Microsoft

Inside OUT

The ultimate in-depth reference

Hundreds of timesaving solutions

Supremely organized, packed with expert advice

SQL Server 2019 Administration

Randolph West ▪ Melody Zacharias ▪ William Assaf
Sven Aelterman ▪ Louis Davidson ▪ Joseph D'Antoni

Microsoft

SQL Server 2019 Administration

Randolph West
Melody Zacharias
William Assaf
Sven Aelterman
Louis Davidson
Joseph D’Antoni

SQL Server 2019 Administration Inside Out
Published with the authorization of Microsoft Corporation by:
Pearson Education, Inc.

ISBN-13: 978-0-13-556108-9
ISBN-10: 0-13-556108-6

Library of Congress Control Number: 2020930782

ScoutAutomatedPrintCode

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.

To Marinus (obviously), but also Larry Toothman, Robert Davis, and Ian van Schalkwyk, because I miss you.

– Randolph West

To my husband Chris and daughters Jessica and Chloe for all your patience and understanding.

– Melody Zacharias

To Christine from her biggest fan

– William Assaf

In Memory of Erick Aelterman
Your example of hard work and perseverance made this possible

– Sven Aeleterman

To the rest of the writing and tech editing team. This edition of the book has been so much more of a joy due to everyone's hard work and commitment to the project. I am very proud to have been chosen to not only tech-edit half of this book, but to also take charge of revising two chapters.

– Louis Davidson

I'd like to thank the SQL Server Product Group for their hard work on the product.

– Joseph D'Antoni

To my dog Buster. Thanks for being patient and dealing with fewer dog park visits while I edited this book.

– Meagan Longoria

Contents at a Glance

Part I Introduction

Part II Deployment

Part III SQL Server management

Part IV Security

Part V Performance

Part VI Cloud

Part VII Big Data and Machine Learning

Table of Contents

About the authors

Randolph West (they / them) is a technologist and Data Platform MVP from Calgary, Alberta, Canada, and hates writing in the third person because pronouns suck. Randolph is the founder of the Calgary Data User Group, and leader of the Calgary PASS User Group. You can see Randolph presenting at various conferences around the world, and acting on stage and screen. Randolph specializes in implementing best practices, performance tuning, disaster recovery, cloud migrations, software development, and yelling at Microsoft Word. If you're interested in becoming a public speaker, check out SpeakingMentors.com. Blog: bornsql.ca/blog. Twitter: @_randolph_west. Not to be trusted around chocolate.

Melody Zacharias has worked with SQL Server since version 7, and she is very passionate about all things data! As a Microsoft MVP Melody loves to share her passion with users around the world by teaching at events, working within the SQL community, and mentoring. When not working with data she loves to hike with her dog in the Canadian Rockies. You can find her on her blog at SQLMelody.com where she shares musings on how to make SQL sing, or sometimes just carry a tune, or on Twitter @SQLMelody.

William Assaf (he/him) is a Microsoft SQL Server consultant, speaker, and manager, and blogs about SQL at sqltact.com. William has been a developer, admin, and consultant for databases around the world, having launched from in-house DBA to consulting DBA thanks to the recession. He has helped develop Microsoft SQL Server certification exams since 2012, has been a Regional Mentor for PASS since 2015, and just stepped down from being the lead organizer of SQLSaturday Baton Rouge after a decade leading one of the largest SQLSat events in the world, and the only one with jambalaya. William enjoys volunteering for STEM initiatives as well as guest lecturing at LSU and SELU, where he gives a shoutout to all the ISDS and CIS students. William lives in continuous awe of his far more accomplished high school sweetheart Christine, and they would like to say that empty nesting is pretty great. Both of them hope to see you at PASS Summit and a future SQLSaturday event.

Sven Aelterman started with SQL Server when he first deployed version 2000 in a failover cluster scenario. Since then, he has worked as IT manager, principal consultant, and IT director. He currently serves the Trojans (students) of Troy University as a lecturer in information systems in the Sorrell College of Business and as director of IT for the College. In addition, he is cloud software architect for Sorrell Solutions, a business services nonprofit through which Trojans can gain real-world business and IT experience. In a fledgling attempt to give back to the community, he has spoken at many SQL Saturdays and code camps in the southeastern United States since 2005. He spoke about SSIS 2012 at Microsoft TechEd 2012. In 2012, he co-authored a book dedicated to SQL Server FILESTREAM. He was a co-author on the 2017 edition of this text. His involvement with Microsoft Azure resulted in the organization of four Global Azure Bootcamp events at Troy University. Sven blogs about a variety of Microsoft technologies at blog.aelterman.com and tweets and retweets about technology @svenaelterman.

Louis Davidson is a data architect for CBN in Virginia Beach, VA; telecommuting from Cleveland, TN (which is not even as glamourous as it sounds.) Louis has written and contributed to many books on SQL Server topics over the past 20 years. His most prominent work has been five editions of his book entitled: "Pro SQL Server Relational Database Design and Implementation" for Apress in 2016, with a new version forthcoming in 2020. Louis has been a speaker at many SQL Saturday events, and has helped organize events in Nashville and Chattanooga, TN. He blogs on Red-Gate's Simple-Talk website here:
https://www.red-gate.com/simple-talk/author/louis-davidson/ and you can visit his website at http://drsql.org.

Joseph D'Antoni is a Principal Consultant and Microsoft Data Platform MVP with over 20 years of experience working in both Fortune 500 and smaller firms. He has worked with SQL Server and Oracle since version 7 of each database and has specific expertise in performance tuning, infrastructure, and disaster recovery.

Meagan Longoria is a Microsoft Data Platform MVP living in Denver, Colorado. She is an experienced consultant who has worked in business intelligence, data warehousing, and database development for over a decade. She enjoys creating solutions in Azure and SQL Server that make data useful for decision makers. 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).

Introduction

Much has changed in the months leading up to the publishing of Microsoft SQL Server 2019 Administration Inside Out, a follow-up edition to this team's successful and well-reviewed Microsoft SQL Server 2017 Administration Inside Out. Among just our writing and editing team, one became an empty nester, another got a new knee, another two new hips. One was evacuated because of forest fires, another left powerless under a hurricane, another twice moved houses, others got new jobs and job roles. Countless naming conventions and product names were changed in the Microsoft data platform, too, causing many late edits to this book. Azure SQL Database managed instance was made generally available, as well as the concepts of Big Data Clusters, Azure DevOps, Azure SQL Database Hyperscale and Azure SQL Database serverless tier, Azure Arc, Azure Cosmos DB, and more. SQL Server 2008 and 2008 R2 hit their end of life support dates and are no longer supported by Microsoft*, new streaming services were launched, the Washington Nationals won the World Series, LSU beat Alabama in football, and Avengers: Endgame finally beat Avatar as the highest grossing film of all time… so yeah, a lot of long-standing states of information have changed.

While no book can keep up with all the changes Microsoft has made even to just the data platform and supporting technologies in Azure, this book does its best to fully outer join the latest guidance on modern data platform solutions with long-standing database administrator wisdom. We'll cover both columnstore and rowstore indexes. Both Linux and Windows. Both Azure and on-prem solutions (and hybrid). Both memory-optimized tables and disk-based tables, with optimistic and pessimistic isolation. This book aims to be a reference, inspiration, early radar warning, or at least a launch pad, for your continued education and data platform solutions development.

Data warehousing has changed, with the traditional extract-transform-load (ETL) strategies giving way to extract-load-transform on-demand (ELT) strategies. SQL Server 2019 has new features and tools to develop these solutions while leveraging your existing Transact-SQL skillset. Features like Big Data Clusters and PolyBase external tables are helping you design fast, scalable, modern solutions where once you had to rely on external products, custom solutions in SQL Server Integration Services (SSIS), or linked server objects. All these solutions have their place, and all will continue to be in the inventory of your DBA utility belt, but your ensured success as a modern, relevant DBA requires adding tools. This book, and its large accompanying repository of sample scripts, aims to be that reference to DBAs as they add modern tools to their skill set.

Threat vectors and regulatory requirements have also changed. That's why SQL Server 2019 has vastly improved data classification capabilities leveraging a new dedicated schema, as well as Always Encrypted now with secure enclaves. There's also a new certificate manager built-in to SQL Server Configuration Manager, and new integrations with Azure Key Vault. We've expanded the SQL Server security and protecting data chapters greatly from the last book, recognizing the DBA's role as key to an enterprise data security team.

This book's content, and especially the Inside OUT features, are derived directly from the skills and experience of a group of seasoned database professionals with many decades of experience in designing, optimizing, developing, and teaching complex modern data platform solutions. It is written for experienced DBAs and developers.

After reading this book, you will be able implement a cloud-based, hybrid, and on-premises data solution using SQL Server 2019.

