....	569

	Azure Arc .....	572
	Azure Site Recovery.....	572
	Azure Network Adapter.....	573
<b>Chapter 18</b>	<b>Windows Subsystem for Linux.....</b>	<b>575</b>
	Linux on Windows Server .....	575
	Installing WSL .....	576
	WSL 2.0 .....	579
<b>Chapter 19</b>	<b>Hardening Windows Server and Active Directory.....</b>	<b>581</b>
	Hardening Active Directory .....	582
	Hardening domain controllers .....	582
	Least privilege.....	583
	Role-Based Access Control .....	584
	Password policies.....	585
	Account security options.....	586
	Protected accounts .....	588
	Authentication policies silos .....	589
	Disable NTLM .....	591
	Block server operators from scheduling tasks .....	592
	Enable Local Security Authority protection.....	592
	KRBTGT account password .....	593
	Enhanced Security Administrative Environment forest .....	594
	Hardening Windows Server .....	596
	User rights .....	596
	Service accounts.....	600
	Just Enough Administration .....	603
	Privileged Access Management .....	609
	Local Administrator Password Solution .....	613
	Advanced auditing .....	615
	Windows Firewall with Advanced Security .....	618
	Shielded VMs .....	628
	Guarded fabric .....	631
<b>Chapter 20</b>	<b>Security systems and services .....</b>	<b>635</b>
	Security Compliance Toolkit.....	636
	Policy Analyzer tool.....	636
	Local Group Policy Object tool .....	638
	Attack Surface Analyzer.....	638
	Credential Guard .....	640
	Windows Defender Application Control .....	642
	Virtualization-based security.....	644
	Controlled Folder Access.....	645
	Exploit Protection .....	647
	Windows Defender .....	650
	Windows Defender SmartScreen .....	651



<b>Chapter 21</b>	<b>Maintenance and monitoring</b>	<b>653</b>
	Data collector sets	653
	Alerts	655
	Event Viewer	655
	Event log filters	655
	Event log views	656
	Event subscriptions	657
	Event-driven tasks	658
	Network monitoring	659
	Resource Monitor	659
	Message Analyzer	660
	Azure Monitor	661
	Windows Server Backup	662
	Backup locations	664
	Backing up data	664
	Role- and application-specific backups	665
	Restore from backups	665
	Restore to an alternative location	666
	Azure Backup	666
	Preparing Azure Backup	667
	Backing up data to Azure Backup Agent	669
	Restore from Azure Backup	669
	Vssadmin	670
	Windows Server Update Services	672
	Products, security classifications, and languages	672
	Autonomous and replica modes	673
	Update files	673
	WSUS security roles	674
	WSUS groups	675
	WSUS policies	675
	Deploying updates	677
	Automatic approval rules	677
	Azure Update Management	679
	Monitoring and maintenance related PowerShell cmdlets	683
	WSUS related PowerShell cmdlets	684
<b>Chapter 22</b>	<b>Upgrade and migration</b>	<b>685</b>
	Supported upgrade and migration paths	685
	Upgrading roles and features	687
	Converting evaluation version to licensed version	688
	Upgrading editions	689
	Windows Server Migration Tools	689
	Active Directory	693
	FRS to DFSR migration	695
	Migrating to a new forest	696

Active Directory Certificate Services .....	699
Preparation .....	700
Migration .....	702
Verification and post migration tasks .....	703
DNS .....	704
DHCP .....	705
Preparing to migrate DHCP .....	706
Migration .....	708
Verification and post migration tasks .....	709
File and storage servers .....	709
Migrate file servers using Storage Migration Service .....	710
Migrate file and storage servers using WSMT .....	718
<b>Chapter 23 Troubleshooting .....</b>	<b>723</b>
Troubleshooting methodology .....	723
Redeployment .....	724
Symptoms and diagnosis .....	725
Dependencies .....	726
Ranking hypothetical solutions .....	727
Applying solutions .....	728
Command-line tools .....	729
Sysinternals tools .....	733
Process Explorer .....	734
Process Monitor .....	735
ProcDump .....	736
PsTools .....	737
VMMap .....	738
SigCheck .....	739
AccessChk .....	740
Sysmon .....	741
AccessEnum .....	744
ShellRunAs .....	745
LogonSessions .....	746
Active Directory Explorer .....	746
Insight for Active Directory .....	749
PsPing .....	750
RAMMap .....	751
<b>Index .....</b>	<b>761</b>



## About the author



**Orin Thomas** is a principal cloud operations advocate at Microsoft and has written more than three dozen books for Microsoft Press on topics including Windows Server, Windows Client, Azure, Office 365, System Center, Exchange Server, Security, and SQL Server. He has authored Azure Architecture courses at Pluralsight, has authored multiple Microsoft Official Curriculum and EdX courses on a variety of IT Pro topics, and is completing his Doctorate in Information Technology on cloud computing security and compliance at Charles Sturt University. You can follow him on twitter at <http://twitter.com/orinthomas>.



# Introduction

This book is primarily written for IT Professionals who work with Windows Server operating systems on a regular basis. As such, it's likely that Windows Server 2019 isn't the first version of Windows Server that you've been responsible for managing. This is because the majority of Windows Server administrators have been working with some version of the operating system for more than a decade, with a good percentage having experience going back to the days of Windows NT 4. With that in mind, this book doesn't spend a great amount of time on introductory concepts or techniques; instead, it aims to provide intermediate to advanced coverage of the most important roles and features available with Windows Server 2019, including its hybrid cloud capabilities.

This book is also written under the assumption that as an experienced IT professional, you know how to use a search engine to find relevant technical information. This leads to an obvious question, "Why would I buy a book if I can find relevant technical information with a search engine?" The answer is that even though you may be good at tracking down technical information and have experience filtering useful knowledge from wildly inaccurate guesses, you can only search for something if you have an idea about it in the first place.

When presenting at conferences and user groups on Windows Server topics, I regularly encounter IT Professionals who have worked with Windows Server for many years who are unaware of specific functionality or techniques related to the product, even if that functionality or technique has been available for many years. This is because Windows Server 2019 includes so many roles, features, and moving parts, you are simply unlikely to know everything about the operating system. My aim in writing this book is to give you comprehensive coverage so that you'll learn things that you didn't know or simply missed, because when you've been solving a critical problem, you've been focused on the specifics of that problem and haven't had time to explore every facet of what the Windows Server operating system is capable of.

## Changes since *Windows Server 2016 Inside Out*

This book includes several new chapters as well as revisions and updates—from moderate to substantive—of chapters that were present in the Windows Server 2016 version of this book. Some of the substantive changes include removing content related to the Nano Server deployment option, which is no longer supported; coverage of Windows Admin Center; a new chapter on Azure IaaS and hybrid services, as well as Windows Subsystem for Linux; and splitting an extension of the security chapter from the 2016 edition into two separate chapters in this text. There is also coverage of new roles and features including Storage Migration Services, Azure File Sync, and Azure Update Management. While there are some chapters where only cosmetic changes were made, from the perspective of total word count, this book is about 15 percent longer than its predecessor, *Windows Server 2016 Inside Out*.

## Acknowledgments

I'd like to thank Rick Kughen, Vince Averello, Dan Foster, Charvi Arora, and Loretta Yates for the assistance they provided in bringing this text to print. I would also like to thank Thomas Maurer for his advice on revisions for this new edition of the text.

## Figure Credits

Figure number	Credit
FIG03-17	Copyright © 2020 Chef Software, Inc. All Rights Reserved.
FIG03-18	Copyright © 2020 Chef Software, Inc. All Rights Reserved.
FIG03-19	Copyright © 2020 Chef Software, Inc. All Rights Reserved.
FIG03-20	Copyright © 2020 Chef Software, Inc. All Rights Reserved.
FIG03-21	Copyright © 2020 Chef Software, Inc. All Rights Reserved.
FIG03-22	Copyright © 2020 Chef Software, Inc. All Rights Reserved.
FIG03-23	Copyright © 2020 Chef Software, Inc. All Rights Reserved.
FIG03-24	Copyright © 2020 Chef Software, Inc. All Rights Reserved.
FIG03-25	Copyright © 2020 Chef Software, Inc. All Rights Reserved.
FIG03-26	Copyright © 2020 Chef Software, Inc. All Rights Reserved.
FIG03-27	Copyright © 2020 Chef Software, Inc. All Rights Reserved.
FIG03-28	© 2020 Puppet
FIG03-29	© 2020 Puppet
FIG03-30	© 2020 Puppet
FIG03-31	© 2020 Puppet
FIG03-32	© 2020 Puppet
FIG11-01	Courtesy of Pearson Education

## Errata, updates, and 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:

*[MicrosoftPressStore.com/WindowsServer2019InsideOut/errata](http://MicrosoftPressStore.com/WindowsServer2019InsideOut/errata)*

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

For additional book support and information, please visit:

*[MicrosoftPressStore.com/Support](http://MicrosoftPressStore.com/Support)*

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>.



Container concepts .....	337	Applying updates.....	356
Isolation modes .....	339	Container networking .....	357
Managing containers with Docker .....	340	Linux containers on Windows .....	363
Managing containers .....	347	Container orchestration.....	364

A container is a portable isolated application execution environment. Containers allow developers to bundle an application and its dependencies in a single image. This image can easily be exported, imported, and deployed on different container hosts, from a developer's laptop computer, to Server Core on bare-metal hardware, and eventually to being hosted and run on Azure. Because an application's dependencies are bundled with the application within a container, IT operations can deploy a container as soon as it is handed off without worrying whether an appropriate prerequisite software package has been installed or if a necessary setting has been configured. Because a container provides an isolated environment, a failure that occurs within one container will only impact that container and will not impact other containerized applications running on the container host.

## Container concepts

Containers can be conceptually challenging. To understand how containers work, you first must understand some of the terminology involving containers. While these concepts will be fleshed out more completely later in this chapter, you should understand the following terms at a high level:

- **Container image.** A container image is a template from which a container is generated. There are two general types of container images, which are usually created for a workload: a base OS image and a specific image. The difference between these containers on Windows Server is as follows:
  - **Container base OS image.** This is an image of the operating system upon which other container images are built. Four Windows OS container images are regularly published and updated by Microsoft. These images are available through an official public container registry.



