The ultimate in-depth reference

Hundreds of timesaving solutions

Supremely well-organized, packed
with expert advice

Microsoft

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

Microsoft

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.

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

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

Part I: Introduction

Part II: Deployment

Part III: SQL Server management

Part IV: Security

Part V: Performance

Part VI: Cloud

Table of Contents

Part I: Introduction

Part II: Deployment

Part III: SQL Server management

Part IV: Security

Part V: Performance

Part VI: Cloud

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 *accitentionaldba.com* and post more often but with much less focus on Twitter as *@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*.

CHAPTER 4

Install and configure SQL Server instances and features

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 shutdown 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 premade 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 autogrowth 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.

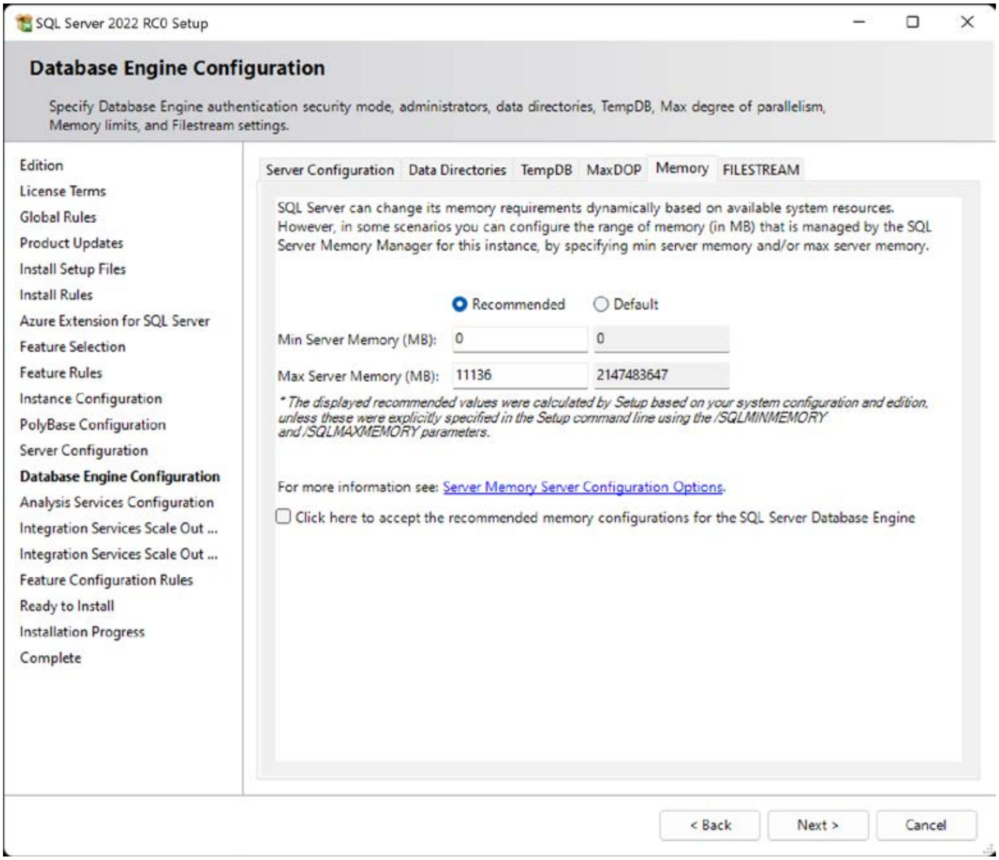
You can configure this Max Server Memory option intelligently at the time of installation. It's important to note (and there's a check box to accept this guess) that SQL Server Setup assumes this SQL Server instance will run alone on this server. If you expect to host other applications on this server, 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.



Database Engine Configuration

Specify Database Engine authentication security mode, administrators, data directories, TempDB, Max degree of parallelism, Memory limits, and Filestream settings.