The velocity of change for the Microsoft SQL Server DBA has increased this decade. Gone are the days when DBAs had between three and five years to soak in and adjust to new tools and tricks in the Database Engine and surrounding features. We had less than a calendar year between the release of SQL Server 2017 and the first technical previews of SQL Server 2019.

This book is written and edited by SQL Server experts with two goals in mind: to deliver a solid foundational skillset for all of the topics covered in SQL Server configuration and administration, and also to deliver awareness and functional, practical knowledge for the dramatic number of new features introduced in SQL Server 2019. We haven't avoided late-breaking content—even content that stretched the boundaries of writing deadlines with late-breaking new releases. You will be presented with not only the "how" of new features, but also the "why" and the "when" for their use.

**There are still a couple ways to get support for SQL Server 2008 / 2008 R2. One involves paying a lot of money, the other involves migrating it all to Azure VMs. Neither are as good an option as just upgrading it already!*

Who this book is for

SQL Server administration was never the narrow niche skillset that our employers might have suspected it was. Even now it continues to broaden, with support for new operating systems and platforms: cloud-based and serverless in addition to on-premises, or maybe, a little of all three. This book is for the DBAs 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. SQL Server administrators should read this book to become more prepared and 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 SQL Server Administration, whether you are implementing, developing, or supporting development.

Part I: Introduction

Chapter 1, “Getting started with SQL Server tools” gives you a tour of modern tooling for SQL Server administrators, from the installation media to the free downloads, not the least of which is the rapidly evolving SQL Server Management Studio and Azure Data Studio, as well as the newly enhanced Configuration Manager, performance and reliability monitoring tools, tools for writing PowerShell, and more.

Chapter 2, “Introducing 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, “Designing and implementing 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), and the new Accelerated Database Recovery feature, one of many new features of SQL Server 2019. We spend time with TempDB and its optimal configuration, and server-level configuration options. Finally, we introduce you to Kubernetes.

Part II: Deployment

Chapter 4, “Installing and configuring SQL Server instances and features” reviews installation of SQL Server for Windows platforms when SQL Server Setup is needed to install SQL Server. We discus 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, as well as configuration and guidance for other features including SSIS, SSAS, and SSRS, as well as PolyBase.

Chapter 5, “Installing and configuring 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, “Provisioning and configuring 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 well as database collations.

Chapter 7, “Understanding table features” completes the drill down from instances to databases to tables, covering table design, data types, keys and constraints. Use of IDENTITY and sequences, computed columns and other column properties, as well as special table types are discussed. We’ll review special types of tables including temporal tables, introduce memory-optimized tables (more on these in Chapter 14), and graph tables. Finally, we’re reviewing FILESTREAM and FileTable for storing blobs, table partitioning for storing and switching large amounts of data, and strategies for tracking data changes.

Part III: SQL Server management

Chapter 8, “Maintaining and monitoring 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 also 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 new SQL Assessment API. We’ll finally discuss the current Microsoft servicing model for SQL Server.

Chapter 9, “Automating SQL Server administration” includes an introduction to 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, Master/Target Agent jobs, proxies, SQL Agent alerts, event forwarding and Policy-Based Management.

Chapter 10, “Developing, deploying, and managing 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. Backups and restores in a hybrid environment, Azure SQL Database recovery, and geo-replication are important assets for the modern DBA.

Chapter 11, “Implementing high availability and disaster recovery” then goes beyond backups and into strategies for disaster recovery from the old (log shipping and replication) to the new (availability groups), as well as monitoring and troubleshooting availability groups. We’ll compare HA and DR strategies and dive into proper architecture for maximizing SQL Server uptime.

Part IV: Security

Chapter 12, “Administering 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, “Protecting data through encryption, privacy 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 including Transparent Data Encryption (TDE) and Always Encrypted, as well as protecting data in motion with TLS. The new Certificate Management feature of SQL Server 2019 Configuration Manager is reviewed, as well as 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 tuning 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'll 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 review important performance-related Dynamic Management Objects (DMOs) and several new SQL Server 2019 performance features, most notably in the Intelligent Query Processing family.

Chapter 15: "Understanding and designing indexes" tackles performance from the angle of indexes, from 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 hash. We review statistics and statistics options, including understanding how they work on a variety of index and table types.

Part VI: Cloud

Chapter 16, "Designing and implementing 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 also Infrastructure-as-a-Service 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 or PowerShell.

Chapter 17, "Provisioning Azure SQL Database" covers the cloud-first database service with a very high degree of compatibility with SQL Server 2019. This platform powers many web-based applications and services, scalable from basic $5/month to 80-vCore and 100 TB. You will learn about the Azure SQL database platform, compatibility, security, and availability. You will learn how to create servers, databases, and elastic pools, and how to perform important management tasks for your databases.

Chapter 18, "Provisioning Azure SQL Database managed instance" details the powerful Azure SQL Database 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, "Migrating 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-prem feature migration strategies to Azure platforms, including how to migrate SSIS packages to the Integration Runtime. We review post-migration steps, best practices for security and resiliency during migration, and the common causes for migration failures.

Part VII: Big Data and Machine Learning

Chapter 20, "Leveraging Big Data and Machine Learning" starts with a primer on big data features in SQL Server, including an introduction to Big Data Clusters. We're review how to operationalize analytics with machine learning services and the benefits of PolyBase external tables for direct access to external data sources, even non-relational data sources.

Acknowledgments

Randolph West They say it's easier the second time around. They are liars. Fortunately, I managed to rope in Melody Zacharias as payback for getting me involved in the last book. Cue evil laugh. After sitting at the kitchen counter for most of the last book, my husband bought me a desk which I never sit at. Thanks anyway! And thank you to my dog Trixie for taking me on long walks to think about better phrasing to keep Meagan happy. I would also like to give special thanks to Anthony Nocentino, Argenis Fernandez, and Glenn Berry for their help, technical advice, and support. Since we're naming names, here's a shout-out to the book team: Meagan, Melody (again), Loretta, Liv, Louis, William, Sven, and Joey. Each of you deserves a medal. This book would not be possible without the contributions of everyone involved.

Melody Zacharias I really want to thank the #SQLFamily for their inspiration, support, friendship and guidance. Particularly Argenis Fernandez who was my PASS Summit big buddy and introduced me to the community in 2011. Because of that experience, my life was forever changed 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.

William Assaf I'd 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, my personal dream team of coauthors on this very book, and finally Patrick Leblanc, a fellow Baton Rouge native, whose friendship has repeatedly challenged and furthered me and my career. I'd also like to thank my father, a rare mechanical and electrical engineer (and a HAM), and both my older brothers who are brilliant software engineers, for repeatedly letting me play games on their computers. I'd finally like to thank Kim Fossey, Casey Phillips, Kristen Reeves, Helena Williams, Cheryl Cummings, Ajayi Anwansedo, Quinton Jason Jr, and other heroes of the STEM education in Baton Rouge, La., and all over. They are doing the hard work of developing our future coworkers and coauthors in my home state's perpetually underfunded, underappreciated, and underestimated public school youth.

Sven Aelterman There are a lot of people who have made my contributions to this book possible. They include those who have given me chances early in my career, enabled me to study abroad, and encouraged career development and growth. In no particular order, I want to recognize Walter Weyne, Yves Sucaet, Judson Edwards, Bill Belcher, Hank Findley, David Phelps, Janet and Jeff Kervin, and the late Don Hines. This "old" and expanded group of co-authors have held me to high standards, as have the editors. Speaking of editors, they're an odd breed. During the writing process: entirely unnecessary; my initial draft is perfect, of course. After the first review round, they're know-it-all, nit-picky experts who just don't see it the way it is. By the second review round, they're usually correct. And when the book is finished, it turns out they were invaluable. Thanks, Meagan and Louis! Most importantly, my family continues to be the driving force behind what I do every day. It's not easy for Ebony, Edward, and Sofia to put up with the late nights of learning and writing. But they do it, with love and a smile (most of the time). I want them to know their support and love are valuable, valued, and acknowledged

Louis Davidson Acknowledgements are super hard, not because I am so great and have no need for others, but rather because I have been surrounded by scores of people who are amazing for so long. The authors of this and the previous edition have taught me a ton along the way. I could not have survived the writing and tech editing processes without the writers on the docs.microsoft.com site and the many blogs in the #sqlfamily community. Additionally, I must acknowledge the PASS and the Microsoft MVP program, because they have helped me to expand my horizons throughout the past 20 years, helping me to get to the point I have reached in my career.

