

The ultimate in-depth reference

Hundreds of timesaving solutions

Supremely well-organized, packed
with expert advice



SQL Server 2022 Administration Inside **OUT**

**Randolph West • William Assaf • Elizabeth Noble
Meagan Longoria • Joey D'Antoni • Louis Davidson**

With contributions from: William Carter, Josh Smith, Melody Zacharias

FREE SAMPLE CHAPTER |



SQL Server 2022 Administration Inside Out

Randolph West
William Assaf
Elizabeth Noble
Meagan Longoria
Joey D'Antoni
Louis Davidson

With contributions from:
William Carter
Josh Smith
Melody Zacharias

SQL Server 2022 Administration Inside Out

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

Copyright © 2023 by Pearson Education.

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.

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: 978-0-13-789988-3

ISBN-10: 0-13-789988-2

Library of Congress Control Number: 2023930547

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 or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearson.com.

Editor-in-Chief: Brett Bartow

Executive Editor: Loretta Yates

Associate Editor: Charvi Arora

Development Editor: Kate Shoup

Technical Reviewers: William Carter, Louis Davidson, Meagan Longoria, Elizabeth Noble, Josh Smith

Managing Editor: Sandra Schroeder

Senior Project Editor: Tracey Croom

Copy Editor: Rebecca Rider

Indexer: Rachel Kuhn

Proofreaders: Audrey Doyle, James Fraleigh

Editorial Assistant: Cindy Teeters

Cover Designer: Twist Creative, Seattle

Compositor: Jeff Lytle, Happenstance Type-O-Rama

Graphics: Jeff Wilson, Happenstance Type-O-Rama

Pearson's Commitment to Diversity, Equity, and Inclusion

Pearson is dedicated to creating bias-free content that reflects the diversity of all learners. We embrace the many dimensions of diversity, including but not limited to race, ethnicity, gender, socioeconomic status, ability, age, sexual orientation, and religious or political beliefs.

Education is a powerful force for equity and change in our world. It has the potential to deliver opportunities that improve lives and enable economic mobility. As we work with authors to create content for every product and service, we acknowledge our responsibility to demonstrate inclusivity and incorporate diverse scholarship so that everyone can achieve their potential through learning. As the world's leading learning company, we have a duty to help drive change and live up to our purpose to help more people create a better life for themselves and to create a better world.

Our ambition is to purposefully contribute to a world where:

- Everyone has an equitable and lifelong opportunity to succeed through learning.
- Our educational products and services are inclusive and represent the rich diversity of learners.
- Our educational content accurately reflects the histories and experiences of the learners we serve.
- Our educational content prompts deeper discussions with learners and motivates them to expand their own learning (and worldview).

While we work hard to present unbiased content, we want to hear from you about any concerns or needs with this Pearson product so that we can investigate and address them.

- Please contact us with concerns about any potential bias at <https://www.pearson.com/report-bias.html>.

Dedications

To Marinus (as always), to the friends we've lost along the way, and to the friends we make by saying "yes" to new opportunities.

—Randolph West

The previous book in this series was published in March 2020, at the advent of a historic disruption: social, political, economic, and educational. I dedicate this book to all those who have been made unsafe and further marginalized by the COVID-19 pandemic.

—William Assaf

To Mind, Danny, Eve, Mom, and Dad, for your encouragement and support.

—Elizabeth Noble

I dedicate this book to my dad, who passed away in 2022. He was always supportive of my educational and professional goals.

—Meagan Longoria

I would like to thank my other authors and my family for helping this work go so smoothly.

—Joey D'Antoni

To all my friends at CBN who provided me with training, inspiration, practice, and until recently, servers to test code on for so many years.

—Louis Davidson

To all those who need someone to believe in them. Know that there are two, Me and You, because as Henry Ford said, "Whether you think you can or you can't, you're right!"

—Melody Zacharias

In memory of Ruby Jean Carter. You were the embodiment of love, patience, and strength.

—William Carter

For data professionals everywhere always looking to learn and grow.

—Josh Smith



Contents at a glance

Foreword	xxix	Chapter 10	
Introduction	xxxi	Develop, deploy, and manage data recovery	445
Part I: Introduction		Chapter 11	
Chapter 1		Implement high availability and disaster recovery	487
Get started with SQL Server tools	1	Part IV: Security	
Chapter 2		Chapter 12	
Introduction to database server components	55	Administer instance and database security and permissions	549
Chapter 3		Chapter 13	
Design and implement an on-premises database infrastructure	93	Protect data through classification, encryption, and auditing	617
Part II: Deployment		Part V: Performance	
Chapter 4		Chapter 14	
Install and configure SQL Server instances and features	129	Performance tune SQL Server	669
Chapter 5		Chapter 15	
Install and configure SQL Server on Linux	195	Understand and design indexes	753
Chapter 6		Part VI: Cloud	
Provision and configure SQL Server databases	215	Chapter 16	
Chapter 7		Design and implement hybrid and Azure database infrastructure	789
Understand table features	249	Chapter 17	
Part III: SQL Server management		Provision Azure SQL Database	829
Chapter 8		Chapter 18	
Maintain and monitor SQL Server	325	Provision Azure SQL Managed Instance	869
Chapter 9		Chapter 19	
Automate SQL Server administration	393	Migrate to SQL Server solutions in Azure	915
		Index	939



Table of Contents

About the Authors.....	.xxi
Acknowledgments.....	xxv
Foreword.....	xxix
Introduction.....	xxxii
Who this book is for.....	.xxxii
How this book is organized.....	.xxxii
Conventions.....	xxxiv
Text conventions.....	xxxiv
Book features.....	xxxv
Errata, updates, and book support.....	xxxv

Part I: Introduction

Chapter 1	Get started with SQL Server tools.....	1
	SQL Server setup.....	1
	Install SQL Server with the Installation Center.....	2
	Plan before an upgrade or installation.....	3
	Install or upgrade SQL Server.....	6
	Tools and services installed with the Database Engine.....	7
	Machine Learning Services.....	7
	Data Quality Services.....	8
	Command line interface.....	9
	SQL Server Configuration Manager.....	13
	Performance and reliability monitoring tools.....	14
	Database Engine Tuning Advisor.....	14
	Extended Events.....	14
	Management Data Warehouse.....	16
	SQL Server Reporting Services (SSRS).....	16
	Installation.....	16
	Report Server Configuration Manager.....	18
	SQL Server Management Studio (SSMS).....	18
	Releases and versions.....	19
	Install SQL Server Management Studio.....	19
	Upgrade SQL Server Management Studio.....	20
	Features of SQL Server Management Studio.....	20

- Additional tools in SQL Server Management Studio 27
- Error logs 31
- Activity Monitor 32
- SQL Server Agent 37
- Azure Data Studio 40
 - User interface 41
 - Highlighted features in Azure Data Studio 43
 - Notebooks in Azure Data Studio 47
- SQL Server Data Tools 48
 - SQL Server Integration Services 48
- SQL Server on Azure Arc-enabled servers 51
- Microsoft Purview 52
- Discontinued and deprecated features 53

Chapter 2 Introduction to database server components. 55

- Memory 56
 - Understand the working set 56
 - Cache data in the buffer pool 56
 - Cached plans in the procedure cache 57
 - Lock pages in memory 58
 - Editions and memory limits 59
- Central processing unit 59
 - Simultaneous multithreading 60
 - Non-uniform memory access 61
 - Disable power saving everywhere 63
- Data storage 63
 - Types of storage 64
 - Configure the storage layer 65
- Connect to SQL Server over the network 70
 - Protocols and ports 71
 - Added complexity with Virtual Local Area Networks 71
- High-availability concepts 72
 - Why redundancy matters 73
 - Disaster recovery 74
 - Clustering 74
 - The versatility of log shipping 77
 - Always On availability groups 78
- Secure SQL Server 82
 - Integrated Authentication and Active Directory 82
 - Azure Active Directory 85
 - Kerberos for Azure SQL Managed Instance 87
- Understand virtualization and containers 87
 - Going virtual 88
 - Provision resources for virtual consumers 89
 - When processors are no longer processors 90
 - The network is virtual, too 92

Chapter 3	Design and implement an on-premises database infrastructure	93
	Introduction to SQL Server database architecture	93
	Data files and filegroups	94
	Group data pages with extents	95
	Contents and types of data pages	96
	Verify data pages by using a checksum	98
	Record changes in the transaction log	99
	Flush data to the storage subsystem with checkpoints	100
	Inside the transaction log file	100
	The Minimum Recovery LSN	103
	Types of database checkpoints	103
	Restart with recovery	105
	MinLSN and the active log	106
	A faster recovery with accelerated database recovery	107
	Partition tables	108
	Compress data	109
	Table and index compression	109
	Backup compression	112
	Manage the temporary database	113
	Storage options for tempdb	113
	Recommended number of files	114
	Configuration settings	115
	Manage system usage with Resource Governor	115
	Configure the operating system page file	116
	Take advantage of logical processors with parallelism	116
	SQL Server memory settings	119
	Allocate CPU cores with an affinity mask	122
	File system configuration	124
Part II: Deployment		
Chapter 4	Install and configure SQL Server instances and features	129
	What to do before installing SQL Server	130
	Decide on volume usage	131
	Important SQL Server volume settings	134
	SQL Server editions	135
	Install a new instance	137
	Plan for multiple SQL Server instances	138
	Install SQL Server on Windows	139
	Install common features	147
	Log SQL Server Setup	158
	Automate SQL Server Setup with configuration files	158
	SQL Server on Azure virtual machines	163
	Post-installation server configuration	163
	Post-installation checklist	163
	Post-installation configuration of other features	176
	SSISDB initial configuration and setup	176
	SQL Server Reporting Services initial configuration and setup	177

	SQL Server Analysis Services initial configuration and setup	180
	Azure Synapse Link for SQL Server	181
	Container orchestration with Kubernetes	182
	Kubernetes support for SQL Server	184
	Deploy SQL Server in containers	185
	Get started with SQL Server on Kubernetes	188
	Deploy SQL Server on Kubernetes	189
	Review cluster health	193
Chapter 5	Install and configure SQL Server on Linux	195
	What is Linux?	195
	Differences between Windows and Linux	196
	Linux distributions supported by SQL Server	199
	Considerations for installing SQL Server on Linux	200
	Configure OS settings	200
	Install SQL Server on Linux	203
	Installation requirements	204
	Download and install packages	204
	Configure SQL Server on Linux	206
	Use mssql-conf to set up and configure SQL Server	207
	Caveats of SQL Server on Linux	212
	Missing SQL Server features on Linux	212
Chapter 6	Provision and configure SQL Server databases	215
	Add databases to a SQL Server instance	215
	Create a database	216
	Move existing databases	220
	Upgrade database compatibility levels	222
	Other considerations for migrating databases	225
	Database-scoped configurations	229
	Database properties and options	230
	Move and remove databases	241
	Move user and system databases	241
	Move databases within instances	242
	Single-user mode	247
Chapter 7	Understand table features	249
	Review table structures	249
	General-purpose data types	250
	Specialized data types	258
	Data type precedence	266
	Constraints	266
	Sequence objects	270
	User-defined data types and user-defined types	273
	Sparse columns	274
	Computed columns	275
	Special table types	276
	System-versioned temporal tables	277
	Memory-optimized tables	282

Graph tables	287
Store large binary objects	292
Understand FILESTREAM	293
FileTable	295
Table partitions	295
Horizontally partitioned tables and indexes	296
Vertical partitions	302
Capture modifications to data	303
Use change tracking	303
Use change data capture	305
Query change tracking and change data capture	307
Compare change tracking, change data capture, and temporal tables	308
Benefits of PolyBase for external data sources and external tables	309
Unified data platform features	309
Install and configure PolyBase	311
More PolyBase examples, architectures including S3 and URL queries	320
PolyBase examples with a generic ODBC driver	322
Azure bulk operations examples	323

Part III: SQL Server management

Chapter 8	Maintain and monitor SQL Server	325
	Detect, prevent, and respond to database corruption	325
	Set the database's page verify option	326
	Repair database data file corruption	329
	Recover from database transaction log file corruption	329
	Database corruption in Azure SQL Database	330
	Maintain indexes and statistics	330
	Change the fill factor when beneficial	331
	Monitor index fragmentation	333
	Maintain indexes	334
	Manage database file sizes	342
	Understand and find autogrowth events	344
	Shrink database files	345
	Monitor activity with DMOs	347
	Observe sessions and requests	347
	Understand wait types and wait statistics	349
	Monitor with the SQL Assessment API	358
	Use Extended Events	361
	View Extended Events data	363
	Use Extended Events to capture deadlocks	367
	Use Extended Events to detect autogrowth events	369
	Use Extended Events to detect page splits	369
	Secure Extended Events	370

- Capture performance metrics with DMOs and data collectors 371
 - Query performance metrics with DMVs 371
 - Capture performance metrics with Performance Monitor 374
 - Monitor key performance metrics 375
 - Monitor key performance metrics in Linux 379
 - Monitor key performance metrics in Azure portal 380
- Protect important workloads with Resource Governor 384
 - Configure the Resource Governor classifier function 386
 - Configure Resource Governor resource pools and workload groups 387
 - Monitor resource pools and workload groups 388
- Understand the SQL Server servicing model 389
 - Updated servicing model 389
 - Plan for the product support life cycle 390

Chapter 9 Automate SQL Server administration 393

- Foundations of SQL Server automated administration 394
 - Database Mail 394
 - SQL Server Agent 400
- Maintain SQL Server 412
 - Basic care and feeding of SQL Server 412
- Use SQL Server maintenance plans 414
 - Cover databases with the maintenance plan 415
 - Maintenance plan tasks 416
 - Maintenance plan report options 423
 - Build maintenance plans using the Maintenance Plan designer in SSMS 424
 - Back up availability groups using a secondary replica 426
- Strategies for administering multiple SQL Servers 428
 - Master/Target servers for SQL Agent jobs 428
 - SQL Server Agent event forwarding 431
 - Policy-based management 431
- Use PowerShell to automate SQL Server administration 434
 - PowerShell basics 436
 - Install the PowerShell SQLServer module 436
 - Use PowerShell with SQL Server 438
 - Use PowerShell with availability groups 442

Chapter 10 Develop, deploy, and manage data recovery 445

- Prepare for data recovery 446
 - A disaster recovery scenario 447
 - Define acceptable data loss: RPO 449
 - Define acceptable downtime: RTO 450
 - Establish and use a runbook 450
- Ransomware attacks 451
- Understand different types of backups 453
 - An overview of SQL Server recovery models 454
 - Full backups 456
 - Differential backups 458

The backup chain	459
File and filegroup backups	461
Additional backup options and considerations	462
Understand backup devices	467
Back up to disk	468
Back up to URL	468
Backup and media sets	469
Back up to S3-compatible storage	472
Create and verify backups	472
Create backups	473
Verify backups	474
Restore a database	475
Restore a database using a full backup	476
Restore a database with differential and log backups	477
Restore a database to a point in time	478
Restore a database piecemeal	480
Define a recovery strategy	481
A sample recovery strategy for our DR scenario	482
Recovery strategies for hybrid and cloud environments	484
Chapter 11 Implement high availability and disaster recovery	487
Overview of high-availability and disaster-recovery technologies	488
Compare HA and DR technologies	489
Understand log shipping	489
Understand the capabilities of failover clustering	492
Understand the capabilities of availability groups	494
Configure failover cluster instances	496
Understand FCI quorum	497
Configure a SQL Server FCI	499
Patch a failover cluster	502
Design availability groups solutions	502
Compare different cluster types	504
Create WSFC for use with availability groups	507
Understand the database mirroring endpoint	509
Recent improvements to availability groups	509
Choose the correct secondary replica availability mode	512
Understand the impact of secondary replicas on performance	513
Understand failovers in availability groups	515
Seeding options when adding replicas	518
Additional actions after creating an availability group	522
Read secondary database copies	524
Query Store on replicas	531
Implement a hybrid availability group topology	531
Understand the Azure SQL Managed Instance link feature	532
Failover and failback to Azure SQL Managed Instance with database portability	534
Provision and scale the Azure SQL Managed Instance link feature	534
Failover and failback tooling and automation	537

Configure availability groups in SQL Server on Linux	537
Understand the differences between Windows and Linux clustering.	537
Set up an availability group in SQL Server on Linux	538
Administer availability groups	543
Analyze DMVs for availability groups	544
Analyze wait types for availability groups	546
Analyze Extended Events for availability groups	547
Alerts for availability groups.	548

Part IV: Security

Chapter 12 Administer instance and database security and permissions 549

Understand authentication modes	549
Windows Authentication	550
SQL Server Authentication	551
Azure Active Directory	551
Advanced types of server principals	553
Authentication to SQL Server on Linux	554
Contained database authentication.	555
Grasp security principals	555
The basics of privileges	557
Configure login server principals	559
Database principals.	582
Understand permissions and authorization	593
Permissions for controlling Data Definition Language and Data Manipulation Language	593
How permissions accumulate.	595
Understand authorization.	597
Perform common security administration tasks	605
Orphaned SIDs	605
Create login with known SID	608
Migrate SQL Server logins and permissions	608
Dedicated administrator connection.	614

Chapter 13 Protect data through classification, encryption, and auditing. 617

Privacy in the modern era	617
General Data Protection Regulation (GDPR)	618
Microsoft Purview overview.	619
Introduction to security principles and protocols.	620
Secure your environment with defense in depth.	621
The difference between hashing and encryption	623
A primer on protocols and transmitting data	625
Digital certificates	630
Protect the data platform	631
Secure the network with TLS	632
Data protection from the OS	633
The encryption hierarchy in detail	634

Use EKM modules with SQL Server	635
Master keys in the encryption hierarchy	637
Encrypt data with TDE	639
Protect sensitive columns with Always Encrypted	642
Row-level security	648
Dynamic data masking	650
Protect Azure SQL Database with Microsoft Defender for SQL	651
Ledger overview	653
Immutable storage	653
Ledger verification	653
Ledger considerations and limitations	654
Data storage requirements	654
Types of ledger tables	655
Audit with SQL Server and Azure SQL Database	657
SQL Server Audit	657
Auditing with Azure SQL	663
Secure Azure infrastructure as a service	664
Network security groups	664
User-defined routes and IP forwarding	666
Additional Azure networking security features	667

Part V: Performance

Chapter 14 Performance tune SQL Server	669
Understand isolation levels and concurrency	671
Understand how concurrent sessions become blocked	676
Change the isolation level	679
Understand and handle common concurrency scenarios	681
Understand row version-based concurrency	692
Understand on-disk versus memory-optimized concurrency	700
Understand durability settings for performance	702
Delayed durability database options	703
How SQL Server executes a query	705
Understand the query execution process	706
View execution plans	708
Understand execution plans	713
Understand parameterization and parameter sniffing	725
Explore the procedure cache	728
Understand parallelism	733
Use advanced engine features to tune queries	735
Internal improvements in SQL Server 2022	736
Recent improvements to tempdb	736
Leverage the Query Store feature	737
Query Store hints	742
Automatic plan correction	745
Intelligent query processing	746

Chapter 15	Understand and design indexes	753
	Design clustered indexes	754
	Choose a proper rowstore clustered index key	754
	The case against intentionally designing heaps	758
	Understand the OPTIMIZE_FOR_SEQUENTIAL_KEY feature	759
	Design rowstore nonclustered indexes	760
	Understand nonclustered index design	761
	Create filtered nonclustered indexes	767
	Understand the missing indexes feature	767
	Understand and provide index usage	771
	Understand columnstore indexes	773
	Design columnstore indexes	774
	Understand batch mode	776
	Understand the deltastore of columnstore indexes	777
	Demonstrate the power of columnstore indexes	778
	Understand indexes in memory-optimized tables	780
	Understand hash indexes for memory-optimized tables	781
	Understand nonclustered indexes for memory-optimized tables	782
	Understand index statistics	783
	Automatically create and update statistics	783
	Manually create statistics for on-disk tables	784
	Understand statistics on memory-optimized tables	785
	Understand statistics on external tables	785
	Understand other types of indexes	786
	Understand full-text indexes	786
	Understand spatial indexes	786
	Understand XML indexes	787

Part VI: Cloud

Chapter 16	Design and implement hybrid and Azure database infrastructure	789
	Cloud computing and Microsoft Azure	789
	Database as a service	790
	Managing Azure with the Azure portal and PowerShell 7	791
	Azure governance	792
	Cloud-first	794
	Resource scalability	794
	Networking in Azure	795
	Cloud models and SQL Server	796
	Infrastructure as a service	797
	Platform as a service	805
	Hybrid cloud with Azure	821
	Cloud security	826
	Other data services in Azure	826
	Azure Synapse Analytics	827
	Non-relational Azure data offerings	827
	Third-party fully managed data platforms	828

Chapter 17	Provision Azure SQL Database	829
	Provision an Azure SQL Database logical server	830
	Create an Azure SQL Database server using the Azure portal	832
	Create a server using PowerShell	833
	Establish a connection to your server	834
	Delete a server	836
	Provision a database in Azure SQL Database	836
	Create a database using the Azure portal	837
	Create a database using PowerShell	839
	Create a database using Azure CLI	840
	Create a database using T-SQL	841
	Scale up or down	842
	Provision a named replica for a Hyperscale database	843
	Provision an elastic pool	844
	Manage database space	845
	Security in Azure SQL Database	846
	Security features shared with SQL Server 2022	846
	Server- and database-level firewall	846
	Integrate with virtual networks	849
	Azure Private Link for Azure SQL Database	850
	Control access using Azure AD	850
	Use Azure role-based access control	853
	Audit database activity	854
	Microsoft Defender for SQL	859
	Prepare Azure SQL Database for disaster recovery	861
	Understand default disaster recovery features	861
	Manually export database contents	862
	Enable zone-redundant configuration	863
	Configure geo-replication	863
	Set up failover groups	865
	Use Azure Backup for long-term backup retention	867
Chapter 18	Provision Azure SQL Managed Instance	869
	What is Azure SQL Managed Instance?	872
	Differences between SQL Server and Azure SQL Managed Instance	873
	Create a SQL managed instance	883
	Select a service tier and service objective	884
	Use the Azure portal to provision a SQL managed instance	886
	Use PowerShell to provision a SQL managed instance	891
	Delete a SQL managed instance	892
	Establish a connection to a SQL managed instance	893
	Create the endpoints via the Azure portal	894
	Create a VPN gateway via PowerShell	894
	Network requirements for SQL managed instances	898
	Migrate data to Azure SQL Managed Instance	900
	Link feature for Azure SQL Managed Instance	900
	Azure Data Migration Service	901

Migrate with backup and restore	901
Managed instance pools	902
Azure SQL Managed Instance administration features	903
High availability	903
Replication	905
Scale up or down	906
Monitor SQL managed instances	906
Link feature for Azure SQL Managed Instance	907
Azure SQL Managed Instance security features	908
Azure Active Directory	908
Azure SQL Managed Instance data protection features	911
Prevent data exfiltration	911
Isolation	912
Auditing	912
Data encryption	912
Row-level security	913
Dynamic data masking	913
Chapter 19 Migrate to SQL Server solutions in Azure	915
Migration services options	915
Microsoft Assessment Planning toolkit	916
Total Cost of Ownership calculator	917
Database Experimentation Assistant	918
Azure Data Migration Assistant	919
Azure Database Migration Service	921
SQL Server Migration Assistant	925
Data Access Migration Toolkit	926
Resolve common migration failures using Database Migration Service	926
Large object columns with data larger than 32 KB	928
Final notes for migration	928
Open source PowerShell migration with dbatools	929
Migrate with Azure Data Factory	933
Azure integration runtime	933
Self-hosted integration runtime	935
Self-hosted IR servers and nodes	935
Azure-SSIS integration runtime	936
Best practices for security and resilience during migration	937
Network security	937
Cloud requirements for application resilience	938
Index	939

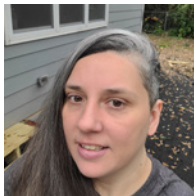
About the Authors



Randolph West (*they/them*) lives in Calgary, Alberta, Canada, with a husband and two dogs. After being a consultant for millennia, Randolph now writes full-time at Microsoft Docs, still yelling at the screen. Occasional voice actor. Occasional blogger at *bornsql.ca*. Not to be trusted around chocolate. Yes, this is a short bio.



William Assaf (*he/him*) is a senior content developer for Microsoft, writing Learn content for SQL Server, Azure SQL Database, Azure Synapse Analytics, and more. A long-time Baton Rougean, William and his adventure buddy Christine moved to Seattle during the pandemic. They love their new home but are still New Orleans Saints fans. Before joining Microsoft, William was a Data Platform MVP, SQL Saturday and SQL community organizer, and a long-time DBA and data consultant. As a consultant for 13 years, he worked with clients across the U.S. on SQL Server and Azure SQL platform optimization, management, data integration, disaster recovery, and high availability, and led a multi-city team of senior consulting SQL DBAs. William has written for Microsoft SQL certification exams since 2011 and was the team lead author of the 2017 and 2019 editions of *SQL Server Administration Inside Out* by Microsoft Press.



Elizabeth Noble is a Director of Database Development, the author of *Pro T-SQL 2019*, and a Microsoft Data Platform MVP. Ze has spoken at several SQL Saturdays across the United States and at PASS Summit. Most of zir topics focus on DevOps, collaboration with other IT departments, and automated database deployments. Zir passion is to help others improve the quality and speed of deploying database changes through automation. When ze is not trying to automate all things, ze can be found spending time with zir dogs, playing disc golf, or paddleboarding (if the weather is right).



Meagan Longoria is a Microsoft Data Platform MVP living in Denver, Colorado. She is an experienced consultant and trainer who has worked with the Microsoft Data Platform for over 15 years. She enjoys creating solutions in Azure, SQL Server, and Power BI that make data useful for decision makers and make the lives of information workers a little bit easier. Meagan enjoys sharing her knowledge with the technical community by speaking at conferences, blogging (*DataSavvy.me*), and sharing tips and helpful links on Twitter (*@mmarie*).



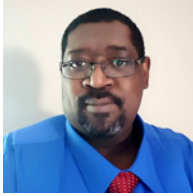
Joseph D'Antoni is a Principal Consultant at Denny Cherry & Associates Consulting. He is recognized as a VMWare vExpert and a Microsoft Data Platform MVP, and has over 20 years of experience working in both Fortune 500 and smaller firms. He has worked extensively on database platforms and cloud technologies and has specific expertise in performance tuning, infrastructure, and disaster recovery.



Louis Davidson has over 20 years as a data architect and technical writer. Recently he joined Redgate as the editor of the Simple Talk website after 20-plus years working for a nonprofit, where he was the lead SQL Server architect and programmer. Louis has been the principal author on many technical books about SQL Server, including six editions of a book on database design. Louis' blog, located at *simple-talk.com* for many years, provides information about technical issues and upcoming presentations, including previewing the thought process that goes into writing presentations, books, and blogs.



Melody Zacharias is a Microsoft MVP for the data platform and Microsoft Regional Director. She has co-written several books on data, including *SQL Server 2019 Administration Inside Out* by Microsoft Press. She speaks at conferences on data, technology, women in Tech, professional development, and more. You can find her on her blog at *sqlmelody.com*, on Twitter *@SQLMelody*, and as */melodyzacharias* on LinkedIn.



William F. Carter (*he/him*) is a technologist and Microsoft SQL Server consultant with 25 years of experience dating back to SQL Server 6.5. Bill is passionate about helping individuals and organizations use technology and data to drive change and bring about successful outcomes. When not managing data and architecting solutions, he loves to model and 3D print props for the local high school's theater department. You can connect with him on LinkedIn at www.linkedin.com/in/william-f-carter/.



Josh Smith has held several titles over the last 20 years, including Stage Manager, Art Director, Teacher, Case Manager, and—for the last 10 years—Database Administrator. They currently infrequently write at accitention-aldba.com and post more often but with much less focus on Twitter as [@sqldeployhelmet](https://twitter.com/sqldeployhelmet). They are team pineapple on pizza, have a completely reasonable fear of spiders, and are the current president of the Inland Northwest Data Professionals Association in Spokane, Washington.

Acknowledgments

Randolph West

It may take a village to raise a child, but it takes a small country to write a book. Five books in, and I still don't understand how it comes together at the end. Thank you to Loretta and Charvi at Microsoft Press, and to William Assaf, for long and thoughtful conversations. My co-authors and technical editors, obviously. To Marinus, thank you for begrudgingly letting me do another book even though I still don't sit at the desk you bought me. Thanks to Trixie for the slower walks, our new puppy Tilley for keeping me on my toes, and Apple for making a quiet laptop.

A lot of us wouldn't be here without the tireless efforts of medical professionals during the global pandemic. Join me in thanking your healthcare friends when you get a chance. Thank your teachers. Thank your first responders. Thank the people who keep the lights on and the water flowing. Hug a queer person.

I would like to extend a special acknowledgment to Melody Zacharias, who has been a major contributor to the Microsoft Data Platform community for a number of years. Melody selflessly introduced me to the community in Canada and even included me as one of the authors in her *Let Them Finish* book. She has helped so many people in our community and also deserves a special mention in the production of this book.

William Assaf

Becoming an empty nester has allowed me to spend more time than ever with my best friend and adventure buddy. Thanks, Christine, for tolerating endless nights of writing, rewriting, and editing this book instead of hiking, exploring, or snuggling.