- **Windows Server Core.** This is an image of the Windows Server operating system in the Server Core configuration. Containers don't run with a GUI, so Server with Desktop Experience is not available. This container image is suitable for traditional .NET Framework applications.
  - **Nano Server.** This is a Windows Server image with all unnecessary elements stripped out. It is suitable for hosting .NET Core applications. The Nano Server image container image is what became of the Nano Server installation option that was available with the RTM version of Windows Server 2016. Because the image is stripped down to the essentials, it is far smaller than the Windows Server Core image and can be deployed and run more quickly.
  - **Windows IoT Core.** This is an image for Internet of Things (IoT) applications.
  - **Windows.** This is an image that provides the full Windows API set but doesn't include all the server roles and features that are available in the Server Core image. You should only use this option if the application you are trying to host has a dependency that is not included in the Windows Server Core container image.
  - **Container image.** A container image stores changes made to a running container base OS image or another container image. For example, you can start a new container from a container base OS image, make modifications such as installing Java or a Windows feature, and then save those modifications as a new container image. The new container image only stores the changes you make, and therefore, it is much smaller than the parent container base OS image. You can then create an additional container image that stores modifications made to the container image that has an installed Java and Windows feature. Each container image only stores the changes made to the image from which it was run.
- **Sandbox.** The sandbox is the environment in which you can make modifications to an existing container image before you commit the changes to create a new container image. If you don't commit those changes to a new image, those changes will be lost when the container is removed.
  - **Image dependency.** A new container image has the container image from which it was created as a dependency—for example, if you create a container image named WebServer-1.0 that has the IIS role installed from the Server Core base OS image, and the Server Core base OS image is a dependency for the WebServer-1.0 image. This dependency is very specific, and you can't use an updated version of the Server Core base OS image as a replacement for the version that you used when creating the WebServer-1.0 image. You can then export this container image that only records the changes made to another container host. You can start the image as long as the dependency container OS base image is present on that container host. You can have multiple images in a dependency chain.

- **Container host.** A container host is a computer that runs containers. A container host can be virtual, physical, or even a cloud-based Platform as a Service (PaaS) solution.
- **Container registries.** Container registries are central storehouses of container images. While it's possible to transfer containers or copy them to file shares using tools like FTP, common practice is to use a container registry as a repository for container images. Public container registries, such as the one that hosts Microsoft's base OS images, also store previous versions of the container base OS images. This allows you to retrieve earlier versions of the container base OS image that other images may depend upon. Docker Hub is the primary public container registry used by Microsoft to publish images, though the container images themselves are hosted in the Microsoft Container Registry at *mcr.microsoft.com*. Container registries can be public or private. When working with your own organization's container images, you have the option of creating and maintaining a private container registry.

## Inside OUT

### *Data and Containers*

Containers offer a non-persistent, isolated application execution environment. An application should run in the container, but it shouldn't store application data in a container. Application data should be stored in a location separate from the container, such as in a back-end database server, Azure storage blob, or other persistent alternative. Containers are good at serving in stateless roles, such as load balanced front-end web servers; this is especially true because you can start new containers quickly to handle additional loads as required.

## Isolation modes

Windows Server supports two container isolation modes: Process Isolation mode and Hyper-V Isolation mode. In previous versions of Microsoft's documentation, these isolation modes were occasionally called "Windows Server containers" and "Hyper-V containers." Windows 10 supports only Hyper-V Isolation mode. Windows Server 2016 and Windows Server 2019 support Process Isolation and Hyper-V Isolation modes.

### Process Isolation mode

Process Isolation mode provides a container with process and namespace isolation. Containers running in this Isolation mode share a kernel with all other containers running on the container host. This is similar to the manner in which containers run on Linux container hosts. Process Isolation is the default mode used when running a container. If you want to ensure that the container is being used, start the container using the `--isolation=process` option.

## Hyper-V Isolation mode

A container running in Hyper-V Isolation mode runs in a highly optimized virtual machine that also provides an isolated application execution environment. Hyper-V Isolation mode containers don't share the kernel with the container host, nor do they share the kernel with other containers on the same container host. You can only use Hyper-V Isolation mode containers if you have enabled the Hyper-V role on the container host. If the container host is a Hyper-V virtual machine, you will need to enable Nested Virtualization. By default, a container uses the Process Isolation mode. You can start a container in Hyper-V Isolation mode by using the `--isolation=hyperv` option.

For example, to create a Hyper-V container from the `microsoft/windowsservercore` image, issue this command:

```
Docker run -it --isolation=hyperv mcr.microsoft.com/windows/servercore cmd
```

### More Info

#### *Container isolation*

You can learn more about container isolation at <https://docs.microsoft.com/en-us/virtualization/windowscontainers/manage-containers/hyperv-container>.

## Managing containers with Docker

Containers on Windows Server are managed using the Docker engine. The advantage is that the command syntax of Docker on Windows is almost identical to the command-line tools in Docker on Linux. While there is a community-maintained PowerShell module for managing containers on Windows Server available on GitHub, PowerShell is not the primary tool for Windows Server container management, and very few people use PowerShell to manage containers.

For Windows Server administrators unfamiliar with Docker syntax, the commands include extensive help support. Typing **Docker** at the command prompt will provide an overview of the high-level Docker functionality. You can learn more about specific commands by using the `--help` command parameter. For example, the following command will provide information about the `Docker Image` command:

```
docker image --help
```

### More Info

#### *Docker Engine on Windows*

You can learn more about Docker Engine on Windows at <https://docs.microsoft.com/en-us/virtualization/windowscontainers/manage-docker/configure-docker-daemon>.

## Installing Docker

Docker is not included with the Windows Server installation media, and you don't install it as a role or feature. Instead, you need to install Docker from the PowerShell Gallery. Although this is unusual for an important role on a Windows Server operating system, it does have the advantage of ensuring that you have the latest version of Docker.

The simplest way to install Docker is to ensure that your Windows Server computer has Internet connectivity. You then run the following command from an elevated PowerShell prompt, as shown in Figure 10-1, to install the Docker Microsoft Provider:

```
Install-Module -Name DockerMsftProvider -Repository PSGallery -Force
```

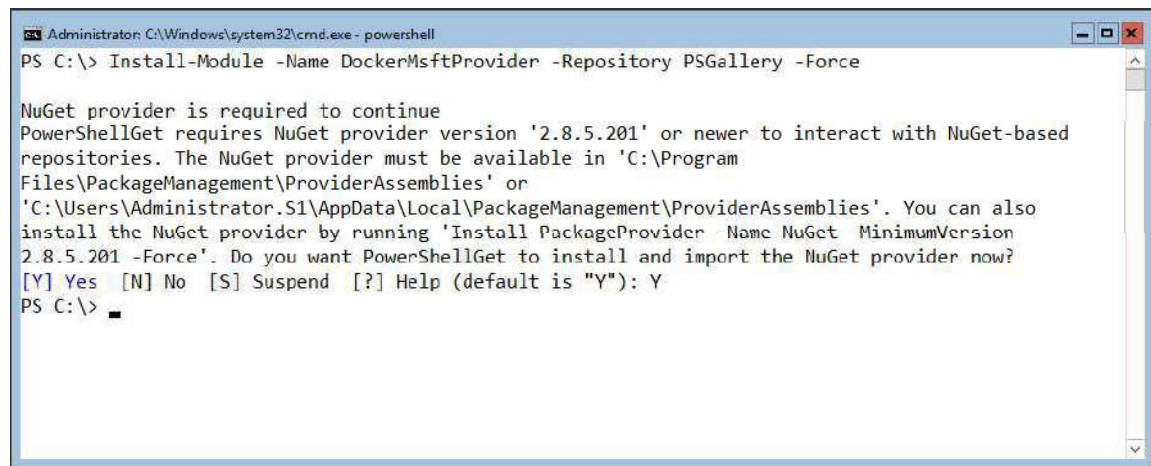


FIGURE 10-1 Install DockerMsftProvider

Next, install the most recent version of Docker. You do this by executing the following command, as shown in Figure 10-2; choose Yes when prompted to install software from a source not marked as Trusted.

```
Install-Package -Name docker -ProviderName DockerMsftProvider
```



FIGURE 10-2 Install Docker

You'll then need to restart the computer that will function as the container host to complete the installation of Docker. You can update the version of Docker when a new one becomes available by rerunning the following command:

```
Install-Package -Name docker -ProviderName DockerMsftProvider
```

## Demon.json

If you want to change the default Docker Engine settings, such as whether to create a default NAT network, you need to create and configure the Docker Engine configuration file. This file doesn't exist by default. When it is present, the settings in the file override the Docker Engine's default settings.

You should create this file in the `C:\ProgramData\Docker\config` folder. Before editing the `Demon.json` file, you'll need to stop the Docker service using the following command:

```
Stop-Service docker
```

You only need to add settings that you want to change to the configuration file. For example, if you only want to configure the Docker Engine to accept incoming connections on port 2701, you add the following lines to `demon.json`:

```
{
  "hosts": ["tcp://0.0.0.0:2701"]
}
```

The Windows Docker Engine doesn't support all possible Docker configuration file options. The ones that you can configure are shown below:

```
{
  "authorization-plugins": [],
  "dns": [],
  "dns-opts": [],
  "dns-search": [],
  "exec-opts": [],
  "storage-driver": "",
  "storage-opts": [],
  "labels": [],
  "log-driver": "",
  "mtu": 0,
  "pidfile": "",

  "cluster-store": "",
  "cluster-advertise": "",
  "debug": true,
  "hosts": [],
  "log-level": "",
  "tlsverify": true,
  "tlscacert": "",
  "tlscert": "",
  "tlskey": "",
```

```

"group": "",
"default-ulimits": {},
"bridge": "",
"fixed-cidr": "",
"raw-logs": false,
"registry-mirrors": [],
"insecure-registries": [],
"disable-legacy-registry": false
}

```

These options allow you to do the following when starting the Docker Engine:

- **authorization-plugins.** Which authorization plug-ins the Docker Engine should load
- **dns.** Which DNS server the containers should use for name resolution
- **dns-opts.** Which DNS options to use
- **dns-search.** Which DNS search domains to use
- **exec-opts.** Which runtime execution options to use
- **storage-driver.** Specify the storage driver
- **storage-opts.** Specify the storage driver options
- **labels.** Docker Engine labels
- **log-driver.** Default driver for the container logs
- **mtu.** Container network MTU
- **pidfile.** Path to use for daemon PID file
- **group.** Specify the local security group that has permissions to run Docker commands
- **cluster-store.** Cluster store options
- **cluster-advertise.** Cluster address to advertise
- **debug.** Enable Debug mode
- **hosts.** Daemon sockets to connect to
- **log-level.** Logging detail
- **Tlsverify.** Use TLS and perform verification
- **tlscacert.** Specify which Certificate Authorities to trust
- **tlscert.** Location of the TLS certificate file