Meagan Longoria I'd like to thank my coworkers at Denny Cherry & Associates Consulting. I learn about the Microsoft Data Platform from them every day, and I enjoy them as humans. Thanks to Melissa Coates for giving me my first technical editing opportunity, as well as for all the friendship and data platform knowledge she has shared over the years. Thanks to Melody Zacharias and Rie Irish, who gave me the opportunity to become a published author and set me on the path to more editing opportunities. Thank you to Bill Fellows, for helping me get involved in the SQL Server community, which changed my life in so many ways, and for being my friend. And of course, I'd like to thank the authors of this book: William Assaf, Randolph West, Sven Aelterman, Melody Zacharias, Louis Davidson, and Joey D'Antoni. I'm so glad to know them and thankful for everything they do in the SQL Server community.

Errata, updates, & book support

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

MicrosoftPressStore.com/SQLServer2019InsideOut/errata

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

For additional book support and information, please visit:

MicrosoftPressStore.com/Support

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

PART I

Introduction

CHAPTER 4

Installing and configuring SQL Server instances and features

In this chapter, we review the process of installing and configuring a Microsoft SQL Server instance as well as the creation or migration of databases. We pay special attention to new features introduced in SQL Server 2019, and other recent features you may not have noticed in SQL Server 2017. We review a post-installation checklist for you to verify every time and, when necessary, direct you to where you can find other sources of information and details for critical steps elsewhere in this book.

The SQL Server Setup tool content of this chapter mainly applies to SQL Server installations on Windows operating systems. The installation or creation is vastly simplified for an Azure SQL Database, Azure SQL Database managed instance, SQL Server on Linux, SQL Server in Docker containers, or Azure VMs running SQL Server. Many recommended settings in this chapter still apply for server-based platforms of SQL Server, such as in Docker containers or SQL Server on Linux. They are, after all, still very much the same SQL Server product that has always existed on Windows.

In Chapter 6, "Provisioning and configuring SQL Server databases," we'll cover the initial creation and configuration of databases inside the SQL Server instance. This chapter, however, focuses on server-level setup and settings.

What to do before installing SQL Server

Before running the SQL Server installer on your Windows Server, there are a number of 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, is not a choice you can easily reverse after installation. (More about the server level collation option later in this chapter, under the "Instance Collation" section.)

However, many mistakes made in installation can be resolved afterwards, likely with some tedium and outages. For example, skipping the initial default data and log directories may land all of your databases on the Operating System 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 Setup to fail. For more information on the risks and limitations, visit: *https://docs.microsoft.com/sql/sql-server/install/hardware-and-software-requirements-for-installing-sql-server-ver15#DC_support*.

We recommend you acquire the contents of this list prior to beginning SQL Server Setup:

- Active Directory service accounts, for the SQL Server service, SQL Agent Service, and other features if needed
- The latest downloaded cumulative update 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 be generating, backups of certificates, keys
- Whether to install the default or a named instance
- A plan for where SQL Server files will go, and each volume formatted to 64KB disk unit allocation size

Let's talk more about that last item now.

Deciding 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 prior to installation.

The examples in this chapter assume that your Windows operating system installation is on the C volume of your server. You should have many other volumes for SQL Server files–we'll review a sample layout soon. One of the basic guiding principles for a SQL Server installation is that anywhere you see "C:\," change it to another volume. This helps minimize SQL Server's footprint on the operating system (OS) volume, especially if you install multiple SQL Server instances, and can have potential disaster recovery implications in terms of volume-level backup 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 will delete temporary files involved in their installation, but not log files or configuration files, which should have a minimal footprint. Outside of log files, we recommend that you do not delete any files installed by SQL Server Setup or cumulative updates. Instead, let's take a look at some proactive steps to move these files off of the C volume.

Some parts of SQL Server Setup will 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\150\Setup Bootstrap* subfolder structure, where 150 is specific to the internal version number (15.0) of SQL 2019. 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 will be, 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 "Automating SQL Server Setup by using configuration files" section 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), default database locations for user database files, and their backups. If this is not the first SQL Server 2019 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 it.

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 2019 can consume more than 14 GB.

Inside OUT

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

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

The temporary storage volume in Azure VMs is a temporary disk that is locally present on the machine hosting your 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 if certain other considerations are taken. For more about this, see Chapter 3: Designing and implementing an on-premises database infrastructure. Otherwise, do not store any non-temporary files in the temporary storage.

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.** OS, some SQL Server files must install 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 file. (Alternatively, use the D: Temporary Storage volume on Azure Windows VMs.)
- **Volume H.** SQL Server backups (if written locally.)

And, 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 onto 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 impact to 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 four important volume configuration settings required to 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 disk support files and volumes larger than 2 TB, whereas 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: 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 prior to installation.

If you notice this on an existing SQL Server instance, your likely resolution steps are 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 of course lose the data.

- 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 a formatting decision, 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.

- There is a hardware-level concept related to file unit allocation size called "disk starting offset" that deals with how Windows, storage, disk controllers, and cache segments align their boundaries. Aligning disk starting offset was far more important prior to 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 DBA's. Still, it should be verified upon first use of a new storage system or 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: Prompt:

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

  A 1024 KB starting offset is a Windows default, which is displayed as 1048576 (bytes) for Disk #0 Partition #0.

Similar to 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 for all of 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; rather, editions are a key piece of knowledge for SQL administrators to understand what features may or may not be available.

- **Enterprise edition.** 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, instead use the free Developer edition. You'll have a far easier time in a licensing audit if your pre-production environment installations are Developer edition.

- **Developer edition.** Appropriate for all preproduction environments, especially those under a production Enterprise edition. Not allowed for production environments. This edition supports the same features and capacity as Enterprise edition and is free.

- **Standard edition.** Appropriate for production environments. Lacks the scale and compliance features of Enterprise edition that are required in some regulatory environments. Limited to the lesser of 4 sockets or 24 cores and also 128 GB of buffer pool memory, whereas Enterprise edition is limited only by the OS for compute and memory.

- **Web edition.** Appropriate for production environments but limited to low-cost server environments for web applications.

- **Express edition.** 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 proof-of-concepts, lightweight applications, or 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 cap).

- **Express with Advanced Services.** Similar to Express edition in all caveats and limitations, this edition includes some additional features including R integration, full-text search, and distributed replay.

- **Evaluation Edition.** Functionally the same as Enterprise edition, and free with a 180-day shutdown timer, but it isn't supported. Can be upgraded to any edition but Express. Do not use it 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 2019 installer, you are prompted to install a number of features outside of 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, and that is if you have licensed all physical cores on a virtual 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.

Changing SQL Server editions and versions

Upgrading editions in-place is supported by a feature of the SQL Server 2019 installer. You can upgrade in the following order: Express, Web, Standard, and Enterprise.

It is important to note that 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 2019, from Enterprise edition to Standard edition.

In-place upgrades for major versions (from 2017 to 2019, 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 2019 are not recommended, they are supported for versions as old as SQL Server 2012 on Windows Server 2016. SQL Server 2019 is the first SQL Server version that does not support Windows Server 2012 or 2012 R2.

A supported upgrade also assumes that the operating system 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 2019, visit *https://docs.microsoft.com/sql/database-engine/install-windows/supported-version-and-edition-upgrades-version-15*.

Beginning with SQL Server 2019, a SQL Server upgrade from SQL Server 2008 is no longer supported, though individual databases can still be migrated to SQL Server 2019. You can attach or restore its databases to SQL Server 2019, although they will be upgraded to compatibility level 100, the version level for SQL 2008.

Installing a new instance

In this section, you learn how to begin a new SQL Server 2019 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 2019. 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 highlight new features from SQL Server 2019.

It's important to note that 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 go over some of the command-line installation methods that you can use to automate the installation of a SQL Server instance.

Planning for multiple SQL Server instances

You can install as many as 50 SQL Server instances on a Windows Server; obviously, we 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, for example, *Servername\InstanceName*. 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*.

Creating an Azure VM running SQL Server

You should not need to install SQL Server on new Azure VMs, because provisioning new VMs with various versions and editions of Windows Server and SQL Server are available in the Azure Marketplace. For more information about bringing your existing SQL Server licensing to Azure VMs, visit *https://azure.microsoft.com/blog/easily-bring-your-sql-server-licenses-to-azure-vms/*.

When manually creating a new VM with SQL Server, however, there are still some configuration choices you can make in the browser. We will walk through some of those decision points in this section, although much of this is already handled for you.

Specifically in the *SQL Server settings* page of the *Create a virtual machine* wizard, you can choose to enable *Mixed Mode Authentication*. If you do, you must specify a username and password, but note that this probably isn't needed for you–administrators can and should use Azure Active Directory authentication when possible with your Azure VMs.

You can configure the Azure VM to be automatically powered down, which is very handy for rarely used VMs in your own sandbox. You can use Azure Automation to easily schedule VMs to run during pre-scheduled hours, thus lowering the monthly cost.

You can also enable Azure-specific integrations with Azure Key Vault, automatic patching, and automatic SQL backups for your new VM from the Azure Portal. Use Tags to organize your Azure infrastructure with information on environments, applications, locations, or other custom departmental metadata as desired, making it easy to monitor and manage objects in groups.