I'd like to thank Loretta Yates, our intrepid and tactful editor throughout our Microsoft Press experience. I'd also like to thank the mentors and managers and colleagues in my professional career heretofore, who affected my trajectory, and to whom I remain grateful for technical and nontechnical lessons learned. I'd like to thank Connie Murla, David Alexander, Darren Schumaker, Ashagre Bishaw, Charles Sanders, Todd Howard, Chris Kimmel, Richard Caronna, Mike Huguet, Mike Carter, Jason Prell, James Sampson, Jason Roth, and finally Patrick Leblanc, a fellow Baton Rouge native, whose friendship has repeatedly challenged me and furthered my career. I'd also like to thank my father, a rare stamped mechanical and electrical engineer (and a HAM), and both my older brothers who are brilliant software engineers, for letting me play games on their 386s all summer long. I'd finally like to thank the STEM educators, nonprofit volunteers, and organizers in my hometown of Baton Rouge, Louisiana. They are doing the hard work of developing our future coworkers and coauthors among my home state's perpetually underfunded, underappreciated, and underestimated public school youth.

Elizabeth Noble

I would like to thank Randolph West and William Assaf for inviting me to collaborate on this book. I also want to thank the entire team that made this book possible. You all were kind, supportive, and encouraging. I'd also like to thank my many mentors (mostly unofficial) including Phil Pledger, Mike Lawell, Ed Watson, and Rob Volk. You each have provided encouragement and guidance over the years. Rie Merritt, thank you for welcoming me to my second user group meeting. If it weren't for you, I would not have come back to my third meeting or have met this wonderful community. I want to thank my family for giving me the time and space to work on this book. To Mom, thank you for being my cheerleader. To Dad, thank you for nudging me every so often to see how the book was going. Also, in memory of Khari, my forever companion, thank you for making sure that I remembered to take care of myself.

Meagan Longoria

First, I'd like to thank the co-authors and co-editors who collaborated on this book. It was a pleasure to work with them and learn from them. I would also like to thank my coworkers at Denny Cherry & Associates Consulting for their technical advice and support. I also want to acknowledge my laptop bag for safely carrying my computer through many states and a couple of countries while I worked on this book. I think everyone should consider a nice laptop bag. Finally, I'd like to thank my dog Izzy for being understanding when dinner was a little late due to writing or editing, and for reminding me to take breaks to go on walks. Life is better with a dog.

Joey D'Antoni

I would like to thank my wife Kelly, and my coworkers at Denny Cherry & Associates Consulting, for helping me and giving me time to work on this project. Also, thanks to the team at Microsoft for answering dumb questions when I had them.

Louis Davidson

I would like to acknowledge the rest of the team on this book for their wonderful work that makes tech editing the chapters I worked on some of the easiest technical book work I have ever participated in.

Melody Zacharias

I really want to thank William Assaf and Randolph West for their push/encouragement to do another book with them. It is always a pleasure to work with some of the best professionals in the industry. There have been a few who are special in the community who made this possible: Argenis Fernandez who introduced me to #SQLFamily, John Morehouse who mentored and encouraged me to do my first presentation, Dave Kawula for encouraging me to write my

first book, and Rie Merritt for keeping me going and inspired to inspire others year after year. I would not be doing this if not for your encouragement. Thank you all for each experience that changed my life for the better. No family is perfect, but my #SQLFamily is an amazingly supportive and inclusive family, and I am so proud to be a member. Thank you to Marsha Pierce and Rob Ludeman, for bringing me into the Pure Family, and for letting me share my crazy obsession with SQL Server with Pure. Most of all, thank you to my family for understanding when I go back into my office after dinner and on weekends to work on this book and all the presentations and other community work I do. Thank you for accepting me as I am.

William Carter

I want to thank my children, Kadence and Kayla, for inspiring me to continue to grow and embrace change. I'd like to thank my mom and dad and my personal village of family, friends, and teachers who instilled in me the passion and perseverance to pursue my dreams. Finally, I want to thank Joel Whittington, Fred Seals, and William Assaf, three of my mentors who I respect and admire a great deal. Each of you directly impacted my professional and personal growth, sharing your wisdom and humor along the way.

Josh Smith

I'd like to thank those in the SQL Server community who have taken the time to provide me with opportunities to learn and grow in my career, including the authors of this book for taking a chance and inviting me to join the other technical editors. I am eternally grateful for the patience of my family throughout the last 10 years, but over this past summer in particular, as we've attempted to do ALL THE THINGS at the same time. Sara, hopefully by the time I am showing you this in print, we've finished unpacking.

Foreword

The world of data is getting more complicated. As threats and technologies evolve, businesses may fall behind.

As a consultant before joining Microsoft, I walked into an industrial plant where the sysadmins did not know what SQL Server backup was. To them, a file system backup or virtual machine (VM) storage snapshot was enough. (It is not enough.) I worked with a small regional bank that thought a 3 TB+ SQL Server transaction log file was just the cost of doing business. I helped a healthcare broker that was being actively probed via SQL injection attacks. In each case, the seeds of business-crippling disaster were planted, waiting to sprout.

As a reader of this book, you likely know the basics of what is necessary to secure your data platform from disaster, whether it is malicious actors or natural disaster. (If not, we have you covered there too.)

As the latest team brought together to write this *Microsoft SQL Server Administration Inside Out* series book, we have always tried to present to you a complete, practical, field-tested picture of administration tasks, including wisdom and tips collected from our own experience as administrators and architects. Just as the authors of this book have brought a collective effort to guide and advise, it has never been more important to organize collective effort across the entire IT department.

In terms of the basics of cybersecurity, Microsoft's own Digital Defense Report makes it clear. The primary protections companies can take are to "patch systems regularly and keep software up to date, and to use MFA" (Microsoft Digital Defense Report, October 2021, <https://go.microsoft.com/fwlink/p/?LinkID=2173952>).

Patching is just the basics. What else?

- Led by Microsoft's Zero Trust model, enable the use of integrated, multifactor, password-less authentication wherever possible, including to our SQL Server instances and Azure SQL platforms.
- Use multifactor authentication (MFA) everywhere possible, including for access to your Active Directory–authenticated resources on desktops, mobile devices, and collaboration applications.
- Immediately replicate database backups to offsite, heterogenous systems to protect them from ransomware encryption attacks.
- Separate day-to-day accounts from administrative accounts to ensure that commonly entered user credentials don't have all the keys to the kingdom.
- Protect from software supply-chain attacks by using secure admin workstations (SAWs) and privileged administrative workstations (PAWs). (See "Protecting high-risk environments

with secure admin workstations,” May 2018, at <https://www.microsoft.com/insidetrack/protecting-high-risk-environments-with-secure-admin-workstations>.)

- Secure network connections by default, and in this new era of widespread fully remote work, consider the security of VPN and virtualized networking software and hardware. Microsoft’s Zero Trust model has led the company to move away internally from a ubiquitous, always-connected global VPN to mitigate the impact of a single compromised device.
- “Antimalware and detection and response technologies should be deployed across the ecosystem [...] from virtual machines and containers to machine learning (ML) algorithms, databases, and applications.” (Microsoft Digital Defense Report, October 2021.)

As data platform professionals, we need modern tools and skills to secure data from attack and recover data in case attacks are successful. Disaster recovery checklists around the world are not only being triggered by annual “100-year” flood events, but by cybercrime and malicious activity, such as ransomware attacks. The landscape of digital security affects the entire enterprise, and your database platform is the crown jewel for attackers.

When it comes to your Microsoft data platform, in the cloud, on-premises, hybrid, as a service, or otherwise, we want to provide you with the tools to administer with confidence. Chapter 10, “Develop, deploy, and manage data recovery;” dives deep into a disaster recovery scenario and the runbook every SQL Server administrator should have ready. New features of Microsoft SQL Server 2022 make it easier than ever to span a durable, secured data infrastructure across hybrid environments, including a new feature to sync bidirectionally between on-premises SQL Server instances and Azure SQL managed instances (see Chapter 18, “Provision Azure SQL Managed Instance”). Not only is what used to be a one-way ticket into Azure SQL Managed Instance now reversible, but failover and failback from SQL Server 2022 to Azure SQL Managed Instance is now possible.

In addition to covering all the new features coming with SQL Server 2022, this book contains more than four years of new features that roll out to Azure SQL platforms outside of the year-numbered SQL Server product. We’ll be sure to point out the new capabilities of Azure SQL Managed Instance, for example, that have arrived in the November 2022 feature wave in conjunction with SQL Server 2022. Many new features simplify our job of securing the data estate, such as the introduction of Windows Authentication to Azure SQL Managed Instance using Azure Active Directory and Kerberos, an important feature that arrived in August 2022.

Like Azure’s infrastructure approach, the authors of this book aim to make it easier for businesses to protect their technology platform and avoid preventable disasters. We want to make it *less likely* that you will leave your data estate vulnerable due to ignorance. We draw from 100+ years of combined experience across this author team, which consists of Microsoft employees, Data Platform MVPs, consultants, entrepreneurs, leaders, on-call DBAs, data architects, and day-to-day SQL Server administrators. We want to inform you of low-hanging fruit to be picked, and of practical, easy wins. We hope this approach gives you—and your data—more confidence and reassurance.

—William Assaf, *Database Docs*, Microsoft

Introduction

Who this book is for

Data platform administration was never the narrow niche skillset that employers or recruiters might have suspected. The job description continues to broaden, with support for new operating systems and platforms: cloud-based and serverless in addition to on-premises, hybrid environments, even on-premises to cloud failover. We wrote this book for data professionals who are unafraid to add these new skillsets and features to their utility belt, and to give courage and confidence to those who are still hesitant. Data platform administrators should read this book to become more prepared and so they are aware of features when talking to their colleagues in application development, data analytics, and system administration.

How this book is organized

This book gives you a comprehensive look at the various features you will use. It is structured in a logical approach to all aspects of Microsoft SQL Server and Azure SQL administration, whether you are architecting, implementing, developing, or supporting development.

Part I: Introduction

Chapter 1, “Get started with SQL Server tools,” gives you a tour of modern tooling for SQL Server administrators, from the installation media and all tooling, including SQL Server Management Studio and Azure Data Studio, to performance and reliability monitoring tools, tools for writing PowerShell, and more.

Chapter 2, “Introduction to database server components,” introduces the working vocabulary and concepts of database administration, starting with hardware-level topics such as memory, processors, storage, and networking. We then move into high availability basics (much more on those later), security, and hardware and OS virtualization.

Chapter 3, “Design and implement an on-premises database infrastructure,” introduces the architecture and configuration of SQL Server, including deep dives into transaction log virtual log files (VLFs), data files, in-memory online transaction processing (OLTP), accelerated database recovery (ADR), and other new features of SQL Server 2022. We also spend time with tempdb and its optimal configuration and server-level configuration options. Finally, we introduce you to Kubernetes.

Part II: Deployment

Chapter 4, “Install and configure SQL Server instances and features,” reviews installation of SQL Server for Windows platforms when SQL Server Setup is needed to install SQL Server. We discuss volume settings and layout for a SQL Server instance, editions, Smart Setup and

unattended setup configuration, and setup logging. Look here also for post-installation checklists and configuration guidance, and for configuration and guidance for other features including SSIS, SSAS, and SSRS, as well as PolyBase.

Chapter 5, “Install and configure SQL Server on Linux,” reviews configuration of SQL Server on Linux instances, including feature differences between Windows and Linux. We’ll provide guidance and caveats on Linux distributions, Linux-specific monitoring and storage considerations, and tooling for setup and administration.

Chapter 6, “Provision and configure SQL Server databases,” reviews creation and configuration of SQL Server databases on any SQL Server platform, including strategies for migrating and moving databases. Database options and properties are discussed, as are database collations.

Chapter 7, “Understand table features,” completes the drill down from instances to databases to tables, covering table design, data types, keys, and constraints. The use of IDENTITY and sequences, computed columns and other column properties, as well as special table types, are discussed. We review special types of tables including temporal tables, introduce memory-optimized tables (more on these in Chapter 14), and graph tables. We review FILESTREAM and FileTable for storing blobs, table partitioning for storing and switching large amounts of data, and strategies for tracking data changes. Finally, we dive deep into PolyBase, the powerful SQL Server feature for virtualization of third-party or non-relational data sources.

Part III: SQL Server management

Chapter 8, “Maintain and monitor SQL Server,” covers the care and feeding of SQL Server instances on both Windows and Linux, including monitoring for database corruption, monitoring index activity and fragmentation, and maintaining and monitoring indexes and index statistics. We dive into Extended Events, the superior alternative to traces, and cover Resource Governor, used for insulating your critical workloads. We review monitoring and data collection strategies based in Windows, Linux, and Azure, as well as the SQL Assessment API. Finally, we discuss the current Microsoft servicing model for SQL Server.

Chapter 9, “Automate SQL Server administration,” introduces automating activities for SQL Server, including maintenance plans, but also custom solutions involving PowerShell, including the latest features available in PowerShell. We also review built-in tools and features needed to automate tasks to your SQL Server, including database mail, SQL Server Agent jobs, proxies, SQL Server Agent alerts, event forwarding, and Policy-Based Management.

Chapter 10, “Develop, deploy, and manage data recovery,” covers the fundamentals of SQL Server database backups in preparation for disaster recovery scenarios, including a backup and recovery strategy appropriate for your environment. We use a memorable narrative to explain various factors, features, and failures in a fictional disaster recovery scenario. We discuss how backups and restores in a hybrid environment, Azure SQL Database recovery, and geo-replication are important assets for the modern DBA.

Chapter 11, “Implement high availability and disaster recovery,” goes beyond backups and into strategies for disaster recovery, from log shipping to availability groups, as well as monitoring and troubleshooting availability groups. We compare HA and DR strategies and dive into proper architecture for maximizing SQL Server uptime.

Part IV: Security

Chapter 12, “Administer instance and database security and permissions,” begins with the basics of authentication: the configuration, management, and troubleshooting of logins and users. Then, we dive into permissions, including how to grant and revoke server and database-level permissions and role membership, with a focus on moving security from server to server.

Chapter 13, “Protect data through classification, encryption, and auditing,” takes the security responsibilities of the SQL Server DBA past the basics of authentication and permissions and discusses advanced topics including the various features and techniques for encryption, such as transparent data encryption (TDE) and Always Encrypted, as well as protecting data in motion with TLS. We cover modern strategies for row-level security and protection of sensitive data. We discuss security measures to be taken for SQL Server instances and Azure SQL databases as well as the SQL Server Audit feature.

Part V: Performance

Chapter 14: “Performance tune SQL Server,” dives deep into isolation and concurrency options, including read committed snapshot isolation (RCSI), and why your developers shouldn’t be using NOLOCK. We discuss various strategies for memory-optimized data, including delayed durability. We review graphical execution plans analysis, the important Query Store feature, and automatic plan correction. We also review important performance-related dynamic management objects (DMOs) and new SQL Server 2022 performance features in the intelligent query processing family, including degree of parallelism (DOP) feedback, cardinality estimation (CE) feedback, and enhancements to memory grant feedback.

Chapter 15: “Understand and design indexes,” tackles performance from the angle of indexes, including their creation, monitoring, and tuning. We review all the various forms of indexes at our disposal, past rowstore clustered and nonclustered indexes and into other types of indexes including columnstore and memory-optimized hashes. We review statistics and statistics options, including how they work on a variety of index and table types, such as the new XML compression feature in SQL Server 2022.

Part VI: Cloud

Chapter 16, “Design and implement hybrid and Azure database infrastructure,” discusses the infrastructure options for Azure-based SQL Server databases, including platform as a service (PaaS) options of Azure SQL Database, Azure SQL Managed Instance, and infrastructure as a service (IaaS) options of Azure VMs running SQL Server instances. We discuss the resource scalability

options for Azure SQL Database, which have dramatically expanded recently. We discuss management and governance in the Azure SQL data platform using the Azure portal and PowerShell.

Chapter 17, “Provision Azure SQL Database,” covers the cloud-first database service without peer in the marketplace. This platform powers many web-based applications and services, scalable from a basic \$5/month plan, to 128-vCore powerhouses, to hyperscale hardware. You will learn about the Azure SQL Database platform, compatibility, security, and availability. You will also learn how to create servers, databases, and elastic pools, and how to perform important management tasks for your databases.

Chapter 18, “Provision Azure SQL Managed Instance,” details the powerful Azure SQL Managed Instance offering, including provisioning, managing, and scaling the instance. We review the service objectives, limitations and advantages, and security features of the managed instance.

Chapter 19, “Migrate to SQL Server solutions in Azure,” covers various strategies for Azure migrations, including the Microsoft tools provided for testing and migrating SQL Server workloads. We review differences and limitations for on-premises feature migration strategies to Azure platforms, including how to migrate SSIS packages to the integration runtime. Finally, we review post-migration steps, best practices for security and resiliency during migration, and the common causes for migration failures.

Conventions

This book uses special text and design conventions to make it easier for you to find the information you need.

Text conventions

The following conventions are used in this book:

- **Boldface type** is used to indicate text that you should type where directed.
- For your convenience, this book uses abbreviated menu commands. For example, “Select Tools > Track Changes > Highlight Changes” means you should select the Tools menu, point to Track Changes, and then select the Highlight Changes command.
- Elements with the Code typeface are meant to be entered on a command line or inside a dialog box. For example, “type `cd \Windows` to change to the Windows subdirectory” means that you should be entering `cd \Windows` with your keyboard or text input device.
- The first letters of the names of menus, dialog boxes, dialog box elements, and commands are capitalized—for example, the Save As dialog box.
- *Italicized type* indicates new terms.

Book features

In addition to the text conventions, this book contains sidebars to provide additional context, tips, or suggestions.

Inside OUT

These are the book's signature tips. In these tips, you'll get the straight scoop on what's going on with the software or service—inside information about why a feature works the way it does. You'll also find field-tested advice and guidance as well as details that give you the edge on deploying and managing like a pro.

READER AIDS

Reader aids are exactly that—Notes, Tips, and Cautions provide additional information on completing a task or specific items to watch out for.

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:

www.MicrosoftPressStore.com/SQLServer2022InsideOut/downloads

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

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



Install and configure SQL Server instances and features

What to do before installing SQL Server.....	130	Post-installation server configuration	163
Install a new instance	137	Post-installation configuration of other features	176
SQL Server on Azure virtual machines.....	163	Container orchestration with Kubernetes.....	182

This chapter reviews the process of installing and configuring a Microsoft SQL Server instance as well as creating and migrating databases. We pay special attention to new features introduced in SQL Server 2022 as well as other recent features you might not have noticed in earlier editions of SQL Server. We also discuss how to deploy SQL Server using containers and Kubernetes.

We present a post-installation checklist for you to use to verify your installation. When necessary, we also direct you to other sources of information and details for critical steps elsewhere in this book.

The content in this chapter related to SQL Server Setup mainly applies to SQL Server installations on Windows operating systems. Provisioning is vastly simplified for Azure SQL Database, Azure SQL Managed Instance, SQL Server on Linux, SQL Server in Linux containers, and Azure virtual machine (VM) images with pre-installed SQL Server from the Azure Marketplace. Even so, many recommended settings in this chapter apply for server-based platforms of SQL Server, such as in Linux containers or SQL Server on Linux. They are, after all, still very much the same SQL Server products that have always existed on Windows.

This chapter focuses on server-level setup and settings. Chapter 6, “Provision and configure SQL Server databases,” covers the initial creation and configuration of databases inside the SQL Server instance.

- For more on SQL Server on Linux, see Chapter 5, “Install and configure SQL Server on Linux.”
- For more on Azure SQL Database, see Chapter 17, “Provision Azure SQL Database.”
- For more on Azure SQL Managed Instance, see Chapter 18, “Provision Azure SQL Managed Instance.”
- For more on database migrations to SQL Server platforms in Azure, see Chapter 19, “Migrate to SQL Server solutions in Azure.”

What to do before installing SQL Server

Before running SQL Server Setup on your Windows Server, there are several factors and settings to consider—some of which you *cannot* easily change after installation. For example, choosing between the default instance and a named instance or choosing an instance collation are not decisions you can easily reverse after installation. (More about the server-level collation option later in this chapter, in the “Instance collation” section.)

However, many mistakes made in installation can be resolved afterward—albeit likely with some tedium and outages. For example, skipping the initial default data and log directories may land all your databases on the operating system (OS) volume. They can be moved to the appropriate volumes later, but it’s best to get it right the first time.

CAUTION

Do not install SQL Server on the same server as a domain controller. In some scenarios, it is not supported, and can even cause SQL Server Setup to fail.

SQL Server 2022 has most of the same hardware and software requirements as SQL Server 2019. There are some differences, however. For example, SQL Server 2022 requires .NET Framework 4.7.2, which you can download from <https://dotnet.microsoft.com/download/dotnet-framework/net472>.

In addition, we recommend you acquire the following before starting SQL Server Setup:

- Active Directory (AD) service accounts for the SQL Server service, SQL Agent service, and other features if needed
- The latest download of cumulative updates to bring the instance up to the latest patch level
- A licensing decision around the number of processors and the edition to buy
- A secure enterprise digital location for various passwords you will generate, backups of certificates, and keys
- A decision as to whether to install the default or a named instance
- A plan for where SQL Server files will go, with each volume formatted to the 64-KB disk unit allocation size (discussed in the next section)

Inside OUT

What are SQL Server service accounts?

SQL Server service accounts are the accounts used to handle the communication of services between the OS and SQL Server. Using AD accounts on Windows for these accounts is a best practice. These can be updated and set up after installation in SQL Server Configuration Manager if needed.

If possible, it is recommended to use managed service accounts (MSAs) or group managed service accounts (gMSAs). These are specially provisioned Windows accounts whose passwords are self-managed by Windows. This means privileged service account secrets no longer need to be secured and managed, providing greater security. For more information, visit <https://learn.microsoft.com/windows-server/security/group-managed-service-accounts/group-managed-service-accounts-overview>.

Decide on volume usage

For many good reasons, various types of SQL Server files should be placed on separate volumes. Although you can move user and system database data and log files to other locations after installation, it's best to plan your volumes before installation.

The examples in this chapter assume your Windows OS installation is on the C: volume of your server. You should have many other volumes for SQL Server files, and we'll review a sample layout soon. One of the basic guiding principles for a SQL Server installation is that anywhere you see "C:\," you should change it to another volume. This helps minimize SQL Server's footprint on the OS volume (especially if you install multiple SQL Server instances), and can have potential disaster recovery implications in terms of volume-level backups and restores.

Inside OUT

What if you are tight on space on the OS volume after installing SQL Server?

There are some easy ways and some tricky ways to minimize the footprint of a SQL Server installation on the OS volume of your server (typically the C: volume, as it is for this example). In general, SQL Server Setup and cumulative updates delete temporary files involved in their installation, but not log files or configuration files, which should have a minimal footprint. Apart from log files, we recommend that you not delete any files installed by SQL Server Setup or cumulative updates. Instead, let's look at some proactive steps to move these files off the C: volume.

Some parts of SQL Server Setup install on the OS volume (typically, and in this and future examples, the Windows C: volume). These files, which are staging areas for SQL Server Setup, are created on the OS volume in a C:\Program Files\Microsoft SQL Server\160\Setup Bootstrap\ subfolder structure, where 160 is specific to the internal version number (16.0) of SQL Server 2022. This folder is used for future cumulative updates or feature changes.

If you're extremely tight on space before installing SQL Server, you will also find that the root binaries installation directory is, by default, C:\Program Files\Microsoft SQL Server\. When you're using the SQL Server Setup user interface, there is no option to change this. You will, however, find this installation directory folder path listed as the INSTANCEDIR parameter in the config file that is generated by SQL Server Setup. How to use the config file to install SQL Server is further covered in the section "Automate SQL Server Setup with configuration files" later in this chapter.

If this is the first SQL Server instance you are installing on a server, you will have the opportunity to change the location of shared features files, the data root directory for the instance (which contains the system databases), and default database locations for user database files and their backups. If this is not the first SQL Server 2022 instance installation on this server, the shared features directory locations (for Program Files and Program Files x86) will already be set for you, and you cannot change them.

You should place as much of the installation as possible on other volumes, not the OS volume. Keep in mind that a full-featured installation of SQL Server 2022 can consume more than 14 GB.

Inside Out

What can you do with the D: volume on an Azure VM?

For Microsoft Azure Windows VMs, do not set the installation directories for any settings on the "Temporary Storage" D: volume. In a Linux VM, the same applies to /dev/sdb1.

The D: volume is the temporary storage volume on an Azure VM. The temporary storage volume is a high-speed disk that is locally present on the machine hosting the Azure VM, so it has better performance and lower latency than the default C: volume. The temporary storage volume contains only the Windows page file by default and is wiped upon server restart, resize, or host migration.

The only possible long-term use for the temporary storage volume is for tempdb files, which can exist on this drive if certain other considerations are taken. Otherwise, do not store any non-temporary files in the temporary storage.

For more details on using the D: volume for tempdb files, see "Locate tempdb files on the VM" in Chapter 16, "Design and implement hybrid and Azure database infrastructure."

The following sample scenario is a good starting point for a volume layout for your SQL Server installation (the volume letters don't matter):

- **Volume C.** The OS. Some SQL Server files must be installed here.
- **Volume E.** SQL Server installation files, log files, SQL Server database data files.
- **Volume F.** SQL Server database log files.
- **Volume G.** SQL Server tempdb data files and log files. (Alternatively, use the D: Temporary Storage volume on Azure Windows VMs.)
- **Volume H.** SQL Server backups (if written locally).

Here are some more advanced volume decisions:

- Use additional volumes for your largest data files (larger than 2 TB) for storage manageability:
 - For the most active databases
 - For FILESTREAM filegroups
 - For database replication snapshot files
 - For the Windows page file, especially for servers with large amounts of memory

Inside OUT

Why separate SQL Server files onto different volumes?

There are good reasons to separate your SQL Server files into various volumes, and not all of them are related to performance. You should still separate your files onto different volumes even if you exclusively use a storage area network (SAN).

More discrete storage I/O on a physical server with dedicated drives means better performance. But even in a SAN, separating files onto different volumes is also done for stability. Think of the volumes as bulkheads on a submarine. If a volume fills and has no available space, files cannot be allocated additional space. On the OS volume, running out of free space would result in Windows Server stability issues—user profile and remote desktop problems at least—and affect other applications.

Important SQL Server volume settings

There are some settings to consider for volumes that host SQL Server data and log files, and this guidance applies specifically to these volumes. For other volumes—for example, those that contain the OS, application files, or backup files—the default Windows settings are acceptable unless otherwise specified.

When adding these volumes to Windows, there are important volume configuration settings that you must examine or discuss with your storage administrator:

- When creating new drives, opt for GUID Partition Table (GPT) over Master Boot Record (MBR) disk types for new SQL Server installations. GPT is a newer disk-partitioning scheme than MBR, and GPT disks support files and volumes larger than 2 TB. In contrast, the older MBR disk type is capped at 2 TB.
- The appropriate file unit allocation size for SQL Server volumes is 64 KB, with few exceptions. Setting this to 64 KB for each volume can have a significant impact on storage efficiency and performance. The Windows default is 4 KB, which is not optimal for SQL Server data and log files.

To check the file unit allocation size for an NT File System (NTFS) volume, run the following from the Administrator: Command Prompt, repeating for each volume:

```
fsutil fsinfo ntfsinfo d:
```

The file unit allocation size is returned with the Bytes Per Cluster; thus, the desired 64 KB would be displayed as 65,536 (bytes). If formatted as the default, this will display 4096. Correcting the file unit allocation size requires formatting the drive, so it is important to check this setting before installation.

If you notice this on an existing SQL Server instance, your likely solution is to create a new volume with the proper file unit allocation size and then move files to the new volume during an outage. Do *not* format or re-create the partition on volumes with existing data; you will lose the data when it is reformatted.

Modern storage devices are currently in a transition between disks that use a Bytes per Physical Sector size of 512 bytes (the old standard) and “4K Native” disks that have both a Bytes per Sector size and a Bytes per Physical Sector size of 4 KB. Usually, a DBA will not notice or even be aware of this difference. When configuring availability groups or log shipping between servers on different storage systems with mixed Bytes per Physical Sector modes, however, this can result in very poor performance, with the transaction logs unable to truncate, and the error message “There have been *nnn* misaligned log IOs which required falling back to synchronous IO.” You may encounter this with hybrid availability groups spanning on-premises and Azure VM-based SQL Server instances, for example.

This cannot be resolved via reformatting, but can potentially be resolved via hardware-level storage or firmware settings. To avoid this, all storage that hosts the transaction log

files of SQL Servers in an availability group or log shipping relationship should have the same Bytes per Physical Sector.

A workaround is to apply Trace Flag 1800 as a startup flag on the SQL Server instances that use storage without having a Bytes per Physical Sector setting of 4K. TF1800 overrides disk default behavior and writes the transaction log in 4-KB sectors, resolving the issue. TF1800 must be enabled on the on-premises SQL Server instances in the case of using the older on-premises and Azure VM availability group.

Check the Bytes per Physical Sector setting of a volume by using the same `fsutil` command noted in the previous code sample.

- A hardware-level concept related to file unit allocation size called *disk starting offset* deals with how Windows, storage, disk controllers, and cache segments align their boundaries. Aligning disk starting offset was far more important before Windows Server 2008. Since then, the default partition offset of 1,024 KB has been sufficient to align with the underlying disk's stripe unit size, which is a vendor-determined value and rarely a concern for DBAs. Still, it should be verified upon first use of a new storage system or upon the migration of disks to a new storage system. This can be verified in consultation with the drive vendor's information.