- **tlskey.** Location of the TLS key file
- **group.** UNIX socket group
- **default-ulimits.** Default ulimits for containers
- **bridge.** Attach containers to network bridge
- **fixed-cidr.** IPv4 subnet for static IP address
- **raw-logs.** Log format used
- **registry-mirrors.** Preferred registry mirror
- **insecure-registries.** Allow insecure registry communication
- **disable-legacy-registry.** Block contacting legacy registries

Once you have made the necessary modifications to the *daemon.json* file, you should start the Docker service by running this PowerShell command:

```
Start-Service docker
```

## Retrieving container OS image

You can retrieve the Server Core base OS container image by running the following command, as shown in Figure 10-3:

```
docker pull mcr.microsoft.com/windows/servercore:ltsc2019
```



```

Administrator: Windows PowerShell
PS C:\> docker pull mcr.microsoft.com/windows/servercore:ltsc2019
ltsc2019: Pulling from windows/servercore
65014b3c3121: Pull complete
e96b0897c5d1: Pull complete
Digest: sha256:e75260361cbd398788c195a159c2c56e3ef4ee011ecd9bd57bdac73ff1a43e33
Status: Downloaded newer image for mcr.microsoft.com/windows/servercore:ltsc2019
mcr.microsoft.com/windows/servercore:ltsc2019
PS C:\>

```

FIGURE 10-3 Pull OS image

You can retrieve the Nano Server base OS container image by running this command:

```
docker pull mcr.microsoft.com/windows/nanoserver:1909
```




Unlike the Server Core image, where you will want to pull the Long Term Servicing Channel (LTSC) version, with Nano Server you'll need to determine the most recent supported version, keeping in mind that Nano Server images are only supported for 18 months. You can do that by looking at the image page on Docker Hub at [https://hub.docker.com/\\_/microsoft-windows-nanoserver](https://hub.docker.com/_/microsoft-windows-nanoserver).

The latest version of the Server Core image is updated frequently. Because of dependency issues when working with older containers, you might need an earlier version of the base OS image. To retrieve all versions of the *WindowsServerCore* image from the public registry, run this command:

```
docker pull -a mcr.microsoft.com/windows/servercore
```

To retrieve all versions of the Nano Server image, as shown in Figure 10-4, run this command:

```
Docker pull -a mcr.microsoft.com/windows/nanoserver
```



```
Administrator: C:\Windows\system32\cmd.exe - powershell
cf62dbf6d334: Pull complete
Digest: sha256:e161c43c9695a20d0b7271e7339bb041026db548667d2d9ecc04e8dc6fba9bed
10.0.14393.206: Pulling from microsoft/nanoserver
5496abde368a: Pull complete
Digest: sha256:1c514beb110052b91235dfd4a9994d7bcadb15682d061be6a6aedc6dfdaae3
10.0.14393.206_cs-cz: Pulling from microsoft/nanoserver
c0917285e823: Pull complete
Digest: sha256:3200a3f3d9c4e898347134f5a04a398b6035587ebc871c100ca749a195f47a47
10.0.14393.206_de-de: Pulling from microsoft/nanoserver
48899ba2f3b3: Pull complete
Digest: sha256:75aad42d69ed123d23b7529980b3bf7b915e7e6838e8679bd827d529721dc3f2
10.0.14393.206_es-es: Pulling from microsoft/nanoserver
b5fa4f3fecd6: Pull complete
Digest: sha256:d1c809c7617e8d8beb825bea2285313e7fbc83c344e4559db255f2e9b2a18afe
10.0.14393.206_fr-fr: Pulling from microsoft/nanoserver
215b98cc20f3: Extracting 243.6 MB/243.6 MB
```

FIGURE 10-4 Pulling all versions of an OS image

## Container registries and images

Container registries are repositories for the distribution of container images. The main container registry that will be of interest to Windows Server administrators is the Docker Hub public repository. Microsoft posts container images on the Docker Hub registry, including the base OS images, images that include evaluation editions of SQL Server, and technologies such as the latest builds of ASP.NET on containers and Azure CLI. You can view all Microsoft's published container images at <https://hub.docker.com/u/microsoft>.

From the Docker Hub registry, you can retrieve the following published Microsoft container images:

- **Nanoserver.** This is the base image for the Nano Server container operating system.

```
docker pull mcr.microsoft.com/windows/nanoserver:1909
```



- **Windows Server Core.** This is the base image for the Windows Server Core container operating system.

```
docker pull mcr.microsoft.com/windows/servercore:ltsc2019
docker pull mcr.microsoft.com/windows/servercore:1909
```

- **Windows IoT Core.** This is the base image for Windows IoT containers.

```
docker pull mcr.microsoft.com/windows/iotcore:1809
```

- **Windows IIS.** This includes the Internet Information Services on Windows Server Core container operating system.

```
docker pull mcr.microsoft.com/windows/servercore/iis:windowsservercore-ltsc2019
docker pull mcr.microsoft.com/windows/servercore/iis:windowsservercore-1909
```

- **Microsoft SQL Server.** This container image contains Microsoft SQL Server on Linux and will only run on Linux container hosts, rather than on Windows Server container hosts, unless you configure LCOW.

```
docker pull mcr.microsoft.com/mssql/server:2019-latest
```

- **Azure Cli.** This includes Azure CLI on a Linux container operating system.

```
docker pull mcr.microsoft.com/azure-cli
```

- **ASP.NET Core Runtime.** This includes ASP.NET Core on the Windows Server Core container operating system.

```
docker pull mcr.microsoft.com/dotnet/core/aspnet:3.1
```

- **Microsoft PowerShell.** This includes PowerShell on cross-platform container images.

```
docker pull mcr.microsoft.com/powershell
```

In cases in which multiple images exist, such as Windows Server Core and Nano Server, you can use the `-a` option with the Docker `pull` command to retrieve all images. This can be helpful if you don't know the image ID of the specific image that you wish to retrieve. For example, use this command when pulling the *Microsoft PowerShell* container image, which includes both Windows Server Core and Nano Server images: `docker pull -a mcr.microsoft.com/powershell`.

When you pull an image from a registry, the action will also download any dependency images that are required. For example, if you pull an image that was built on a specific version of the Windows Server Core base image and you don't have that image on the container host, that image will also be downloaded from a container registry.

You can view a list of images that are installed on a Windows Server container host by using the following command:

```
Docker image list
```

You can use Windows Admin Center to view a list of container images that are installed on a Windows Server container host, as shown in Figure 10-5. You can also use Windows Admin Center to delete images that are installed on a Windows Server container host.

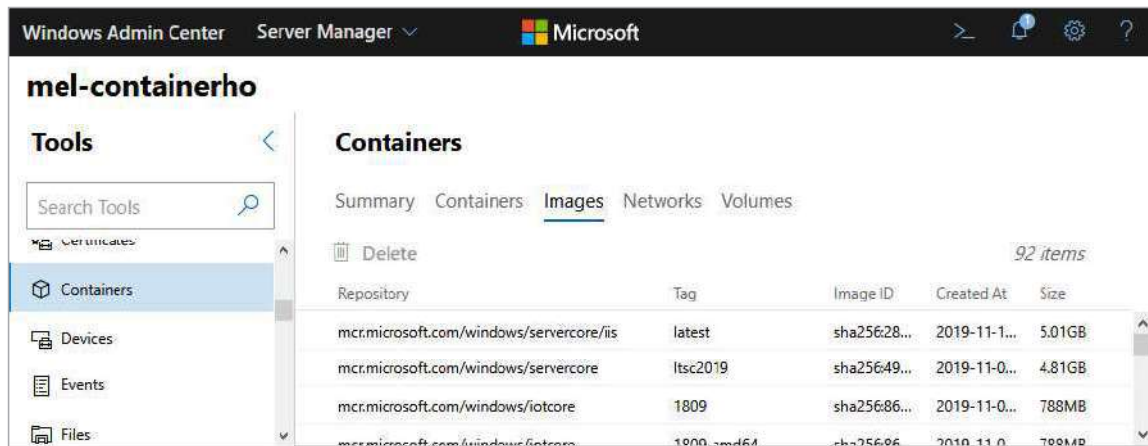


FIGURE 10-5 List of container images

## More Info

### Official Microsoft Container Images

You can see which container images Microsoft has made available at <https://hub.docker.com/u/microsoft/>.

## Managing containers

You use Docker to perform all container management tasks on computers running Windows Server. At present, the container management functionality available in Windows Admin Center is limited, but it's likely that over time, most tasks that you can perform from the Docker prompt will be available in WAC.

### Starting a container

You create a new container by specifying the container image from which you wish to create the container. You can start a container and run an interactive session either by specifying *cmd.exe* or *PowerShell.exe* by using the *-it* option with *docker run*. Interactive sessions allow you to directly interact with the container through the command line from the moment the container starts. Detached mode starts a container, but it doesn't start an interactive session with that container.

For example, to start a container from the *Microsoft/windowsservercore* image and to enter an interactive PowerShell session within that container once it is started, use this command:

```
Docker run -it mcr.microsoft.com/windows/servercore:ltsc2019 powershell.exe
```

Also important is that, by default, containers use network address translation. This means that if you are running an application or service on the container that you want to expose to the network, you'll need to configure port mapping between the container host's network interface and the container. For example, you would run the following command if you had downloaded the *Microsoft/iis* container image and you wanted to start a container in Detached mode and map port 8080 on the container host to port 80 on the container (see Figure 10-6).

```
Docker run -d -p 8080:80 mcr.microsoft.com/windows/servercore/iis:windowsservercore-ltsc2019
```

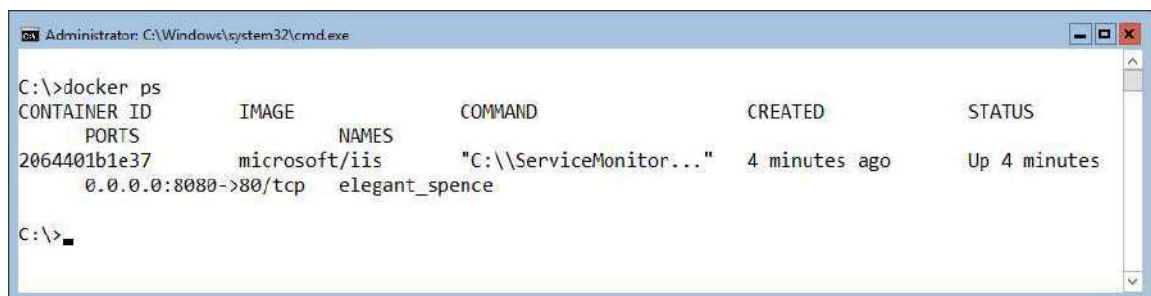


```
Administrator: Windows PowerShell
PS C:\> Docker run -d -p 8080:80 mcr.microsoft.com/windows/servercore/iis:windowsservercore-ltsc2019
8aa8ff8163f71de882037d725b21098c4cb9c7e5ea1109e5cf62f030a5fe53a3
PS C:\>
```

FIGURE 10-6 Port mapping

This is only the very basic sort of information you'd need to get started with a container, and you will learn more about container networking later in this chapter.