In the *Review + create* page, you can review the expected hourly spend and all of your Azure VM settings. Don't miss this detail, which is similar to SQL Server setup: there is a link to download a template for automation. An Azure VM template is a .json file that stores the options needed to create Azure resources, so that you can reference it within the Azure Portal, PowerShell, the Azure CLI (-Line Interface) and others to create Azure VMs with the same choices. Use the .json file to create a new Azure VM, for example, by choosing "Create a Resource" in Azure, then "Template deployment (deploy using custom templates)".

- ➢ **You'll find much more on Azure infrastructure, including the differences between Azure VMs, Azure SQL Database, and Azure SQL Database Managed Instances in Chapter 16 "Designing and implementing hybrid and Azure database infrastructure."**

Inside OUT

How does licensing work in Azure VMs?

This isn't an authoritative book on Azure or SQL Server licensing, but speaking broadly, 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:

- **Bring-your-own-license (BYOL) VM images available for you to provision using the same process and then later associate your existing Enterprise license agreements. The image names you're looking for here are prefixed with BYOL.**
- **Manually upload an .iso to the VM and install SQL Server 2019 as you would on any other Windows Server.**
- **Upload an image of an on-premises VM to provision the new Azure VM.**

It is important for you to keep in mind that 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 prior to creating your Azure VM.

Installing SQL Server on Windows

The rest of this chapter is dedicated to installations of SQL Server not part of a pre-made Azure Marketplace VM and apply to 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.

Launching SQL Server Setup

You should not run Setup with the installation media mounted over a remote network connection, via 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 file folder local to the server, instead of mounting a file or launching Setup from a location over the network.

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 then, on the shortcut menu that appears, click **Run As Administrator**.

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

In the tab pane on the left, click **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 when upgrading from prior versions of SQL Server.

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

- A link to launch the relatively painless Edition Upgrade Wizard. This is only for promoting your existing installation's edition as we discussed earlier.
- The Repair feature is not a commonly used feature. Its use is necessitated by a SQL Server with a corrupted installation. You might also need to repair an instance of SQL Server when the executables, .dll files, or Registry entries have become corrupted or damaged by disk corruption, antivirus, 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.
- Note that whereas 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.
- On the *Advanced* page, there is 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 *Automating SQL Server Setup by* using 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, "Implementing high availability and disaster recovery."**

CHAPTER 4

Windows Update in the SQL Server Setup

Since SQL Server 2012, the SQL Server installer has had the ability to patch itself while 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, and so a SQL Server 2019 Setup with Internet connectivity is the easiest way to carry out the installation. This also could be described as a way to “slip-stream” 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 should 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 “Installing by using a configuration file” section.

Regardless, after installation is complete and before the SQL Server enters further use in your team, verify that the latest SQL Server patches have been applied. For SQL Server 2019, see the official build versions site:
https://support.microsoft.com/help/4518398/sql-server-2019-build-versions.

Installing SQL Server *stand-alone installation*

Although what follows in this chapter is not a step-by-step walk-through, we’ll 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 is 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 needing 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.

Granting Perform Volume Maintenance Tasks

On the same Server Configuration page on which service accounts are set, you will see a check box labeled *Grant Perform Volume Maintenance Task privilege to the SQL Server Database Engine Service*. This option was added to SQL Server Setup in SQL Server 2016.