To access the disk starting offset information, run the following from the Administrator: Command Prompt:

```
wmic partition get BlockSize, StartingOffset, Name, Index
```

A 1,024-KB starting offset is a Windows default, which is displayed as 1048576 (bytes) for Disk #0 Partition #0.

Like the file unit allocation size, the only way to change a disk partition's starting offset is destructive: You must re-create the partition and reformat the volume to align with the vendor-supplied offset.

SQL Server editions

The following are brief descriptions of all the editions in the SQL Server family, including past editions that you might recognize. It's important to use the appropriate licenses for SQL Server, even in preproduction systems.

NOTE

This book is not intended to be a reference for licensing or sales-related documentation; still, editions are a key piece of knowledge for SQL Server administrators to understand what features may or may not be available.

- **Enterprise.** Appropriate for production environments; not appropriate for preproduction environments such as user acceptance testing (UAT), quality assurance (QA), testing, development, or a sandbox. For these environments, you should instead use the free

Developer edition. You'll have a far easier time in a licensing audit if your preproduction environment installations are Developer edition.

- **Developer.** Appropriate for all preproduction environments, especially those under a production Enterprise edition. Not allowed for production environments. This edition has the same features and capacity as Enterprise edition and is free.
- **Standard.** Appropriate for production environments. Lacks the scale and compliance features of Enterprise edition required in some regulatory environments. Limited to the lesser of 4 sockets or 24 cores and 128 GB of buffer pool memory, whereas Enterprise edition is limited only by the OS for compute and memory.
- **Web.** Appropriate for production environments but limited to low-cost server environments for web hosting.
- **Express.** Not appropriate for most production environments or preproduction environments. Appropriate only for environments in which data size is small, is not expected to grow, and can be backed up with external tools or scripts (because Express edition has no SQL Server Agent to automate backups). The free Express edition is ideal for production proofs-of-concept, lightweight applications, and student projects. It lacks some critical features and is severely limited on compute (lesser of 1 socket or 4 cores), available buffer pool memory (1,410 MB), and individual database size (10-GB capacity).
- **Express with Advanced Services.** Like Express edition in all caveats and limitations, this edition includes some additional features, including R integration and full-text search.
- **Evaluation.** Functionally the same as Enterprise edition, and free with a 180-day shut-down timer. Evaluation edition isn't supported. This edition can be upgraded to any edition except for Express. Do not use this edition if you plan for a clustered installation, because an upgrade in that case is not supported.

It's worth noting that the hardware limitations of SQL Server editions have not changed since SQL Server 2016.

NOTE

When you run the SQL Server 2022 Setup, you can choose to install several features outside the core database features. Installing SQL Server features on multiple Windows servers requires multiple licenses per server, even if you intend to install each SQL Server instance's features only once.

There is an exception to this rule, however: If you have licensed all physical cores on a host server for SQL Server Enterprise edition, and purchased Software Assurance, you can install any number or combination of SQL Server instances and their standalone features on virtual guests.

Change SQL Server editions and versions

Upgrading editions in-place is supported by a feature of the SQL Server 2022 installer. You can upgrade in the following order: Express, Web, Standard, and Enterprise.

You cannot downgrade a SQL Server version or licensed edition. This type of change requires a fresh installation and migration. For example, you cannot downgrade in-place from SQL Server 2022 Enterprise edition to Standard edition.

In-place upgrades for major versions (from 2019 to 2022, for example) is supported but not recommended. Instead, we strongly recommend that you perform a fresh installation of the newer version and then migrate from old to new instances. This method offers major advantages in terms of duration of the planned outage, rollback capability, and robust testing in parallel.

Although in-place upgrades to SQL Server 2022 are not recommended, upgrades are supported for versions as old as SQL Server 2012 SP4. You can even migrate databases using detach and reattach, from older versions of SQL Server to SQL Server 2022, as long as the source database compatibility level is 90 or higher. Databases with a compatibility of 90 (SQL Server 2005) will be automatically upgraded to compatibility level 100. Databases already at compatibility level 100 will not change.

A supported upgrade also assumes that the OS and previous version of SQL Server are not 32-bit installations. Beginning with SQL Server 2016, SQL Server is available only for 64-bit platforms. For more information on upgrades to SQL Server 2022, visit <https://learn.microsoft.com/sql/database-engine/install-windows/supported-version-and-edition-upgrades-2022>.

Install a new instance

In this section, you learn how to begin a new SQL Server 2022 instance installation, upgrade an existing installation, or add features to an existing instance.

The instructions in this chapter are the same for the first installation or any subsequent installations, whether it is for the default or any named instances of SQL Server 2022. As opposed to an exhaustive step-by-step instruction list for installations, we've opted to cover the important decision points and the information you need and to highlight new features from SQL Server 2022.

Even though you can change *almost* all of the decisions you make in SQL Server Setup after installation, those changes potentially require an outage or server restart. Making the proper decisions at installation time is the best way to ensure the least administrative effort. Some security and service account decisions should be changed only via the SQL Server Configuration Manager application, not through the Services console (services.msc). This guidance will be repeated elsewhere for emphasis.

We begin by going through the typical interactive installation. Later in this chapter, we will go over some of the command-line installation methods that you can use to automate the installation of a SQL Server instance.

Plan for multiple SQL Server instances

You can install as many as 50 SQL Server instances on a Windows Server, although we obviously do not recommend this. In a Windows failover cluster, the maximum number of SQL Server instances is reduced by half if you're using shared cluster drives.

Only one of the SQL Server instances on a server can be the default instance. All, or all but one, of the SQL Server instances on a SQL Server will be named instances. The default instance is reachable by connecting to the name of the Windows Server, whereas named instances require an instance name. The SQL Browser service is required to handle traffic for named instances on the SQL Server.

For example, you can reach the default instance of a SQL Server by connecting to `servername`. All named instances have a unique instance name, such as `servername\instancename`.

NOTE

If the Browser service is not turned on, this does not mean you cannot reach the instance, but that you will need to know the specific port on which it is listening. You reach the instance using `servername, portnumber` in place of the instance name.

Inside OUT

What is different about SQL Server on an Azure VM?

The Azure Marketplace provides VM images that are pre-installed with SQL Server, with a wide selection of edition and compute options, so you usually won't install SQL Server yourself. However, the default configuration might require some tweaking.

When it comes to licensing, there are two types of SQL Server licensing agreements for Azure VMs. SQL Server VM images in the Azure Marketplace contain the SQL Server licensing costs as an all-in-one billing package.

Alternatively, if you'd like to leverage your existing Enterprise licensing agreement using the Azure Hybrid Benefit, there are three options:

- Choose bring-your-own-license (BYOL) VM images using the same process, then later associate your existing Enterprise license agreements. The image names you're looking for here are prefixed with BYOL.

- Manually upload an .iso file to the VM and install SQL Server 2022 as you would on any other Windows Server.
- Upload an image of an on-premises VM to provision the new Azure VM.

You cannot change from the built-in licensing model to the BYOL licensing model after the VM has been provisioned. You need to make this decision before creating your Azure VM.

Install SQL Server on Windows

The rest of this chapter is dedicated to installations of SQL Server that are not part of a pre-made Azure Marketplace VM and apply to the installation of SQL Server on any Windows Server.

While logged in as a local Windows administrator, begin by mounting the installation .iso to the Windows server. These days, this rarely involves inserting a physical disc or USB flash drive, although you can use them if necessary.

Launch SQL Server Setup

You should not run SQL Server Setup with the installation media mounted over a remote network connection, a shared remote desktop drive, or any other high-latency connection. For a faster SQL Server Setup experience, unpack the contents of the .iso file to a physical folder local to the server.

Start setup.exe on the SQL Server Setup media, running the program as a Windows user with administrator privileges. If AutoPlay is not turned off (it usually is), setup.exe will start when you first mount the media or double-click to open the .iso. Instead, as a best practice, right-click **setup.exe** and select **Run As Administrator** on the shortcut menu that appears.

We'll review here a few items (not all) in the SQL Server Installation Center worth noting before you begin an installation.

In the pane on the left, select **Planning** to open a long list of links to Microsoft documentation websites. Most helpful here might be a standalone version of the System Configuration Checker, which you run during SQL Server Setup later, but it could save you a few steps if you review it now. A link to download the Data Migration Assistant (DMA) is also present, which is a helpful Microsoft-provided tool for upgrading from prior versions of SQL Server.

On the **Maintenance** page, you will find the following:

- The **Edition Upgrade Wizard** is relatively painless. This is only for promoting your existing installation's edition, as discussed earlier.

- The **Repair** feature is not commonly used except in the case of an instance with a corrupted installation. You might also need to repair an instance of SQL Server when the executables, .dll files, or Windows Registry entries have become corrupted or damaged by disk corruption, antimalware, malware, or malicious activity. A failed SQL Server in-place upgrade or cumulative update installation might also require a repair, which could be better than starting from scratch.
 - Removing a node from an existing SQL Server failover cluster is an option in the **Maintenance** page. Adding a node to an existing SQL Server failover cluster is an option in the **Installation** page.
 - The **Advanced** page features a link to perform an installation based on a configuration file. We will discuss how to easily generate and use a configuration file later in this chapter, in the section “Automate SQL Server Setup with configuration files.” If you are tasked with installing multiple SQL Servers with mostly common settings, consider this time-saving method. There are also links to wizards for advanced failover cluster installations.
- We discuss failover cluster instances (FCIs) in Chapter 11, “Implement high availability and disaster recovery.”

Windows Update in the SQL Server Setup

Since SQL Server 2012, the SQL Server installer has had the ability to patch itself within the Setup wizard. The **Product Updates** page is presented after the **License Terms** page, and, after you accept it, it is downloaded from Windows Update (or Windows Server Update Services) and installed along with other SQL Server Setup files.

This is recommended, so a SQL Server 2022 Setup with Internet connectivity is the easiest way to carry out the installation. This also could be described as a way to “slipstream” updates, including hotfixes and cumulative updates, into the SQL Server installation process, eliminating these efforts post-installation.

For servers without Internet access, there are two setup.exe parameters that support downloading these files to an accessible location and making them available to Setup. When starting setup.exe from Windows PowerShell or the command line (you can read more about this in the next section), you set the /UpdateEnabled parameter to FALSE to turn off the download from Windows Update. The /UpdateSource parameter can then be provided as an installation location of .exe files. Note that the /UpdateSource parameter is a folder location, not a file. You will find more on these two parameters later in the “Install by using a configuration file” section.

Regardless, after installation is complete, and before the SQL Server enters further use, verify that the latest SQL Server patches have been applied. For SQL Server 2022, see the official build versions at <https://support.microsoft.com/help/4518398>.

Install SQL Server stand-alone installation

Although what follows in this chapter is not a step-by-step walk-through, we do cover key new features and decision points of the **New SQL Server Stand-Alone Installation** option of the SQL Server Installation Center.

Inside OUT

Where are the installers for SQL Server Management Studio and SQL Server Data Tools?

SQL Server Management Studio, SQL Server Data Tools (for Visual Studio 2015 and higher), and SQL Server Reporting Services are no longer installed with SQL Server's traditional setup media. These products are now updated regularly (as often as monthly) and available for download.

You should keep up-to-date versions of SQL Server Management Studio (SSMS) on administrator workstations and laptops.

Avoid installing SSMS locally on the SQL Server if possible. In fact, avoid the need to use Remote Desktop Connection to manage and administer the SQL Server altogether. For all SQL Server platforms, try to use SSMS, Azure Data Studio, PowerShell, and other tools to do as much of your work on SQL Server remotely as possible.

PolyBase Services

Immediately after the instance configuration is a new configuration for the port range of PolyBase services. This is where you choose a range of ports to use for this service. If you plan to use PolyBase, the ports typically used are TCP ports between 16450 and 16460, of which there must be at least six ports. These should be allowed through the firewall if needed. This option was added to SQL Server Setup in SQL Server 2022.

Grant Perform Volume Maintenance Tasks

On the same **Server Configuration** page on which service accounts are set, notice the **Grant Perform Volume Maintenance Task privilege to the SQL Server Database Engine Service** check box. Selecting this check box automates what used to be a standard post-installation checklist step for SQL DBAs beginning with Windows Server 2003.

The reason to grant this permission to use instant file initialization is to speed the allocation of large database data files, which could dramatically reduce the Recovery Time Objective (RTO) capacity for disaster recovery. This can mean the difference between hours and minutes when

restoring a very large database. It can also have a positive impact when creating databases with large initial sizes, or in large autogrowth events—for example, with multiple data files in the tempdb (more on this next). It is recommended that you allow SQL Server Setup to turn on this setting.

Inside OUT

How can you verify that instant file initialization is enabled?

IFI is granted to a SQL Server service account via the Perform Volume Maintenance Tasks permission in Local Security Policy on the Windows server. But it's straightforward to verify whether IFI is in place for the SQL Server service, via the `sys.dm_server_services` dynamic management view:

```
SELECT servicename, instant_file_initialization_enabled
FROM sys.dm_server_services
WHERE filename LIKE '%sqlservr.exe%';
```

- For more information on instant file initialization, see Chapter 3, “Design and implement an on-premises database infrastructure.”

Instance collation

The **Collation** tab on the **Server Configuration** page allows you to choose a collation for the Database Engine. The collation determines how character data is stored, sorted, and compared. For more information, see the section on collation in Chapter 7, “Understand table features.”

Initially, the instance collation provided in SQL Setup is the default collation for the server’s regional settings, but you might need to change this collation based on vendor or developer specifications.

While changing the collation of a database is easy, the instance collation is important to get right at the time of SQL Server installation, as changing the instance collation is quite difficult.

The server collation you set here acts as the collation for all system databases as well as the default for any newly created user databases. For new application development, you may choose to take advantage of UTF-8 collations as the server default, introduced in SQL Server 2019.

Inside OUT

How do you change the server collation after installing SQL Server?

This is one of those things you want to get right at the time of installation. To change the collation of the SQL Server instance, reference this lengthy and difficult Microsoft guide, at <https://learn.microsoft.com/sql/relational-databases/collations/set-or-change-the-server-collation>.

In the case of Azure SQL Managed Instance, you cannot change the server-level collation after it is created. For more information, visit <https://learn.microsoft.com/sql/relational-databases/collations/set-or-change-the-server-collation#setting-the-server-collation-in-managed-instance>.

Mixed Mode authentication

SQL Server supports two modes of authentication: Windows Authentication and SQL Authentication. Windows Authentication is preferable to SQL Authentication, and in multiple places in this book we will emphasize this.

- You can read more on this topic in Chapter 12, “Administer instance and database security and permissions,” but it is important to note this decision point here.

Ideally, all authentication is made via Windows Authentication, through types of server principals called *logins* that reference Windows accounts—ideally, AD domain accounts or, starting with SQL Server 2022, Azure Active Directory (Azure AD) principals. These are created by your existing enterprise security team, which manages password policy, password resets, password expiration, and so on.

A redundant security model for connecting to SQL Server also exists within each instance: SQL Server Authenticated logins. Logins are maintained at the SQL Server level, are subject to local policy password complexity requirements, are reset/unlocked by SQL DBAs, have their own password change policy, and so forth.

Enabling Mixed Mode (SQL and Windows Authentication Mode) activates SQL Authenticated logins. Be aware that SQL Authentication is not on by default, and isn’t the recommended method of connection. The recommended Windows Authentication cannot be turned off. When possible, applications and users should use Windows Authentication.

Enabling Mixed Mode also activates the *sa* account, which is a special built-in SQL Server Authentication that is a member of the server sysadmin role. Setup will ask for a strong password to be provided at this time.

- You can learn more about the sa account and server roles in Chapter 12.

If you find you have an actual need to enable SQL Server Authentication, but didn't do this during SQL Server Setup, you can do it later by connecting to the SQL Server instance via Object Explorer in SQL Server Management Studio. To do so, right-click the server name and select **Properties** from the shortcut menu. Then, select the **Security** page and change to **Mixed Mode**. You must perform a SQL Server service restart to effect this change.

Default settings for the tempdb database

Starting with SQL Server 2016, SQL Server Setup provides a more realistic default configuration for the number and size of tempdb data files. This has been a common to-do list for all post-installation checklists for DBAs since the early days of SQL Server.

The TempDB database page in SQL Server Setup provides not only the ability to specify the number and location of the tempdb's data and log files, but also their initial size and auto-growth rates. The best number of tempdb data files is almost certainly greater than one and less than or equal to the number of logical processor cores, including hyper-threading for local machines. For example, with 16 logical processors, SQL Server Setup will default the installation to have eight tempdb data files.

Adding too many tempdb data files can degrade SQL Server performance—perhaps severely. For example, with 20 logical processors, SQL Server Setup will still default the installation to have 8 tempdb data files. If you add 20 tempdb data files, SQL Server may struggle to respond.

- For more information on the best number of tempdb data files, see Chapter 3.

Specifying tempdb's initial size to a larger, normal operating size is important and can improve performance after a SQL Server restart when the tempdb data files are reset to their initial size. Setup accommodates an individual tempdb data file initial size up to 256 GB. For data file initial sizes larger than 1 GB, you will be warned that SQL Server Setup can take a long time to complete if instant file initialization is not turned on.

Since SQL Server 2016, all tempdb files autogrow at the same time, keeping file sizes the same over time, which is critical to the way multiple tempdb data files are used. This is superior to the old way of ensuring tempdb data files stay the same size: using the server-level setting via server Trace Flag 1117, which applied the data file growth behavior to all databases. Trace Flag 1117 is no longer necessary.

Also note the naming convention for the second tempdb data file and beyond: *tempdb_mssql_n.ndf*. A SQL Server uninstallation will automatically clean up tempdb data files with this naming convention. For this reason, we recommend that you follow this naming convention for tempdb data files.

- The tempdb system database is discussed in detail in Chapter 3.

Default settings for MAXDOP

New in SQL Server 2019 were defaults for the configuration of the server-wide **Maximum Degree of Parallelism (MAXDOP)** setting on the **Database Engine Configuration** page under the new **MaxDOP** tab.

In the same way that new tempdb defaults since SQL Server 2016 are dependent on the detected processors, a suggested default MAXDOP is also configured based on the number of logical processors. For many servers with 16 or fewer virtual processor cores, the default is the same as the number of cores, effectively the same as a MAXDOP setting of 0, which allows for unlimited parallelism.

For example, with 8 logical processors, SQL Server Setup will default the installation to use a MAXDOP of 8. With over 16 logical processors, SQL Server Setup may default to half the number of logical processors—at most 16. For example, with 20 logical processors, SQL Server Setup will default the installation to use a MAXDOP of 10.

- For more recommendations about MAXDOP, visit Microsoft Support at <https://support.microsoft.com/help/2806535>. See also the section on “Max degree of parallelism” in Chapter 3.

You can always reconfigure the MAXDOP after installation without a restart, though not without potential disruption. Although changing the server-wide (or database-level) MAXDOP setting takes effect immediately, it is definitely not advisable to do so during normal production operating hours, because it can lead to widespread plan recompilation and a heavy CPU spike. This server-wide MAXDOP setting can be overridden at the database, query, or Resource Governor group level. The **MaxDOP** tab in the **Database Engine Configuration** tab has a recommended MAXDOP setting of 8 for a server with eight virtual cores. This is effectively the same as a MAXDOP of 0, but offers the administrator an option to potentially change the MAXDOP at the time of installation.

NOTE

Some applications recommend disabling parallelism on their databases. Consult your vendor’s specifications and recommendations documentation. MAXDOP can be set at the server level now, then configured and overridden at each database level after SQL Server Setup is complete using a database scoped configuration.

- For much more information on performance tuning, parallelism, and the MAXDOP setting, see Chapter 14, “Performance tune SQL Server.”

Default settings for Maximum Server Memory

New in SQL Server 2019 were defaults for the configuration of the instance-level **Max Server Memory** option, a common post-installation checklist item, under the **Memory** tab of the **Database Engine Configuration** page. SQL Server Setup makes a guess based on total server memory for an appropriate option. In previous versions of SQL Server, it was important to remember to change the Max Server Memory setting after installation was complete; otherwise, SQL Server memory would be uncapped and have access to all memory on the server.

You can configure this Max Server Memory option intelligently at the time of installation. It's important to note (and there's a check box to accept this guess) that SQL Server Setup assumes this SQL Server instance will run alone on this server. If you expect to host other applications on this server, or to run memory-heavy features of SQL Server on the same server such as SSAS or SSRS, you should further reduce the Max Server Memory setting for the SQL Server instance.

- Chapter 3 discussed the Max Server Memory setting, in the “Configuration settings” section.

Let's use an example of the new Max Server Memory recommendation for a Windows Server with one SQL Server instance and 16 GB of memory. SQL Server Setup recommends a Max Server Memory setting of 12672 MB. The Min Server Memory setting, which establishes a floor for memory allocation, is set to 0. It is generally unnecessary to change this setting from the default. You might find this setting useful for situations in which the total system memory is insufficient and many applications, including SQL Server instances, are present. The Min Server Memory setting is not immediately allocated to the SQL Server instance upon startup; instead, it does not allow memory below this level to be freed for other applications. Figure 4-1 shows the Memory tab of the Database Engine Configuration page, with the Min Server Memory and Max Server Memory settings visible.

After installation, server memory settings are accessible via SQL Server Management Studio, in Object Explorer, and on the Server Properties page.

You should ensure that SQL Server leaves enough memory for the OS and other applications. Keep in mind that SQL Server will slowly consume more memory over time and may take hours or days, depending on your business cycle, for the SQL Server instance to consume the maximum amount of memory made available. Lowering this setting after installation and during operation does not return SQL Server memory back to the OS immediately; rather, it does so over time during SQL Server activity. Increasing this setting will not immediately show the effect of a change in memory use.

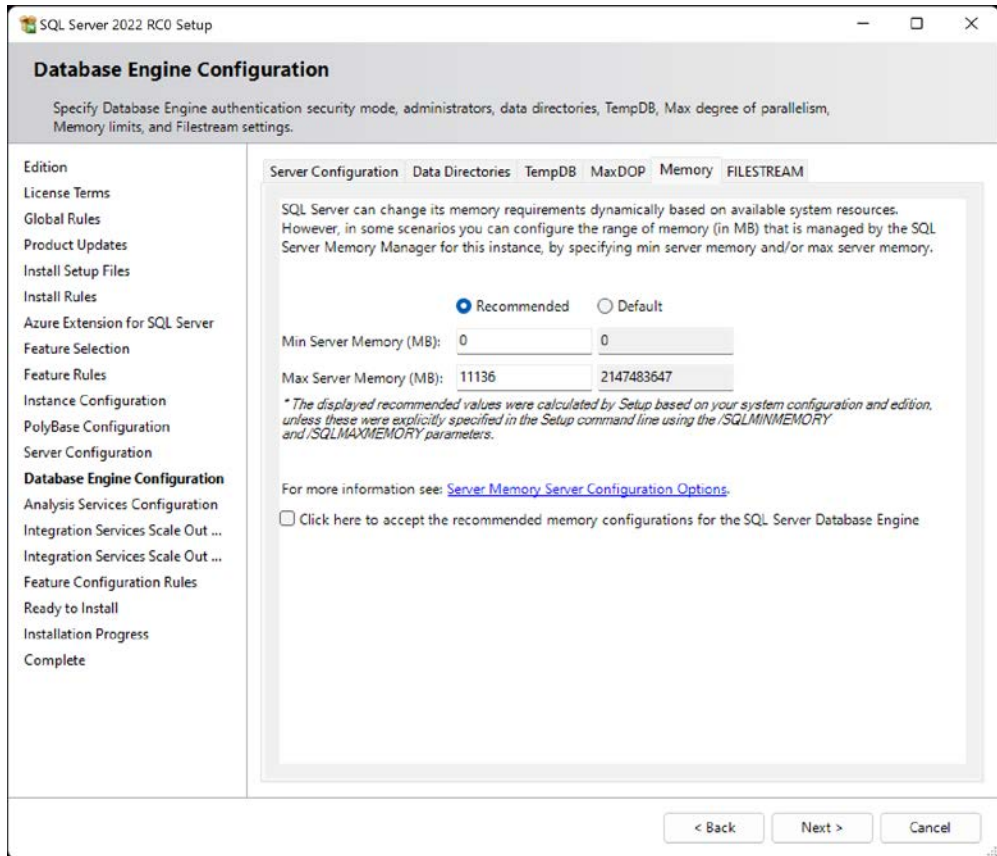


Figure 4-1 This figure displays the minimum and maximum server default memory settings for the SQL Server setup.

Install common features

Aside from the SQL Server service itself, other features of the product might be common to your installations. For example, SQL Server Analysis Services, SQL Server Integration Services, and SQL Server Reporting Services are part of the license and are provided at no additional cost. This section covers the installation of these features using SQL Server Setup. Later, this chapter covers the post-installation steps necessary to use them.

Install SQL Server Analysis Services

Installing SQL Server Analysis Services (SSAS) requires you to decide ahead of time which mode to install. Each instance of SSAS can be in only one mode, which means that with a single license, you can run either Multidimensional mode, the newer Tabular mode (introduced in SQL Server 2012), or Power Pivot mode.

Ask your business intelligence (BI) decision makers which platform you should use. For most new development, Tabular mode is popular and recommended. Tabular mode databases can also run in Azure Analysis Services. Brief descriptions of each mode follow:

- **Multidimensional.** This is the SSAS setup that was introduced in SQL Server 2000 and helped revolutionize the data-warehousing industry. This is also the only mode to support data mining and other features on which existing SSAS data models predating SQL Server 2012 may be dependent. The primary language for building and querying multidimensional models is MDX.
 - **Tabular.** This is the newer and recommended SSAS setup introduced in SQL Server 2012, using the in-memory VertiPaq processing engine. Since SQL Server 2017, this has been the default installation mode selected on the Analysis Services Configuration page of SQL Server Setup. The primary language for building and querying tabular models is DAX, which is similar to the Excel function language.
 - **Power Pivot.** This mode installs SSAS in Power Pivot for SharePoint mode. Power Pivot workbooks use both DAX and MDX. Note that Analysis Services Power Pivot for SharePoint support for Microsoft SharePoint 2019 has been discontinued.
- For more on the differences between these SSAS installation options, visit <https://learn.microsoft.com/analysis-services/comparing-tabular-and-multidimensional-solutions-ssas>.

Inside OUT

What if you choose the wrong SSAS mode?

If you choose one SSAS mode at installation, but your BI developers want another mode, the supported option is to uninstall and reinstall the SSAS feature. However, changing the SSAS mode from Multidimensional to Tabular, or vice versa, after installation is not supported, and administrators are specifically warned not to do this.

Packages developed for each mode are not supported for the other. If no databases have been deployed to the SSAS server instance, changing the `DeploymentMode` property in the `MSMDSRV.ini` file should make it possible to change an existing instance. But again, this is not a supported change. The file is located in `%Programfiles%\Microsoft SQL Server\MSAS15.instanceName\OLAP\Config\`.

Install SQL Server Integration Services

The SQL Server Integration Services (SSIS) instance is installed once per server per version, not once per instance like other features. Starting in SQL Server 2017, however, a new Integration

Services Scale Out Configuration became available. We discuss this new feature further in the next section.

A 64-bit version of SSIS is installed on 64-bit operating systems. If you worry about connecting to 32-bit servers, data sources, or application installations (such as Microsoft Office), you don't need to. Those connections are not dependent on the 32-bit/64-bit installation and are handled at the package or connection-string level. Unlike other features, you can install SSIS on a 32-bit OS; however, we do not recommend this.

Installations of different versions of SSIS are installed side by side on a server. Specifically, SSIS 16.0 is compatible with prior versions.

Apart from configuring the service account, you need not do any additional configuration when installing SSIS during SQL Server Setup. The default virtual service account is NT Service\MsDtsServer160.

Inside OUT

Should you install SSIS alone on a server?

A standalone installation of SSIS without a matching Database Engine instance is possible but not recommended. For the modern Project Deployment model of SSIS, the storage and logging of packages will still be dependent on a SQL Server Database Engine, and the execution of packages on a schedule still requires a SQL Agent service.

So, the SSIS workload is not best isolated in this way. A dedicated installation including the SQL Server Database Engine and SQL Server Agent is a better configuration to isolate SSIS package runtime workloads from other database workloads. Both options carry the same licensing cost.

Install SQL Server Integration Services Scale Out configuration

Since SQL Server 2017, SSIS supports a Scale Out configuration, by which you can run a package on the same or multiple SQL Server instances. This also allows for high availability of SSIS, and a similar architecture allows for integration and “lift and shift” code deployments from on-premises SSIS to the Azure Integration Runtime.

- Additional information on integration runtimes can be found in Chapter 19.

The master node talks to worker nodes in an SSIS Scale Out system, with the communication over a port (8391 by default) and secured via a new Secure Sockets Layer (SSL) certificate. The SQL Server installer can automatically create a 10-year self-signed certificate and endpoint for communication when the master node is set up.

When adding another SSIS installation as a Scale Out Worker, start the new SSIS Manage Scale Out window via SQL Server Management Studio. To do so, right-click the catalog you have created and select **Manage Scale Out**. At the bottom of the page, select the + button to add a new Scale Out Worker node.

Next, you provide the server name on which to connect. If using a named instance, provide only the server name of the node; do not include the instance name. A dialog box confirms the steps taken to add the Scale Out Worker node, including copying and installing certificates between the Worker node and Master node, updating the endpoint and `HttpsCertificateThumbprint` of the worker, and restarting the Worker node's Scale Out service.