SQL Server can change its memory requirements dynamically based on available system resources. However, in some scenarios you can configure the range of memory (in MB) that is managed by the SQL Server Memory Manager for this instance, by specifying min server memory and/or max server memory.

● Recommended ○ Default

Min Server Memory (MB): 0 0

Max Server Memory (MB): 11136 2147483647

** The displayed recommended values were calculated by Setup based on your system configuration and edition, unless these were explicitly specified in the Setup command line using the /SQLMINMEMORY and /SQLMAXMEMORY parameters.*

For more information see: Server Memory Server Configuration Options.

□ Click here to accept the recommended memory configurations for the SQL Server Database Engine

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 `HttpsCertThumbprint` 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.

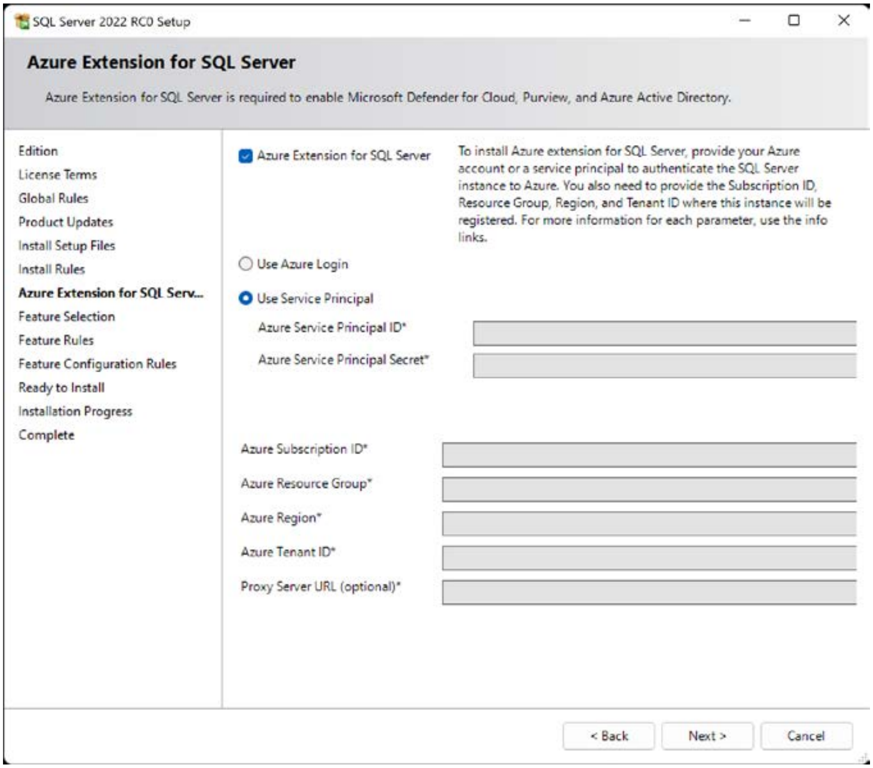
➤ **For information on creating a Microsoft Purview account, see *https://learn.microsoft.com/azure/purview/create-catalog-portal*.**

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.



SQL Server 2022 RC0 Setup

Azure Extension for SQL Server

Azure Extension for SQL Server is required to enable Microsoft Defender for Cloud, Purview, and Azure Active Directory.

☑ Azure Extension for SQL Server

To install Azure extension for SQL Server, provide your Azure account or a service principal to authenticate the SQL Server instance to Azure. You also need to provide the Subscription ID, Resource Group, Region, and Tenant ID where this instance will be registered. For more information for each parameter, use the info links.