This automates what used to be a standard post-installation checklist step for SQL DBAs since 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 SQL Server service account via the *Perform Volume Maintenance Tasks* permission in Local Security Policy on the Windows server. But it's actually quite easy to verify whether or not IFI is in place for the SQL Server service via the `sys.dm_server_services` **dynamic management view. The column instant_file_initialization_enabled was first added in a series of patches to supported SQL Servers: in SQL Server 2012 SP4, SQL Server 2014 SP2, SQL Server 2016 SP1.**

```
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.**

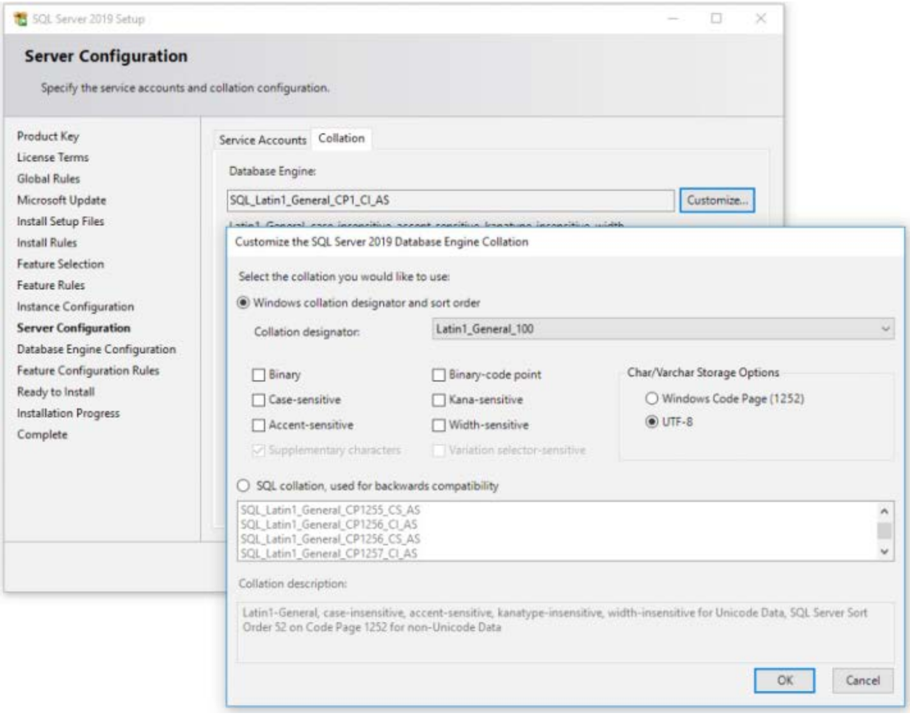
Instance Collation

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 Chapter 7: "Understanding table features" in the section on Collation.

Initially the instance collation provided in SQL Setup is the default collation for the server's regionalization, but you may 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. To change the collation of the SQL Server instance, reference this lengthy and difficult Microsoft guide: *https://docs.microsoft.com/sql/relational-databases/collations/set-or-change-the-server-collation*. Note that in the case of Azure SQL managed instances, you cannot change the server-level collation after it is created. For more information, visit: *https://docs.microsoft.com/sql/relational-databases/collations/set-or-change-the-server-collation#setting-the-server-collation-in-managed-instance*.

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. In SQL Server 2019 and for new application development, you may choose to take advantage of the new support for UTF-8 collations as the server default. Pictured below is the new interface in SQL Server Setup for choosing a UTF-8 collation.



Server Configuration

Specify the service accounts and collation configuration.

Database Engine: SQL_Latin1_General_CP1_CI_AS

Customize the SQL Server 2019 Database Engine Collation

Select the collation you would like to use:

● Windows collation designator and sort order

Collation designator: Latin1_General_100

□ Binary □ Binary-code point
□ Case-sensitive □ Kana-sensitive
□ Accent-sensitive □ Width-sensitive
☑ Supplementary characters □ Variation selector-sensitive

Char/Varchar Storage Options
○ Windows Code Page (1252)
● UTF-8

○ SQL collation, used for backwards compatibility

SQL_Latin1_General_CP1255_CS_AS
SQL_Latin1_General_CP1256_CI_AS
SQL_Latin1_General_CP1256_CS_AS
SQL_Latin1_General_CP1257_CI_AS

Collation description:

Latin1-General, case-insensitive, accent-sensitive, kanatype-insensitive, width-insensitive for Unicode Data, SQL Server Sort Order 52 on Code Page 1252 for non-Unicode Data

Figure 4-1 The options to set a server-level Windows collation using the new UTF-8 collation options now available in SQL Server 2019 Setup.

Mixed Mode authentication

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

➤ **You can read more on this topic in Chapter 12, "Administering 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, which are ideally Active Directory domain accounts. 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. It is important to note that SQL Authentication is not on by default, and isn't the recommended method of connection. By default, the recommended Windows Authentication is turned on and 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 7.**

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 this later on by connecting to the SQL Server instance via Object Explorer in SQL Server Management Studio. To do so, right-click the server name and then, on the shortcut menu that opens, click **Properties**, and then click the **Security** page, change to Mixed Mode. You must perform a SQL Server service restart to make this change effective.

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 hyperthreading for local machines. For example, with 16 logical processors, SQL Server Setup will default the installation to have 8 TempDB data files.

Adding too many TempDB data files can degrade SQL Server performance, perhaps severely if you add too many. 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.

Note also the new 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.

➤ TempDB is discussed in greater detail in Chapter 3.

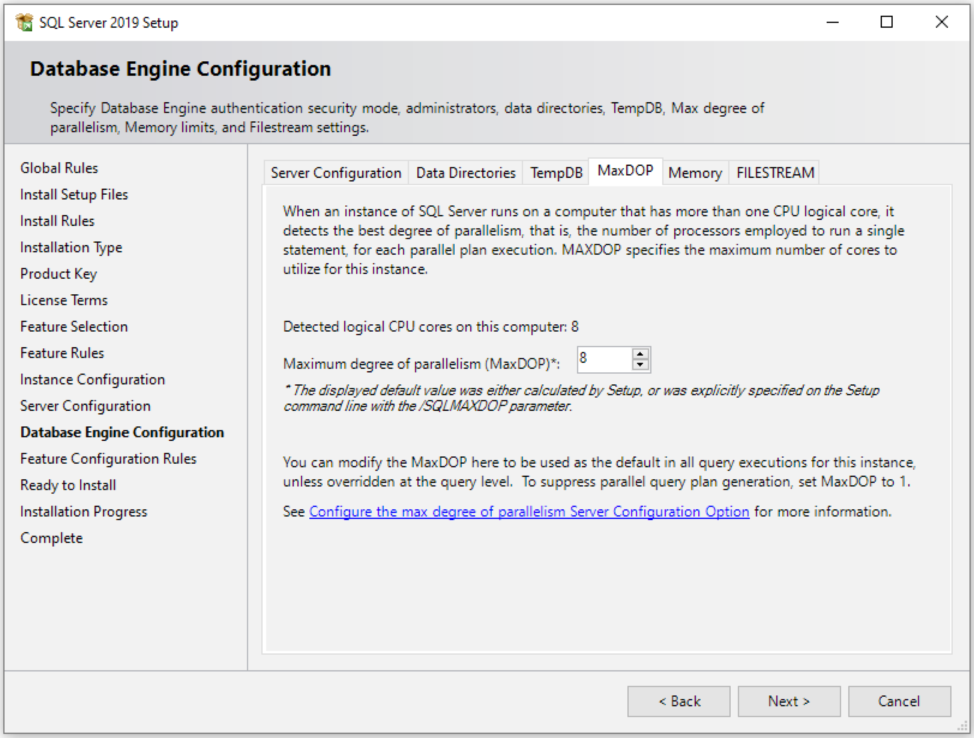
Default settings for MaxDOP

New in SQL Server 2019 are defaults for the configuration of the server-wide Maximum Degrees of Parallelism setting, included on the *Database Engine Configuration* page under the new *MaxDOP* tab.

Similar to how 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 the cores, effectively the same as *MaxDOP* setting = 0, which allows for unlimited parallelism.

For example, with 8 logical processors, SQL Server Setup will default the installation to use MaxDOP 8, as pictured in Figure 4-2. 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 MaxDOP 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. While 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.



Database Engine Configuration

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

When an instance of SQL Server runs on a computer that has more than one CPU logical core, it detects the best degree of parallelism, that is, the number of processors employed to run a single statement, for each parallel plan execution. MAXDOP specifies the maximum number of cores to utilize for this instance.

Detected logical CPU cores on this computer: 8

Maximum degree of parallelism (MaxDOP)*: 8

** The displayed default value was either calculated by Setup, or was explicitly specified on the Setup command line with the /SQLMAXDOP parameter.*

You can modify the MaxDOP here to be used as the default in all query executions for this instance, unless overridden at the query level. To suppress parallel query plan generation, set MaxDOP to 1.

See Configure the max degree of parallelism Server Configuration Option for more information.

Figure 4-2 The new MaxDOP tab in the Database Engine Configuration tab in SQL Server 2019 Setup, which has recommended a MaxDOP of 8 for a server with 8 virtual cores. This is effectively the same as MaxDOP 0, but offers the administrator an option to potentially change MaxDOP at the time of installation.

NOTE

Some applications, including Microsoft SharePoint and other third-party vendor applications, recommend that parallelism be disabled on their databases. This can be accomplished at the server level now, or, can be configured and overridden at each database level after SQL Server Setup is complete. Consult your vendor's specifications and recommendations documentation.

➤ **For much more information on performance tuning, parallelism, and the MaxDOP setting, see Chapter 14, "Performance tuning SQL Server."**

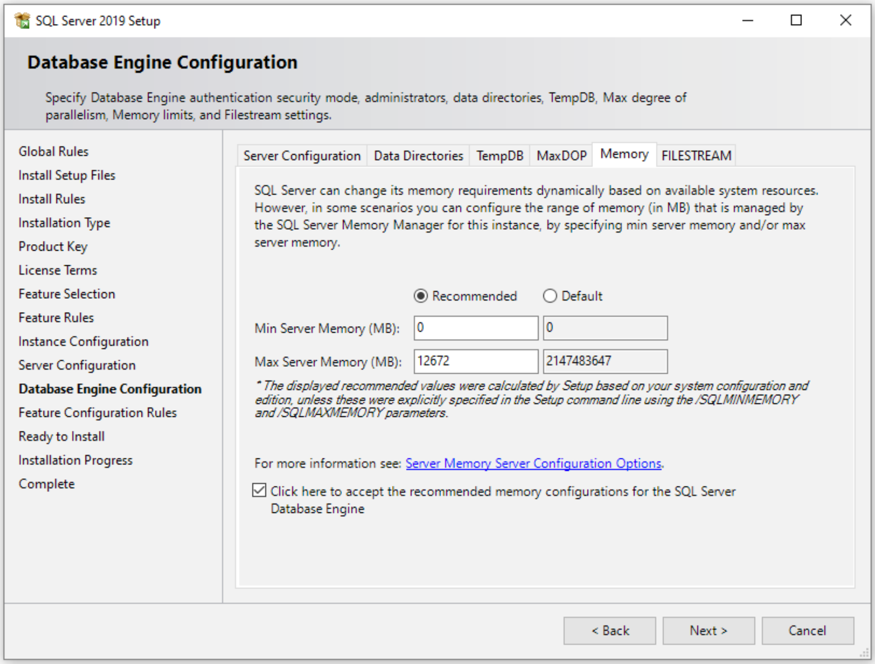
Default settings for Maximum Server Memory

New in SQL Server 2019 are defaults for the configuration of the instance-level Maximum 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 go and 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.

Now in SQL Server 2019, this max server memory option can be configured intelligently at the time of installation. It's important to note (and there's a checkbox to accept this guess) that SQL Server Setup is assuming this SQL Server instance will run alone on this server. If you expect to host other applications on this server, or memory-heavy features of SQL Server on the same server, such as SSAS or SSRS, you should reduce the maximum server memory for the SQL Server instance further.

➤ **We discussed the Maximum Server Memory setting in Chapter 3, in the "Configuration Settings" section.**

An example of the new max server memory recommendation configuring 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, as seen in Figure 4-3.



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):	12672	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 4-3 The new Memory tab in the Database Engine Configuration tab in SQL Server 2019 Setup.

Just above the *Maximum Server Memory* setting is the *Minimum Server Memory* setting, which establishes a floor for memory allocation. It is generally unnecessary to change this setting from the default of 0. 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 minimum server memory 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.

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

You should assure that SQL Server leaves enough memory for the operating system 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. Similarly, 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. Similarly, increasing this setting will not immediately show the effect of a change in memory utilization.

Installing common features

Aside from the SQL Server service itself, other features of the product might be common to your installations. 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. If you need them, this section covers installing these features using Setup. Later in this chapter, we cover the post-installation steps necessary to use them.

Installing SQL Server Analysis Services

Installing SQL Server Analysis Services (SSAS) requires you to make a decision at installation time regarding the mode in which it can be installed. Each instance of SQL Server Analysis Services can be in only one mode, which means that with a single license, you can run only Multidimensional mode, the newer Tabular mode (introduced in SQL 2012), or the Power Pivot mode.

Ask your business intelligence decision makers which platform you should use, though for most new development, Tabular mode is popular and recommended. Tabular mode databases can also run in Azure Analysis Services. The following are brief descriptions of each mode:

- **Multidimensional mode.** This is the SSAS setup that was first introduced in SQL 2000. This is also the only mode to support data mining and other features that existing SSAS data models predating SQL Server 2012 may be dependent on. The primary language for building and querying multidimensional models is MDX.

  - **Tabular mode.** This is the newer and recommended SQL Server Analysis Services setup that was first introduced in SQL 2012, using the in-memory VertiPaq processing engine. Since SQL Server 2017, this is the default installation mode selected on the Analysis Services Configuration page of Setup. The primary language for building and querying tabular models is DAX, which is similar to the Excel function language.

  - **Power Pivot mode.** This mode installs SQL Server Analysis Services in the Power Pivot for SharePoint mode. Power Pivot workbooks use both DAX and MAX. 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://docs.microsoft.com/analysis-services/comparing-tabular-and-multidimensional-solutions-ssas*.

Inside OUT

What if you choose the wrong SQL Server Analysis Services mode?

If you choose one SQL Server Analysis Services mode at installation but your business intelligence developers want another mode, the supported option is to uninstall and

reinstall the SQL Server Analysis Services feature. Changing the SQL Server Analysis Services 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 SQL Server Analysis Services 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\*.

Installing SQL Server Integration Services

The SQL Server Integration Services instance for SQL Server 2019 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 is available. We discuss this new feature further in the next section.

A 64-bit version of SQL Server Integration Services is installed on 64-bit operating systems. If you worry about connecting to 32-bit servers, data sources, or applications installations (such as Microsoft Office), don't—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 SQL Server Integration Services on a 32-bit OS; however, we do not recommend this.

Installations of different versions of SQL Server Integration Services are installed side-by-side on a server; specifically, the service SQL Server Integration Services 15.0 is compatible with prior versions.

Outside of configuring the service account, you do not need any additional configuration when installing SQL Server Integration Services during SQL Server Setup. The default virtual service account is *NT Service\MsDtsServer150*.

Inside OUT

Should you install SQL Server Integration Services (SSIS) alone on a server?

A standalone installation of SQL Server Integration Services without a matching SQL Server Database Engine is possible but not recommended. For the modern Project Deployment model of SQL Server Integration Services, 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 isolation of the SQL Server Integration Services 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 SQL Server Integration Services package runtime workloads from other database workloads. Both of these options would carry the same licensing cost.

Installing SQL Server Integration Services Scale Out configuration

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

The master node talks to worker nodes in a SQL Server Integration Services Scale Out, 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 at the time the master node is set up.

When adding another SQL Server Integration Services installation as a Scale Out Worker, start the new SQL Server Integration Services Manage Scale Out window via SQL Server Management Studio. Right-click the Catalog you have created, and then click **Manage Scale Out**. At the bottom of the page, click the **+** button to add a new Scale Out Worker node. 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 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’s Scale Out service. After the worker node is added, refresh the Worker Manager page, and then click the new Worker node entry, which will be red. You must turn on the Worker Node by clicking **Enable Worker**.

You also can copy and install the certificates manually between servers. You will find them in: *%program files%\Microsoft SQL Server\150\DTS\Binn\*. For more information on certificates between servers, visit *https://docs.microsoft.com/sql/integration-services/scale-out/deal-with-certificates-in-ssis-scale-out*. For a Microsoft-provided walkthrough of setting this up, visit *https://docs.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 SQL Server Integration Services Service Account doesn’t run packages or need permission to do very much, the Scale Out Master and Worker service accounts actually do run packages. The SQL Server Integration Services 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 2019) but do appear in Services.msc. By default, these services run under virtual accounts *NT Service\SSISScaleOutMaster150* and *NT Service\SSISScaleOutWorker150*, 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.

Installing SQL Server Reporting Services

Starting with SQL Server 2017, SQL Server Reporting Services is no longer found in the SQL Server Setup media; it is instead available as a simplified, unified installer and a small download. SQL Server Reporting Services 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 SQL Server Reporting Services isn't free, and that the separate installer isn't a licensing change, though SQL Server Express with Advanced Services offers some limited SSRS support. For more information on the limitations of SSRS with SQL Server Express license, view *https://docs.microsoft.com/sql/reporting-services/reporting-services-features-supported-by-the-editions-of-sql-server-2016*.

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

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

You'll notice the Report Server Configuration Manager 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\150\Tools\Binn\RSConfigTool.exe*). The Report Server Configuration Manager application itself is largely unchanged since SQL 2008.

The virtual service account *NT SERVICE\SQLServerReportingServices* is the default SQL Server Reporting Services service account. It is a second-best option, however: we recommend that you create a new domain service account to be used only for this service; for example, *Domain*\svc_*ServerName*_SSRS or 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 SQL Server Reporting Services service account later, use only the Reporting Services Configuration Manager tool to make this change. Like other SQL Server services, 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 SQL Server Reporting Services Execution Account, email settings, or backing up the encryption key using Reporting Services Configuration Manager.

SQL Server 2019 Reporting Services also can integrate with Microsoft Power BI dashboards. A page in the Report Server Configuration Manager supports registering this installation of SQL Server Reporting Services with a Power BI account. You will be prompted to sign into Azure Active Directory. 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 SQL Server Reporting Services, via Report Manager, and a member of the sysadmin role in the SQL Server that hosts the Report Server database.

Inside OUT

Where is SQL Server Reporting Services SharePoint Integrated mode?

Starting with SQL Server 2017, there is no more SharePoint Integrated mode. 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 SQL Server Reporting Services native mode with on-premises SharePoint sites via embedded SQL Server Reporting Services reports, including SQL Server Reporting Services reports stored in the Power BI Report Server.

Similarly, there is no future support for SQL Server Reporting Services integration with SharePoint Online.

Installing machine learning features

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

Data scientists can take advantage of this feature to build advanced analytics, data forecasting, and algorithms for machine learning.

In SQL Server 2019, the Instance Feature Machine Learning Services (In-Database) is renamed to Machine Learning Services and Language Extensions, while the name for the standalone Shared Feature Machine Learning Server (Standalone) is the same.

Machine Learning Services and Language Extensions support Java code execution inside the Database Engine. By default it installs the Zulu Open JRE if you select the Java checkbox under "Machine Learning Services and Language Extensions" in the feature selection page. If you must use a different Java JRE because of an application dependency, such as the official Oracle JRE, you should add it via a home variable and the icacls Windows command, as detailed here: *https://docs.microsoft.com/sql/language-extensions/install/install-sql-server-language-extensions-on-windows*.

➤ **For more on Installing and Leveraging Machine Learning Services, see Chapter 20, "Leveraging Big Data and Machine Learning."**

Installing 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, MongoDB, Cloudera, or Apache Hadoop.

Using PolyBase EXTERNAL tables, we can use SQL data types and T-SQL queries to seamless 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 against non-Microsoft database platforms like Oracle and DB2, but also for Hadoop nonrelational data or 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.

New to SQL Server 2019 is the ability to install the *Java connector for HDFS data sources* with SQL Server Setup. This 7GB+ option doesn't replace the requirement to install the Oracle JRE in order to install the PolyBase Query Service feature.

➤ **For more on the PolyBase Query Service feature, see Chapter 20.**

Logging SQL Server Setup

SQL Server Setup generates a large number of logging files for diagnostic and troubleshooting purposes. These logs should be the first place you go when you have an issue with Setup.

First, a System Configuration Check report .htm file is generated each time you run Setup, so 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.

NOTE

At the time of this writing, there is an outstanding installation issue with SQL Server 2019, installation media mounted on removeable media, and SQL Server Management Studio versions 18.0 through 18.3. The problem and solution are detailed here in a link provided by Microsoft: *https://techcommunity.microsoft.com/t5/SQL-Server-Support/SQL-Server-2019-Installation-Error-An-error-occurred-for-a/ba-p/998033*. In short: copy the files locally, or, uninstall the SQL Server 2012 Native Client, then re-try installation of SQL Server 2019. You should always try to run the latest version of SSMS.

Automating SQL Server Setup by using 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.

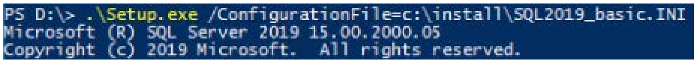
Values provided in configuration files can prepopulate or override Setup settings. They also can configure Setup to run with the normal user interface, or silently without any interface.

Starting 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 that you always use the Administrator level for these two shells. The title on each application window should be preceded by *Administrator:*, such as: *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 command prompt as Administrator, in the Start menu, search for the desired application (see Figure 4-4), right-click it, and then, on the shortcut menu that opens, select **Run As Administrator**.