After the worker node is added, refresh the **Worker Manager** page. Then select the new Worker node entry, which will be red. Finally, turn on the Worker node by selecting **Enable Worker**.

You also can copy and install the certificates manually between servers. You will find them in `%Program Files%\Microsoft SQL Server\160\DTS\Binn\`.

- For more information on certificates between servers, visit <https://learn.microsoft.com/sql/integration-services/scale-out/deal-with-certificates-in-ssis-scale-out>. For a Microsoft-provided walk-through of setting this up, visit <https://learn.microsoft.com/sql/integration-services/scale-out/walkthrough-set-up-integration-services-scale-out>.

One major security difference with Scale Out is that even though the SSIS service account doesn't run packages or need permission to do very much, the Scale Out Master and Worker service accounts *do* run packages. The SSIS service account is different from the Scale Out Master and Scale Out Worker service accounts.

The Worker and Master nodes do not appear in SQL Server Configuration Manager (as of SQL Server 2019) but do appear in the Services console (`services.msc`). By default, these services run under virtual accounts `NT Service\SSISScaleOutMaster160` and `NT Service\SSISScaleOutWorker160`, but you might want to change these to a Windows-authenticated domain service account that will be used to run packages across the Scale Out.

Install SQL Server Reporting Services

Starting with SQL Server 2017, SQL Server Reporting Services (SSRS) is no longer found in the SQL Server Setup media; it is instead available as a simplified, unified installer and a small download. SSRS is now a 95+MB download named `SQLServerReportingServices.exe` but still needs a SQL Server Database Engine instance as part of the license to host the two Report Server databases. Note that SSRS isn't free, and that the separate installer isn't a licensing change—although SQL Server Express with Advanced Services offers some limited SSRS support.

- For more information on the limitations of SSRS with SQL Server Express license, see <https://learn.microsoft.com/sql/reporting-services/reporting-services-features-supported-by-the-editions-of-sql-server-2016>.

To install SSRS, you need to provide a license key upon installation in a production environment. You can choose a free edition to install (Evaluation, Developer, or Express), but you should note that Developer edition is not allowed in a production environment.

The “native mode” of SSRS is now the only mode since SQL Server 2017. If you are familiar with Reporting Services Report Manager from the past, accessible via the URL *servername/Reports*, that is the “native mode” installation of Reporting Services.

Report Server Configuration Manager is in a new location, in its own Program Files menu: Microsoft SQL Server Reporting Services. After installation, start the Report Server Configuration Manager (typically installed in a path like `\Program Files (x86)\Microsoft SQL Server\160\Tools\Binn\RSConfigTool.exe`). The Report Server Configuration Manager application itself is largely unchanged since SQL Server 2008.

The default SSRS service account is the virtual service account called `NT SERVICE\SQLServerReportingServices`. It is a second-best option, however. We recommend that you instead create a new domain service account to be used only for this service—for example, `Domain\svc_ServerName_SSRS` or something with a similar naming convention. You will need to use a domain account if you choose to configure Report Server email with Report Server service account (NTLM) authentication.

If you choose to change the SSRS service account later, you must use the Reporting Services Configuration Manager tool. As with other SQL Server services, you should never use the Services console (`services.msc`) to change service accounts.

After installation, you will need to follow up on other changes and necessary administrative actions—for example, configuring the SSRS Execution Account and email settings or backing up the encryption key using Reporting Services Configuration Manager.

SSRS can also integrate with Microsoft Power BI dashboards. A page in the Report Server Configuration Manager supports the registration of this installation of SSRS with a Power BI account. You will be prompted to sign into Azure AD. The account you provide must be a member of the Azure tenant where you intend to integrate with Power BI. The account should also be a member of the system administrator in SSRS, via Report Manager, and a member of the `sysadmin` role in the SQL Server that hosts the Report Server database.

Inside OUT

Where is SSRS SharePoint Integrated mode?

Starting with SQL Server 2017, SharePoint Integrated mode has been removed. The simplified “native” mode is the only installation available. This matches the moves that Microsoft has made in other areas that step away from the SharePoint on-premises product in favor of SharePoint Online features and development.

Instead, you can integrate SSRS native mode with on-premises SharePoint sites via embedded SSRS reports, including SSRS reports stored in the Power BI Report Server.

Similarly, there is no future support for SSRS integration with SharePoint Online.

Install machine learning features

The Machine Learning Services feature makes it possible for developers to integrate with the R and Python language extensions using standard Transact-SQL (T-SQL) statements.

Data scientists can take advantage of this feature to build advanced analytics, data forecasting, and algorithms for machine learning. Data engineers can leverage these languages to integrate predictive analytic and machine learning. The scripts you create can be executed in-database without having to move data. You can prepare, clean, train, evaluate, perform feature engineering, and deploy machine learning models where the data resides. This eliminates the transfer of data across the network to another server.

Machine Learning Services is not a standalone feature. It requires a Database Engine instance. Also, it is now only available in the Instance Features section, and is no longer available in the Shared Features section.

Beginning with SQL Server 2022, runtimes for R, Python, and Java are no longer installed with SQL Setup. You must run the SQL Setup Wizard to install Machine Learning Services and Language Extensions. Then you must install your desired R, Python, or Java runtime(s) and packages.

You can install and use your open-source package and framework of choice, such as PyTorch, TensorFlow, and others. Machine Learning Services use an extensibility framework to run Python and R scripts.

NOTE

After installing your desired runtime(s), be sure to enable the external scripting feature using the following T-SQL command:

```
EXEC sp_configure 'external scripts enabled';
```

Then restart the SQL Server service.

- Note that there are separate Microsoft Docs articles for installation of Machine Learning Services on SQL Server 2019 and prior, and for SQL Server 2022. For installation on SQL Server 2022 on Windows, visit <https://learn.microsoft.com/sql/machine-learning/install/sql-machine-learning-services-windows-install-sql-2022>. For information about installing Machine Learning Services for SQL Server 2022 on Linux, see <https://learn.microsoft.com/sql/linux/sql-server-linux-setup-machine-learning-sql-2022>.

Availability groups are supported for Machine Learning Services, to ensure business continuity by configuring packages on each node, and failover cluster instances are supported from SQL Server 2019 onward.

You can execute Python and R scripts on a SQL Server instance with the stored procedure `sp_execute_external_script`.

You can find more details on each framework for this evolving feature in these Microsoft Docs articles:

- **Extensibility framework.** <https://learn.microsoft.com/sql/machine-learning/concepts/extensibility-framework>
- **Python extension.** <https://learn.microsoft.com/sql/machine-learning/concepts/extension-python>
- **R extension.** <https://learn.microsoft.com/sql/machine-learning/concepts/extension-r>

Install PolyBase Query Service for External Data

The PolyBase connector is a much-marketed feature for allowing native connectors for external data sources—even non-Microsoft or non-relational database platforms like Oracle, Teradata, and MongoDB.

Using PolyBase EXTERNAL tables, we can use SQL data types and T-SQL queries to seamlessly query data sources in-place in what Microsoft calls *data virtualization*. This eliminates the need for complex heterogeneous data movement and reduces the need for developers to have knowledge of other external query languages.

The PolyBase Query Engine feature is specifically designed for read and write queries on non-Microsoft database platforms like Oracle and DB2, but also for Azure Blob Storage files, MongoDB, and more. This is a superior alternative to linked servers to the same external data sources, because PolyBase allows “push down” computation for these external sources, reducing the amount of data transferred and increasing the performance of analytical-scale queries.

Install Azure extension for SQL Server

A new feature for SQL Server 2022 is extensibility for Azure features. This is in large part where the connections are initially set up for the features that make up the most Azure-connected

version of SQL Server to date. Let's look at the most common ones available so you understand what you are setting up.

Azure Arc-enabled servers

Azure Arc-enabled SQL Server instances are on-premises but still managed by Azure. This extends the services of Azure to the datacenter or wherever it is needed.

Azure Arc-enabled servers are supported only for the following operating systems:

- Windows Server 2012 R2 and higher
- Ubuntu 16.04 and 18.04 (x64)
- Red Hat Enterprise Linux (RHEL) 7 (x64)
- SUSE Linux Enterprise Server (SLES) 15 (x64)

NOTE

SQL Server instances on Azure Arc-enabled servers are not currently supported in Linux containers.

To perform all the actions needed to connect an Azure Arc-enabled server to Azure, you need an account with all of the following privileges:

- Microsoft.HybridCompute/machines/extensions/read
- Microsoft.HybridCompute/machines/extensions/write
- Microsoft.HybridCompute/machines/extensions/delete
- Microsoft.HybridCompute/machines/read
- Microsoft.HybridCompute/machines/write
- Microsoft.GuestConfiguration/guestConfigurationAssignments/read
- Microsoft.Authorization/roleAssignments/write
- Microsoft.Authorization/roleAssignments/read

To enable the services so that Azure Arc recognizes your instance, you need to register it for the services you want to take advantage of. There are a few steps to follow, both in Azure and on the server itself, for existing instances. Detailed instructions on how to do this can be found at <https://learn.microsoft.com/sql/sql-server/azure-arc/overview>.

Inside OUT

What are Azure Arc-enabled servers?

Azure Arc-enabled servers are servers that are managed by Azure but reside outside of Azure. These can reside on a private network, corporate network, or other public cloud. The experience is designed to be like how you would manage an Azure VM.

Azure Arc-enabled servers unlock additional features and advantages, including unified cloud manageability, but also, for example, the ability to use Azure AD-integrated authentication in on-premises SQL Servers. Azure Arc is a continuously evolving and developing technology, with new announcements arriving regularly.

Microsoft Defender for Cloud

Microsoft Defender for Cloud is a Cloud Security Posture Management (CSPM) and Cloud Workload Protection Platform (CWPP) that can be run in Azure but has been extended to on-premises and third-party clouds for multi-cloud opportunities with Azure Arc. The purpose of Defender is to assess, secure, and defend from threats. It does this by:

- Continuously assessing your security posture so you can identify opportunities, track vulnerabilities, and report
 - Securing resources and checking best practices to provide cloud recommendations
 - Defending from, detecting, alerting on, and resolving threats in real-time so you can prevent security events from happening
- For more detailed steps on setting up Microsoft Defender for Cloud, see the Microsoft Cloud Guide tutorial at <https://mslearn.cloudguides.com/guides/Protect%20your%20multi-cloud%20environment%20with%20Microsoft%20Defender%20for%20Cloud>.

Microsoft Defender is only supported for SQL Server on Windows machines and must have one of the RBAC roles assigned to it, as described in the next paragraph.

- Details on how to install Defender on your Azure Arc-enabled server for SQL Server can be found at <https://learn.microsoft.com/sql/sql-server/azure-arc/configure-advanced-data-security>.

Microsoft Defender for Cloud uses Azure role-based access control (RBAC)—a built-in set of roles assigned to users, groups, and services in Azure—to assess, manage, and access resources. Users require the Assignments role with write permissions, such as a User Access Administrator or Owner. You can access information related to a resource when you are assigned the role of Owner, Contributor, or Reader for the subscription or the resource's resource group.

Other built-in roles are specific to Microsoft Defender for Cloud:

- Security Reader users have viewing rights, which lets them view recommendations, alerts, security policies, and security states, but not make changes.
- Security Admin users have the same rights as Security Reader users but can also update the security policy, dismiss alerts and recommendations, and apply recommendations.
- ▶ For detailed instructions on how to assign roles in the Azure portal, see “Assign Azure roles using the Azure portal” at <https://learn.microsoft.com/azure/role-based-access-control/role-assignments-portal>.
- ▶ Find instructions for assigning administrator roles in Azure AD at <https://learn.microsoft.com/azure/active-directory/roles/manage-roles-portal>.

Azure AD Authentication

New with SQL Server 2022, you can authenticate SQL Server with Azure AD using the following methods:

- Azure AD Password
- Azure AD Integrated
- Azure AD Universal with Multi-Factor Authentication
- Azure AD access token

Azure AD support makes hybrid integrations with Azure Synapse Analytics, Azure SQL Managed Instance, Azure Arc, and other services easier. If your Windows Server AD is federated with Azure AD, users can use those credentials to sign into SQL Server. However, Azure AD authentication does not support service accounts or other complex architectures of AD.

Azure AD support requires that both SQL Server and the host server (Windows or Linux) be registered with Azure Arc.

- ▶ For more details on Azure AD, see Chapter 12.

Microsoft Purview

Microsoft Purview is a data-governance tool designed to support organizations in finding, understanding, governing, and consuming data stores. Microsoft Purview has been a cloud-first feature for some time, and has come to SQL Server on-premises with SQL Server 2022.

As with many other Azure hybrid features, SQL Server must be registered with Azure Arc to use Microsoft Purview. In addition, you will need to create a Microsoft Purview account and enable Azure AD.

- For information on creating a Microsoft Purview account, see <https://learn.microsoft.com/azure/purview/create-catalog-portal>.

CAUTION

Take care when assigning permissions for Microsoft Purview. There are inherent risks with the various admin roles, and these should be shared among different people in your organization. To prevent policies from being modified, you can use Azure Resource Manager (ARM) locking. More details on setting up Purview for an Azure Arc server are available at <https://learn.microsoft.com/azure/purview/how-to-data-owner-policies-arc-sql-server#configuration>.

Azure extension for SQL Server

To connect SQL Server to Azure Arc and take advantage of Microsoft Defender, Azure AD, and Microsoft Purview, you must install the Azure extension for SQL Server during SQL Server Setup on the Azure Extension for SQL Server page (see Figure 4-2). You can use your existing Azure credentials or an Azure Service Principal, and then complete the required fields such as Azure Research Group, Azure Region, and Azure Tenant ID. If you are not interested in connecting your SQL Server instance to Azure Arc, simply deselect the Azure Extension for SQL Server check box.

Figure 4-2 The Azure Extension for SQL Server page displays a number of required fields to enable Azure Arc features.

Log SQL Server Setup

SQL Server Setup generates many logging files for diagnostic and troubleshooting purposes. These logs should be the first place you go if you have an issue with Setup.

First, a System Configuration Check Report .htm file is generated each time you run Setup. You can view this report in SQL Server Setup near the start of the installation steps.

A new timestamp-named folder of log files is generated for each launch of SQL Server Setup. After you proceed past the **Ready to Install** page, and regardless of whether Setup was a complete success, it generates a number of log files in the following folder:

```
%programfiles%\Microsoft SQL Server\150\Setup Bootstrap\Log\YYYYMMDD_HHMMSS\
```

However, when you run Setup using the /Q or /QS parameters for unattended installation, the log file is written to the Windows %temp% folder.

A log summary file of the installation is created that uses the following naming convention:

```
Summary_instancename_YYYYMMDD_HHMMSS.txt
```

Setup generates similar files for the Component and Global Rules portions of Setup, as well as a file called Detail.txt in the same folder. These files might contain the detailed error messages you are looking for when troubleshooting a failed installation. The Windows Application Event log might also contain helpful information in that situation.

You'll also find the new SQL Server instance's first error log encoded at UTC time in this folder, showing the log from startup, similar to the normal SQL Server Error Log.

Automate SQL Server Setup with configuration files

Let's dig more into what you can do with setup.exe outside of the user interface. You can use configuration files to automate the selection process when installing SQL Server, which helps to create a consistent configuration.

Values provided in configuration files can prepopulate or override Setup settings. They also can configure Setup to run with the normal user interface or silently without any interface.

Start SQL Server Setup from the command line

You can start `setup.exe` from either Windows PowerShell or the command prompt, providing repeatability and standardization of parameter options. You also can use it to prefill sections of the Setup wizard or to change the default behavior of Setup.

For the purposes of the installer, ensure you always use the Administrator level for these two shells. The title on each application window should be preceded by *Administrator:*—for example, Administrator: Windows PowerShell.

Sometimes you also might find it necessary to start Setup from the command line or Windows PowerShell because of a workaround for a specific problem or to automate and standardize future SQL Server installations. To start Windows PowerShell or the command prompt as Administrator, in the **Start** menu, search for the desired application, right-click it, and then select **Run As Administrator** on the shortcut menu that opens.

From the location of the SQL Server Setup installation files—for example, the mounted .iso file—execute the following command with PowerShell or the Windows Prompt:

```
.\setup.exe /ConfigurationFile=c:\install\SQL2019_basic.INI
```

This sample script, and all scripts for this book, are available for download at <https://www.MicrosoftPressStore.com/SQLServer2022InsideOut/downloads>. The preceding code sample uses a configuration file to pre-select installation choices—for example, features to be installed. Let's talk more about configuration files.

Generate a configuration file

Writing a configuration file by hand is not necessary, and can be tedious. Instead of going through that effort, you can let SQL Server Setup create a configuration file for you.

Work your way through the normal SQL Server Setup user interface, completing everything as you normally would, but pause when you get to the **Ready to Install** page. Near the bottom of this page is a path (see Figure 4-3). At that location, you'll find a generated configuration file, ready for future use and modification if needed.

For example, the first modification you need to make to the .ini file is to accept the SQL Server license terms via the `IACCEPTSQLSERVERLICENSETERMS` parameter, which isn't automatically provided in the automatically generated .ini file. Unless you modify an .ini file to provide this, it isn't possible to run the installer without user interaction.

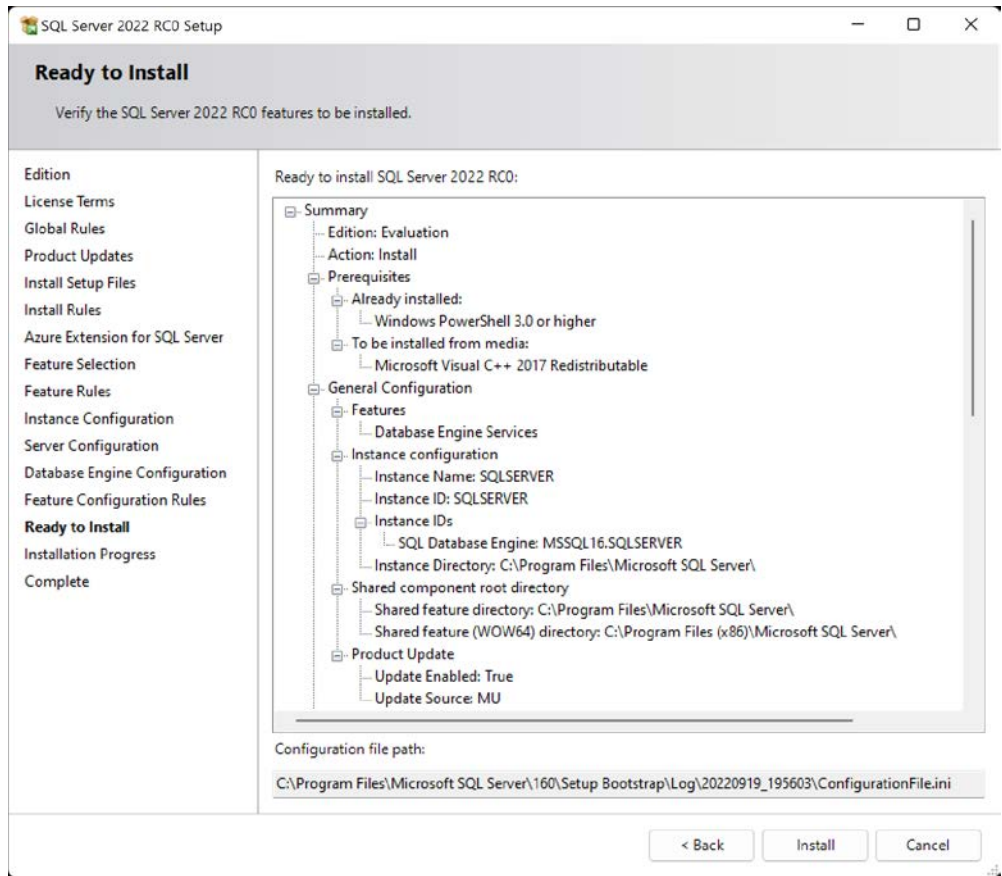


Figure 4-3 The Ready to Install page displays a summary of the installation steps as well as the path to the configuration file that has been prepared.

Install by using a configuration file

Now that you have a configuration file generated using the previous walk-through, you can take the next step to automate or standardize your installation.

You can start `setup.exe` from a command prompt with a configuration file by using the `/CONFIGURATIONFILE` parameter of `setup.exe`. Or you can launch SQL Server Setup with a configuration file by navigating to the **Advanced** page of the SQL Server Installation Center that starts with `setup.exe` in Windows. Once there, select the **Install Based On A Configuration File** check box. A message appears, asking you to browse to the `.ini` file. After you select the appropriate file, `setup.exe` will start with those options.

One thing to keep in mind is that configuration files generated by `setup.exe` do not and should not store the passwords you provided for any service accounts. If you do want to configure

service account credentials in your configuration file, for security reasons, do not store the service account passwords in plain text in a configuration file. Instead, store passwords separately and securely, and provide them when you run setup.exe.

Each service's account parameters are available in a setup.exe runtime parameter, which is listed in Table 4-1.

Table 4-1 Common setup.exe parameters and their purposes

Service	Parameter name	Description
SQL Server Database Engine	/SQLSVCPASSWORD	Password for the SQL Server Database Engine Services service account. This is the service account for sqlservr.exe. It is required if a domain account is used for the service.
SQL Server Agent	/AGTSVCPASSWORD	Password for the SQL Server Agent service account. This is the service account for sqlagent.exe. It is required if a domain account is used for the service.
sa password	/SAPWD	Password for the sa account. It is required when /SECURITYMODE=SQL is used, which enables Mixed Mode authentication.
Integration Services	/ISSVCPASSWORD	Password for the Integration Services service. It is required if a domain account is used for the service.
Reporting Services (Native)	/RSSVCPASSWORD	Password for the Reporting Services service. It is required if a domain account is used for the service.
Analysis Services	/ASSVCPASSWORD	Password for the Analysis Services service account. It is required if a domain account is used for the service.
PolyBase	/PBDMSVCPASSWORD	Password for the PolyBase service account.
Full-Text filter launcher service	/FTSVCPASSWORD	Password for the Full-Text filter launcher service.

For example, in the snippet that follows, the PROD_ConfigurationFile_Install.INI provides the account name of the SQL Server Database Engine service account, but the password is provided when setup.exe runs in the command prompt or PowerShell:

```
setup.exe /SQLSVCPASSWORD="securepwd" /ConfigurationFile="d:\SQL\PROD_Install.INI"
```

You can provide further parameters like passwords when you run Setup. Parameter settings provided override any settings in the configuration file, just as the configuration file's settings override any defaults in the Setup operation. Table 4-2 lists and describes the parameters.

Table 4-2 Common setup.exe parameters of which you should be aware

Parameter usage	Parameter	Description
Unattended installations	/Q	Specifies Quiet Mode with no user interface and user interactivity allowed.
Unattended installations	/QS	Specifies Quiet Mode with user interface but no user interactivity allowed. Will fail if all needed information or parameters are not provided.
Accept license terms	/IACCEPTSQLSERVERLICENSETERMS	Must provide in any configuration file looking to avoid prompts for installation.
R open license terms	/IACCEPTROPENLICENSETERMS	Must provide for any unattended installation involving the R language option for Machine Learning Services.
Python open license terms	/IACCEPTPYTHONLICENSETERMS	Must provide for any unattended installation involving the Python language option for Machine Learning Services.
Instant file initialization	/SQLSVCINSTANTFILEINIT	Set to true to grant Perform Volume Maintenance Task privilege to the Database Engine service account (recommended).
Windows accounts to provision as members of the sysadmin role	/SQLSYSADMINACCOUNTS	Must provide groups or service accounts to specify as the initial members of the sysadmin role.
Provision the user running SQL Server Setup as a member of the sysadmin role	/ADDCURRENTUSERASSQLADMIN	If desired, specify the current local Windows Server user running SQL Server Setup as an initial member of the sysadmin role. Not desired if using a personal named account; use a group instead.
tempdb data file count	/SQLTEMPDBFILECOUNT	Set to the number of desired tempdb data files to be installed initially.
Enable TCP/IP	/TCPENABLED="1"	Disabled by default and used in many installations. Enable TCP/IP here to save yourself a step in Configuration Manager later on.

By default, the `/UpdateEnabled` parameter is enabled and doesn't need to be specified, and SQL Server will include updates found via Windows Update. If you choose to disable this behavior by providing `/UpdateEnabled=False`, you can also specify `/UpdateSource` as the location of the cumulative update or other SQL patch file executables to be included in the installation.

SQL Server on Azure virtual machines

Azure options are continuously evolving, making it hard to comprehensively cover them in any one book. SQL Server 2022 is touted as the most Azure-connected version to date. It is Microsoft's way of bringing you hybrid flexibility from ground to cloud, so it is worthwhile covering some of those intersections here.

At the time of writing, there are three options:

- **Azure VMs.** VMs hosted in Azure. They function very similarly to VMs in your on-premises environment, except they are hosted in Azure. You have the same responsibilities for protection and management, but with the utilities and services of Azure at your disposal.
- **SQL Server on Azure VMs.** Azure VMs with a preset configuration of SQL Server you choose based on your desired workload. The default workload environment is production, but there are options for dev/test as well. You can choose between different performance tiers; some focus on CPU-intensive workloads, while others focus on memory-optimized workloads, with variations in between. These tiers provide a wide selection of virtual hardware to run enterprise applications, relational databases, analytics, in-memory workloads, and intensive batch processing.
- **Azure Arc VMs.** VMs that can be created in one of your non-Azure environments. Typically, this is on-premises, but it could also be another cloud provider, public or private.

Post-installation server configuration

After you install SQL Server, there are several changes to make or confirm on the OS and in settings for SQL Server.

Post-installation checklist

You should run through the following checklist on your new SQL Server instance. The order of these items isn't necessarily specific. Many deal with SQL Server and/or Windows configuration settings. You want to evaluate whether these are appropriate for your environment, but you should consider and apply them to most SQL Server installations.

- Check your SQL Server patch level version and apply patches if necessary.
- Review maximum server memory settings for other features.

- Review the surface area configuration facet.
- Set up SQL Agent.
- Turn on TCP/IP if needed.
- Verify server power options.
- Configure antivirus exclusions for SQL Server processes and files.
- Evaluate whether Lock Pages in Memory is necessary.
- Review the size and location of the Windows page file.
- Set up scheduled backups, index maintenance, log retention maintenance, and integrity checks.
- Back up service master and database master keys.
- Increase SQL Agent and SQL Error Log retention from the defaults.
- Suppress successful backup messages.
- Increase default SQL Agent history retention.

Let's look at each of these in more detail in the following subsections.

Check your SQL Server patch level version and apply patches if necessary

After you install SQL Server, check the version number against the latest cumulative updates list—especially if you did not opt to or could not use Windows Update during SQL Server Setup. You can view the version number in SQL Server Management Studio's Object Explorer or via a T-SQL query on either of the following built-in functions:

```
SELECT @@VERSION;  
SELECT SERVERPROPERTY('ProductVersion');  
SELECT SERVERPROPERTY('Edition');
```

While you're at it, double-check that you installed the right edition of SQL Server, too!

NOTE

Take the opportunity before your SQL Server enters production to patch it. For information about the latest cumulative updates for SQL Server, visit <https://learn.microsoft.com/troubleshoot/sql/general/determine-version-edition-update-level#sql-server-complete-version-list-tables>, and select your version or build.

Review maximum server memory settings for other features

Other features of SQL Server have their own maximum server memory settings. As you will notice by their default settings, for servers on which both the Database Engine and SSAS and/or SSRS are installed, competition for and exhaustion of memory is possible. It is recommended that you protect the Database Engine by lowering the potential memory impact of other applications.

Limit SSAS memory

SQL Server Analysis Services (SSAS) has not just one maximum server memory limit, but five, and you can enforce limits by hard values in bytes or by a percentage of total physical memory of the server.

You change these memory settings via SSMS by connecting to the SSAS instance in Object Explorer. To start, right-click the server and select **Properties** on the shortcut menu. Some of the memory settings described here are identical for Multidimensional and Tabular installations, whereas others are for Tabular mode only:

- **LowMemoryLimit.** A value that serves as a floor for memory, but also the level at which SSAS begins to release memory for infrequently used or low-priority objects in its cache. Below this level, no memory maintenance is performed by SSAS. The default value is 65, or 65 percent of total server physical memory (or technically the virtual address space, but SSAS, among other features, is no longer supported on 32-bit systems, so this is not a concern).
- **TotalMemoryLimit.** A value that serves as a threshold for SSAS to begin to release memory for higher priority requests. This is not a hard limit. The default is 80 percent of total server memory.
- **HardMemoryLimit.** A hard memory limit that leads to more aggressive pruning of memory from cache and potentially to the rejection of new requests. By default, this is displayed as 0, and is effectively the midway point between the TotalMemoryLimit and the server physical memory. The TotalMemoryLimit must always be less than the HardMemoryLimit.
- **VertiPaqMemoryLimit.** For SSAS installations in Tabular mode only, a value that serves as a memory limit for the VertiPaq processing engine. The default is 60, or 60 percent of server physical memory. Above this percentage, and only if VertiPaqPagingPolicy is turned on (it is by default), SSAS begins to page data to the hard drive using the OS page file. Paging to a drive can help prevent out-of-memory errors when the HardMemoryLimit is met.
- **QueryMemoryLimit.** A value that can limit the amount of memory used by individual DAX queries, preventing any one query from dominating memory. For any individual query, this setting can be overridden by a new XMLA property, DbpropMsmRequestMemoryLimit, specified for the query connection. This setting can be specified as a percentage (values ≤ 100) or as a number of bytes greater than 100. The default setting of 0 implies no limit to the memory of individual queries.