You can verify which containers are running by using the *docker ps* command, as shown in Figure 10-7, or by using Windows Admin Center. The problem with the simple *docker ps* command option is that this will only show you the running containers and won't show you any that are in a stopped state. You can see which containers are on a container host, including containers that aren't currently running, by using the *docker ps -a* command.



```
Administrator: C:\Windows\system32\cmd.exe
C:\>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS              NAMES
2064401b1e37       microsoft/iis      "C:\\ServiceMonitor..." 4 minutes ago      Up 4 minutes
0.0.0.0:8080->80/tcp elegant_spence
```

FIGURE 10-7 Listing containers

You can also view the containers that are on a Windows Server container host using Windows Admin Center, if you have loaded the Containers extension, as shown in Figure 10-8.

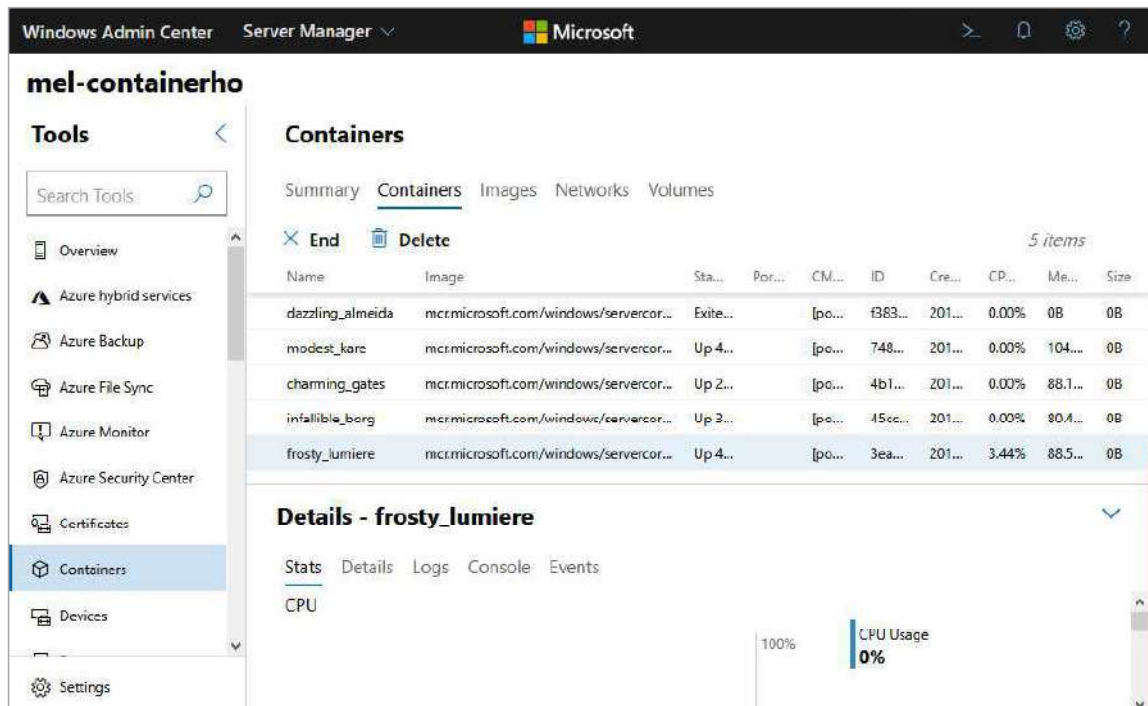


FIGURE 10-8 Windows Admin Center container list

One thing that you'll notice about containers is that they appear to be assigned random names, such as *sarcastic\_hedgehog*, *dyspeptic\_hamster*, and *sententious\_muppet*. Docker assigns random names, rather than asking you for one, because containers are a more ephemeral type of application host than a VM; because they are likely to only have a short lifespan, it isn't worth assigning any name that you'd need to remember. The reason for the structure of the random names is that they are easy to remember in the short term, which makes containers that you must interact with on a short-term basis easier to address than when using hexadecimal container IDs. Figure 10-9 shows the output of the `docker ps -a` command and shows the names of containers including: *charming\_gates*, *infallible\_borg*, *frosty\_lumiere*, *modest\_kare*, and *dazzling\_alemida*.



FIGURE 10-9 List all images

Earlier in the chapter, you learned that it was possible to start a container in Detached mode. If you want to start an interactive session on a container that you started in Detached mode, use the `docker exec -i <containername> powershell.exe` command. Figure 10-10 shows entering an interactive session on a container started in Detached mode. The `hostname` command displays the container's ID.



```

Administrator: Windows PowerShell
PS C:\> docker ps
CONTAINER ID        IMAGE                                     PORTS                NAMES
-----
8aa8ff8163f7       mcr.microsoft.com/windows/servercore/iis:windowsservercore-ltsc2019  0.0.0.0:8080->80/tcp  gracious_robinson
PS C:\> docker exec -i gracious_robinson powershell.exe
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\> hostname
hostname
8aa8ff8163f7
PS C:\>

```

FIGURE 10-10 Starting an interactive session

In some cases, it will be necessary to start a stopped container. You can start a stopped container using the `Docker start <containername>` command.

## Modifying a running container

Once you have a container running, you can enter the container and make the modifications that you want to make to ensure that the application the container hosts will run correctly. This might involve creating directories, using the `dism.exe` command to add roles, or using the `wget` PowerShell alias to download and install binaries such as Java. For example, the following code, when run from inside a container, downloads and installs an older version of Java into a container based on the Server Core base OS container:

```

wget -Uri "http://javadl.sun.com/webapps/download/AutoDL?BundleId=107944" -outfile
javaInstall.exe -UseBasicParsing
REG ADD HKLM\Software\Policies\Microsoft\Windows\Installer /v DisableRollback /t REG_
DWORD /d 1 | Out-Null ./javaInstall.exe /s INSTALLDIR=C:\Java REBOOT=Disable | Out-Null

```

Once you are finished modifying the container, you can type **Exit** to exit the container. A container must be in a shut-down state before you can capture it as a container image. You use the `docker stop <containername>` cmdlet to shut down a container.

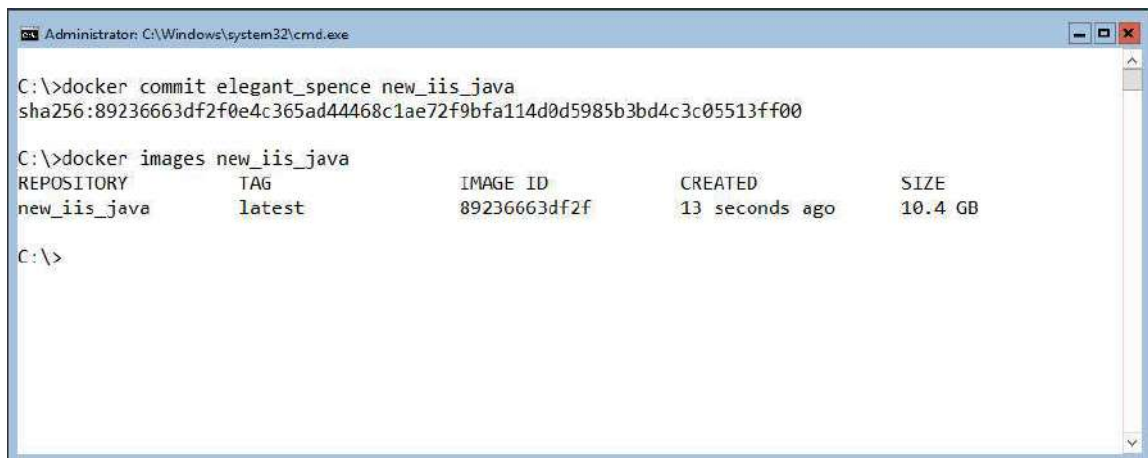
## Inside OUT

### *Containers and Chocolatey*

Chocolatey offers another method of installing software on containers. You can spin up a new container, install Chocolatey, and then create a new image from that container. The process of installing additional software will be simpler each time you spin up a new container from that new image that includes Chocolatey.

## Creating a new image from a container

Once the container is in the desired state and shut down, you can capture the container to a new container image. You do this using the `docker commit <container_name> <new_image_name>` command. Figure 10-11 shows committing a modified container to a new container image named `new_iis_java` and verifying the properties of the image using the `docker images` command.



```

Administrator: C:\Windows\system32\cmd.exe

C:\>docker commit elegant_spence new_iis_java
sha256:89236663df2f0e4c365ad44468c1ae72f9bfa114d0d5985b3bd4c3c05513ff00

C:\>docker images new_iis_java
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
new_iis_java        latest      89236663df2f     13 seconds ago  10.4 GB

C:\>

```

FIGURE 10-11 Docker commit

Once you have committed a container to a container image, you can remove the container using the `docker rm` command. For example, to remove the `elegant_spence` container, issue this command:

```
Docker rm elegant_spence
```

## Using Dockerfiles

Dockerfiles are text files that allow you to automate the process of creating new container images. You use a Dockerfile with the `docker build` command to automate container creation, which is very useful when you need to create new container images from regularly updated base OS container images.



Dockerfiles have the elements shown in Table 10-1.

**Table 10-1** Dockerfile Element

Instruction	Description
FROM	Specifies the container image used in creating the new image creation. For example: <i>FROM mcr.microsoft.com/windows/servercore:ltsc2019</i>
RUN	Specifies commands to be run and captures them into the new container image. For example: <i>RUN wget -Uri "http://javadl.sun.com/webapps/download/AutoDL?BundleId=107944" -outfile javaInstall.exe -UseBasicParsing</i> <i>RUN REG ADD HKLM\Software\Policies\Microsoft\Windows\Installer /v DisableRollback /t REG_DWORD /d 1   Out-Null</i> <i>RUN ./javaInstall.exe /s INSTALLDIR=C:\Java REBOOT=Disable   Out-Null</i>
COPY	Copies files and directories from the container host filesystem to the filesystem of the container. For windows containers, the destination format must use forward slashes. For example: <i>COPY example1.txt c:/temp/</i>
ADD	Can be used to add files from a remote source, such as a URL. For example: <i>ADD https://www.python.org/ftp/python/3.5.1/python-3.5.1.exe</i>
WORKDIR	Specifies a working directory for the <i>RUN</i> and <i>CMD</i> instructions.
CMD	A command to be run when deploying an instance of the container image.