```
PS D:\> .\Setup.exe /ConfigurationFile=c:\install\SQL2019_basic.INI
Microsoft (R) SQL Server 2019 15.00.2000.05
Copyright (c) 2019 Microsoft.  All rights reserved.
```

Figure 4-4 Starting Setup.exe from PowerShell.

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

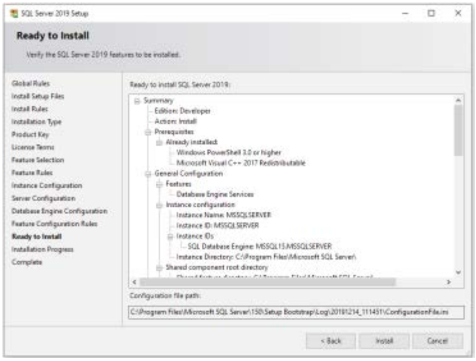
```
.\Setup.exe /ConfigurationFile=c:\install\SQL2019_basic.INI
```

The previous sample script and all scripts for this book are available for download at *https://www.MicrosoftPressStore.com/SQLServer2019InsideOut/downloads*. The above code sample uses a configuration file to pre-select installation choices, for example, features to be installed. Let's talk more about configuration files.

Generating 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 4-5). At that location, even before you hit the **Install** button, you'll find a generated configuration file, ready for future use and modification if needed.



SQL Server 2019 Setup

Ready to Install

Verify the SQL Server 2019 features to be installed.

Ready to install SQL Server 2019:

```
Summary
  Edition: Developer
  Action: Install
  Prerequisites
    Already installed:
      Windows PowerShell 3.0 or higher
      Microsoft Visual C++ 2017 Redistributable
  General Configuration
    Features
      Database Engine Services
    Instance configuration
      Instance Name: MSSQLSERVER
      Instance ID: MSSQLSERVER
      Instance IDs
        SQL Database Engine: MSSQL15.MSSQLSERVER
      Instance Directory: C:\Program Files\Microsoft SQL Server\
    Shared component root directory
```

Configuration file path:

C:\Program Files\Microsoft SQL Server\150\Setup Bootstrap\Log\20181214_111451\ConfigurationFile.ini

Figure 4-5 The Ready To Install page displays a summary of the installation steps as the Configuration File Path that has been prepared.

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.

Installing by using a configuration file

Now that you have a configuration file generated by using a previous walk-through of Setup, you can take the next step to automating or standardizing your installation.

You can start Setup.exe from a 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. 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 mind, however, 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. You should 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 engine 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` has provided the account name of the 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"
```

The previous sample script and all scripts for this book are available for download at *https://www.MicrosoftPressStore.com/SQLServer2019InsideOut/downloads*.

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

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 provided `/UpdateEnabled=False`, you can also specify `/UpdateSource` the location of cumulative update or other SQL patch file executables to be included in the installation.

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 this parameter for any unattended installation involving the R language option for Machine Learning Services.

Python open license terms	`/IACCEPTPYTHONLICENSETERMS`	Must provide this parameter 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 SQL Server Database Engine Service (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, also 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 the TCP/IP Protocol	`/TCPENABLED="1"`	Disabled by default and used in many installations, enable the TCP/IP protocol here to save yourself a step in Configuration Manager later on.

Post-installation server configuration

After you install SQL Server, there are a number of changes to make or confirm on the Windows Server 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, and many deal with SQL server and/or Windows configuration settings. Evaluate whether these are appropriate for your environment, but you should consider and apply them to most SQL Server installations.

1. Check your SQL Server patch level version, and apply patches if necessary.
2. Review Maximum server memory settings for other features.
3. Review Surface Area Configuration.
4. Set up SQL Agent.

5. Turn on TCP/IP if needed.
6. Verify power options for the server.
7. Configure antivirus exclusions for SQL Server processes and files.
8. Enable the server setting “optimize for ad hoc workloads.”
9. Evaluate whether Lock pages in memory is necessary.
10. Review size and location of the Windows page file.
11. Set up scheduled backups, index maintenance, log retention maintenance, and integrity checks.
12. Backup service master and database master keys.
13. Increase SQL Agent and SQL Error log retention from the defaults.
14. Suppress successful backup messages.

Let’s take a look at each of these in more detail in the subsections that follow.

Check your SQL Server version

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 the following built-in functions:

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

While you’re at it, doublecheck 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, search for KB321185 or visit *https://support.microsoft.com/help/321185/*.**

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 SQL Server Database Engine and SQL Server Analysis Services and/or SQL Server Reporting Services 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.

Limiting SQL Server Analysis Services 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 can change these memory settings via SSMS by connecting to the SSAS instance in Object Explorer. Right-click the server, and then, on the shortcut menu, click **Properties**. Some of the memory settings described here are identical for Multidimensional and Tabular installations, some 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 the Virtual Address Space technically, but SSAS, among other features, is no longer supported on 32-bit systems, and so this is not a concern).

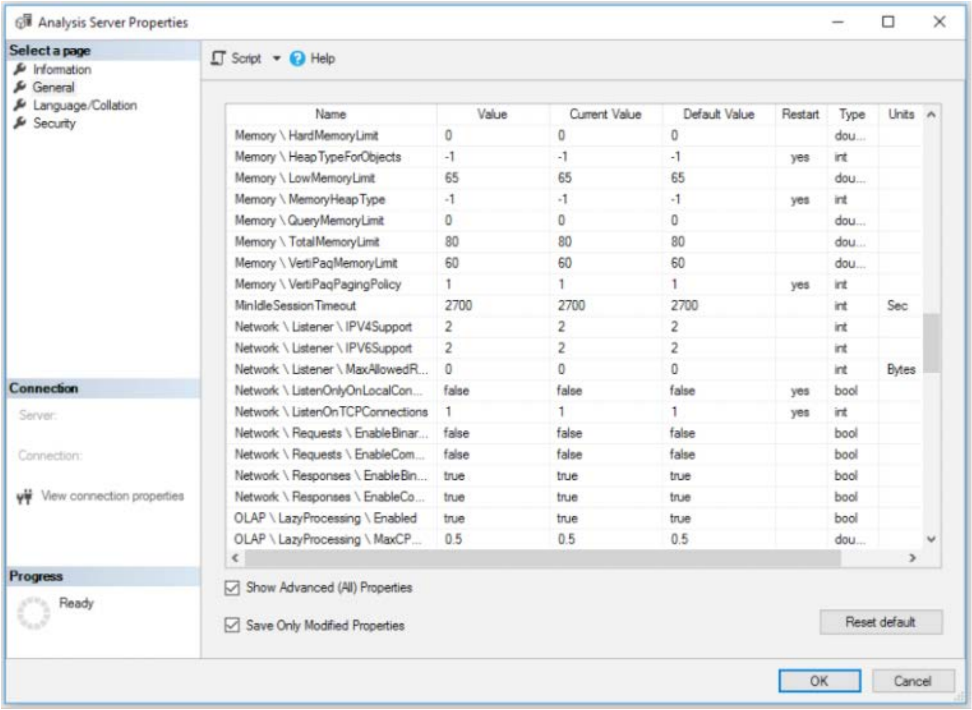
- **TotalMemoryLimit.** A value that serves as a threshold for SSAS to begin to release memory for higher priority requests. It's important to note that this is not a hard limit. The default is 80 percent of total server memory.

- **HardMemoryLimit.** This is a hard memory limit that will lead 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, this limit has a default of 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.** New to SQL Server 2019, this 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 4-6 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 option to “Show Advanced (All) Properties” 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 4-6 The General page in the Analysis Server Properties dialog box showing the default settings.

Limiting SQL Server Reporting Services memory

Four options are available for limiting SQL Server Reporting Services memory utilization, and all 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 that is 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 take a look at each one:

- **MemorySafetyMargin.** The percentage of `WorkingSetMaximum` that SQL Server Reporting Services will use before taking steps to reduce background task memory utilization 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 SQL Server Reporting Services will deny new requests, slow down existing requests, and page memory to a hard drive until memory conditions improve.

Two more settings are instead 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 those two values have been ignored since SQL Server 2008.

- **WorkingSetMaximum.** By default, this is the total server's 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 SQL Server Reporting Services will be aware. This value is expressed in kilobytes of memory.
- **WorkingSetMinimum.** By default, this value is 60 percent of the `WorkingSetMaximum`. If SQL Server Reporting Services 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 Server Reporting Service's memory utilization.

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>
```

Limiting Machine Learning Server memory

Similar to 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=SQL2K19
```

```
TRACE_LEVEL=1
JOB_CLEANUP_ON_EXIT=1
USER_POOL_SIZE=0
WORKING_DIRECTORY=C:\Program Files\Microsoft SQL
Server\MSSQL15.SQL2K19\MSSQL\ExtensibilityData
PKG_MGMT_MODE=0
MEMORY_LIMIT_PERCENT=25
```

The previous sample script and all scripts for this book are available for download at *https://www.MicrosoftPressStore.com/SQLServer2019InsideOut/downloads*.

Surface Area Configuration

If you are a veteran SQL Server DBA, you will remember when SQL Server Surface Area Configuration was a separate application. Surface Area Configuration is now considered a facet, accessed via the Facets dialog in SSMS starting with SQL Server 2008.

To view Surface Area Configuration in SSMS, in Object Explorer, connect to the *SQL Server*, right-click the server, and then, on the shortcut menu, click **Facets**. (Note that this window sometimes takes a moment to load.) In the dialog box that opens, change the value in the list box to **Surface Area Configuration**.