Figure 4-4 shows the General page of the Analysis Server Properties dialog box, as started in Object Explorer in SSMS, and the locations of the preceding memory configuration properties with their defaults in SQL Server 2019 for a Tabular mode installation of SSAS. Note that the Show Advanced (All) Properties check box is checked.

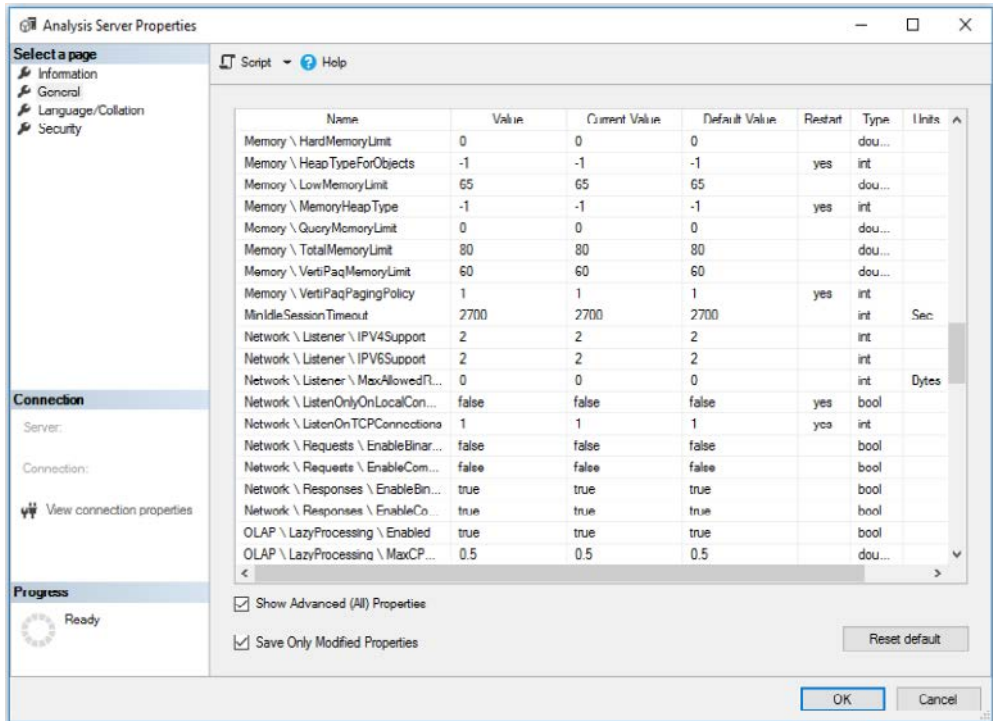


Figure 4-4 The General page in the Analysis Server Properties dialog box showing the default settings.

Limit SSRS memory

Four options are available for limiting SQL Server Reporting Services (SSRS) memory: `MemorySafetyMargin`, `MemoryThreshold`, `WorkingSetMaximum`, and `WorkingSetMinimum`. All four are based on numbers contained in tags within a config file, so be sure to make a backup of it before editing. You can configure memory settings only in the `RSReportServer.config` file, which is a text file stored at `%ProgramFiles%\Microsoft SQL Server Reporting Services\SSRS\ReportServer`.

NOTE

This location has changed from previous versions, but the config file name has not.

Two of the settings are in the config file by default; two more are available to administrators to use in advanced scenarios. Let's look at each one:

- **MemorySafetyMargin.** The percentage of `WorkingSetMaximum` that SSRS will use before taking steps to reduce background task memory use and prioritize requests coming from the web service, attempting to protect user requests. User requests could still be denied.
- **MemoryThreshold.** The percentage of `WorkSetMaximum` at which SSRS will deny new requests, slow down existing requests, and page memory to a hard drive until memory conditions improve.

Two more settings are given values automatically upon service startup, but you can override them in the config file. Two older memory settings from SQL Server 2005 with which SQL DBAs might be familiar are `MemoryLimit` and `MaximumMemoryLimit`, but these two values have been ignored since SQL Server 2008.

- **WorkingSetMaximum.** By default, this is the server's total physical memory. This setting does not appear by default in the config file, but you can override it to reduce the amount of memory of which SSRS will be aware. This value is expressed in kilobytes of memory.
- **WorkingSetMinimum.** By default, this value is 60 percent of `WorkingSetMaximum`. If SSRS needs memory below this value, it will use memory and not release it due to memory pressure. This setting does not appear by default in the config file, but you can override it to increase the variability of SQL SSRS's memory use.

These four settings can appear in the `rsreportserver.config` file. As demonstrated here, you could override the default settings to 4 GB maximum and 2 GB minimum (each expressed in KB):

```
<MemorySafetyMargin>80</MemorySafetyMargin>
<MemoryThreshold>90</MemoryThreshold>
<WorkingSetMaximum>4194304</WorkingSetMaximum>
<WorkingSetMinimum>2097152</WorkingSetMinimum>
```

Limit Machine Learning Server memory

Like SSAS and SSRS, the Machine Learning Server has a config file at `%ProgramFiles%\Microsoft SQL Server\MSSQL15.instance\Binn\mlauncher.config`.

By default, Machine Learning Server is similar to 20 percent of total server memory. You can override this by adding a tag to the config file to provide a value for `MEMORY_LIMIT_PERCENT`. This value is not in the config file by default.

Remember to make a backup of this config file before editing. The following is an example of the contents of the `rlauncher.config` file, with the default memory limit changed to 25 percent:

```
RHOME=C:\PROGRA~2\MICROS~1\MSSQL1~4.SQL\R_SERV~2
MPI_HOME=C:\Program Files\Microsoft MPI
INSTANCE_NAME=SQL2K22
TRACE_LEVEL=1
JOB_CLEANUP_ON_EXIT=1
USER_POOL_SIZE=0
WORKING_DIRECTORY=C:\Program Files\Microsoft SQL
Server\MSSQL16.SQL2K22\MSSQL\ExtensibilityData
PKG_MGMT_MODE=0
MEMORY_LIMIT_PERCENT=25
```

Review the surface area configuration

If you are a veteran SQL Server DBA, you will remember when the SQL Server Surface Area Configuration was a separate application. Now, surface area settings are a facet, accessed via the Facets dialog box in SSMS starting with SQL Server 2008.

To view surface area configuration settings in SSMS, open Object Explorer, connect to the SQL Server, right-click the server, and select **Facets** on the shortcut menu. (The Facets window sometimes takes a moment to load.) Then, in the dialog box that opens, change the value in the list box to **Surface Area Configuration**.

Most of these options should remain off unless needed because they present a specific potential for misuse by an administrator or unauthorized user. In typical installations of SQL Server 2022, however, you should consider enabling three of these options:

- **Database Mail.** This should be enabled on most instances to allow SQL Server to, at the very least, send out a message in case of a high-severity incident or job failure, and to allow developers to send custom email messages using the system procedure `sp_send_dbmail`. You also can turn this setting on or off via the Database Mail XPs option in `sp_configure`. (More about this setting in Chapter 9, “Automate SQL Server administration.”)
- **Remote Dedicated Admin Connection.** This could be particularly useful for bypassing a malfunctioning login trigger or Resource Governor. You also can turn this setting on or off via the remote admin connections option in `sp_configure`. (More on this setting in Chapter 13: “Protect data through classification, encryption, and auditing.”)
- **CLR Integration.** Turn on if you need to use SSIS or to write CLR objects. You also can turn this setting on or off via the `clr_enabled` option in `sp_configure`.

You should turn on other options in the Surface Area Configuration only if they are specifically required by an application and you are aware of the potential security concerns.

Set up SQL Agent

There are several post-installation tasks to set up in SQL Agent before SQL Server can begin to help you automate, monitor, and back up your new instance.

- Chapter 8, “Maintain and monitor SQL Server,” and Chapter 9 cover SQL Agent and monitoring topics in detail.

You will likely want to do the following:

1. Change the SQL Agent service from Manual to Automatic startup.
2. Set up a Database Mail account and profile (see Chapter 9) to send email notifications for alerts or job status notifications.
3. Set up an operator for a distribution group of IT professionals in your organization who would respond to a SQL Server issue.
4. Configure SQL Server Agent to use Database Mail, including a fail-safe operator.
5. Set up SQL Server Alerts for desired errors and high severity (Severity 21+) errors.

At the very least, these steps are put in place so that SQL Server can send out a call for help. Even if you have centralized monitoring solutions in place, the most rare and severe of errors should be important enough to warrant an email.

You can choose to configure many Windows Management Instrumentation (WMI) conditions, Performance Monitor counter conditions, and SQL Server Error messages by number or severity in SQL Server Alerts. However, do not overcommit your inboxes, and do not set an inbox rule to Mark As Read and file away emails from SQL Server. By careful selection of emails, you can assure yourself and your team that emails from SQL Server will be actionable concerns that rarely arrive.

- For much more information on maintaining and monitoring SQL Server, see Chapter 8.

Turn on TCP/IP if needed

Depending on the edition you have installed, the common network protocol TCP/IP is off by default. The only protocol that *is* on is Shared Memory, which allows only local connections. You will likely not end up using Shared Memory alone to connect to the SQL Server for common business applications that use multiple servers for database, web, and application tiers.

NOTE

It is possible to enable TCP/IP by default at the time of installation if using a configuration file for SQL Server Setup, but this option does not appear in the UI for SQL Server Setup. It must be changed after installation is complete.

When you connect to SQL Server using SSMS while local to the server, you connect to the Shared_Memory endpoint whenever you provide the name of the server, the server\instance, localhost, dot character (.), (local), .\instance, or (local)\instance.

TCP/IP, however, is ubiquitous in many SQL Server features and functionality. Many applications will need to use TCP/IP to connect to the SQL Server remotely. Many SQL Server features require TCP/IP to be enabled, including the Remote Dedicated Admin Connection (DAC), the availability groups listener, and Kerberos authentication.

To configure TCP/IP, open the SQL Server Configuration Manager application locally on the server. Then, in the left pane, select **SQL Server Network Configuration**. Browse to the protocols for your newly installed instance of SQL Server. The default instance of SQL Server, here and in many places, will appear as **MSSQLSERVER**.

You can also enable TCP/IP for a SQL Server instance with PowerShell:

```
Import-Module SqlServer
$wmi = new-object('Microsoft.SqlServer.Management.Smo.Wmi.ManagedComputer')
#Path to the local server
$path = "ManagedComputer[@Name='$env:COMPUTERNAME']/"
$path = $path+"ServerInstance[@Name='SQL2K22']/ServerProtocol[@Name='Tcp']"
#Enable the TCP protocol on the local server, on the named instance SQL2K22
$TCPIP = $wmi.GetSmoObject($path)
$TCPIP.IsEnabled = $true
$TCPIP.Alter()
$TCPIP.IsEnabled
#Restart SQL Server Database Engine service to apply the change
```

After turning on TCP/IP, regardless of what method you use, you need to restart the SQL Server Database Engine service for it to take effect.

NOTE

Turning on Named Pipes is not required or used unless an application specifically needs it.

Verify server power options

The Windows Server Power Options setting should be set to High Performance for any server hosting a SQL Server instance.

In other power plans, Windows might not operate the processor at maximum frequency during normal or even busy periods of SQL Server activity. This applies to physical or virtual Windows servers.

Review this setting and ensure that the group policy will not change it back to Balanced or another setting. Also ensure that group preferences are configured with High Performance selected for new SQL Servers. Finally, you may also need to check that the BIOS is also configured for High Performance.

Configure antivirus exclusions for SQL Server processes and files

Configure any antivirus software installed on the SQL Server to ignore scanning files with extensions used by your SQL Server data and log files. Typically, these will be MDF, LDF, and NDF files.

Also, configure any antivirus programs to ignore folders containing SQL Server files. These could include:

- Full-text catalog files
- Backup files
- Replication snapshot files
- SQL Server trace (TRC) files
- SQL Audit files
- Analysis Services database
- Log and backup files
- FILESTREAM and FileTable folders
- SSRS temp files and log files

Processes might also be affected, so set antivirus programs to ignore the programs for all instances of the SQL Server Database Engine service, Reporting Services service, Analysis Services service, and R Server (RTerm.exe and BxlServer.exe).

In SQL Server FCIs and availability groups, configure antivirus software to exclude the MSCS folder on the quorum drive if in use, the MSDTC directory on the MSDTC share, and the Windows\Cluster folder on each cluster node, if they exist.

Inside OUT

What if you suspect antivirus or antimalware software is interfering with SQL Server?

This is one of the more challenging troubleshooting exercises: a strange error message, DLL error, or file accessibility issue. It is critical to configure antivirus to exclude SQL Server files and folders from on-access scans, exclusive-lock scans, and more.

If you notice, for example, random databases failing to recover upon SQL Server startup, or error messages like “File activation failure” or “Unable to open the physical file,” sqlservr.exe may not be able to gain exclusive access to the files because they are being scanned by another application. Use the Windows Sysinternals Process Explorer application to search for handles, including your SQL Server files, and potentially catch that other

application accessing the file. Download the Sysinternals Process Explorer at <https://aka.ms/processexplorer>.

Antivirus applications may also interfere with service packs and cumulative updates if those files, even if they are signed by Microsoft, have not been pre-approved for execution in the production environment. Communicate with the teams that control antivirus, antiransomware, or antimalware solutions in your enterprise.

Optimize for ad hoc workloads

The server-level setting Optimize for Ad Hoc Workloads doesn't have the most intuitive name. We are not optimizing ad hoc queries; we are optimizing SQL Server memory usage to prevent ad hoc queries from consuming unnecessary cache.

- For more about the Optimize for Ad Hoc Workloads setting, see Chapter 3.

For the unlikely scenario in which a large number of queries are executed only two times, setting this option to **True** would be a net negative for performance. Enabling this setting can also affect performance tuning for single-use queries.

- For more about cached execution plans, read Chapter 14.

Evaluate whether Lock Pages in Memory is necessary

Consider using the Lock Pages in Memory policy for environments in which instances of SQL Server are expected to experience memory pressure due to other applications, server limitations, or overallocated virtualized systems. This is an in-depth topic to be carefully considered.

- For more about the Lock Pages in Memory setting, see Chapter 2, "Introduction to database server components."
- For more about the Windows page file, see Chapter 3.

Inside OUT

How can you tell if the Lock Pages in Memory policy is in effect?

Starting with SQL Server 2016 SP1, you can check whether the Lock Pages in Memory policy has been granted to the Database Engine using the following query:

```
SELECT sql_memory_model_desc
FROM sys.dm_os_sys_info;
--CONVENTIONAL = Lock pages in memory privilege is not granted
--LOCK_PAGES = Lock pages in memory privilege is granted
--LARGE_PAGES = Lock pages in memory privilege is granted in Enterprise mode
-- with Trace Flag 834 ON
```

Review the size and location of the Windows page file

The page file is used to page out system memory. It can also capture a system memory dump for crash forensic analysis, a factor that dictates its size on modern operating systems with large amounts of memory. Therefore, the general recommendation for the system page file is that it should be at least the same size as the server's amount of physical memory. This is also why the page file is best moved to its own volume, away from the OS volume, so that it does not unexpectedly grow and create space issues.

- For more guidance on the operating system page file, see the section “Configure the operating system page file” in Chapter 3.

Set up scheduled backups, index maintenance, log retention maintenance, and integrity checks

Backups are a critical part of disaster recovery. They should begin as soon as possible after installation, and before users or applications begin to use the SQL Server.

Generate database backups, at least of the master and msdb system databases, right away. You should also back up other SQL Server Setup–created databases, including ReportServer, ReportServerTempDB, and SSISDB, as soon as possible.

- For more information on backups, index maintenance, and monitoring, see Chapter 11.

As soon as your new SQL Server instance has databases in use, regularly perform selective index maintenance and integrity checks that consider the current fragmentation levels of indexes rather than performing index maintenance on entire databases. In many cases, statistics maintenance may be more effective in the shorter term.

- For more information on automating maintenance, see Chapter 9.

Back up service master and database master keys

You should back up service master keys and any database master keys as they are created, securely storing their information.

- For more information on service master and database master keys, see Chapter 13.

To back up the instance service master key, use the following command:

```
BACKUP SERVICE MASTER KEY TO FILE = 'localfilepath_or UNC'
ENCRYPTION BY PASSWORD = 'complexpassword'
```

As soon as database master keys come into existence in each user database—for example, as you implement features like transparent data encryption (TDE) or column data encryption, back up individual database master keys as follows:

```
BACKUP MASTER KEY TO FILE = 'localfilepath_or UNC' ENCRYPTION BY PASSWORD =
'complexpassword'
```

If you implement TDE, Always Encrypted, native backup encryption, column encryption, or any other native or external solutions that generate certificates, keys, and/or passwords, develop a secure storage and retrieval method inside your enterprise. Failure to back up master and database master keys could compromise future disaster recovery attempts!

Increase SQL Agent and SQL error log retention from the defaults

By default, SQL Server maintains the current SQL Server Error Log plus six more historical error logs. Logs are cycled each time the SQL Server service is started.

One fun weekend of server troubleshooting or maintenance where the SQL Server service is restarted many times could wipe out a significant amount of your error history. This could make the task of troubleshooting periodic or business cycle–related errors difficult or impossible.

You need visibility into errors that occur only during a monthly processing, monthly patch day, or periodic reporting, for example. Follow these steps:

1. In SQL Server Management Studio, in Object Explorer, connect to the SQL Server instance.
2. Expand the **Management** folder, right-click **SQL Server Logs**, and select **Configure** in the shortcut menu.
3. Select the **Limit the Number of Error Logs Before They Are Recycled** check box.
4. For the **Maximum Number of Error Log Files** setting, type a value larger than 6. You might find that a value between 25 and 50 will result in more useful log history contained for multiple business cycles.

On SQL Server instances that generate a large amount of log noise, consider other options to reduce the clutter of the SQL Server Error Log, including Trace Flag 3226 to suppress the logging of successful backup operations. (Much more on this in the next section.)

You may also choose to configure a SQL Agent Job to manually cycle the SQL Server Error Log using `sp_cycle_errorlog` so that no one log file contains so much data it becomes unwieldy for scan and analysis. Consider scheduling `sp_cycle_errorlog` to execute weekly, and keep 50 SQL Agent error jobs, leaving at most 50 weeks of history.

Suppress successful backup messages

By default, SQL Server writes an event to the SQL Server error log upon a successful database backup, whether it be FULL, DIFFERENTIAL, or TRANSACTION LOG.

On instances with many databases, and with many databases in the full recovery model with regular transaction log backups, the amount of log activity generated by just their successful frequent log backups could flood the log with clutter, lowering log history retention.

NOTE

You can review successful backup history by querying the `msdb` system database. It has a series of tables dedicated to storing the backup history for all databases, including `msdb.dbo.backupset` and `msdb.dbo.backupmediafamily`. The built-in “Backup and Restore Events” report in SQL Server Management Studio provides access to this data, as well.

- For more on backups, see Chapter 10, “Develop, deploy, and manage data recovery.”

SQL Server Trace Flag 3226 controls an option at the instance level to suppress successful backup notifications.

There are many trace flags available to administrators to alter default behavior—many more options than there are user interfaces to accommodate them in SQL Server Management Studio. Take care when turning them on and understand that many trace flags are intended only for temporary use when aiding troubleshooting. Because Trace Flag 3226 is intended to be a permanent setting, simply enabling the trace flag by using `DBCC TRACEON` is not sufficient, as the trace flag will no longer be active following a SQL Server service restart. Instead, add the trace flag as a startup parameter to the Database Engine service by using SQL Server Configuration Manager. In the **Properties** of the SQL Server service, go to the **Startup Parameters** tab, and use the syntax `-Tflagnumber`. This field is essentially adding parameters that are passed to the `sqlserver.exe` executable. For example, enter **-T3226**, then select **Add**. The change will not take effect until the SQL Server Database Engine service is restarted.

- For more information on SQL Server Configuration Manager, refer to Chapter 1, “Get started with SQL Server tools.”

Increase default SQL Agent history retention

Similarly, you might find that the SQL Server Agent history is not sufficient to cover an adequate amount of job history, especially if you have frequent job runs.

You can use SSMS to change the job history settings for SQL Server Agent. In Object Explorer, connect to the SQL Server instance. Then right-click **SQL Server Agent**, select **Properties** from the shortcut menu, and select the **History** page.

This page is not intuitive and can be confusing. The first option, **Limit Size of Job History Log**, is a rolling job history retention setting. Consider adding zeros to increase the maximum log history size in rows from the default of 1,000 to 10,000 or more, and increasing the maximum job history per job in rows from the default of 100 to 2,000 or more. This data is stored in the `msdb` system database and will cause that database to grow larger over time. Consider pre-allocating some additional file space to the `msdb` data file now.

Heads up: The second option on the **History** page, **Remove Agent History**, along with its corresponding **Older Than** text box, is *not* a rolling job history retention setting. Rather, it is

an immediate and manual job history pruning. Select this second check box and select **OK**. When you return to the **History** page, you will find the second check box is cleared. Behind the scenes, SQL Server Management Studio immediately ran the `msdb.dbo.sp_purge_jobhistory` stored procedure to remove the job history once.

Post-installation configuration of other features

SQL Server Database Engine installation is now complete, but three other features require post-installation configuration: SSIS, SSRS, and SSAS. You will need to perform the steps detailed in this section before use if these features are installed.

SSISDB initial configuration and setup

Among the best features added by SQL Server 2012 were massive improvements to SSIS—specifically a new server-integrated deployment, built-in performance data collector, environment variables, and more developer quality-of-life improvements. For these reasons, you should use the new Project Deployment Model and built-in SSISDB for all new development.

When the Integration Services Catalog is created, a new user database called SSISDB is also created. You should back it up and treat it as an important production database.

You must create the SSISDB catalog soon after installation and before an SSIS development can take place. You will need to create the catalog only once. Because this involves potential surface area configuration changes and the creation of a new strong encryption password, a SQL DBA, not an SSIS developer, should perform this step and should store the password securely alongside others generated at the time of installation.

To create the catalog, in Object Explorer, connect to your instance, right-click **Integration Services Catalog**, and select **Create Catalog** in the shortcut menu that appears. In the single-page setup window, select the **Enable CLR Integration** check box, decide whether SSIS packages should be allowed to be run at SQL Server Startup (we recommend this due to the maintenance and cleanup performed then), and provide an encryption password for the SSISDB database.

The encryption password is for the SSISDB database master key. After you create it, you should back up the SSISDB database master key and securely store the SSISDB database password where it can be retrieved along with other disaster-recovery information for this server.

► For more on database master keys, see Chapter 13.

The SSISDB database will contain SSIS packages and their connection strings. The SSISDB encryption would not allow these sensitive contents—a treasure trove of connections to other servers—to be decrypted by a malicious user who gains access to the database files or backups. This SSISDB password will be required if the database is restored to a new server, so you should store it in a secure location within your enterprise.

NOTE

If you receive an error when creating the SSSIDB catalog that reads “The catalog backup file ‘C:\Program Files\Microsoft SQL Server\150\DTS\Binn\SSISDBBackup.bak’ could not be accessed” or similar, it is probably because SSIS was not actually installed. Most likely, the 6-MB template database backup was not copied from the SQL Server media, probably because the SSIS feature was not a selected feature. To rectify this, you can run SQL Server Setup again or copy the SSISDBBackup.bak file from another SQL Server installation of the same version.

SQL Server Reporting Services initial configuration and setup

There are still tasks to perform upon first installation of an SSRS native-mode installation from the downloaded installer file, `SQLServerReportingServices.exe`.

- **Get the latest installer and see what’s new in SSRS** at <https://learn.microsoft.com/sql/reporting-services/what-s-new-in-sql-server-reporting-services-ssrs>.

At the end of the Microsoft SQL Server 2022 Reporting Services installer wizard, on the Setup Completed screen, select the **Configure Report Server** button to open the Reporting Services Configuration Manager application. Then connect to the newly installed SSRS instance and review the following options, from top to bottom:

- **Service Account.** You can change the SSRS service account here. Remember that you should use only the Reporting Services Configuration Manager tool to make this change, never `services.msc`.
- **Web Service URL.** The web service URL is not for user interaction; rather, it is for Report Manager and custom applications to programmatically connect to the SSRS instance.

By default, a web service on TCP Port 80 is created called `ReportServer`. For named instances, the web service will be called `ReportServer_instancename`. The URL for the webservice would then be:

`servername/ReportServer`

or:

`servername/ReportServer_instancename`

To accept defaults, at the bottom of the application window, select **Apply**.

You can optionally configure an SSL certificate for a specific URL for the Web Portal in the **Advanced** section here. Choose an identity and an HTTPS certificate that’s been loaded to the server, and the Reporting Services Configuration Manager will make the necessary changes.

- For more information on configuring SSL connections for the SSRS Web Service and Web Portal, visit <https://learn.microsoft.com/sql/reporting-services/security/configure-ssl-connections-on-a-native-mode-report-server>.

- **Database.** Each SSRS instance requires a pair of databases running on a SQL Server instance. Executing the SSRS installer alone does not configure the databases for SSRS; you need to configure them via Reporting Services Configuration Manager. The database names by default are ReportServer and ReportServerTempDB, or, for a named instance, ReportServer\$*InstanceName* and ReportServer\$*InstanceName*TempDB. Both of these databases are important, and you should create a backup schedule for each. The ReportServerTempDB is not a completely transient database like the SQL Server instance's tempdb system database.

The databases for SSRS can be hosted on an on-premises SQL Server instance or Azure VM-hosted SQL Server instance or, since SQL Server 2019, an Azure SQL Managed Instance.

To set the databases for a new instance of SSRS, in the **Database** page of the Reporting Services Configuration Manager, select **Change Database**, and then follow the Report Server Database Configuration Wizard.

- **Web Portal URL.** The web portal URL is the user-friendly website that hosts links to reports and provides administrative features to the SSRS instance. This is the link to share with users if you will be using the SSRS portal. By default, the URL for the web portal is *servername/Reports* for the default instance, or *servername/Reports_InstanceName* for named instances. You can change the name from the default if desired. To proceed, at the bottom of the application window, select **Apply**.
- **Email Settings.** You use these email settings to send reports to user subscribers via email. SSRS uses its own email settings and does not inherit from the SQL Server instance's Database Mail settings. This setting is optional if you do not intend to send reports to subscribers via email.

SSRS can authenticate to an SMTP server using anonymous (No Authentication), Basic, or NT LAN Manager (NTLM) authentication, which uses the SSRS service account to authenticate to the SMTP server.

Modern email systems likely require at least TLS 1.2. For example, with Office 365, TLS 1.0 and 1.1 have been deprecated since 2020. Older versions of Windows and SQL Server may need to be patched to support TLS 1.2.

- If you suspect the TLS 1.2 requirement is preventing SSRS from sending emails, review <https://support.microsoft.com/topic/kb3135244-tls-1-2-support-for-microsoft-sql-server-e4472ef8-90a9-13c1-e4d8-44aad198cdbe>.

SQL Server 2022 introduces support for TLS 1.3 as well. Leverage this when you can, including with SMTP connections.

NOTE

Enterprise SMTP servers usually have an allow list of IP addresses. You will need to add this server's IP to this list to relay email.

- **Execution Account.** You can optionally provide this domain account to be used when reports are configured to run unattended on a schedule or to connect to remote servers for external images. This credential must be a domain account.

To follow the principle of least privilege, the execution account should not be the same as the SSRS service account. Further, this account should have minimal read-only access to any data sources that will require it. You also can give it EXECUTE permissions to stored procedures that serve as data sources for reports, but you should never give it additional SQL Server permissions or let it be a member of any server roles, including sysadmin.

- **Encryption Keys.** Immediately after installation, and after the two SSRS databases have been created, you should back up this instance's encryption keys. These are used to encrypt sensitive information such as connection strings in the two databases. If the databases are restored to another server and this key is not available from the source server, credentials in connection strings will not be usable, and you will need to provide them again for the reports to run successfully on a new server.

If you can no longer locate the backup of a key, back it up again. Alternatively, rotate the key by using the **Change** operation on this page and then back it up.

To restore the original key to a new server to which the databases have been moved, use the Restore operation on this page.

NOTE

SSRS key encryption is different from TDE, which was first supported for SSRS databases in SQL Server 2019. SSRS encryption keys are used inside tables to secure connection strings, while TDE is used to secure database files from being restored or attached to other SQL Servers.

- **Subscription Settings.** Use this page to specify a credential to reach file shares to which report subscriptions can be written. Reports can be dropped in this file share location in PDF, Microsoft Excel, or other formats for consumption. Multiple subscriptions can employ this file share credential, which can be used on this page in a central location. This account should be different from the SSRS execution account, to follow the principle of least privilege.
- **Scale-Out Deployment.** Visit this page on multiple SSRS instances to join them together. By using the same SSRS databases for multiple SSRS instances, multiple front ends can provide processing for heavy reporting workloads, including heavy subscription workloads. The server names can optionally be used in a network load balancer such

as Network Load Balancing (NLB), or you can distribute workload to each SSRS instance from different applications.

Upon first installation, the **Scale-Out Deployment** page will show that the instance is “joined” to a single server scale-out. Each scale-out instance of SSRS must use the same settings on the **Database** page of the Reporting Services Configuration Manager. Connect to each instance in the scale-out and visit this page by opening it on each SSRS instance to view the status, add servers to the scale-out, or remove servers.