For example, the following Dockerfile will create a new container from the *mcr.microsoft.com/windows/servercore:ltsc2019* image, create a directory named *ExampleDirectory*, and then install the IIS Webserver feature.

```
FROM mcr.microsoft.com/windows/servercore:ltsc2019
RUN mkdir ExampleDirectory
RUN dism.exe /online /enable-feature /all /featurename:iis-webserver /NoRestart
```

To create a container image named *example\_image*, change into the directory that hosts the *Dockerfile* (no extension) file and run the following command:

```
Docker build -t example_image
```

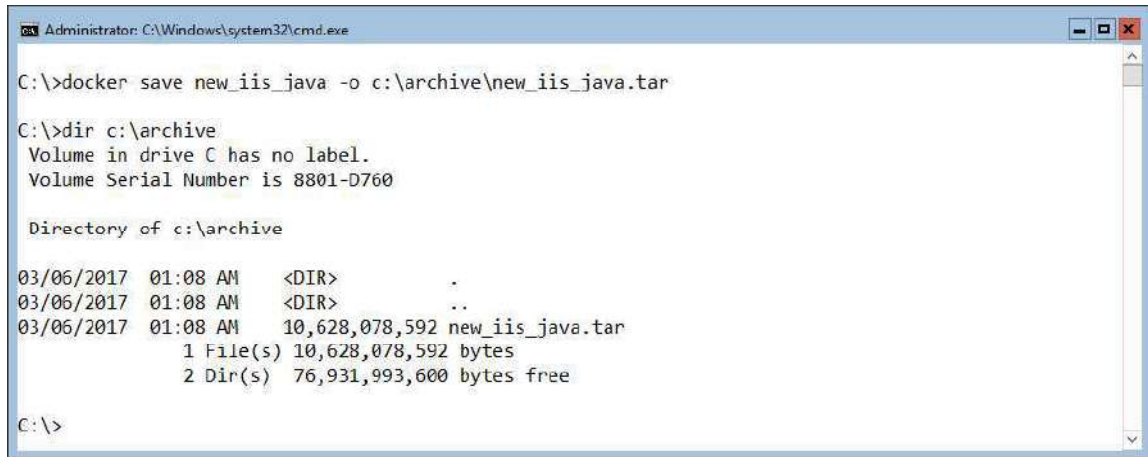
## More Info

### *Dockerfile with Windows*

You can learn more about using Dockerfile with Windows at <https://docs.microsoft.com/en-us/virtualization/windowscontainers/manage-docker/manage-windows-dockerfile>.

## Managing container images

To save a Docker image for transfer to another computer, use the `docker save` command. When you save a Docker image, you save it in `.tar` format. Figure 10-12 shows exporting the `new_iis_java` image to the `c:\archive\new_iis_java.tar` file.



```
Administrator: C:\Windows\system32\cmd.exe

C:\>docker save new_iis_java -o c:\archive\new_iis_java.tar

C:\>dir c:\archive
Volume in drive C has no label.
Volume Serial Number is 8801-D760

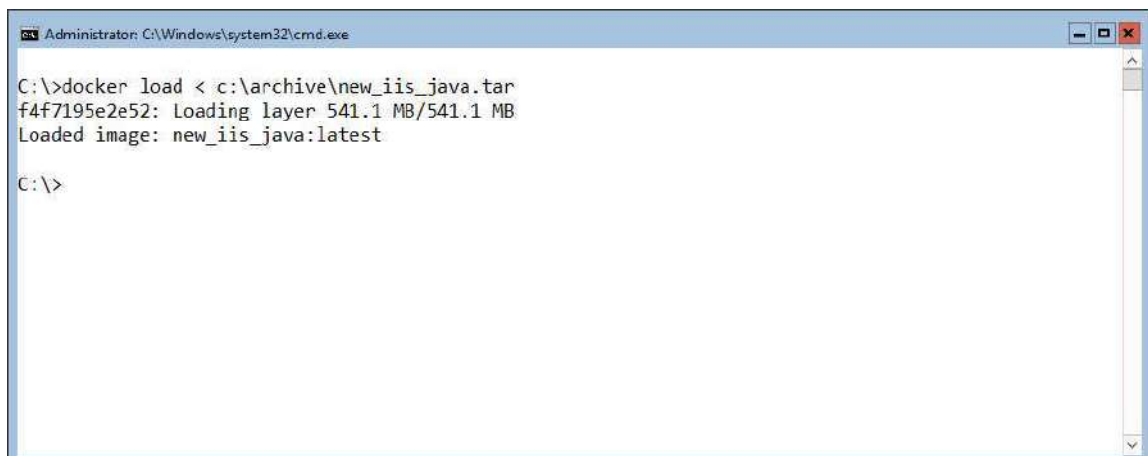
Directory of c:\archive

03/06/2017  01:08 AM  <DIR>          .
03/06/2017  01:08 AM  <DIR>          ..
03/06/2017  01:08 AM  10,628,078,592 new_iis_java.tar
               1 File(s) 10,628,078,592 bytes
               2 Dir(s)  76,931,993,600 bytes free

C:\>
```

FIGURE 10-12 `Docker save` command

You can load a Docker image from a saved image using the `docker load` command. For example, Figure 10-13 shows loading a container image from the `c:\archive\new_iis_java.tar` file created earlier.



```
Administrator: C:\Windows\system32\cmd.exe

C:\>docker load < c:\archive\new_iis_java.tar
f4f7195e2e52: Loading layer 541.1 MB/541.1 MB
Loaded image: new_iis_java:latest

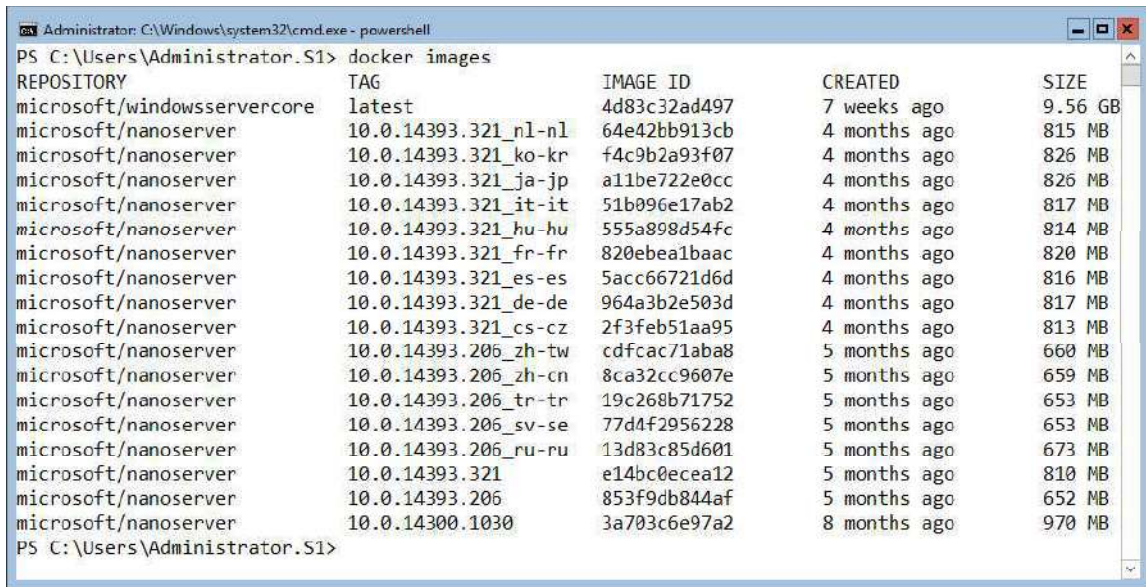
C:\>
```

FIGURE 10-13 Load archived image

When you have multiple container images that have the same name, you can remove an image by using the image ID. You can determine the image ID by running the `docker images` command, as shown in Figure 10-14. You can also use Windows Admin Center to view this



information. You can't remove a container image until the last container created from that image has been deleted either directly or indirectly.



```

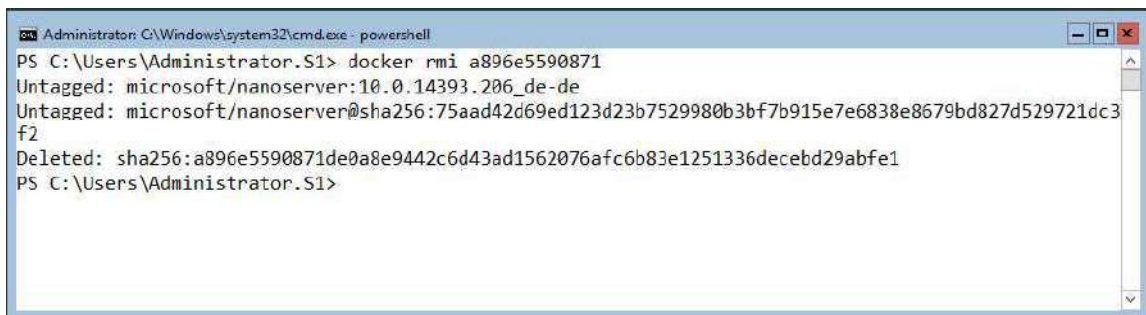
Administrator: C:\Windows\system32\cmd.exe - powershell
PS C:\Users\Administrator.S1> docker images
REPOSITORY          TAG                IMAGE ID           CREATED           SIZE
microsoft/windowsservercore  latest            4d83c32ad497      7 weeks ago     9.56 GB
microsoft/nanoserver  10.0.14393.321_n1-n1  64e42bb913cb      4 months ago     815 MB
microsoft/nanoserver  10.0.14393.321_ko-kr  f4c9b2a93f07      4 months ago     826 MB
microsoft/nanoserver  10.0.14393.321_ja-jp  a11be722e0cc      4 months ago     826 MB
microsoft/nanoserver  10.0.14393.321_it-it  51b096e17ab2      4 months ago     817 MB
microsoft/nanoserver  10.0.14393.321_hu-hu  555a898d54fc      4 months ago     814 MB
microsoft/nanoserver  10.0.14393.321_fr-fr  820ebea1baac      4 months ago     820 MB
microsoft/nanoserver  10.0.14393.321_es-es  5acc66721d6d      4 months ago     816 MB
microsoft/nanoserver  10.0.14393.321_de-de  964a3b2e503d      4 months ago     817 MB
microsoft/nanoserver  10.0.14393.321_cs-cz  2f3feb51aa95      4 months ago     813 MB
microsoft/nanoserver  10.0.14393.206_zh-tw  cdfcac71aba8      5 months ago     660 MB
microsoft/nanoserver  10.0.14393.206_zh-cn  8ca32cc9607e      5 months ago     659 MB
microsoft/nanoserver  10.0.14393.206_tr-tr  19c268b71752      5 months ago     653 MB
microsoft/nanoserver  10.0.14393.206_sv-se  77d4f2956228      5 months ago     653 MB
microsoft/nanoserver  10.0.14393.206_ru-ru  13d83c85d601      5 months ago     673 MB
microsoft/nanoserver  10.0.14393.321        e14bc0ecea12      5 months ago     810 MB
microsoft/nanoserver  10.0.14393.206        853f9db844af      5 months ago     652 MB
microsoft/nanoserver  10.0.14300.1030      3a703c6e97a2      8 months ago     970 MB
PS C:\Users\Administrator.S1>