○ Use Azure Login

● Use Service Principal

Azure Service Principal ID*

Azure Service Principal Secret*

Azure Subscription ID*

Azure Resource Group*

Azure Region*

Azure Tenant ID*

Proxy Server URL (optional)*

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
```

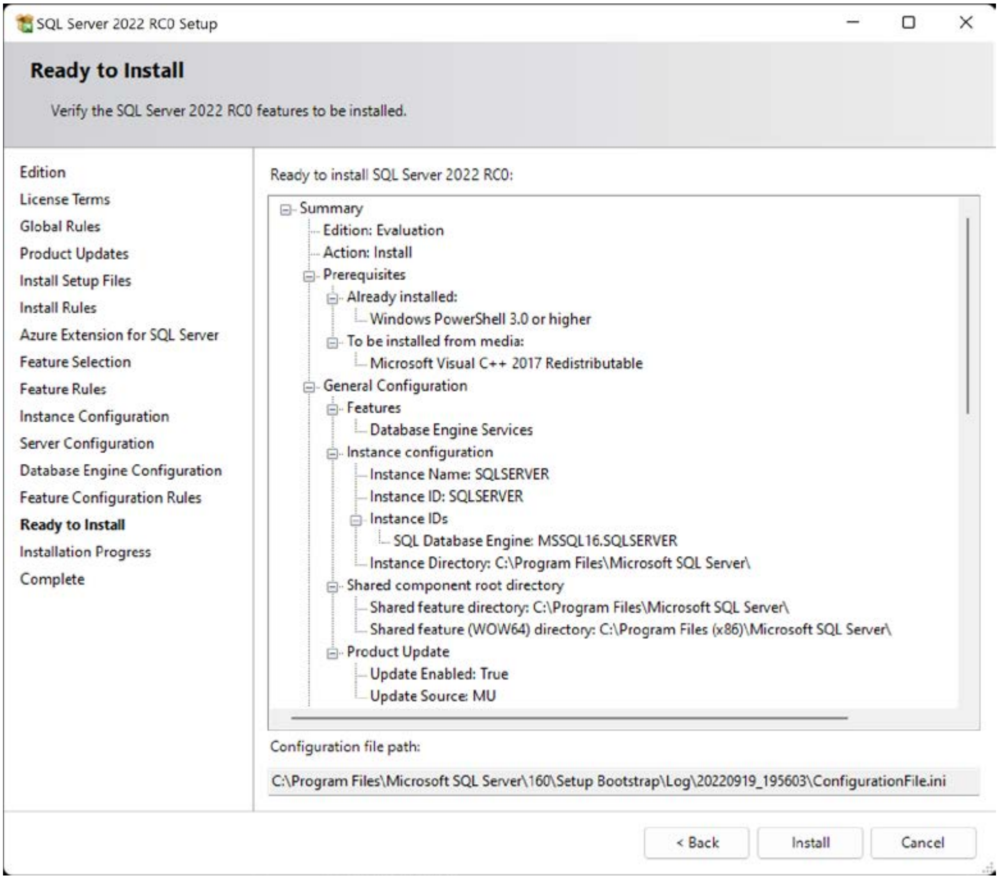
This sample script, and all scripts for this book, are available for download at *https://www.MicrosoftPressStore.com/SQLServer2022InsideOut/downloads*. The preceding code sample uses a configuration file to pre-select installation choices—for example, features to be installed. Let's talk more about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



Ready to Install

Verify the SQL Server 2022 RC0 features to be installed.

Ready to install SQL Server 2022 RC0:

```
Summary
  Edition: Evaluation
  Action: Install
  Prerequisites
    Already installed:
      Windows PowerShell 3.0 or higher
    To be installed from media:
      Microsoft Visual C++ 2017 Redistributable
  General Configuration
    Features
      Database Engine Services
    Instance configuration
      Instance Name: SQLSERVER
      Instance ID: SQLSERVER
      Instance IDs
        SQL Database Engine: MSSQL16.SQLSERVER
      Instance Directory: C:\Program Files\Microsoft SQL Server\
    Shared component root directory
      Shared feature directory: C:\Program Files\Microsoft SQL Server\
      Shared feature (WOW64) directory: C:\Program Files (x86)\Microsoft SQL Server\
    Product Update
      Update Enabled: True
      Update Source: MU
```

Configuration file path:

C:\Program Files\Microsoft SQL Server\160\Setup Bootstrap\Log\20220919_195603\ConfigurationFile.ini

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	`/PBDMSSVCPASSWORD`	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.

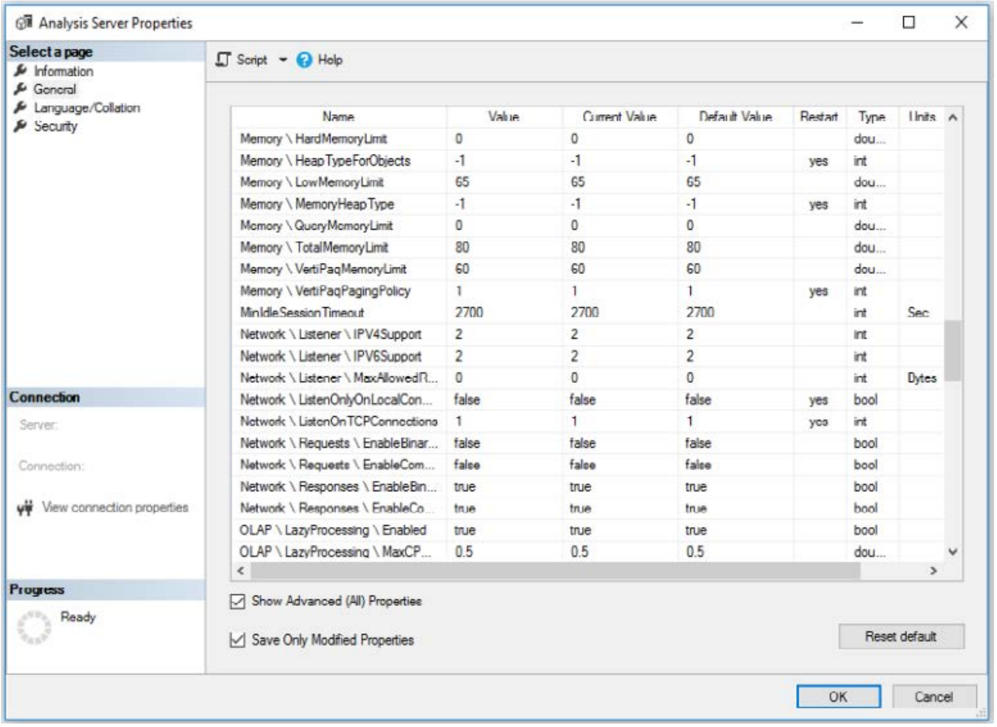
Limit SSAS memory

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, `DbpropMsmdRequestMemoryLimit`, 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.



Name	Value	Current Value	Default Value	Restart	Type	Units
Memory \ HardMemoryLimit	0	0	0		dou...	
Memory \ HeapTypeForObjects	-1	-1	-1	yes	int	
Memory \ LowMemoryLimit	65	65	65		dou...	
Memory \ MemoryHeapType	-1	-1	-1	yes	int	
Memory \ QueryMemoryLimit	0	0	0		dou...	
Memory \ TotalMemoryLimit	80	80	80		dou...	