- ▶ For more detail on scale-out deployments of SSRS, visit <https://learn.microsoft.com/sql/reporting-services/install-windows/configure-a-native-mode-report-server-scale-out-deployment>.
- **Power BI Integration.** Use this page to associate the SSRS instance to a Microsoft Power BI account—specifically to an account in Azure AD. The administrator joining the Power BI instance to the SSRS instance must be:
 - A member of the Azure AD
 - A member of the system administrator role of the SSRS instance
 - A sysadmin on the SQL Server instance that hosts the SSRS databases
- ▶ For the latest information on Power BI/SSRS integration and the latest Azure authentication features, visit <https://learn.microsoft.com/sql/reporting-services/install-windows/power-bi-report-server-integration-configuration-manager>.

SQL Server Analysis Services initial configuration and setup

No additional steps are required after setup to begin using a new SSAS instance.

You can initiate manual backups of SSAS databases in Object Explorer in SQL Server Management Studio as well as restore SSAS databases. Because of the nature of SSAS databases, their size, and how they are populated, they are not typically backed up on a schedule, but you can do so by passing an XMLA command via a SQL Server Agent job step by typing **SQL Server Analysis Services**.

When installing SSAS, a security group should have been chosen to grant permissions to SSAS server administrators, granting a team full access to the server.

If you need to add a different group to the administrator role of the SSAS instance, open SQL Server Management Studio. Then, in Object Explorer, connect to the Analysis Services instance. Right-click the server and select **Properties** on the shortcut menu. Then, on the **Security** page, add Windows-authenticated accounts or groups to the administrator role.

Azure Synapse Link for SQL Server

This feature replicates operational data into a dedicated SQL pool in Azure Synapse Analytics, directly from SQL Server 2022.

Azure Synapse Analytics is an enterprise analytics service in the Azure cloud running on both serverless and dedicated resource models. Azure Synapse Analytics is a combination of broad technologies, including relational data warehousing, serverless data pools for nonrelational data, built-in machine learning, and other big data technologies.

Far outside the scope of this book, Azure Synapse Analytics accelerates insights into data for logs, time series data, and data integrations. The built-in streaming and deep integrations with Power BI, Cosmos DB APIs, and Azure Machine Learning (AzureML), as well as other analytics tools and pipelines, provide convenient access to cloud resources for all kinds of workloads.

Azure Synapse Link connection

The Azure Synapse Link connection initially does a bulk upload, and then a continuous incremental upload of change feed data on a regular basis. The link between the SQL Server 2022 database and the dedicated SQL pool is mapped and can be changed. This ensures the ability to create, manage, monitor, and delete link connections or add and delete tables to the connection. To access corporate data inside a firewall, it is recommended to use a self-hosted integration runtime (IR).

- For step-by-step details on how to create a self-hosted IR, read the documentation here: <https://learn.microsoft.com/azure/data-factory/create-self-hosted-integration-runtime?tabs=synapse-analytics>.

NOTE

Self-hosted IRs created for Azure Synapse workspaces currently cannot be shared across Azure data factories or between other Synapse workspaces, unlike other self-hosted integration runtimes.

Azure Synapse Link landing zone

The landing zone is an intermediate staging location required to hold the data as it comes in from the SQL Server and before it is loaded into the Synapse dedicated SQL pool.

You must provide an Azure Data Lake Storage (ADLS) Gen2 account to be used as a landing zone, and this landing zone cannot be used for anything else. It must be different from the account created with the Azure Synapse Analytics workspace. An unexpired shared access signature (SAS) token for the ADLS Gen2 account is also crucial, because without it, the data will fail to replicate.

Ensure your database in SQL Server 2022 has a master key created by running the following command:

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '<a new password>'
```

- For current detailed steps on how to set up your own Azure Synapse link for SQL Server 2022, see <https://learn.microsoft.com/azure/synapse-analytics/synapse-link/connect-synapse-link-sql-server-2022#create-your-target-synapse-dedicated-sql-pool>.

The Synapse Link feature is also available for data in Microsoft Dataverse for the Power Platform, Cosmos DB APIs, and Azure SQL Database.

- For all the known limits and issues outstanding with Synapse Link, review the list here: <https://learn.microsoft.com/azure/synapse-analytics/synapse-link/synapse-link-for-sql-known-issues>.

NOTE

The Azure Synapse Link for SQL Server 2022 is in preview at the time of this writing and important details may change.

Container orchestration with Kubernetes

SQL Server in Linux containers is almost identical to SQL Server on Windows or Linux. As noted in Chapter 2, the Database Engine is just the same.

This section discusses how to deploy SQL Server in containers on Kubernetes, a container orchestration system initially developed by Google. Processes like fault tolerance, workload schedule, and even networking are all provided by the management layer of Kubernetes.

Once the orchestration is set up, you can configure and manage the databases inside the containers like any other SQL Server instances. One reason Kubernetes (also known as K8s, because there are eight letters between the K and the s) has become a staple in modern datacenters is its flexibility in container orchestration and management. It provides enterprise-level infrastructure functionality to the container development process favored by most DevOps organizations, making it, as Google describes it, the “operating system for the data center.”

Let’s use the analogy of an actual orchestra, comprising a conductor, instrument sections, musicians, their instruments, and their sheet music. While no analogy is perfect, this might help you picture things more easily. At the bottom are your musical instruments. Each instrument needs to be played by a musician, guided by their sheet music. Groups of musicians play together in a section. Finally, the conductor oversees the entire performance. In our analogy, Kubernetes is the conductor, containers are the instruments, clusters are the instrument families (like the string section or the brass section), and musicians are the pods with their sheet music.

Inside OUT

What is the difference between a Kubernetes cluster, a node, and a pod?

Kubernetes clusters consist of two types of nodes: *masters* and *workers*. The master nodes run cluster operations and schedule work, while the worker nodes run the container workloads. While the lowest unit of deployment is a container, it is important to note that containers are always deployed into higher-level pods. Pods provide location affinity within the cluster, meaning dependent workloads are deployed together.

Kubernetes relies on a software-defined infrastructure (the sheet music in our analogy). When you deploy your containers, you use a YAML (a recursive acronym standing for *YAML Ain't Markup Language*) file that defines:

- The container image you are using
- Any storage you are persisting
- The container CPU and memory configuration of the pod
- The networking configuration
- Other metadata about the deployment

The deployment manifest is converted from YAML to JSON by `kubectl` and then deployed to the Kubernetes API, where it is parsed and then deployed into a key-value store (called `etcd`) that stores the cluster metadata. The objects in the manifest are deployed in their respective pods, services, and storage. The cluster controller (part of the *control plane*) ensures that the manifest is running and is in a healthy application state, and redeploys the manifest in the event of node failure or an unhealthy application state. The cluster will always attempt to maintain the desired state of the configuration, as defined in the deployed manifests.

Inside OUT

What is the Kubernetes control plane?

The Kubernetes control plane is a set of processes and pods that control cluster management. These services record all the Kubernetes objects in the system and execute the desired state configuration for all objects within the cluster.

Kubernetes support for SQL Server

Microsoft introduced support for Kubernetes after the release of SQL Server 2017 (see Figure 4-5). Early releases of Kubernetes lacked support for persisted storage, which is an obvious problem for database containers. The implementation uses a Kubernetes service to act as a persisted front-end name and IP address for the container. In this scenario, if the pod fails, the service stays running, and a new copy of the pod is launched and then pointed at the persisted storage. This is nearly analogous to the architecture of a SQL Server failover cluster instance (FCI).

► Refer to Chapter 2 for a more in-depth discussion on FCIs.

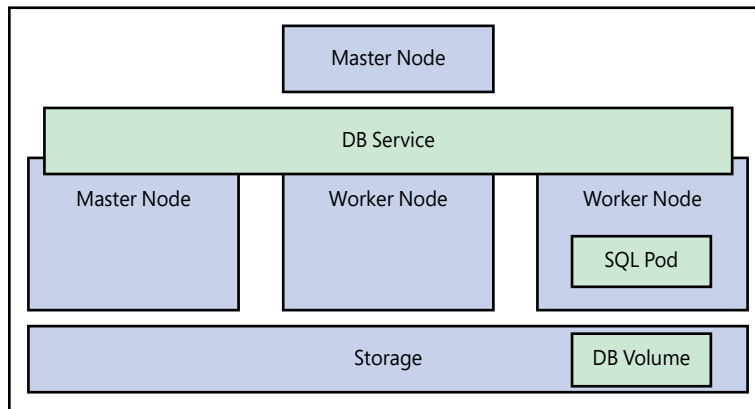


Figure 4-5 SQL Server on Kubernetes architecture.

The SQL Server Kubernetes deployment provides for just a single container per SQL Server pod. Services provide load balancing and persistent IP addressing, while *persistent volume claims* ensure that storage is persisted across container failures or node movement. By defining a persistent volume claim, you align a specific disk volume to your pod deployment to persist data files.

Recent releases of both Kubernetes and Windows Server allow Kubernetes to support both Windows nodes and Windows containers, but SQL Server currently only supports containers on Linux. Kubernetes is also much more broadly used on Linux, so community support will be much more prevalent on that platform.

► To learn more about Kubernetes, read *The Kubernetes Book* (2022) by Nigel Poulton and *Kubernetes: Up and Running* (2022) by Brendan Burns et al.

Inside OUT

What is OpenShift?

Many organizations deploy open-source software with a support agreement in place. Some common examples of this are Red Hat Enterprise Linux (RHEL) and Percona for MySQL databases. Red Hat has also introduced its own Kubernetes offering called OpenShift. While OpenShift is mainly core Kubernetes components, it also introduces some additional tooling into the space for licensed customers—specifically a project called Istio, which offers a service mesh management layer across Kubernetes clusters.

- You can find out more about RHEL in Chapter 5.

Deploy SQL Server in containers

As mentioned, SQL Server runs on Windows, on Linux, and in containers. When originally released with SQL Server 2017, container support was touted for use in development. After all, there was limited support in the container world for persisted storage at the time, and SQL Server lacked support for an enterprise orchestration framework like Kubernetes.

While database containers still make for a fantastic development environment, the support in SQL Server for availability groups and AD authentication means that container deployment is quickly becoming an option for production workloads as well.

- You can read more about availability groups in Chapter 11.

Get started with SQL Server in a Docker container

One of the biggest attractions of running SQL Server in a container is that your choice of OS does not matter. While the container images of SQL Server use Linux as their base, your host machine can run Windows, Linux, or macOS.

While containers can run on almost all host operating systems, SQL Server in containers is only supported for production on Linux hosts running Intel or AMD 64-bit CPU architecture.

NOTE

SQL Server containers do not run on Apple silicon. While SQL Server can run on Apple silicon using the Rosetta 2 emulator and a compatible container host, it is not a supported configuration.

First, you will need to install Docker Desktop on your workstation.

- Download Docker Desktop from <https://www.docker.com/products/docker-desktop>.

After you have Docker installed, you can deploy a SQL Server container with the following steps:

1. Pull a copy of the container image from the Microsoft Container Registry (MCR) to your local environment. To do so, run this command from either a bash shell on Linux or macOS, or an elevated PowerShell prompt on Windows:

```
sudo docker pull mcr.microsoft.com/mssql/server:2022-latest
```

► You can find out more about the bash shell in Chapter 5 and about PowerShell in Chapter 9.

2. Use the following command to deploy the container. Note that the backslash in this command is a way to split a single bash command across multiple lines:

```
sudo docker run -e 'ACCEPT_EULA=Y' -e 'MSSQL_SA_PASSWORD=<YourStrong!Passw0rd>' \
  -p 1433:1433 --name sql2022 \
  -v /users/andrew/mssql:/mssql \
  -d mcr.microsoft.com/mssql/server:2022-latest
```

CAUTION

There is currently no secure way to obfuscate the SA password in a Docker deployment. Microsoft recommends that you change your SA password after you have deployed your container. Do not store the SA password in any saved configuration files. Be careful not to accidentally commit passwords in configuration files to source control.

You may be curious what these parameters (also called *switches* in Linux) mean:

- The `docker pull` command downloads the container image to your local container repository.
- The `docker run` command is where the deployment takes place.
- The `-e` switch allows for an environmental variable to be passed into the deployment. (Chapter 5 covers environment variables.) In this case, you are accepting the End-User License Agreement (EULA) for SQL Server, and providing a strong password for the SA account.
- The `-p` (or `--publish`; note the double-dash before the parameter) switch publishes the container's public and private port for your container. To run multiple SQL Server containers simultaneously, specify a TCP port other than 1433 for the subsequent containers that are deployed.
- The `--name` switch (note the double-dash before the parameter) specifies the name of your container. This is not required, but if it is not specified, the system will generate a name.
- The `-v` switch is probably the most important in terms of database use. It allows a persistent volume to be mounted from your local machine to your container. In this case,

the local directory `/users/randolph/mssql` will appear in the container as `/mssql`. Use this directory to store database backups or data files to be mounted to the container.

- The `-d` switch refers to the container image you are deploying. In this case you are deploying a SQL Server 2022 container from the MCR.
- These are only a few of the command-line parameters that you might need. The full list is documented here: <https://docs.docker.com/engine/reference/commandline/run/>.

NOTE

Docker on macOS does not support persistent volumes. Microsoft recommends that you use separate data container volumes to persist data files that are stored in `/var/opt/mssql/data`. You can read the background on this issue at <https://github.com/microsoft/mssql-docker/issues/12>, and you can learn more about data container volumes at <https://docs.docker.com/storage/volumes/>.

Inside OUT

Can you use containers in development?

Yes. Development is one of the main uses for containers. Given the ease of deploying SQL Server in a container, you can envision a process where a software vendor builds orchestration to perform automated regression tests against every cumulative update (CU) of a release of SQL Server, or across multiple releases. This is just one excellent use case for databases in containers.

After the container is deployed, execute the `docker ps` command (which lists all the running containers) to confirm that your container is running. (In some environments you may need to run `sudo docker ps`.) Also, you can connect to your container using SQL Server tools like SSMS or Azure Data Studio, or `sqlcmd`, by connecting to `localhost`. This is possible because when you deployed the container, you configured it to run on TCP port 1433, which is the default SQL Server port.

```
randolph@charon ~$ docker ps
CONTAINER ID   IMAGE                                COMMAND
CREATED       STATUS      PORTS                    NAMES
24a2a3bda258   mcr.microsoft.com/mssql/server:2019-latest  "/opt/mssql/bin/perm..."
4 minutes ago   Up 4 minutes    0.0.0.0:1433->1433/tcp    sql2019
randolph@charon ~$ sqlcmd -Usa -S localhost
Password:
1>
```

Figure 4-6 A screenshot of `docker ps` output and the `sqlcmd` connection.

NOTE

If you use a custom TCP port (or deploy multiple SQL Server containers) you can connect to `localhost` followed by the port number, separated with a comma. For example, `localhost,1455`.

You can also connect into your container with an interactive shell and execute `sqlcmd`. The first command will launch the bash shell within your container:

```
sudo docker exec -it sql1 "/bin/bash"
```

After launching the interactive bash shell within your container, you then call `sqlcmd` using the full path, since it is not in the path by default:

```
/opt/mssql-tools/bin/sqlcmd -S localhost -U SA -P '<YourNewStrong!Passw0rd>'
```

Once your SQL Server container is deployed, you can execute T-SQL just like it was any other SQL Server.

Get started with SQL Server on Kubernetes

Although running SQL Server in a single Docker container is relatively easy, running SQL Server on a Kubernetes infrastructure is more challenging.

Kubernetes as part of Docker Desktop

You can install Kubernetes with Docker Desktop. However, as mentioned, persistent volumes are not supported on Intel-based macOS. If you are using Docker on Windows and you are running Windows 10 or Windows 11 Professional, you can configure Kubernetes after enabling Hyper-V.

- You can find the instructions for deploying Docker with Kubernetes at <https://docs.docker.com/desktop/kubernetes>.

Kubernetes using minikube

Another commonly used option for development and testing of Kubernetes is minikube, which runs across Windows, Linux, and macOS. minikube is an open-source project that allows for a deployment to your local workstation.

- Configuring minikube is part of the main Kubernetes documentation, available at <https://kubernetes.io/docs/tutorials/hello-minikube>.

Kubernetes using the Azure Kubernetes Service

If you need to simulate a production environment, we recommend deploying using Azure Kubernetes Service (AKS). (See Figure 4-7.) AKS is a managed service that allows you to quickly deploy a Kubernetes cluster of 1 node or up to 100 nodes.

- Configuring AKS is part of the main Azure documentation, available at <https://learn.microsoft.com/azure/aks/learn/quick-kubernetes-deploy-cli>.

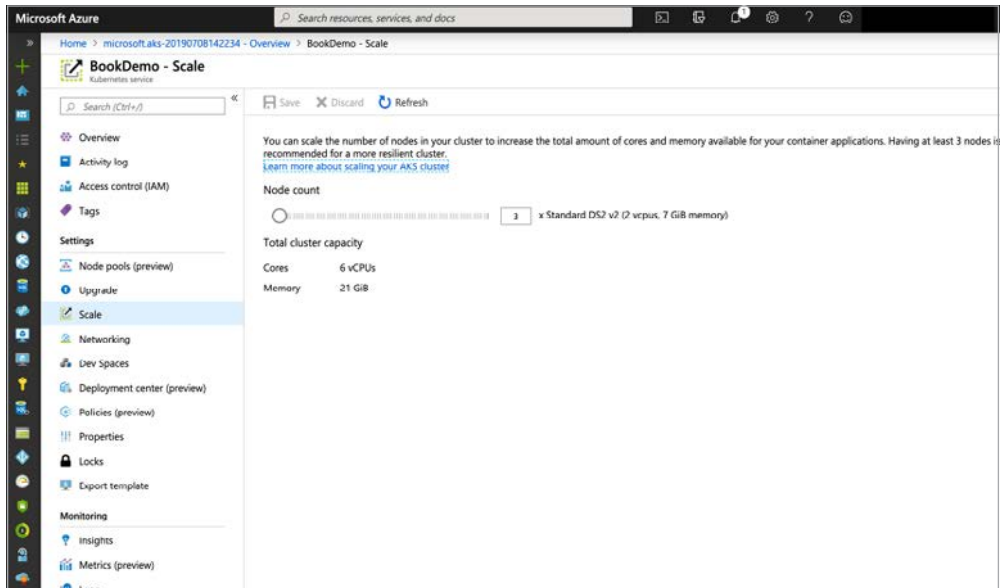


Figure 4-7 A screenshot of the Azure portal showing AKS scale options.

AKS offers the benefit of hosting a highly available control plane for the cluster in Azure, as well as deploying the latest release of Kubernetes without installing software, worrying about dependencies, or finding hardware to build on. The other benefit of AKS is that the service itself is free. You are charged only for the underlying VM compute costs. Storage in AKS is provided by using either Azure Managed Disks or the Azure File service that acts as a file share.

Deploy SQL Server on Kubernetes

Once you have a Kubernetes cluster or simulated cluster like minikube, you can start deploying SQL Server. First, you will need to create a secret in Kubernetes to store your SA password:

```
kubectl create secret generic mssql --from-literal=MSSQL_SA_PASSWORD=<password>
```

If `kubectl` (the Kubernetes command-line tool) is not installed on the machine where you are managing your cluster, you will need to install it to manage your deployment.

- Instructions for installing `kubectl` are available at <https://kubernetes.io/docs/tasks/tools/install-kubectl>.

Next, you will create a persistent volume claim (PVC). As mentioned, containers were originally designed to be ephemeral and not persist data across restarts or failures. A PVC will ask the

cluster to provide a mapping to a persistent volume (PV). A PV can be statically or dynamically provisioned.

- A statically provisioned PV is defined by the cluster administrator. A PVC will be matched to that PV based on size and access mode.
- A dynamically provisioned PV is provisioned from a cluster-defined storage class. A PV asks the storage class to provision the volume from the underlying storage subsystem of the cluster. This can be a cloud provider's persistent volume such as Azure Managed Disks, or even an on-premises SAN.

If you are using Azure Kubernetes Services, save the following code to a file called `pvc.yaml`:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: azure-disk
provisioner: kubernetes.io/azure-disk
parameters:
  storageaccounttype: Standard_LRS
  kind: Managed
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: mssql-data
  annotations:
    volume.beta.kubernetes.io/storage-class: azure-disk
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi
```

This code defines the Azure storage class, and then an 8-GB volume. This code example uses Azure Storage, which is how you would implement on AKS. You will use slightly different code if you are using storage local to your cluster, like you do when using minikube or Docker:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: mssql-data-claim
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 8Gi
```

Just like in the Azure example, save this file to `pvc.yaml` and then deploy using this `kubectl apply` command:

```
kubectl apply -f C:\scripts\pvc.yaml
```

CAUTION

`ReadWriteOnce` is one of three access modes available for persistent volumes. It is the only option that allows both writes and single-node mounting. You will corrupt your databases if a volume is mounted by multiple writers.

The next step is to deploy the SQL Server service and the pod itself. In the following code, you specify the load balancer service as well as the container running SQL Server. Kubernetes can use extensive metadata to describe and categorize your environment, as you will note from the metadata and label fields in the following YAML. Much like in the Docker script earlier, you define a port, passing in the SA password you defined in the secret and accepting the EULA. Finally, in the last section, you define the load balancer, which gives you a persistent IP address for your SQL instance.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mssql-deployment
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: mssql
    spec:
      terminationGracePeriodSeconds: 10
      containers:
      - name: mssql
        image: mcr.microsoft.com/mssql/server:2022-latest
        ports:
        - containerPort: 1433
        env:
        - name: MSSQL_PID
          value: "Developer"
        - name: ACCEPT_EULA
          value: "Y"
        - name: MSSQL_SA_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mssql
              key: MSSQL_SA_PASSWORD
        volumeMounts:
        - name: mssqldb
```


```

    mountPath: /var/opt/mssql
  volumes:
  - name: mssqldb
    persistentVolumeClaim:
      claimName: mssql-data
---
apiVersion: v1
kind: Service
metadata:
  name: mssql-deployment
spec:
  selector:
    app: mssql
  ports:
  - protocol: TCP
    port: 1433
    targetPort: 1433
  type: LoadBalancer

```

You can save this YAML as `sql.yaml`. Then, using the same `kubectl apply -f` command, you can deploy it from where you manage Kubernetes.

Congratulations, you now have SQL Server running on Kubernetes. You can run the `kubectl get pods` and `kubectl get services` commands as shown in Figure 4-8 to see your deployment.



```

randolph@charon ~$ kubectl get services
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes          Cluster-IP    10.0.0.1      <none>         443/TCP          83m
mssql-deployment    LoadBalancer 10.0.149.78   52.183.90.238  1433:32495/TCP  31m
randolph@charon ~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
mssql-deployment-5bc9cc6d54-m9zlr  1/1     Running   0          31m
randolph@charon ~$

```

Figure 4-8 A screenshot showing the load balancer and SQL Server pod in a Kubernetes deployment.
© 2023 The Linux Foundation

If you review the output of the `kubectl get services` command, you will see the external IP address of your SQL Server service. You can now use any SQL Server client tool to connect to that address with the SA password you created in the secret.

CAUTION

This configuration exposes port 1433 to the Internet and should only be used for demonstration purposes. To secure your cluster for production usage, review AKS networking best practices at <https://learn.microsoft.com/azure/aks/best-practices>.

Review cluster health

Kubernetes provides a built-in dashboard for monitoring the overall health of your cluster and its resource use. If you are using AKS, you can view this by running the `az aks browse` command with the resource group and cluster names. Depending on the configuration and version of your cluster, you may need to create a cluster role binding to view the dashboard, as shown in Figure 4-9.

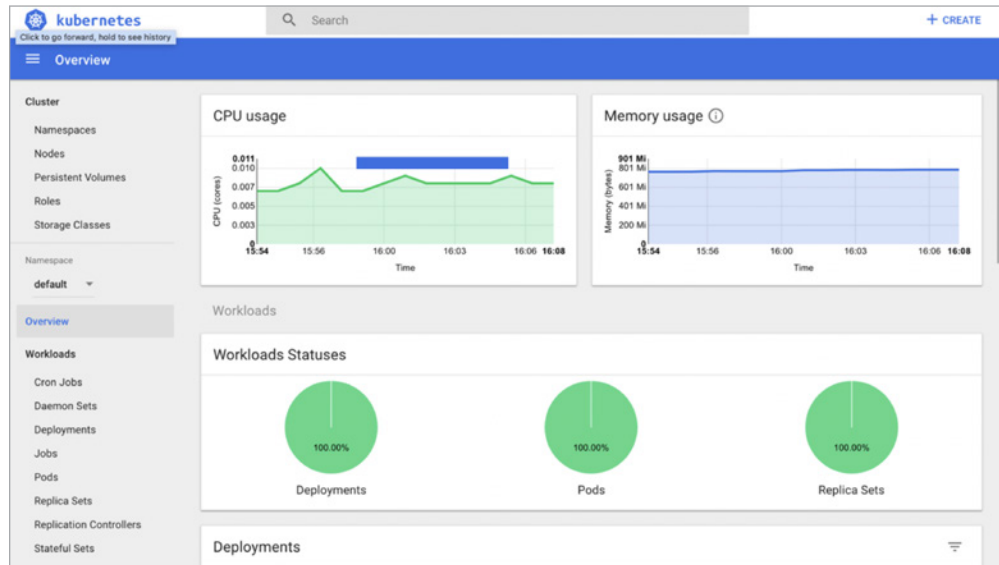


Figure 4-9 A screenshot of the Kubernetes web dashboard.

- If you are not using AKS, you can find instructions on installing and configuring a dashboard for your cluster at <https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard>.

Kubernetes deployments move all your infrastructure into scripts. For some automation-focused administrators, this may be the holy grail that they have been waiting for. But it is important to manage these scripts just as you would your application code. They should be version-controlled in a source control system like Azure DevOps or GitHub. If you are hosting your own repositories, you should ensure they are backed up and highly available.