```

FIGURE 10-14 Image list

You then remove the image by using the `docker rmi` command with the image ID. For example, Figure 10-15 shows the removal of the image `hu` with the ID `a896e5590871` using this command:

```
docker rmi a896e5590871
```



```

Administrator: C:\Windows\system32\cmd.exe - powershell
PS C:\Users\Administrator.S1> docker rmi a896e5590871
Untagged: microsoft/nanoserver:10.0.14393.206_de-de
Untagged: microsoft/nanoserver@sha256:75aad42d69ed123d23b7529980b3bf7b915e7e6838e8679bd827d529721dc3f2
Deleted: sha256:a896e5590871de0a8e9442c6d43ad1562076afc6b83e1251336decebd29abfe1
PS C:\Users\Administrator.S1>

```

FIGURE 10-15 Remove image

You can also remove a container image by selecting the image in the list of containers in Windows Admin Center and clicking Delete.

## Service accounts for Windows containers

Although containers based on the Server Core and Nano Server operating systems have most of the same characteristics as a virtual machine or a bare-metal deployment of the Server Core or Nano Server versions of Windows Server, one thing that you can't do with containers is to join them to a domain. This is because containers are supposed to be temporary, rather than permanent, and domain-joining them would clog up Active Directory with unnecessary computer accounts.

While containers can't be domain-joined, it is possible to use a group-managed service account (gMSA) to provide one or more containers with a domain identity similar to that used by a device that is realm-joined. Performing this task requires downloading the Windows Server Container Tools and ensuring that the container host is a member of the domain that hosts the gMSA. When you perform this procedure, the container's LocalSystem and Network Service accounts use the gMSA. This gives the container the identity represented by the gMSA.

To configure gMSA association with a container, perform the following steps:

1. Ensure that the Windows Server container host is domain-joined.
2. Add the container host to a specially created domain security group. This domain security group can have any name.
3. Create a gMSA and grant gMSA permission to the specially created domain security group of which the container host is a member.
4. Install the gMSA on the container host.
5. Use the *New-CredentialSpec* cmdlet on the container host to generate the gMSA credentials in a file in JSON format. This cmdlet is located in the *CredentialSpec* PowerShell module, which is available as a part of the Windows Server Container tools. For example, if you created a gMSA named *MelbourneAlpha*, you would run the following command:

```
New-CredentialSpace -Name MelbourneAlpha -AccountName MelbourneAlpha
```

6. You can verify that the credentials have been saved in JSON format by running the *Get-CredentialSpec* cmdlet. By default, credentials are stored in the *C:\ProgramData\Docker\CredentialSpecs\* folder.
7. Start the container using the *--security-opt "credentialspec="* option and specify the JSON file containing the credentials associated with the gMSA. For example, run the following command if the credentials are stored in the file *tw\_t\_webapp01.json*:

```
docker run --security-opt "credentialspec=file://tw_t_webapp01.json" --hostname webapp01 -it mcr.microsoft.com/windows/servercore:1tsc2019 powershell
```

Once you've configured the container to indirectly use the gMSA for its Local System and Network Service accounts, you can provide the container with permissions to access domain resources by providing access to the gMSA. For example, if you wanted to provide the container with access to the contents of a file share hosted on a domain member, you can configure permissions so that the gMSA has access to the file share.

## More Info

### *Active Directory Container Service Accounts*

You can learn more about Active Directory Container Service Accounts at <https://docs.microsoft.com/en-us/virtualization/windowscontainers/manage-containers/manage-serviceaccounts>.

## Applying updates

One of the concepts that many IT operations personnel find challenging is that you don't update a container that is deployed in production. Instead, you create a fresh container from the original container image, update that container, and then save that updated container as a new container image. You then remove the container in production and deploy a new container to production that is based on the newly updated image.

For example, you have a container that hosts a web app deployed from a container image named *WebApp1* that is deployed in production. The developers in your organization release an update to *WebApp1* that involves changing some existing settings. Rather than modifying the container in production, you start another container from the *WebApp1* image, modify the settings, and then create a new container image named *WebApp2*. You then deploy a new container into production from the *WebApp2* container image and remove the original un-updated container.

While you can manually update your container base OS images by applying software updates, Microsoft releases updated versions of the container base images each time a new software update is released. Once a new container OS base image is released, you or your organization's developers should update existing images that are dependent on the container OS base image. Regularly updated container base OS images provide an excellent reason for eventually moving toward using Dockerfiles to automate the process of building containers. If you have a Dockerfile configured for each container image used in your organization, updating your Container base OS images when a new Container base OS image is released is a quick, painless, and automated process.

## Inside OUT

### *Keep old OS images*

You'll need to keep your old container OS base images around until all the container images that are dependent on that old container OS base image have been replaced by newly updated container images.

## Container networking

Each container has a virtual network adapter. This virtual network adapter connects to a virtual switch, through which inbound and outbound traffic is sent. Networking modes determine how network adapters function in terms of IP addressing—meaning whether they use NAT or are connected to the same network as the container host.

Windows Containers support the following networking modes:

- **NAT.** Each container is assigned an IP address from the private 172.16.0.0 /12 address range. When using NAT, you can configure port forwarding from the container host to the container endpoint. If you create a container without specifying a network, the container will use the default NAT network. The Docker service creates its own default NAT network. When the container host reboots, the NAT network will not be created until the Docker service has restarted. Any container that was attached to the existing NAT network and that is configured to persist after reboot (for example, because it uses the *-restart always* option) will reattach to the NAT network that is newly created when the Docker service restarts.
- **Transparent.** Each container endpoint connects to the physical network. The containers can have IP addresses assigned statically or through DHCP.
- **Overlay.** Use this mode when you have configured the Docker Engine to run in Swarm mode. Overlay mode allows container endpoints to be connected across multiple container hosts.
- **L2 Bridge.** Container endpoints are on the same IP subnet used by the container host. IP addresses must be assigned statically. All containers on the container host have the same MAC address.
- **L2 Tunnel.** This mode is only used when you deploy containers in Azure.

You can list available networks using the following command, as shown in Figure 10-16:

```
docker network ls
```

```

Administrator: C:\Windows\system32\cmd.exe

C:\>docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
8d46d3fdd012       nat                 nat                 local
32322690f4d0       none                null                local

C:\>

```

FIGURE 10-16 Network list

You can also view a list of container networks in Windows Admin Center, as shown in Figure 10-17.

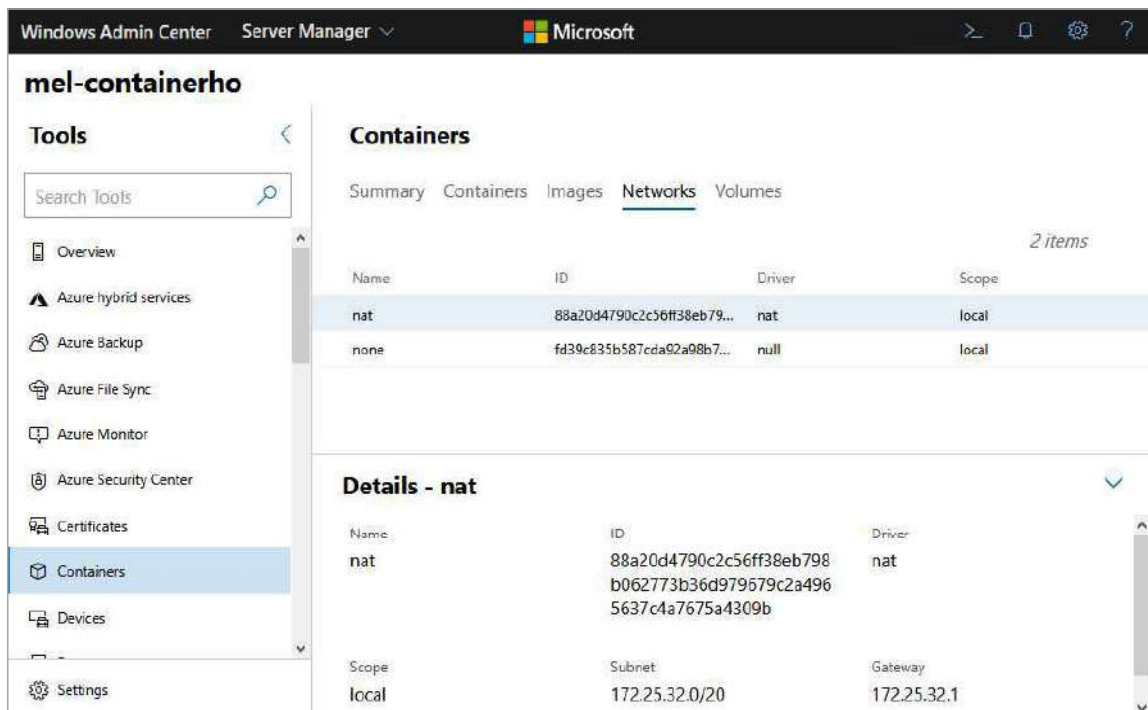


FIGURE 10-17 Windows Admin Center network list

To view which containers are connected to a specific network, run this command:

```
Docker network inspect <network name>
```

You can create multiple container networks on a container host, but you need to keep in mind the following limitations:

- If you are creating multiple networks of the transparent or L2 bridge type, you need to ensure that each network has a separate network adapter.
- If you create multiple NAT networks on a container host, additional NAT network prefixes will be partitioned from the container host's NAT network's address space. (By default, this is 172.16.0.0/12.) For example, these will be 172.16.1.0/24, 172.16.2.0/24, and so on.