Memory \ VertiPaqMemoryLimit	60	60	60		dou...	
Memory \ VertiPaqPagingPolicy	1	1	1	yes	int	
MinIdleSessionTimeout	2700	2700	2700		int	Sec
Network \ Listener \ IPV4Support	2	2	2		int	
Network \ Listener \ IPV6Support	2	2	2		int	
Network \ Listener \ MaxAllowedR...	0	0	0		int	Bytes
Network \ ListenOnlyOnLocalCon...	false	false	false	yes	bool	
Network \ ListenOnTCPConnections	1	1	1	yes	int	
Network \ Requests \ EnableBinar...	false	false	false		bool	
Network \ Requests \ EnableCom...	false	false	false		bool	
Network \ Responses \ EnableBin...	true	true	true		bool	
Network \ Responses \ EnableCo...	true	true	true		bool	
OLAP \ LazyProcessing \ Enabled	true	true	true		bool	
OLAP \ LazyProcessing \ MaxCP...	0.5	0.5	0.5		dou...	

☑ Show Advanced (All) Properties

☑ Save Only Modified Properties

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.instancename\MSSQL\Binn\rlauncher.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 unwieldly 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 -T*flagnumber*. 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 preallocating 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.**

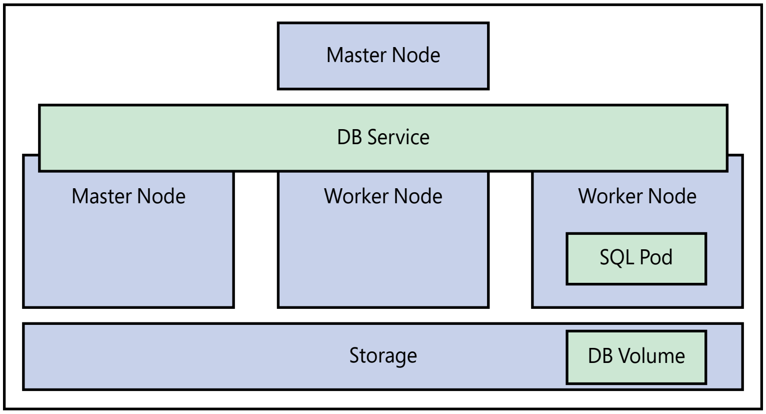


Figure 4-5 SQL Server on Kubernetes architecture.

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/randolph/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

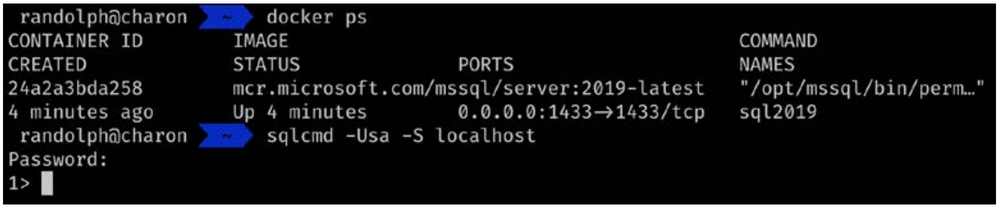