Index

A

accelerated database recovery (ADR),
103, 107, 239–240, 335

access and authentication

anonymous, 397–398

Azure AD, 850–853, 909

backup, 467

database migration and, 226–227

integrated authentication, 82–87

Mixed Mode, 143

modes of, 549–555

permissions and authorization,
593–605

right of access, 618

role-based access control, 155, 794,
850, 853–854

Shared Access Signature (SAS),
313–315, 468–469

security principals, 555–593, 621

security tasks, 605–615

SMTTP server, 399

SQL Server via Linux, 206

access token, 553

ACID properties, 669

Active Expensive Queries, 35–36

active log, 106

Activity Monitor, 32–36

actual execution plans, 706, 709–710

ad hoc queries, 122, 172

addresses, TCP/IP, 627

administration/administrator

availability group, 543–548

Azure AD, 850–851

Azure SQL Managed Instance,
903–908

dedicated connections, 614–615

footprint, 2

foundations of automated, 394–412

logins, 579–582

Maintenance Plans, 414–425

multiple-server, 428–434

passwords, 209

PowerShell, 434–444

privileges, 199, 573–577

security tasks, xxix, 605–615

tools for, xxxi, 1

Advanced Data Security, 860

**Advanced Encryption Standard
(AES), 463, 637, 641, 645**

affinity masks, 122–124

agent servers, 819, 920

alert notifications

administrator, 38–39

availability group, 548

Azure SQL Managed Instance, 875

SQL Server Agent, 407–410

alphanumeric data types, 250

Always Encrypted, 12, 642–648

Always On, 74

analytics, 309, 827, 906–907

antivirus exclusions, 171

append-only ledger tables, 655

**application security groups (ASGs),
795, 796**

**applications, 675, 698, 701, 896–
898, 905, 928, 938**

architecture, Linux vs. Windows, 197

architecture, on-premises database

configuration settings, 115–127

data compression, 109–112

data files and filegroups, 93–98

file types, 93

partition tables, 108–109

temporary database management,
113–114

transaction log, 99–108

assessment, upgrade, 4, 5

atomicity, 669, 671

attacks

brute-force and dictionary, 623

DDoS, 667–668

ransomware, 451–453

SQL injection, 599, 622, 652, 860, 912

auditing, 657–664, 854–859, 912

**authentication. *See* access and
authentication**

**authorization and permissions,
593–605, 621**

auto close, 233

auto_create_stats, 420

auto_update_stats, 341, 784

autogrowth events, 344–345, 369

automatic failovers, 515–516

automatic plan correction, 745

automatic seeding, 518–521

automating scripts, 10

availability groups (AGs)

administration, 543–548

Always On, 74

asynchronous commits, 79,
512–517, 546

Azure SQL Database, 797

backing up, 426–428

capabilities of, 494–496

designing, 502–532

failovers in, 515–518

hybrid topology, 531–532

Linux, 81–82, 537–543

multiserver administration in, 430

PowerShell with, 442–444

read-scale, 80, 510

recent improvements, 509–512

resource creation, 542

seeding options, 518–522

server-level options, 522–523

service accounts, 579

SQL Server Agent, 411–412

synchronizing database
replicas, 525

synchronous commits, 79,
512–517, 546

types of, 80–81

WSFC with, 507–508

**Average Disk seconds per Read or
Write, 375–376**

Azure

- cloud, 71
- command line interface, 840–841
- data services, 826–828
- DBaaS, 790
- governance, 792–794
- hybrid cloud, 821–825
- infrastructure as a service, 797–805
- key features, 789–790
- networking in, 795–796, 824–825
- platform as a service, 805–821
- portal, 803–805, 832–833, 860, 886
- PowerShell for, 791
- SSIS IR, 936–937

Azure Active Directory (Azure AD)

- access via, 12, 82–85, 850–853
- AD DS vs., 197
- authentication, 156, 551–553, 879
- Azure SQL Managed Instance, 908–911
- B2B/guest users, 848–849
- credentials, 852
- Domain Services (AD DS), 197
- FCI configuration, 496
- function of, 85–87
- Linux support, 83
- setup, 910
- subscription, 793

Azure Arc, 51–52, 197

- Azure Arc-enabled servers, 51, 154–155, 823**

Azure Backup, 867–868

- Azure Blob Storage, 322–323, 798, 901, 929–933**

Azure Cosmos DB, 827**Azure Data Factory, 933–937**

- Azure Data Migration Assistant (DMA), 916, 919–921**

Azure Data Studio (ADS)

- command line, 46
- Extended Events, 364
- extensions, 45–46
- features, 43–45
- function of, 40–41
- migrations, 922–925
- notebooks, 47–48
- open-source, 41
- SQLCMD mode, 10
- user interface, 41–43

- Azure Database Migration Service (DMS), 223, 901, 916, 921–933**

- Azure ExpressRoute, 824**

- Azure Hybrid Benefit, 821**

- Azure integration runtime (IR), 933–934**

- Azure Key Vault, 636–637**

- Azure Kubernetes Service (AKS), 188, 193**

- Azure Monitor, 380–383, 906**

- Azure SQL Advanced Threat Protection (ATP), 652, 854, 859, 860, 861**

- Azure SQL Analytics, 384, 907**

Azure SQL Database

- audits, 663–664
- cloud models, 796–821
- copying databases, 842
- data management, 796–797
- database provisioning, 836–844
- disaster recovery prep, 861–868
- elastic jobs, 393
- elastic pools, 817–818
- elastic queries, 818–819
- Export Service, 863
- features, 620, 829, 871
- limitations, 812–813, 817
- Log Replay Service, 490
- logical server provisioning, 830–836
- login migration, 612–613
- logins, 86–87
- Microsoft Defender protection, 651–653
- migrations, 6
- multi-modal capabilities, 827
- new features, 830
- platform as a service, 805–806
- Private Link for, 850
- purchase models, 806–812
- restoring corrupt, 330
- security, 846–861
- SQL Server vs., 812–818
- third-party fully-managed data platforms, 827

- Azure SQL Edge, 212, 822**

Azure SQL Managed Instance

- administration, 903–908
- connecting instances, 893–899
- creating instances, 883–892
- data migration, 900–903
- data protection features, 911–913
- deleting instances, 892–893
- function of, 870, 871, 872–873
- Kerberos for, 87

- link feature, 532–537, 870, 874, 876, 888, 900–901, 907–908

- migration, 916

- new features, 869–870

- protection of, 651

- provisioning, 886–890

- security, 908–911

- server-level collation, 143

- SQL Server vs., 873–883

- unified data platform, 309

Azure Stack, 823**Azure Storage**

- backups, 469
- costs, 800
- deleting old backups, 421
- Kubernetes code, 190
- SQL Server data files in, 801
- SSMS and, 469

- Azure Synapse Analytics, 309, 827**

- Azure Synapse Link, 181–182, 827**

Azure VMs

- authentication, 554
- automatic backups, 822
- Azure SQL Managed Instance connections, 896
- bandwidth, 800
- confidential, 805
- D: volume, 132
- IP forwarding, 666
- NSGs, 664–665
- sizing, 801–802
- SQL Server FCIs, 502
- SQL Server on, 138–139, 163

B

- B+ tree, 764**

backups

- access to, 125
- automatic, 822
- Azure SQL Managed Instance, 876–879, 901
- backup set, 471
- backup to URL, 468–469
- chains, 459–461
- checks/tests, 483
- compressing, 112
- copy policy, 467
- creating/verifying, 472–475
- database, 422–423, 426–428
- database moves and, 243–244
- deleting old, 421–422
- devices, 467–472

- immutable, 452
- long-term retention, 867–868, 878
- master key, 173–174
- migration, 5
- options for, 67
- RAID and, 67
- recommended frequency, 473
- redundancy, 877
- restoring, 220–221
- reviewing successful, 174–175
- scheduling, 173
- security of offsite, 483
- types of, 453–467
- Backup-SqlDatabase, 439**
- BACPAC file, 222, 862, 863**
- bash shell, 186, 207**
- basic availability groups, 81, 506**
- batch mode, 746, 776–777**
- batch requests, 377**
- BCP (Bulk Copy Program), 9, 11–12**
- benign waits, 352–353**
- binary data type, 258**
- binary file execution, 208**
- binary large objects (BLOBs), 292–295, 798**
- blockchain, 653**
- blocked connections, 675, 676–679**
- Border Gateway Protocol (BGP), 628**
- branch predictors, 61**
- browser, 7, 213**
- brute-force attacks, 623, 624**
- buffer pool, 56–57, 70, 99, 635, 640, 736**
- buffers, backup, 474**
- built-in database roles, 586–588**
- built-in server roles, 563–572**
- BUILTIN\Administrators group login, 578**
- bulk-logged recovery model, 454, 455**
- business critical service tier, 904**
- C**
- cache**
 - data, 56–57
 - drive caching, 801
 - procedure, 56, 57–58, 728–732
 - sequence objects, 271
- cardinality estimation (CE), 722–725, 747**
- catalog views, PolyBase, 319–320**
- Central Management Servers, 24**
- certificates, digital, 630–631**
- certification authority (CA), 630–631**
- chains, backup, 459–461**
- change data capture, 303–308**
- change tracking, 303–308**
- character data types, 255**
- check constraints, 269**
- Check Database Integrity task, 417**
- CHECK_POLICY, 561**
- checkpoints, database**
 - defined, 238
 - indirect, 238–239
 - memory-optimized tables, 286
 - performance, 100
 - types of, 103–105
- CHECKSUM, 98, 228, 326, 464–465**
- claims, 86**
- classification, 115**
- classifier function, 386**
- clean pages, 70**
- clock cycles, 60, 91**
- cloud computing**
 - application resilience, 938
 - cloud witness, 498
 - cloud-first development, 794
 - data protection in, 631
 - defined, 88
 - key features, 789–790
 - Microsoft Defender for Cloud, 155–156
 - networking, 71
 - public environments, 789
 - recoveries, 484–485
 - security, 636–637, 826, 829
 - See also* Azure
- cloud models, 796–825**
- clustering**
 - cluster health reviews, 193
 - indexes, 111, 753, 754–760
 - Kubernetes, 183
 - nodes, 74–76
 - Windows vs. Linux, 537–538
 - See also* failover cluster instances (FCIs)
- cmdlets, 436–444**
- code**
 - compatibility, 223
 - IntelliSense, 27–28
 - migration, 930
- collation**
 - constraints, 543
 - data type, 251–253
 - instance, 142, 143
 - three levels of, 230–231
 - UTF-8, 251–252
- Collation tab, 142**
- columns**
 - Always Encrypted, 642–648
 - column sets, 261
 - computed, 275–276
 - partitioning, 295–303
 - sparse, 274–275
 - width of, 252
- columnstore indexes**
 - clustered, 754, 773–775, 781, 783
 - compress data with, 109
 - difference between rowstore and, 753, 770–771
 - maintenance of, 341–342
 - ordered clustered, 747, 774
 - performance tuning with, 718
 - understand, 773–780
- command line**
 - ADS, 46
 - Azure interface, 791, 840–841
 - container deployment from, 186–187
 - downloading updated tools, 12
 - Linux interface, 8–9, 198
 - start SQL Server Setup from, 159
- committed transactions, 98, 99, 105**
- compatibility levels**
 - Query Store for, 237
 - setting, 232
 - upgrading, 222–225, 724–725
- compound primary keys, 267**
- compression**
 - backup, 112, 423
 - data, 112
 - transaction log data, 512
 - XML, 787–788
- compute tier, 810**
- concurrency**
 - on-disk vs. memory-optimized, 700–702
 - isolation levels and, 671–702
 - optimistic, 263, 286
- Configuration Checker, 3–4**

Configuration Manager. *See also* Report Server Configuration Manager; mssql-conf for SQL Server on Linux; SQL Server Configuration Manager

configuration settings

automating SQL Server Setup, 158
 Azure SQL Managed Instance vs. SQL Server, 880–882
 database, 229–240
 FCI, 496–502
 log shipping, 491
 migration and, 227–229
 PolyBase, 312
 post-install, 163–176
 on-premises database architecture, 115–127
 Query Store, 738
 SQL Server Agent, 400
 SQL Server on Linux, 204, 206–212
 SQL Server volume, 134–135
 SSISDB, 176–177
 SSRS, 177–180
 surface area, 168
 zone-redundant, 863

connectivity issues, 905

consistency, 669

constraints

collation, 543
 edge, 288, 289
 table, 266–269

contained availability groups, 81, 506

contained databases, 555, 584

contained users, 584, 880

containers

Azure, 793
 defined, 87
 Kubernetes, 182–193
 Linux, 56
 SQL Server use in, 185–188
 types of, 87–92, 227, 232–233

control plane, 183

Copy activity, 934

copy-only backups, 465

core counts, 62–63, 90, 91, 114

core processors, 122–124

corruption, database. *See also* DBCC

CHECKDB

checking for, 475

detect/prevent/respond to, 325–330
 recovering from, 464–465

cost, logical server provisioning, 831

cost threshold for parallelism, 117

cost-based optimizers, 117

CPU (central processing unit)

core allocation, 122–124
 function of, 59–63
 high performance settings, 200
 pressure, 121
 virtualizing, 87–92

crash recovery, 105–106. *See also* failures; recovery

CREATE and ALTER operations, 339

CREATE DATABASE statement, 218

creation, database, 216–220

credentials, 636, 852

cumulative updates, 217

customization, 18, 29–31

CXCONSUMER wait type, 354–355.

See also wait types and statistics

D

daemons, 210

data

analytics, 309
 Azure services, 826–828
 caches, 56–57
 capturing modifications to, 303–308
 collectors, 371–384
 compression, 109–112
 constraints, 266–269
 data in motion, 632, 635
 exfiltration, 911
 external, 153, 316–317
 file size management, 342–347
 flows, 934
 loss, 72–73, 449
 masking, 650–651
 migration, 4–6, 900–903, 915–938
 protection, xxix–xxx, 618–668, 911–913
 security principals, 620–631
 storage, 63–70
 synchronization, 79
 temporal table movement, 278
 types, 250–266, 273–274
 virtualization, 153, 309

Data Access Migration Toolkit (DAMT), 916, 926

Data Definition Language (DDL), 593–594, 620

Data Encryption Standard (DES), 637

Data File I/O, 35

data files

access to, 125
 adding, 218–219
 Azure Storage, 801
 moving, 241
 recommended number of, 114
 restoring corrupt, 329
 server architecture and, 93–98
 volume settings, 134

Data Manipulation Language (DML), 593, 594, 862

Data Migration Assistant (DMA), 4–6, 223, 916

data pages, 96–98

Data Profiling Task, 50

Data Quality Client, 8–9

Data Quality Services, 8–9

Data Tools

database deployment with, 220
 installation, 141
 list of, 48–51

data transfer speeds, 825

data types

general-purpose, 250–258
 preferences, 266
 specialized, 258–266
 user-defined, 273–274

database as a service (DBaaS), 790, 805

Database Engine

full installation, 6
 network access to, 70
 NUMA nodes, 62
 running queries, 57
 Server Registration with, 22
 TDE in, 639
 tools and services, 7–14

Database Engine Tuning Advisor, 14–16

Database Experimentation Assistant (DEA), 916, 918, 919

database infrastructure, 55

Database Mail, 394–400

database master key (DMK), 463, 634, 637, 638–639, 641

Database Migration Service, 926–933

database transaction units (DTUs),
806–808, 812

databases

accessing, 82–87
audits, 854–859
backing up, 422–423
compatibility levels, 222–225
consolidating, 818
contained, 555
creating new, 216–220, 416, 837–838
detaching, 243
edge computing, 52
files, shrinking, 345–346
firewall, 846–849
geo-replication, 863–865
life cycle, 831
manual exports, 862
master keys, 312
memory-optimized tables prep, 283
Microsoft Purview for, 52
migrating, 225–229, 931–932
mirroring, 78, 213, 240
moving existing, 220–222
moving/removing, 241–248
options for, 230–240
ownership, 218
on-premises architecture, 93
principles, 582–593
provisioning Azure, 836–844
restoring, 475–481
scoped credentials, 315–316
settings, 217, 229–240
sharding, 820–821
snapshots, 462–463
space management, 845
SSRS, 178
VLDBs, 453, 462, 468, 810

datetime data types, 256–258

DBCC CHECKDB

with checksum enabled, 475
integrity checks, 98, 164, 173, 330,
417, 734
determining timing of, 329
resources required, 327
scheduling, 173, 326–327

deadlocks, capturing, 367–369

decompression, 112

**dedicated administrator connection
(DAC), 614–615**

defaults

collation, 231
configuration settings, 117
constraint, 269
database settings, 216–217
disaster recovery features, 861–862
LPIM policy off, 58
tempdb database, 144–145

defense in depth, 621–622

degree, 747

degree of parallelism (DOP), 748.

See also max degree of parallelism (MAXDOP)

delayed durability, 80, 99, 702–705

delegation, 85, 892–893

deltastore data, 777–778

**developers/development. *See also*
applications**

cloud-first, 794
containers in, 187
Extended Events for, 370
privileges for, 573, 591

devices, backup, 467–472

dictionary attacks, 623

**differential backups, 458–459,
475, 477**

differential bitmaps, 97, 458, 465

digital certificates, 630–631

Digital Defense Report, 451

direct-attached storage (DAS), 65, 66

directory structures, 197

DirectQuery, 7

dirty pages, 70, 77, 238, 671

disaster recovery

availability groups, 495
Azure SQL Database, 861–868
defined, 74
scenario, xxx, 447–449
site security, 484
technologies, 488–496
See also failures; recovery

discontinued features, 53

disks

Azure storage, 798–800
backup, 468
C: volume, 131
disk starting offset, 135
D: volume, 132
settings, 202
striping options, 800

**distributed availability groups,
80–81, 506–507**

**distributed transactions, 511,
881–882**

**distributed-denial-of-service
(DDoS) attacks, 667–668**

**distributions, Linux, 195–196, 198,
199–200, 206**

Docker, 1, 88, 89, 185–188, 199

downtime, 450, 502

drive caching, 801

drivers. *See* ODBC; OLE DB

durability, 669, 702–705

dynamic data masking, 650–651, 913

**dynamic management objects
(DMOs), 347–358, 371–384**

**dynamic management views
(DMVs), 409, 544**

dynamic quorum management, 499

E

edge computing, 52, 822

edge tables, 287–289, 292

EKM modules, 635–636

elastic pools, 795, 817–818, 844–845

elimination, partition, 108, 109

email

Database Mail, 394–400
history, 398
settings, 178

enclaves, secure, 643, 644, 647

encryption

backup, 463–464
data, 912
encryption keys, 179, 645–646
hashing vs., 623–625
hierarchy, 630, 631, 634–635,
637–639

network, 71

protocol, 629

ransomware attacks, 452

sensitive data, 619

symmetric/asymmetric, 629–630

TDE, 227, 466, 521, 639–642

encryption keys

automatic rotation of, 912

public and private, 629–630

secret, 629

See also master keys

endpoints

- Azure SQL Managed Instance, 894
- mirroring, 509, 539
- Private Link, 850
- security isolation, 912
- service, 667, 796, 899

enlightenment, 69**erasure, right of, 619****Error Logs, 31–32****errors**

- failure options, 72–82
- log retention, 174
- out-of-memory (OOM), 56, 282

estimated execution plans, 706, 708–709**estimation, cardinal, 722–725****event alerts, 38, 408****event forwarding, 431****EXECUTE AS statement, 600****execution account, 179****execution plan, 705–735****exfiltrated data, 911****exponential backoff, 905****Export Service, 863****ExpressRoute, 824****Extended Events**

- availability groups, 547–548
- GUI, 14–15, 361–370
- XEvent Profiler, 362, 363

extensions

- ADS, 45–46
- Azure, 153–157
- buffer pool, 57

extents

- backups and, 458
- tempdb, 114
- uniform and mixed, 95

external cluster management, 504**external file formats, 317–318****external scripting, 8****external tables, 309–323, 785****F****failover cluster instances (FCIs)**

- Always On, 76–77
- availability groups, 507
- capabilities of, 492–494
- configuring, 496–502
- defined, 75

failover groups, Azure SQL**Database**

- sample, 886–890
- set up, 865–866

failovers

- acceptable downtime and, 450–451
- application resilience and, 938
- automating, 443
- availability group, 515–518
- availability group planned, 516
- availability group forced, 516–518
- Azure SQL Managed Instance, 534, 537, 904, 908
- compare technology, 489
- document scenarios for, 522–524
- failover cluster instance, 77, 504–507
- failover_mode_desc, 544
- forced vs. unforced, 904
- geo-replication, 864–866
- impact on secondary replicas, 512–515
- link feature for Azure SQL Managed Instance, 533–534, 900
- long-term backup retention and, 867–868
- multisubnet, 528–530
- SQL Server Agent job considerations, 411
- Replica read-write traffic redirection, 509
- tooling and automation, 537
- unplanned on a failover cluster instance, 286

failures

- antivirus interference, 171–172
- audit initiation, 658–659
- cumulative update, 217
- data recovery prep, 446–451
- delayed durability and, 99
- Linux node, 75
- migration, 926–933
- node restart after, 542
- partial restores for, 95
- reducing downtime from, 92
- types of, 72–73
- See also* recovery

faults, page, 377–378**federation, 86****Fibre Channel (FC), 68****file system**

- configuration, 124–127
- Linux setup, 201–202
- Windows vs. Linux, 197–198

filegroups, 93–98, 461–462**files**

- data files. *See* data files
- editing Linux, 202
- file backups, 461–462
- generating configuration, 159–160
- instant file initialization (IFI), 106
- locating database, 241–242
- page files, 116, 173
- separating SQL Server, 133
- size management, 342–347
- system, 245
- tempdb, 114
- test, 926
- transaction log storage, 99
- user, 244

FILESTREAM, 292–295**FileTable, 295****fill factor, 331–333****filter drivers, 474****filtered nonclustered indexes, 767****filtering objects, 25–26, 649–650****firewall**

- built-in protection, 652
- Linux, 206
- perimeter security via, 621
- server- and database-level, 846–849

firmware, 61**flash memory, 64–65****floating-point values, 254****forced failovers, 516–518, 904****forced query plans, 728, 740, 741–742****foreign keys, 266–268****fragmentation, 331, 333–334, 419****full backups, 456–458, 459, 475, 476****full recovery model, 454, 455, 481****full seeding, 521–522****full-text indexes, 786****G****General Data Protection Regulation (GDPR), 618–619****geometry/geography data type, 259–260****geo-replication, 863–865, 867, 905****GNU packages, 195–196****GNU/Linux. *See* Linux****Google, 86, 182, 628****governance, cloud, 792–794****graph tables, 287–292**

graphical execution plans, 713–722
 graphics processing units (GPUs),
 624, 625

H

Hadoop, 309, 314–315, 317, 319

HADRON, 495, 525

hard drives, 64

hardware

Azure SQL Managed Instance service tier, 885–886

basic components, 55

CPU, 59–63

memory, 56–59

networks, 70–72

security threats, 61

SQL Server 2022 requirements, 130

storage, 63–70

tips for choosing, 838

virtualization and, 88

hash indexes, 781–782

hashing, 623–624

heaps, 331, 753, 758–759

hierarchyid data type, 264–265

high availability (HA)

availability groups, 78–82, 495

Azure SQL Managed Instance, 873,
 885–886, 903–905

defined, 72, 488

disaster recovery features, 861

failure points, 72–73

FCIs, 76–77

log shipping, 77–78

offerings, 55

recovery technologies, 488–496

redundancy, 73–74

tools, 72–82

high performance, 200

history

History Cleanup task, 421

retention, 174–176

Horizontal Fusion, 7

horizontal partitioning, 108,

296–302

hybrid environments

AG topology, 531–532

hybrid buffer pool, 70

hybrid cloud, 797, 821–825

recovery in, 484–485

Hyperscale service tier, 810–812,
 843–844

Hyper-Threading, 60, 91, 114, 144, 779

hypervisors, 87, 89, 90, 91

I

IaaS, 664–668

idempotent functions, 683

identifiers, unique, 756–757

identities, managed, 552–553, 852

immutable backups, 452

immutable storage, 653

INCLUDE statement, 764–766

incremental statistics, 234

indexes

adding to secured columns, 646

clustered, 754–760, 770

columnstore, 109, 341–342, 718,
 753–754, 770–771, 773–780

compressing, 109–112

full-text indexes, 786

maintaining, 330–342

memory-optimized, 419, 780–783

nonclustered, 753, 759–772, 782–783

partitioning, 297–298

primary keys and, 267

query, 717

rebuilding, 336–339

reorganizing/rebuilding, 418–420

rowstore, 753–754

scheduling maintenance, 173

spatial, 786–787

statistics for, 783–785

unique constraint and, 267

usage, 771–772

XML indexes, 787–788

infrastructure as a service (IaaS),

664–668, 790, 796, 797–805

initialization, instant file, 106, 125–127

In-Memory OLTP, 59, 102, 670,

700, 927

installation

Azure VM, 163

Installation Center, 2–3

Kubernetes, 182–193

Linux installs, 200–213

new instances, 137–163

plan for, 3–6

PolyBase, 311

post-install configuration, 163–182

PowerShell module, 436–438

pre-install considerations, 130–137

SSMS, 19

using configuration file, 160–163

Installation Center, 2–3

instances

adding databases to, 215–240

configuring SQL Server, 209–211

creating new, 883–892

failovers to managed, 532–537

instance collation, 142, 230–231

moving databases within, 242–247

multiple, 138

named, 203

new instance installation, 137–163

restoring, 876

stacking, 7

See also Azure SQL Managed Instance; failover cluster instances (FCIs)

instant file initialization (IFI), 106,
 125–127, 142

integrated authentication, 82–87, 552

integration runtimes, 933–937

integration services, 11, 13, 48–51

integrity checks. *See also* DBCC CHECKDB

Intel QuickAssist Technology (Intel QAT), 112

intelligent query processing (IQP),
 620, 705, 746–752, 871

IntelliSense, 10, 27–28

interleaved execution, 750

Internet Information Services (IIS), 16

Internet of Things (IoT), 627

Internet Small Computer Systems Interface (iSCSI), 68

Invoke-Command, 443

Invoke-Sqlcmd, 440–441

isolation

ACID properties of a relational database, 669

levels and concurrency, 671–702

security, 912

snapshot isolation (SI), 235, 285

J

Java, 7, 8, 152, 197, 818, 926

jobs

agent, 920

automating, 37–38

elastic database, 819–820

Job Activity Monitor, 38

join operators, 720
JSON, 261–262, 358

K

Kerberos, 83–85, 87, 911
kernel support, 47
kernels, 58, 195, 202
keyboard shortcuts, 28
keys. *See also* encryption keys;
referential integrity
Kirby, 558, 559, 564, 573, 581,
606–608, 615, 834, 839, 841
Kubernetes, 1, 182–193
Kubernetes minikube, 188
Kusto Query Language (KQL), 45,
382, 859, 907

L

languages
.NET, 274
external, 7–8
Java, 7, 8, 152, 197, 818, 926
Markdown, 47, 48
large value data, 253–254, 928
lazy commit, 80, 99
ledger, 653–657
tampering, 654
licensing
affinity masking and, 124
installation and, 136
open-source software (OSS), 197
terms, 209
link feature for Azure SQL Managed
Instance, 532–537, 870, 874,
876, 888, 900–901, 907–908
Linux
Active Directory on, 83
affinity configuration, 124
availability groups, 81–82, 537–543
command line interface, 9
command line tool updates, 12
containers, 56
Database Mail, 397
failover clusters, 75, 79, 497
file editing, 202
memory management, 58
migrations, 920–921
navigating, 198
patches, 390
performance metrics, 379–380

system overview, 195–196
Windows vs., 196–199
WSFC and, 510
See also SQL Server on Linux
live execution plans, 706, 711
load-balanced read-only routing, 530
Local Server Groups, 22–23
lock modes, 675, 676–679, 689
lock pages in memory (LPIM)
configuration settings, 121–122
evaluating necessity of, 172
function of, 58–59
on/off checks, 172
space and, 116
Log Analytics, 384, 858–859
log files, 158
log maintenance, 173
Log Replay Service (LRS), 490
log sequence number (LSN), 100
log shipping, 77–78, 489–492
log truncation, 101–102
logical processors, 116–117, 122
logical server provisioning, 87,
830–836
logins
administrator, 579–582
configuring, 559–582
creating with known SIDs, 608
database migration and, 226
database user, 582, 585
migrating, 608–614
orphaned SIDs, 605–608
Pacemaker, 542
special purpose, 577–579
user identifiers for, 556–557
logs, error, 31–32, 174
looping code, 721
lost updates, 683
M
Machine Learning Services, 7–8,
152–153, 167–168, 852
macOS
command line interface, 9
command line tool updates, 12
Docker on, 187
Server setup, 1
SQLCMD mode, 10
maintenance
Azure SQL Managed Instance, 875

data collectors, 371–384
database corruption response,
325–330
DMOs to monitor activity, 347–358
Extended Events, 361–370
file size management, 342–347
index and statistics, 330–342
Maintenance Plans, 414–425
Resource Governor, 384–389
servicing model, 389–391
SQL Assessment API, 358–361
SQL Server, 37, 173, 203, 235, 325,
393, 412–414
Window (Azure SQL), 838, 875–876,
885, 890
malware, 451
managed disks, 800
managed identities, 552–553, 852
Managed Instance. *See* Azure SQL
Managed Instance
management, temporary database,
113–114
Management Console, 13
Management Data Warehouse
(MDW), 16
Management Studio. *See* SQL Server
Management Studio (SSMS)
Markdown language, 47, 48
masking data, 650–651, 913
master files, 246
master keys
backups, 173–174
column, 645–646
database, 463, 634, 637,
638–639, 641
encryption hierarchy and, 637–639
number of possible, 312
protection of, 634
service, 173–174, 634, 637, 638
Master/Target (MSX/TSX) servers,
428–430
MATCH subclause, 290–291
max degree of parallelism (MAX-
DOP), 118–119, 145. *See also*
degree of parallelism (DOP)
Max Server Memory, 119–121,
146–147, 165
max worker threads, 120, 121
mechanical drives, 64
media sets, 470–471

memory

- buffer pool, 56–57
- limits by SQL edition, 59
- Linux management, 58
- lock pages in memory (LPIM), 58–59
- max settings, 165–168
- memory grant feedback, 748–749
- In-Memory OLTP, 59
- non-uniform memory access, 61–62
- OOM errors, 56, 282
- overcommitting, 89
- performance metrics, 377, 378
- persistent, 64–65, 69, 70
- procedure cache, 57–58, 728
- settings, 119–122
- storage, 63–70
- working set, 56

memory-optimized objects, 97–98, 113**memory-optimized tables**

- backing up, 465
- delayed durability and, 704
- function of, 282–287
- index maintenance, 419
- indexes in, 780–783
- isolation levels and, 695
- metadata storage, 113
- queries using, 700
- statistics, 785

menus, customizing SSMS, 29–31**merge join operators, 720****metadata**

- cached plan, 732
- key, 646
- memory-optimized, 113

metrics, performance, 371–384**Microsoft Assessment Planning (MAP) toolkit, 915, 916–917****Microsoft default settings, 117****Microsoft Defender for Cloud, 155–156****Microsoft Defender for SQL, 651–653, 854, 859–861, 867****Microsoft Purview, 52, 156–157, 619–620****migration**

- Azure Data Factory, 933–937
- common failures, 926–933
- data, 4–6, 900–903
- database, 220–229, 926–933
- login, 608–614

security and resilience during, 937–938

service options, 915–926

SQL Server-authenticated logins, 899

time, 922

Minimum Recovery LSN, 103, 106**mirroring endpoint, 509, 539****missing indexes, 767–771****mistakes, installation, 130****mixed extents, 95, 114****Mixed Mode authentication, 143****model database, 94****modern interactive flow, 910****modifications, capturing data, 303–308****module, PowerShell, 791–792****moving/removing databases, 220, 241–248****msdb database, 399****Msodbcsql: See ODBC (Open Database Connectivity)****MSOLEDBSQL 529, See also OLE DB****mssql-cli, 10–11****mssql-conf, 203, 207, 212****multifactor authentication (MFA), xxix, 83, 396, 452, 551, 792, 908****multi-select tool, 26****multi-server administration, 428–434****MultiSubnetFailover, 528–530. See also availability groups (AGs)****multitenant architecture, 817–818****N****name identifiers, 853****natively compiled stored procedures, 283–285****.NET, 273, 274, 691–692****network interface cards (NICs), 69, 92****network security group (NSG), 664–666, 795, 898****network-attached storage (NAS), 68–69****networks**