## NAT

Network Address Translation (NAT) allows each container to be assigned an address in a private address space, while connecting to the container host's network uses the container host's IP address. The default NAT address range for containers is 172.16.0.0 /16. If the container host's IP address is in the 172.16.0.0 /12 range, you will need to alter the NAT IP prefix. You can do this by performing the following steps:

1. Stop the Docker service.
2. Remove any existing NAT networks using the following command:
 

```
Get-ContainerNetwork | Remove-ContainerNetwork
```
3. Perform one of the following actions:
  - Edit the *daemon.json* file and configure the *"fixed-cidr": "< IP Prefix > /Mask"* option to the desired network address prefix.
  - Edit the *daemon.json* file and set the *"bridge": "none"* option, and then use the *docker network create -d* command to create a network. For example, use the following command to create a network that uses the 192.168.15.0/24 range, the default gateway of 192.168.15.1, and is named *CustomNat*:
 

```
Docker network create -d nat --subnet=192.168.15.0/24 --gateway=192.168.15.1 CustomNat
```
4. Start the Docker service.

You can also allow connections to custom NAT networks when a container is run by allowing the use of the *--Network* parameter and specifying the custom NAT network name. To do this, you need to have the *"bridge: none"* option specified in the *daemon.json* file. Use the following command to run a container and join it to the *CustomNat* network created earlier:

```
Docker run -it --network=CustomNat <ContainerImage> <cmd>
```



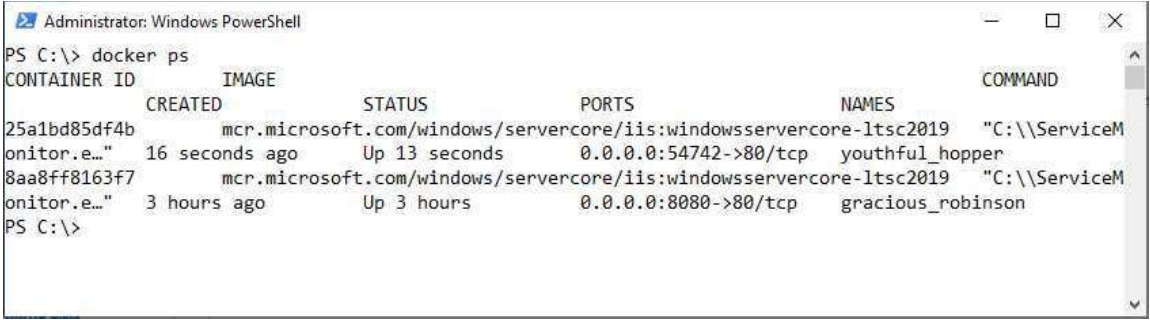
Port mappings allow ports on the container host to be mapped to ports on the container. For example, use the following command to create the container to map port 8080 on the container host to port 80 on a new container created from the *windowsservercore* image and to run *powershell.exe* interactively on the container:

```
Docker run -it -p 8080:80 microsoft/windowsservercore powershell.exe
```

Port mappings must be specified when the container is created or when it is in a stopped state. You can specify them using the *-p* parameter or the *expose* command in a Dockerfile when using the *-p* parameter. A random port will be assigned if you do not specify a port on the container host, but you do specify one on the container itself. For example, run this command:

```
Docker run -itd -p 80 mcr.microsoft.com/windows/servercore/iis:windowsservercore-1tsc2019 powershell.exe
```

A random port on the container host can be mapped through to port 80 on the container. You can determine which port is randomly assigned using the *docker ps* command. Figure 10-18 shows a container named *youthful\_hopper* that has had the port 54742 on the container host assigned to port 80 on the container.



```

Administrator: Windows PowerShell
PS C:\> docker ps
CONTAINER ID        IMAGE                                     STATUS              PORTS              NAMES
25a1bd85df4b      mcr.microsoft.com/windows/servercore/iis:windowsservercore-1tsc2019   Up 13 seconds      0.0.0.0:54742->80/tcp   youthful_hopper
8aa8ff8163f7      mcr.microsoft.com/windows/servercore/iis:windowsservercore-1tsc2019   Up 3 hours         0.0.0.0:8080->80/tcp   gracious_robinson
PS C:\>

```

FIGURE 10-18 *docker ps* command

When you configure port mapping, firewall rules on the container host will be created automatically that will allow traffic through.

## Transparent

Transparent networks allow each container endpoint to connect to the same network as the container host. You can use the Transparent Networking mode by creating a container network that has the driver name *transparent*. The driver name is specified with the *-d* option. Use this command:

```
docker network create -d transparent TransparentNetworkName
```

If the container host is a virtual machine running on a Hyper-V host, and you want to use DHCP for IP address assignment, it's necessary to enable *MACAddressSpoofing* on the VM network adapter, as shown in Figure 10-19. The Transparent Network mode supports IPv6. If you are using the Transparent Network mode, you can use a DHCPv6 server to assign IPv6 addresses to containers.

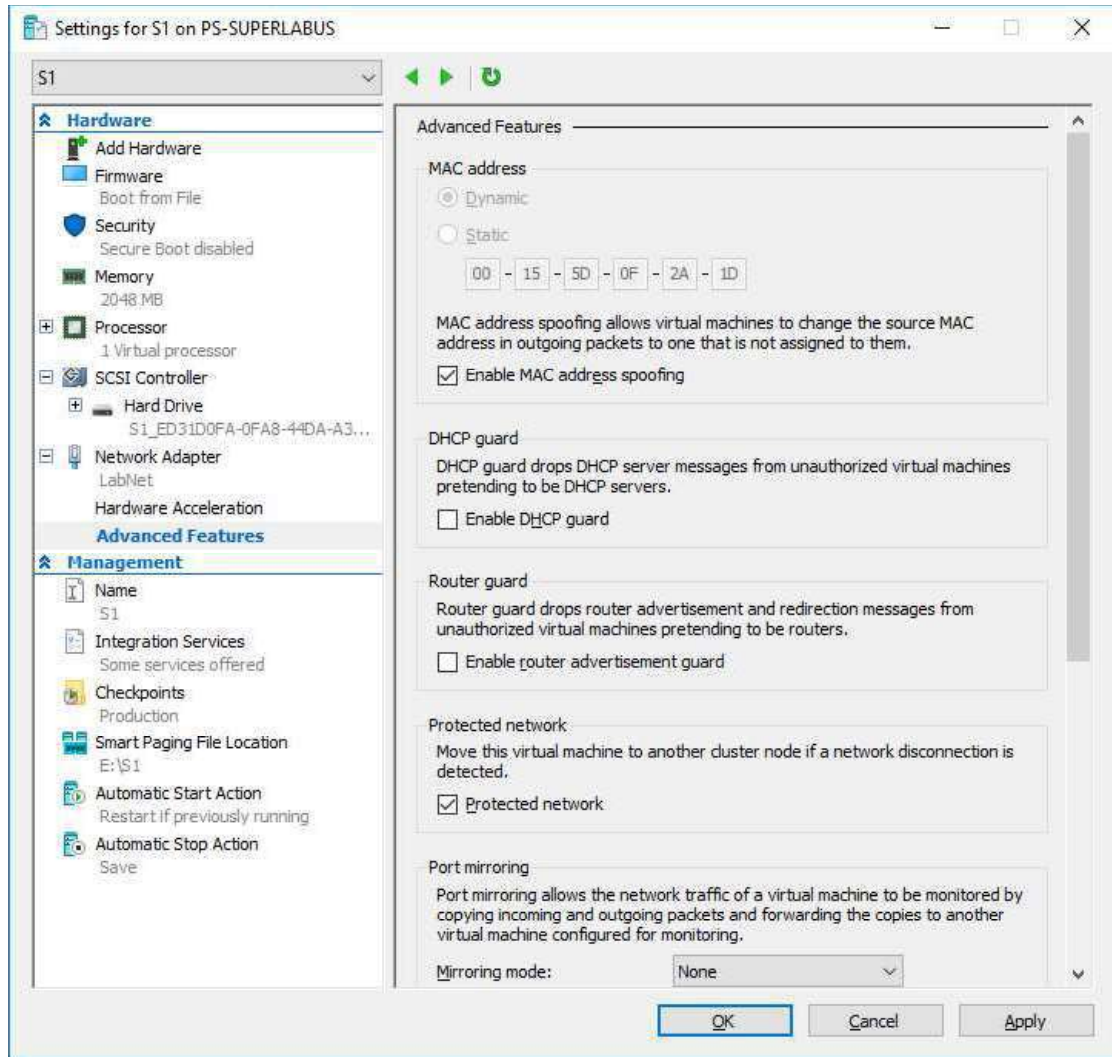


FIGURE 10-19 Enabling MAC address spoofing

If you want to manually assign IP addresses to containers, you must specify the subnet and gateway parameters when you create the transparent network. These network properties need to match the network settings of the network to which the container host is connected.



For example, let's say your container host is connected to a network that uses the 192.168.30.0/24 network and uses 192.168.30.1 as the default gateway. Run this command to create a transparent network that will allow static address assignment for containers on this network called *TransNet*:

```
Docker network create -d transparent --subnet=192.168.30.0/24 --gateway=192.168.30.1 TransNet
```

Once the transparent network is created with the appropriate settings, you can specify an IP address for a container using the `--ip` option. For example, to start a new container from the *microsoft/windowsservercore* image, enter the command prompt within the container. Run this command to assign it the IP address *192.168.30.101* on the *TransNet* network:

```
Docker run -it --network:TransNet --ip 192.168.30.101 microsoft/windowsservercore cmd.exe
```

Just like when you use a transparent network, containers are connected directly to the container host's network; you don't need to configure port mapping into the container.

## Overlay

You can only use Overlay Networking mode if the Docker host is running in Swarm mode as a Manager node. Each overlay network you create on a swarm cluster has its own IP subnet defined by an IP address prefix in the private address space. You create an overlay network by specifying *overlay* as the driver. For example, you can create an overlay network for the subnet 192.168.50.0/24 with the name *OverlayNet* by running the following command from a Swarm Manager node:

```
docker network create --driver=overlay --subnet=192.168.50.0/24 OverlayNet
```

## Layer 2 Bridge

Layer 2 Bridge (L2 Bridge) networks are similar to transparent networks in that they allow containers to have IP addresses on the same subnets as the container host. Layer 2 Bridge networks differ in that IP addresses must be assigned statically; this is because all containers on the container host that use an L2 Bridge network have the same MAC address.

When creating an L2 Bridge network, you must specify the network type as *l2bridge*. You must also specify subnet and default gateway settings that matches the subnet and default gateway settings of the container host. For example, use the following command to create an L2 Bridge network named *L2BridgeNet* for the IP address range 192.168.88.0/24 and with the default gateway address 192.168.88.1:

```
Docker network create -d l2bridge --subnet=192.168.88.0/24 --gateway=192.168.88.1 L2BridgeNet
```

## More Info

### Container networking

You can learn more about networking of containers at <https://docs.microsoft.com/en-us/virtualization/windowscontainers/container-networking/architecture>.