Docker on macOS does not support persistent volumes. Microsoft recommends that you use separate data container volumes to persist data files that are stored in /var/opt/mssql/data. You can read the background on this issue at *https://github.com/microsoft/mssql-docker/issues/12*, and you can learn more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

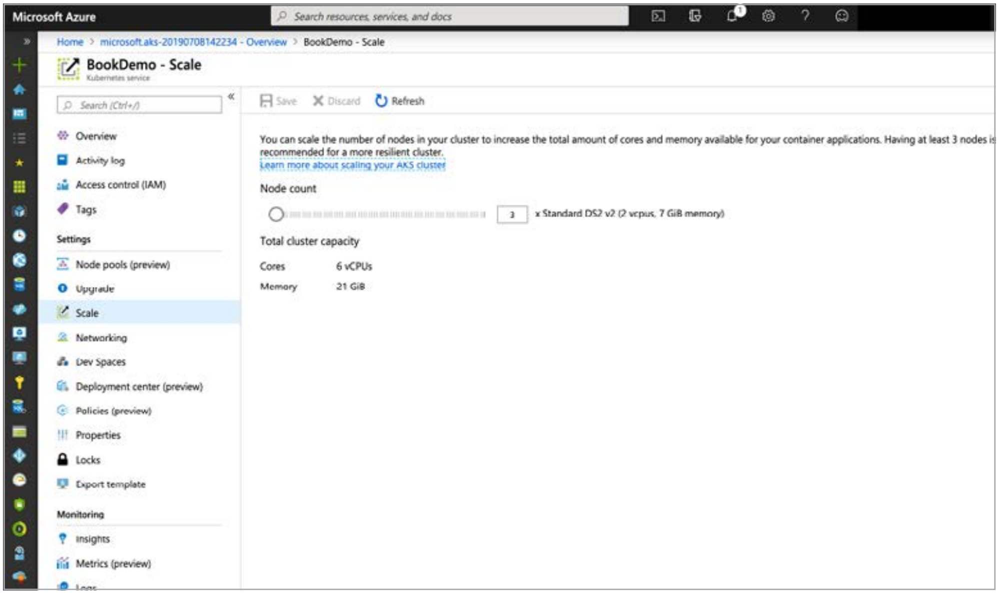
Another commonly used option for development and testing of Kubernetes is minikube, which runs across Windows, Linux, and macOS. minikube is an open-source project that allows for a deployment to your local workstation.

➤ **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*.**



You can scale the number of nodes in your cluster to increase the total amount of cores and memory available for your container applications. Having at least 3 nodes is recommended for a more resilient cluster.

Node count

3 x Standard DS2 v2 (2 vcpus, 7 GiB memory)

Total cluster capacity

Cores 6 vCPUs

Memory 21 GiB

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.

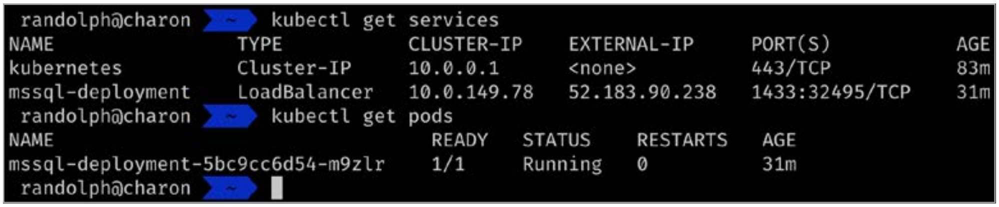
The next step is to deploy the SQL Server service and the pod itself. In the following code, you specify the load balancer service as well as the container running SQL Server. Kubernetes can use extensive metadata to describe and categorize your environment, as you will note from the metadata and label 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.

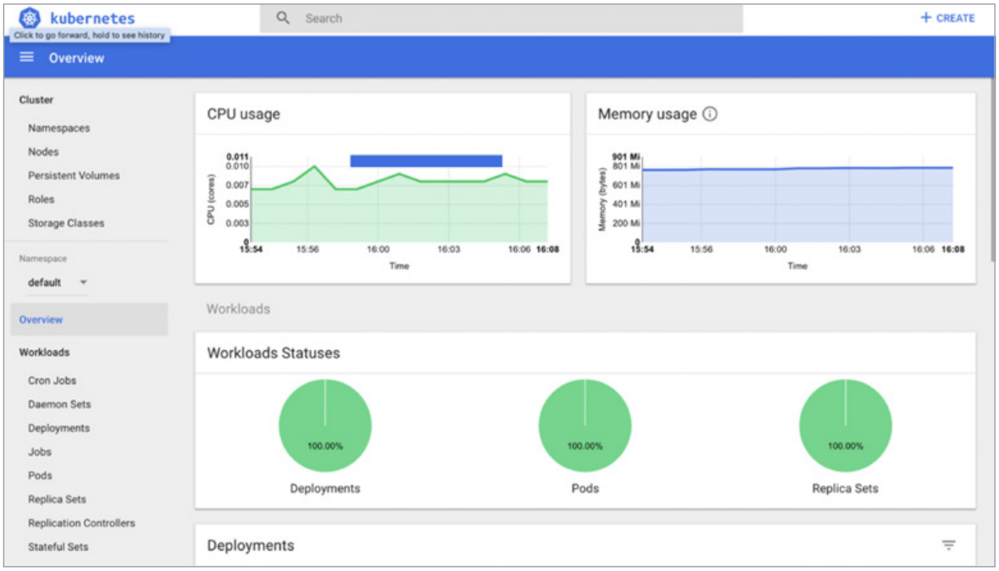


Figure 4-9 A screenshot of the Kubernetes web dashboard.

➤ **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