Azure, 795–796

Azure/on-premises, 824–825

Azure SQL Managed Instance, 893–899

BGP routing, 628

cluster security, 192

data transfer speeds, 825

enabling TCP/IP, 169–170

security, 664–668, 937

SQL Server connection, 70–72

transient connectivity issues, 905

virtual, 92

See also virtual network (VNet)

new databases, 215–240**NEXT VALUE FOR, 272****nodes**

clustering, 74, 75, 76, 496

fencing, 75

graph table, 287, 288, 289

intermediate/non-leaf, 111

Kubernetes, 183

node majority, 498

self-hosted integration runtime, 935–936

nonclustered indexes, 753, 759–772, 782–783**non-relational data**

management, 827

nonrepeatable reads, 684–687**non-uniform memory access (NUMA), 61–62, 91****normalization, 266****notebooks, ADS, 47–48****NT LAN Manager (NTLM), 83, 84****NTFS allocation unit size, 127****NUMA nodes**

affinity masks and, 123, 124

Linux configuration, 201

parallelism and, 118, 119

numeric data types, 254–255**nvarchar data type, 253****O****Object Explorer, 20–21****objective, service. See service tiers/objective****objects**

AG object creation, 508

binary large objects (BLOBs), 292–295

DMOs, 347–358, 371–384

filtering, 25–26

ledger, 657

limiting access to, 604–605

memory-optimized, 97–98, 113

migrating server-level, 932

moving security, 613

objects (continued)

- multi-select tool, 26
- sequence, 270–273
- ODBC (Open Database Connectivity)**, 10, 322–323, 633, 645, 937
- offline migrations, 242, 929
- OLE DB**, 356, 529, 633, 937. *See also* MSOLEDBSQL
- on-disk concurrency, 700
- on-disk tables, 784–785
- OpenShift, 185
- open-source software (OSS), 41, 197, 929
- operating systems**
 - command line tool updates, 12
 - data protection from, 633–634
 - Linux, 195, 200
 - macOS, 1, 9, 10, 12, 187
 - page files, 116
 - reserving memory for, 120
 - SQL Server setup, 129, 131
 - SSMS support for, 19
- operators**
 - new, 407
 - notifications for, 40
 - query execution plans, 715–717, 719
 - SQL Server Agent, 406–407
- optimistic concurrency, 263, 286
- Optimization Level, 716
- Optimize for Ad Hoc Workloads, 122
- OPTIMIZE_FOR_SEQUENTIAL_KEY**, 759–760
- orchestration, container, 182–193
- ordered clustered columnstore index, 747, 774
- orphaned SIDs, 605–608
- Outlook web mail, 396
- out-of-memory (OOM), 56, 282
- overcommitting memory, 89
- Overview, Activity Monitor, 33
- ownership**
 - AG replica, 503
 - authorization as, 597
 - chaining, 598–599
 - database, 218
 - migration and, 227
 - SA login, 577–578
 - total cost of, 915

P**Pacemaker**

- cluster management via, 75, 77, 79
- cluster setup, 540, 541
- configuration, 541
- external resource, 504, 514, 518
- login, 542
- scoring system, 540, 543
- package managers**, 195, 198
- packets, network**, 626
- page faults**, 377–378
- page files**, 116, 173
- page life expectancy (PLE)**, 376
- page reads**, 376–377
- page splits, tracking**, 332–333, 369
- PAGELATCH_* waits**, 356, 357
- pages**
 - automatic repair events, 545
 - compressing, 110–111
 - leaf-level and non-leaf-level, 110–111
- parallelism**, 116–119, 721–722, 733–735, 748
- Parameter Sensitive Plan (PSP) optimization**, 749–750
- parameter sniffing**, 715, 725–727
- parameterization**, 122, 622, 725–727
- partial restores**, 95, 480–481
- partition tables**, 108–109, 234
- partitioning**, 76, 295–303
- passwords**
 - administrator, 209
 - authenticating, 552
 - GPUs for cracking, 625
 - hashing to secure, 623–624
 - login, 143
 - policy enforcement, 560
 - SA passwords, 186
 - secure backups, 484
 - tips on, 209, 624
- patches**
 - applying, 164
 - failover cluster, 502
 - importance of, xxix
 - SSRS, 380
- performance**
 - alerts, 38
 - allocation unit size, 127
 - availability group, 545

- Azure Storage and, 801
- Azure VM, 797–798
- backups and, 426
- checkpoints for, 100
- durability settings, 99, 702–705
- failures, 72–82
- isolation levels and concurrency, 671–702
- memory-optimized table, 700–702
- metrics via DMOs, 371–384
- power-saving settings and, 63
- query execution plan, 705–735
- rows read, 710
- secondary replica impacts, 513–514
- separating Server files, 133
- simultaneous multithreading for, 60
- SQL Server 2022 upgrades, 736
- SQL Server Agent, 409
- stability vs., 58
- strategies, xxxiii
- table partitioning for, 296
- tempdb, 113
- tuning, 669–752
- Performance Monitor**, 120–121, 374–375
- perimeter security**, 621
- permissions**
 - accumulation of, 595–597
 - AG object creation, 508
 - authorization and, 593–605
 - cached plan metadata access, 732
 - data migration, 5–6
 - execution plan viewing, 712
 - migrating logins and, 608–614
 - role-based access control, 853–854
 - server privileges, 573–577
 - SQL Server Agent, 403
 - types of, 558
- persistent memory (PMEM)**, 64–65, 69, 70
- persistent volume claim (PVC)**, 184, 189, 190
- phantom rows**, 687–688
- plan cache**, 56
- planned failovers**, 516
- platform as a service (PaaS)**, 790, 796, 805–821, 829, 872
- Pods**, Kubernetes, 183
- point-in-time restores**, 478–480, 481

policy-based management (PBM), 431–434

PolyBase, 13, 141, 153, 309–323

polymorphism, 292

pools

- elastic pools, 795, 817–818, 844–845
- managed instance, 902–903
- resource, 387–389

ports, 71, 206, 314–315

Power BI, 151, 180

power saving, 63

power supply, 482

PowerShell

- administration, 434–444
- Azure SQL Managed Instance provisioning, 891–892
- database creation, 839–840
- function of, 199
- logical server creation, 833–834
- migrations, 929–933
- PowerShell 7, 437, 791
- Provider, 12
- scripting, 829
- VPN gateway via, 894

precedence, data type, 266

primary keys, 266–268

principles, database, 582–593

principles, security, 549, 555–593, 609, 620–631

privacy, 617–619

private key, 629–630

Private Link, 796, 850

privileges, 557–559, 591–593

procedure cache, 56, 57–58, 728–732

Processes, Activity Monitor, 33–34

processors, parallelism for logical, 117

production environment, 580

production instances, 19

Project Jupyter, 47

properties, database, 230–240

protocols, security, 620–631

provisioning

- Azure SQL Database databases, 836–844
- Azure SQL Database logical servers, 830–836
- Azure SQL Managed Instance, 534–536, 886–890
- elastic pools, 844–845

resources, 89–90

time, 875

pseudonymization, 618

public key, 629–630

purchase models, Azure, 806–808, 812

pushdown computation, 310

Python, 7, 8, 207

Q

queries

advanced tuning features, 735–752

Azure SQL Database elastic, 818–819

change tracking/data capture, 307–308

DMOs, 347, 348

execution plan, 702–705

graph data, 290

optimizing for ad hoc, 7, 122, 172

performance counters, 409

PolyBase, 310

temporal table, 279–281

Query Optimizer, 57, 233, 235, 705, 720

query plans

configuration settings and, 229

execution of, 670

parallel, 116–117

parameterizing, 122

tuning, 735–752

Query Store

configuration, 738

enabling, 746

forced query plans, 728

function of, 237, 737–742

hints, 742–745

parameterization hints, 727

replicas, 82

replicas on, 531

Query Tuning Assistant (QTA), 29, 237, 724

quorums

clustering and, 74

resolving partition issues with, 76

strategy, 507–508

WSFC and, 497–499, 516

R

R language, 7, 8

RAID storage, 66–67

rainbow tables, 624

random-access memory (RAM), 56, 89

ransomware attacks, xxx, 451–453

READ COMMITTED, 672–674, 679–680, 682–685, 689–696

read committed snapshot isolation (RCSI)

- columnstore index support, 773
- enabling, 236
- isolation levels and, 672, 673, 674, 691
- lock waits and, 355
- optimistic concurrency via, 263
- secondary databases, 525
- tempdb for, 113, 114
- workload, 696

read operations, 649–650

read scale-out replicas, 816–817

READ UNCOMMITTED, 673, 680, 689–691, 702

read-only replicas, 80, 237, 524–530

READPAST table hint, 681

reads

- nonrepeatable, 684–687
- phantom, 688–689
- read scale-out replicas, 816–817
- secondary replica, 743
- writes blocking, 683–684

read-scale availability groups, 80, 510

read-write redirection, 509

REBUILD index, 335, 336

Recent Expensive Queries, 35–36

recovery

- accelerated database recovery (ADR), 239–240
- backup types, 453–467
- checkpoints and, 100, 238–239
- hybrid/cloud environments, 484–485
- log truncation delays and, 101
- MinLSN, 103, 106
- model, 228, 232, 454–456
- preparation for, 446–451
- preparing Azure database for, 861–868
- ransomware attacks, 451–453
- restart recovery, 105–106
- scenario, 447–449, 482
- strategy design, 445, 481–485

recovery (*continued*)

- transaction log file corruption, 329–330
- See also* backups; disaster recovery; failures; restores

recovery model

- bulk-logged recovery model, 454, 455
- full recovery model, 454, 455, 481
- simple recovery model, 454, 456, 481

Recovery Point Objective (RPO), 55, 73–74, 446, 449–450, 473, 485, 487, 495, 811, 862–866

Recovery Time Objective (RTO), 55, 74, 104, 141, 414, 446–448, 450, 461, 475, 478, 482–483, 485, 487, 495, 545, 811

Red Hat Enterprise Linux (RHEL), 1, 3, 75, 154, 185, 199, 205, 540

Redo phase, 100, 105, 107

redundancy, 73–74, 877

referential integrity, 267

- primary and foreign, 266–268

RegisterAllProvidersIP, 528–530

registered servers, 23

relational tables, 290

relationships, 287

remote connection, 615

removal, database, 241–248

Remove-Item, 440

reorganizing indexes, 339–340, 418

replicas

- Azure SQL Managed Instance data, 905
- databases, 426–428, 503
- failover partners, 515
- limits for, 511
- minimum synchronized, 511
- performance impacts, 513–514
- provisioning Hyperscale, 843–844
- read scale-out, 816–817
- read-only, 524–530
- read-write traffic, 509–510
- secondary, 512–513, 743
- seeding options, 518–522

Report Server Configuration Manager, 18, 151

Reporting Services. *See* SQL Server Reporting Services (SSRS)

repositories, third-party, 204

reserved capacity, 809

resilience, migration, 937–938

Resource Governor, xxxii, 115, 384–389

Resource Waits, 34–35

resources

- Activity Monitor use of, 36
- auto close, 233
- Azure resource groups, 793–794
- Azure SQL Database limits, 812–813, 817
- name identifiers, 853
- provisioning, 89
- Resource Governor, 115, 384–389
- scalability, 794–795
- shared via VLANs, 71–72
- table-valued functions (TVFs), 59
- Uniform Resource Identifier, 313, 314
- virtual consumer, 88, 89
- wait, 352

restores

- automated restore scripts, 484
- Azure SQL Managed Instance, 876–879, 901
- backup, 220–221, 452
- data file corruption, 329
- database, 475–481
- database moves and, 243
- encryption and, 227
- partial, 480–481
- single-user mode, 247
- See also* failures; recovery

retention, long-term backup, 867–868

retry logic, 905, 938

right of access, 618

ring buffer, 374

role-based access control (RBAC), 155, 794, 850, 853–854

roles, database, 585–593

roles, server, 562–573

rolled back transactions, 98, 99, 105

root account, 199

rows

- compression of, 110
- concurrent updates, 681–682, 692
- Number of Rows Read, 710
- phantom, 687–688
- row-level security, 648–650, 913

rowstore data, 109, 262–263, 686, 746

rowstore indexes

- clustered, 754–758
- columnstore vs., 753–754

- nonclustered, 760–772
- rowstore clustered index, 754–758

RPO. *See* Recovery Point Objective (RPO)

RTO. *See* Recovery Time Objective (RTO)

run commands, 199

runbooks, 47, 450–451, 482, 483

runtime, integration, 933–937

S

S3-compatible object storage, 320–322, 472

SA passwords, 186, 577–578

salt, random, 623, 624

saves, data, 63

scalar UDF inlining, 750–752

Scale Out configuration, 149–150, 180

scaling up/out/down, 794–795, 817, 842–843, 905

scans, 717–718

scheduler flexibility, 124

scoped credentials, 315–316

secondary replicas, 743, 816–817

Secure Sockets Layer (SSL), 71

security

- admin tasks, xxxiii, 605–615
- auditing, 657–664
- Azure, 793
- Azure SQL Database, 846–861
- Azure SQL Managed Instance, 879–880, 908–911
- cloud, 826, 829
- CPU vulnerabilities, 60–61
- data platform protection, 631–653
- data protection, 911–913
- disaster recovery site, 484
- firewall, 846–849
- IaaS, 664–668
- IFI risk, 125–126
- ledger, 653–657
- logical server provisioning, 831
- Microsoft Defender for Cloud, 155
- Microsoft Defender for SQL, 859–861
- Microsoft Purview, 619–620
- migration, 937–938
- network, 70–71
- offsite backup, 483
- principles, 549, 555–593, 609, 620–631
- ransomware attacks, 451–453
- remote connection, 120

- row-level, 648–650, 913
- SQL Server Agent, 401–403
- TLS, 632–633
- Trustworthy setting, 228–229, 237
- worst practices, 589
- See also* access and authentication
- Security Technical Implementation Guide (STIG), 826**
- seek operators, 717
- seek time, 64
- self-hosted integration runtime, 935
- sensitive data, 619
- sequence objects, 270–273
- SERIALIZABLE isolation level, 673, 674, 679, 680
- Server Management Objects (SMO), 12**
- Server Power Options, 170**
- serverless compute tier, 810
- servers**
 - authentication for, 559–561
 - Azure Arc-enabled, 51, 154–155, 823
 - connecting to, 834–836
 - copying, 842
 - deleting, 836
 - firewall for, 846–849
 - groups, 22–23, 24
 - principles, 553
 - privileges, 573–577
 - provisioning logical, 830–836
 - role memberships, 611
 - self-hosted integration runtime, 935–936
 - Server Registration, 21–22
 - server-level collation, 143, 230–231
- service accounts, 131, 177, 500, 578–579**
- Service Broker, 399**
- service master key (SMK), 173–174, 634, 637, 638**
- service principal name (SPN), 83, 84**
- service tags, 795**
- service tiers**
 - basic, 862
 - business critical, 282, 718, 808–809, 816, 863, 880, 884–885, 904, 906
 - general purpose, 718, 808, 845, 863, 884–885, 903, 906
 - hyperscale, 808–812, 816, 843–844
 - premium, 282, 816, 863
 - standard, 718
- service tiers/objective**
 - Azure SQL Managed Instance, 884–886, 903–905
 - database copies and, 842
 - DTU, 807
 - performance and, 813
 - scaling down, 842
 - secondary database, 864, 865, 866
- Service-Level Agreement (SLA), 445**
- Services Manager, 13**
- servicing model, 389–391**
- SET TRANSACTION ISOLATION LEVEL, 679**
- settings. *See* configuration settings**
- setup**
 - Installation Center, 2–3
 - operating systems and, 1, 3
 - plan for upgrade/installs, 3–6
 - Windows SQL Server, 139–140
 - See also* installation
- setup.exe, 139–140, 159–162**
- sharding databases, 820–821**
- shared access signature (SAS), 313–315, 468–469, 931**
- SharePoint**
 - Integrated mode, 152
 - new database creation, 218, 416
 - Power Pivot for, 148
 - statistics creation, 233, 235, 784
 - unique identifiers and, 757
- shortcuts**
 - customizing, 29–31
 - SSMS keyboard, 28
- shredding, 787–788**
- shrinks**
 - auto, 234
 - file, 345–347
 - Shrink Database task, 418
 - when to shrink database, 418
- Simple Mail Transfer Protocol (SMTP), 394, 399**
- simple recovery model, 454, 456, 481**
- simultaneous multithreading (SMT), 60, 91**
- single-user mode, 247–248**
- site-to-site VPNs, 824**
- size, file, 342–347**
- sliding window partition strategy, 300–302**
- SMB 3.0 file share, 69**
- SMO, 359**
- SNAC: *See* SQL Native Client (SNAC)**
- snapshot isolation (SI), 235, 285, 692–700**
- snapshots, database, 462–463**
- snippets, 27–28**
- solid-state drives, 64, 113**
- space management, 845**
- sparse columns, 261, 274–275**
- spatial data types, 259–260**
- spatial indexes, 786–787**
- speculative execution vulnerabilities, 60–61**
- spinlock algorithms, 736**
- split brain. *See* partitioning**
- split-merge databases, 820–821**
- SQL Assessment API, 358–361**
- SQL Native Client (SNAC) 27, 529**
- SQLNCLI. *See* SQL Native Client (SNAC)**
- SQLNCLI11. *See* SQL Native Client (SNAC)**
- SQL Platform Abstraction Layer (SQLPAL), 212**
- SQL Server**
 - adding databases to, 215–240
 - audits, 657–663
 - authentication, 551
 - Azure SQL Database vs., 812–818
 - Azure SQL Managed Instance vs., 873–883
 - Azure Storage, 799, 801
 - Azure VM with, 138–139, 163
 - core counts by edition, 62–63
 - discontinued features, 53
 - editions, 59, 63, 135–137, 208, 226
 - encrypting, 630, 631
 - FCIs, 499–502
 - installation. *See* installation
 - Linux distributions support, 199–200
 - login migration, 611
 - maintenance, xxxii, 412–425
 - PowerShell and, 438–442
 - services, 209, 871
- SQL Server 2022**
 - Azure SQL Managed Instance compatibility, 874
 - new features, 230
 - performance improvements, 736
 - security features, 846
 - upgrades to, 136, 137

SQL Server Agent

- alternatives, 813
- automated admin via, 394, 400–412
- enabled on Linux, 203
- enabling, 400
- error log retention, 174
- event forwarding, 431
- function of, 37–40
- history retention, 175–176
- jobs, 400–401, 403–406, 411, 422
- Master/Target servers, 428–430
- security, 401–403
- services of, 813
- setup, 169

SQL Server Administration Inside Out

- book organization, xxi–xxiv
- signature tips, xxxv
- support for, xxxv
- text conventions, xxxiv

SQL Server Analysis Services (SSAS)

- alternatives, 814
- configuration, 180
- installation, 147–148
- memory limits, 165–166
- services of, 813
- version 2022, 7

SQL Server Configuration Manager

- enable availability groups with, 505
- enable startup trace flags with, 175
- import certificates with, 635
- manage services with 13
- manage service accounts with, 131, 578
- manage database files with, 242, 246,
- manage protocols and ports with, 71, 170
- manage SQL Server Agent with, 400

SQL Server Integration Services (SSIS)

- alternatives, 814
- FCI and, 494
- function of, 48–51
- installation, 148–149
- integration runtimes, 936–937
- Linux use of, 213
- login migration with, 609–610
- Scale Out configuration, 149–150
- services of, 813
- SSISDB, 176–177

SQL Server Management Studio (SSMS)

- Activity Monitor, 32–36
- additional tools, 27–31
- database creation with, 218, 219
- deprecated features, 27
- Error Logs, 31–32
- features of, 20–27
- function of, 18–40
- installation, 2, 19–20, 141
- keyboard shortcuts, 28
- log shipping, 492
- Maintenance Plan building, 424–425
- Master/Target server creation, 429
- model database connections, 218
- releases/versions, 19
- server audits in, 659, 660, 661–662
- server-level firewall, 849
- SQL Server Agent, 37–40
- SQLCMD mode, 10
- upgrades, 20
- v19 tools, 27
- vulnerability assessment from, 860
- XEvent Profiler, 362, 363, 364

SQL Server Migration Assistant (SSMA), 916, 925**SQL Server on Linux**

- authentication, 554
- caveats of, 212–213
- configuration settings, 206–212
- data protection on, 634
- installation, 1, 3, 200–213

SQL Server Reporting Services (SSRS)

- alternatives, 814
- configuration, 177–180
- FCI and, 494
- installation, 16–17, 150–151
- memory limits, 166–167
- patches, 380
- in-place upgrades and, 6
- Report Server Configuration Manager, 18
- report subscription emails, 399
- services of, 813
- SharePoint Integrated mode, 152
- version 2022, 7
- Web Portal URL tab, 17, 178
- web service URL, 177

SQL Server Setup

- automating, 158–163
- launching, 139–140
- logging files, 158

sql_variant data type, 265–266**SQLCMD utility, 9–10****SSISDB, 176–177, 225****Stand-Alone Installation, 140****Standard HDD storage, 799****static code, 47****statistics**

- index, 783–785
- maintaining, 330–342
- table, 233–235
- Update Statistics task, 420
- updating index, 340–341
- viewing live query plan, 711
- wait, 349–358

storage

- audit storage account, 855–858
- Azure disk, 798–800, 801
- Azure SQL Managed Instance backup, 876
- Azure VM, 132
- bandwidth, 800
- binary large objects (BLOBs), 292–295
- common terms, 63–64
- configurations, 65–70
- delayed durability, 80
- immutable, 653
- ledger, 654
- migration, 930
- procedure cache, 729
- provisioning virtual, 90
- S3-compatible object, 320–322, 472
- shared, 493
- space management, 845
- Storage Spaces, 68–69
- storage-area network (SAN), 68
- tempdb, 113
- transaction log file, 99–100, 102–103
- types of, 64–65
- usage for installation, 131–132
- volume settings, 134
- stored procedures, 283–285, 602–604**
- stretch clusters, 497**

strings, 250
 subnets, 664–665, 891–892, 898
 subscription settings, 179
 support
 distributed transaction, 511
 Kubernetes, 184
 life cycle, 390–391
 surface area settings, 168
 SUSE Linux Enterprise Server (SLES),
 3, 199, 205, 311, 504
 swap files, 116
 switches, 186
 synchronization, data, 79, 511
 sys.server_resource_stats, 906
 system databases, 215–216,
 241–242, 465–466
 system files, 245, 246, 399
 system libraries, 195
 system usage, 115
 system-versioned temporal tables,
 277–282

T

table variable deferred compilation,
 750
 tables
 accessing, 604
 binary large objects (BLOBs),
 292–295
 clustered indexes for, 754
 compressing, 109–112
 computed columns, 275–276
 constraints, 266–269
 on-disk, 784–785
 external, 318–319, 785
 graph, 287–292
 ledger, 655–657
 partitioning, 108–109, 295–303
 PolyBase for external, 309–323
 sequence objects, 270–273
 sparse columns, 274–275
 statistics, 233
 structures, 249–276
 table hints, 680
 temporal, 277–282
 See also memory-optimized tables
 tail-log backups, 457–458
 Target server (TSX), 428–430
 TARGET_RECOVERY_TIME, 239

TCP/IP, 169–170, 625–628
 tempdb
 collation, 230–231
 D: volume for, 132
 default settings, 144–145
 improvements to, 736–737
 rebuilding indexes via, 337
 space concerns, 696–697
 TDE's effect on, 641–642
 VM location, 802–803
 working area, 113
 temporal tables, 277–282, 308
 temporary database management,
 113–114
 test files, 926
 thick/thin provisioning, 90
 third-party repositories, 204, 827
 threads, max worker, 120, 121
 ticket granting server (TGS), 83,
 84, 85
 time data type, 256–258
 timeouts, 675
 toolbars, customizing, 29
 tools
 Database Engine, 7–14
 database migration, 222
 discontinued, 53
 elastic database, 818
 performance/reliability monitor-
 ing, 14–16
 SQL Server Data Tools (SSDT), 48–51
 SSMS, 18–40
 SSRS, 16–18
 Total Cost of Ownership (TCO) cal-
 culator, 915, 917–918
 trace flags
 Trace Flag 1118 and 1117, 114
 Trace Flag 1800, 135
 Trace Flag 3226, 175
 tracking changes, 303–308
 traffic redirection, 509–510, 526–528
 transaction log files
 active log, 106
 adding, 218–219
 benefits of IFI to, 125
 corruption repair, 329–330
 data files and, 93, 94
 Maintenance Plans, 416
 moving, 241

 recording changes via, 99–108
 shrinking, 346–347
 tempdb, 114
 transaction log backups, 457,
 475, 477
 volume settings, 134
 transactions, distributed, 881–882
 Transact-SQL functions, 112
 transient connectivity issues, 905
 Transmission Control Protocol
 (TCP), 71
 transparent data encryption (TDE),
 227, 466, 521, 639–642, 846
 Transparent Hugepages (THP), 201
 Transport Layer Security (TLS), 71,
 178, 632–633, 888, 937
 Triple Data Encryption Standard
 (3DES), 637
 troubleshooting
 antivirus interference, 171–172
 Database Mail, 398–399
 Extended Events, 15
 log files for, 158
 options, 240
 Query Store, 739
 TRUNCATE command, 594
 Trustworthy setting, 237
 T-SQL code
 changes to, 223
 database creation with, 841–842
 login migration, 610–612
 server audits with, 660, 661, 662
 statement task, 423
 tuning query plans, 735–752
 U
 Ubuntu, 1, 3, 75, 199, 205
 Undo phase, 105, 107
 unforced failovers, 904
 Unicode, 250, 251
 uniform extents, 95, 114
 Uniform Resource Identifier (URI),
 313, 314, 622
 unique constraints, 267, 268
 unique identifiers, 264, 756–757
 Universal Authentication with
 MFA, 551
 unmanaged disks, 799
 updateable ledger tables, 655. *See*
 also ledger

updates

- antivirus interference, 171–172
- applying patches, 164
- auto update statistics, 234–235
- concurrent, 681–682
- cumulative, 217
- index statistics, 340–341, 783
- isolation conflicts, 697–700
- packages on Linux, 205
- Update Statistics task, 420
- Windows Update, 140

upgrades

- compatibility levels, 237
- Management Studio, 20
- in-place, 6
- plan for, 3–6
- Query Tuning Assistant (QTA), 29
- rolling, 502
- SSISDB database, 225

usage

- system, 115
- volume, 131–133

user files, 244**user-defined data types (UDTs), 273–274****user-defined service roles, 572–573****user-defined traffic routes, 666–667, 899****users**

- Azure AD, 41–43, 910
- Azure interface, 791
- B2B/guest users, 848–849
- concurrent updates by, 681–683
- contained, 584, 880
- database, 582–585
- login identifiers, 556–557
- orphaned SIDs, 605–608
- right of access/erasure, 618–619
- single-user mode, 247
- user-defined data types, 273–274
- See also* access and authentication

UTF-8 collation support, 231–232, 251–252**V****varbinary(max) data type, 253, 258, 293, 294****varchar data type, 250****vCore purchasing model, 808, 812, 884****verification**

- backup, 474
- data change, 686
- data page, 98, 228, 236, 326
- ledger, 653

vertical partitioning, 302–303**vertices, 287****very large databases (VLDBs), 453, 462, 468, 810****views**

- audit log, 662–663
- DMVs, 409
- execution plan, 708–712
- ledger, 656–657
- partitioning, 109
- PolyBase, 319–320
- testing permissions with, 600–602

virtual consumers

- hardware sharing for, 87, 88
- networking for, 92
- provision resources for, 89–90

virtual CPU (vCPU), 91**virtual IP resource, 542–543****virtual local area networks (VLANs), 70, 71–72****Virtual Log File (VLF), 100–101, 346, 355, 736****virtual machines (VMs)**

- Azure portal management, 803–805
- Azure SQL Managed Instance connections, 896
- containers and, 56, 87
- D: volume on, 132
- disabling SMT for, 60, 61
- Enterprise edition for, 63
- installing separate, 203
- performance of Azure, 797–798
- See also* Azure VMs

virtual network name (VNN), 75, 496**virtual network (VNet)**

- Azure, 795
- Azure SQL Managed Instance, 891–892
- database integration with, 849–850
- NSGs for, 664–666

- peering in, 535
- security, xxx, 651, 652
- servers associated with, 846
- service endpoints, 667

virtual private network (VPN), 824, 825, 894**virtualization**

- containers and, 87–92
- data, 153, 309
- defined, 87
- simultaneous multithreading (SMT), 60

Visual Studio Code, 436, 442, 926**volumes**

- Grant Perform Volume Maintenance Tasks, 141–142
- usage plan, 131–133

votes, quorum

- dynamic management and, 499
- forced failovers and, 516–517
- majority, 74, 76, 498
- WSFC setup and, 507–508

VSS Writer, 7**vulnerability assessment, 652, 860****W****wait types and statistics, 349–358, 546****Windows**

- authentication, 550, 583, 610, 911
- command line tool updates, 12
- Linux vs., 196–199, 212–213, 537–538
- PowerShell, 437
- Server Power Options, 170
- SQL Server setup, 129
- Windows Registry, 212
- Windows Services Manager, 13
- Windows Update, 140

Windows Management Instrumentation (WMI) alerts, 39, 169, 407, 410**Windows Server Failover Clustering (WSFC)**

- AG operation via, 79
- availability groups and, 507–508
- defined, 75, 504
- function of, 492, 493
- majority votes, 76

quorums and, 497
witness, 498
working set, 57, 201
workload groups, 115, 387–389
write-ahead logging (WAL), 99
writes, reads blocked by, 683–684

X

xml data type, 260–262
XML indexes, 787–788

Y

YAML files, 183, 190–192

Z

zeroing out, 106, 125
Zero Trust model, xxix, xxx
zone-redundant configuration, 837,
863, 876, 877, 887