## Linux containers on Windows

If you configure Docker to run in Experimental mode, you can run Linux containers on a Windows Server computer that supports the Hyper-V Isolation mode concurrently with containers based on Windows Server images.

To configure Docker to run in the Experimental mode, edit the *daemon.json* in the *c:\program-data\docker\config* folder, and add this line:

```
{
  "experimental":true
}
```

You'll then need to run the following command from an elevated command prompt to set the appropriate environment variable:

```
[Environment]::SetEnvironmentVariable("LCOW_SUPPORTED", "1", "Machine")
```

You'll also need to go to the following GitHub repository at <https://github.com/linuxkit/lcow/releases> and download the files named *initrd.img* and *kernel* and place them in the *C:\Program Files\Linux Containers* folder, as shown in Figure 10-20.

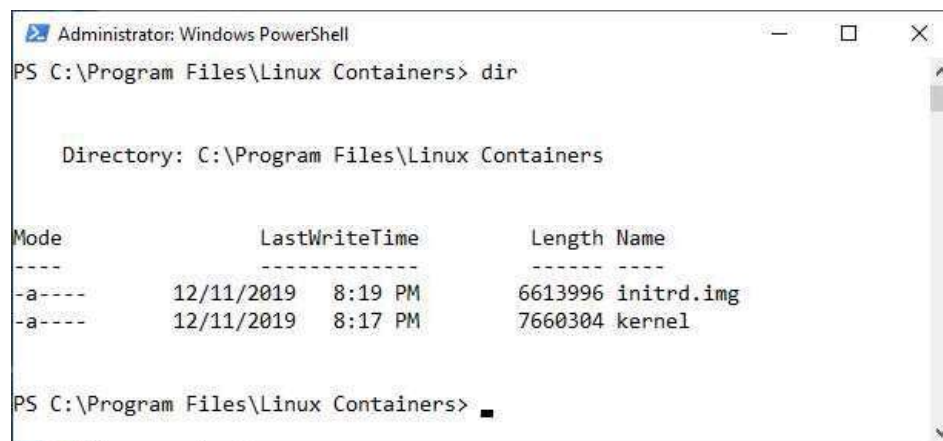


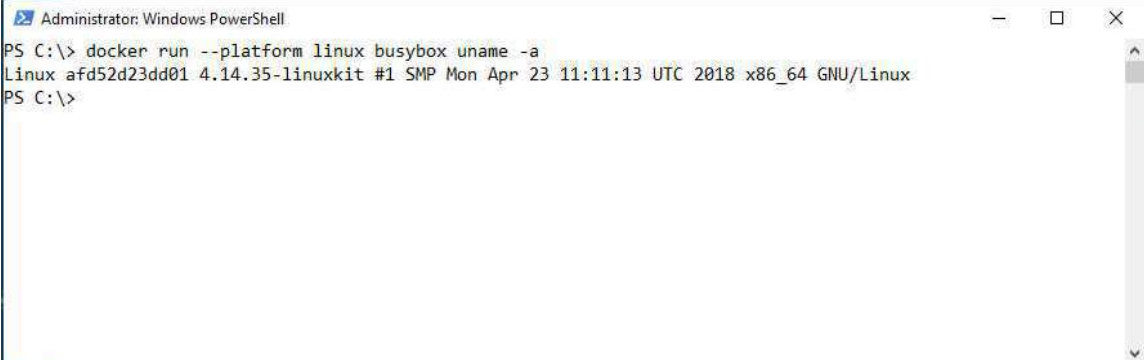
FIGURE 10-20 Files required for Linux Containers on Windows

Once you've performed these steps, restart the Docker service by running this command:

```
Restart-service *docker*
```

Once you've taken these steps, you'll be able to execute the following command to pull and run a Linux image, such as *busybox*, as shown in Figure 10-21.

```
docker run --platform linux busybox uname -a
```



The screenshot shows a Windows PowerShell window titled "Administrator: Windows PowerShell". The command entered is `docker run --platform linux busybox uname -a`. The output is `Linux afd52d23dd01 4.14.35-linuxkit #1 SMP Mon Apr 23 11:11:13 UTC 2018 x86_64 GNU/Linux`. The prompt `PS C:\>` is visible before and after the command.

FIGURE 10-21 A Linux container running on Windows Server

You only need to use the `--platform` option the first time you run a container that needs to be retrieved from a repository.

## Container orchestration

Container Orchestration allows you to perform the following tasks with containers:

- **Scheduling.** Allows container instances to be started on specific container hosts that participate in the orchestration based on the container host load.
- **Affinity.** Allows you to ensure that container instances that need to be near each other are kept in proximity. Can also be configured using anti-affinity to ensure that container instances that need to be highly available are kept on separate container hosts.
- **Health monitoring.** Determines which containers may have become unresponsive and replaces them with functional container instances.
- **Failover.** Tracks the container hosts. If a container host fails, the container instances that were running on that host are started on another host.
- **Scaling.** Allows new instances of containers to be started or stopped to meet the demand on the application. As load increases, the orchestrator starts containers. As load decreases, the orchestrator stops containers.

- **Networking.** Creates a special overlay network that allows containers to communicate when hosted on separate container hosts.
- **Service discovery.** Allows containers to track other containers as those containers are shifted between container hosts and as they change IP addresses.
- **Application upgrades.** Orchestration can ensure that containers are upgraded in such a way that application down time is minimized. Orchestration can also ensure that roll-back occurs in the event that an upgrade causes problems.

Docker on Windows Server supports two forms of orchestration: Kubernetes and Docker Swarm.

## More Info

### *Container orchestration*

You can learn more about container orchestration at: <https://docs.microsoft.com/en-us/virtualization/windowscontainers/about/overview-container-orchestrators>.

## Kubernetes

A Windows Server container host can participate in a Linux-based Kubernetes orchestration cluster. This allows Windows Server–based container images to take advantage of Kubernetes orchestration. In Windows Server 2019, a Kubernetes cluster must be managed by a Linux server, and you cannot deploy a Windows Server–only Kubernetes cluster. Both Linux worker container hosts and Windows Server container hosts can participate in the same Kubernetes cluster.

## More Info

### *Kubernetes and Windows*

You can learn more about Kubernetes and Windows at <https://docs.microsoft.com/en-us/virtualization/windowscontainers/kubernetes/getting-started-kubernetes-windows>.

## Docker Swarm

Docker Swarm mode provides container orchestration. This includes native clustering of container hosts and scheduling of container workloads. You can configure a collection of container hosts to form a swarm cluster. To do this, you need to configure the Docker Engine on the container hosts to run together in Swarm mode.

Swarms include two types of container hosts:

- **Manager nodes.** Swarms are initialized from Manager nodes. All Docker commands for the control and management of the swarm must be run from a Manager node. A swarm must have at least one Manager node, but it's possible for a swarm to have multiple Manager nodes. Manager nodes ensure that the state of the swarm matches the intended state. They keep track of services running on the swarm.
- **Worker nodes.** Manager nodes assign Worker nodes tasks that the Worker nodes execute. Worker nodes use a *join* token, which are generated during swarm initialization to join a swarm.

Swarm mode has the following requirements:

- Docker Engine v1.13.0 or later
- TCP port 2377 for cluster management communication open on each node
- TCP and UDP port 7946 for intra-node communication
- TCP and UDP port 4789 for overlay network traffic

## Creating swarm clusters

Run the following command on the container host that will function as a Manager node and specify the IP address of the container host:

```
Docker swarm init --advertise-addr <container_host_ip> --listen-addr
<container_host_ip>:2377
```

The container host will now function as the first Manager node in a new Docker swarm. Running the *Docker swarm init* command will also generate a *join* token that you will use to add other nodes to the swarm. You can retrieve the Worker node *join* token at any point in the future from the Manager node by running the following command:

```
Docker swarm join-token worker -q
```

Joining additional Manager nodes requires the Manager node *join* token. You can retrieve the Manager node *join* token at any point by running the following command on the Manager node.

```
Docker swarm join-token manager -q
```

You can add container hosts to the swarm as Worker nodes by running the following command:

```
Docker swarm join --token <worker-join-token> <manager_container_host_ip>
```

You can add container hosts as additional Manager nodes to the swarm by running the following command:

```
Docker swarm join --token <manager-join-token> <manager_container_host_ip>
```

You can view which nodes are members of a swarm by running the command:

```
Docker node ls
```

## Creating overlay networks

Overlay mode allows container endpoints to be connected across multiple container hosts. In a multi-node environment, VXLAN (Virtual Extensible LAN) encapsulation occurs in VFP (Virtual Filtering Platform) forwarding extensions included in the Hyper-V virtual switch. Inter-host communications directly reference IP endpoints. In single-node environments, intra-host communication occurs through a bridged connection on the Hyper-V virtual switch.

You can create an overlay network by running the following command from a Manager node, where 10.0.10.0/24 defines the overlay network address space:

```
Docker network create --driver=overlay --subnet=10.0.10.0/24 <OverlayNetworkName>
```

## Deploying and scaling swarm services

A swarm service consists of a specific container image that you want to run in multiple instances across nodes in the swarm cluster. Docker Swarm allows you to scale the service as required. For example, if you had a container image that allowed you to deploy a front-end web server for a multi-tier application, you can deploy that container image as a swarm service. As requirements dictate, you can increase the number of container instances by scaling the swarm service. Traffic can be automatically load balanced across the swarm service using DNS Round-Robin. This is substantially simpler than starting new container instances across separate container hosts and manually configuring load balancing to take account of the new container instances.

To create a swarm service, you use the *Docker service create* command to specify the service name, network name, and container image name, and to set the Endpoint mode to DNS Round-Robin. For example, to create a service named *SWARMPROXY* using the *WEBFRONTEND* container image and the *OverNet* overlay network, issue this command:

```
Docker service create --name=SWARMPROXY --endpoint-mode=dnsrr --network=OverNet
WEBFRONTEND
```

Once the service is created, you can scale it using the *Docker service scale* command and specifying the number of container instances. For example, to scale the *SWARMPROXY* service to 10 instances, run this command:

```
Docker service scale SWARMPROXY=10
```

You can have more than one service present on a Docker Swarm. You can view which services are present by running the following command on a Manager node:

```
Docker service ls
```

You can view which container instances make up a service, including which swarm nodes individual containers are running on, by running the following command:

```
Docker service ps <servicename>
```

## More Info

### *Swarm Mode*

You can learn more about Swarm mode at <https://docs.microsoft.com/en-us/virtualization/windowscontainers/manage-containers/swarm-mode>.