Keep in mind that 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 2019, however, you should consider enabling three of these options:

- Database Mail (more about this in Chapter 9: "Automating SQL Server administration"). This should be enabled on most instances, to allow SQL Server at the very least to send out a message in case of a high severity or job failure, and also to allow developers to send custom email message 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`.

- Remote Dedicated Admin Connection (more on this in Chapter 13: "Protecting data through encryption, privacy and auditing"). This could be particularly useful for bypassing a malfunctioning login triggers or Resource Governor. You also can turn this setting on or off via the remote admin connections option in `sp_configure`.

- CLR Integration, which you will need to turn on if you need to use SQL Server Integration Services, or if you need 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 Surface Area Configuration only if they are specifically required by an application and you are aware of the potential security concerns.

Setting 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.

➤ **Chapters 8 and 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. To send email notifications for alerts or job status notifications, you must set up a Database Mail account and profile (see Chapter 9).
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 a large number of Windows Management Instrumentation (WMI) conditions, Perfmon 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 Managing and Monitoring SQL Server, see Chapter 13.**

Enabling the TCP/IP network protocol

The common network protocol TCP/IP is off by default, and 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 the TCP/IP protocol 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, the dot character ("."), or (local), or .\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 Always On availability groups listener, and Kerberos authentication.

To configure the TCP/IP protocol, open the SQL Server Configuration Manager application locally on the server, in the left pane, click **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 the TCP/IP protocol 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='SQL2K19']/ServerProtocol[@Name='Tcp']"
#Enable the TCP protocol on the local server, on the named instance SQL2K19
$TCPIP = $wmi.GetSmoObject($path)
$TCPIP.IsEnabled = $true
$TCPIP.Alter()
$TCPIP.IsEnabled
#Restart SQL Server Database Engine service to apply the change
```

The previous sample script and all scripts for this book are available for download at *https://www.MicrosoftPressStore.com/SQLServer2019InsideOut/downloads*.

After turning on the TCP/IP protocol via any method, 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.

Verifying 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 this setting back to Balanced or another setting. Also ensure that group preferences are configured for High Performance for new SQL Servers.

Configuring antivirus exclusions

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.

Also, configure any antivirus programs to ignore folders containing SQL Server files. This 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
- SQL Server Reporting Services 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 Failover Cluter Instances (FCIs) and availability groups, also 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 regarding *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.

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, anti-ransomware, or anti-malware solutions in your enterprise.

Enable 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 2: Introducing database server components.**

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. However, like other design concepts in databases, we find that there are either one or many. There is no two.

➤ **To read more about cached execution plans, see Chapter 14.**

Lock Pages in Memory

You should consider using the *Lock Pages in Memory* setting 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.**

➤ **For more about the Windows page file, see Chapter 3.**

Inside OUT

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

Starting with SQL Server 2016 SP1, you can check whether the *Lock Pages in Memory* permission has been granted to the SQL Server Database Engine service:

```
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
```

The previous sample script and all scripts for this book are available for download at *https://www.MicrosoftPressStore.com/SQLServer2019InsideOut/downloads*.

Backups, index maintenance, and integrity checks

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

Begin taking database backups, at least of the master and msdb databases. You should also back up other SQL Server Setup-created databases, including ReportServer, ReportServer-TempDB, and SSISDB right away.

➤ **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 take into account the current fragmentation levels of indexes, rather than performing index maintenance on entire databases.

➤ **For more information on automating maintenance, see Chapter 13.**

Backing 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 6.**

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

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

And 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'
```

The previous sample script and all scripts for this book are available for download at *https://www.MicrosoftPressStore.com/SQLServer2019InsideOut/downloads*.

If you implement Transparent Data Encryption, 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!

Increasing default error and agent history retention

By default, SQL Server maintains the current SQL Server error log plus six more error logs of history. Logs are cycled each time the SQL Server service is started.

One eventful, 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.

In SQL Server Management Studio, in Object Explorer, connect to the SQL Server instance. Expand the Management folder, right-click **SQL Server Logs**, and then, on the shortcut menu, click **Configure**. Select the *Limit The Number Of Error Logs Before They Are Recycled* check box and 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 Servers that generate a large amount of log “noise”, consider other options to reduce the clutter of the SQL Error Log, including Trace Flag 3226 to suppress the logging of successful backup operations (much more on that in the next section).

You may also choose to configure a SQL Agent Job to manually cycle the SQL Error Log using `sp_cycle_errorlog`, so that no one log file contains too much data to be unwieldly for scan and analysis.

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.

To change the history settings for SQL Server Agent, in SSMS, in Object Explorer, connect to the SQL Server instance. Right-click **SQL Server Agent**, and then click **Properties**. Click the

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

The second option, *Remove Agent History*, along with its companion *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 then click **OK** and return to this page; you will find the second check box is cleared. Behind the scenes, SQL Server Management Studio ran the `msdb.dbo.sp_purge_jobhistory` stored procedure to remove job history manually.

Suppress successful backup messages

By default, SQL Server writes an event to the 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 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
>
> **It is important to note that you can review successful backup history by querying the msdb system database, which 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, "Developing, deploying, and managing 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.

As Trace Flag 3226 is intended to be a permanent setting, simply starting the trace by using `DBCC TRACEON` is not sufficient, given that 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 SQL Server 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, add "-T3226", then click Add. The change will not take effect until the SQL Server Database Engine service is restarted.

➤ **For more information on SQL Server Configuration Manager, see Chapter 1.**

Post-installation configuration of other features

SQL Server Database Engine installation is now complete, but three other features require post-installation configuration, including SQL Server Integration Services, SQL Server Reporting Services, and SQL Server Analysis Services. 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 SQL Server Integration Services, specifically with a new server-integrated deployment, built-in performance data collector, environment variables, and more developer quality-of-life improvements. For these reasons, use the new Project Deployment Model and the 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 should create the SSISDB catalog soon after installation and before a SQL Server Integration Services development can take place. You will need to create the catalog only once. Because this will involve potential Surface Area Configuration changes and the creation of a new strong encryption password, a SQL DBA, not a SQL Server Integration Services developer, should perform this and store the password securely alongside others generated at the time of installation.

In Object Explorer, connect to your instance, right-click **Integration Services Catalog**, and then, on the shortcut menu, click **Create Catalog**. In this single-page setup, you must select the *Enable CLR Integration* check box, decide whether SQL Server Integration Services 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 then back up the SSISDB database master key.

➤ **For more on database master keys, see Chapter 13.**

The SSISDB database will contain SQL Server Integration Services packages, their connection strings, and more data about the packages. The encryption would not allow these sensitive contents to be decrypted by a malicious user who gains access to the database files or backups. This password would be required if the database were moved to another 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 likely because SQL Server Integration Services was not actually installed. It's likely that the 6MB template database backup was not copied from the SQL Server media, perhaps because the SSIS feature was not installed on this instance. 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 a SQL Server Reporting Services native-mode installation from the downloaded installer file, SQLServerReportingServices.exe. At the end of the Microsoft SQL Server 2019 Reporting Services installer wizard, on the Setup Completed screen, click the **Configure report server** button to open the Reporting Services Configuration Manager application. Connect to the newly installed SQL Server Reporting Services instance, and then review the following options, from top to bottom:

- **Service Account.** You can change the SQL Server Reporting Services service account here. Remember that you should use only the Reporting Services Configuration Manager tool to make this change.

- **Web Service URL.** The web service URL is not for user interaction; rather, it is for the Report Manager and custom applications to programmatically connect to the SQL Server Reporting Services 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, click **Apply**.

  You can optionally configure an SSL certificate to 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 changes necessary. For more information on configuring SSL connections for the SSRS

Web Service and Web Portal, visit: *https://docs.microsoft.com/sql/reporting-services/security/configure-ssl-connections-on-a-native-mode-report-server*.

- **Database.** Each instance of SSRS requires requires a pair of databases running on a SQL Server instance. Simply executing the SSRS installer alone does not configure the databases for SSRS, you need to configure them via the 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 a on-premises SQL Server instance or Azure VM-hosted SQL Server instance or, new to SQL Server 2019, hosted on 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, click 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 for administrative features to the SQL Server Reporting Services instance. This is the link to share with users if you will be using the SQL Server Reporting Services portal.

  By default, the URL for the web portal is */Reports*

  servername/Reports for the default instance

  or:

  servername/Reports_*InstanceName* for named instances

  You can change the name from the default here if desired. To proceed, at the bottom of the application window, click **Apply**.

- **Email Settings.** You use these email settings for sending reports to user subscribers via email. SQL Server Reporting Services 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.

  SQL Server Reporting Services can authenticate to an SMTP server using anonymous (No Authentication), Basic, or NT LAN Manager (NTLM) authentication, which will use the SQL Server Reporting Services service account to authenticate to the SMTP server.

  NOTE Keep in mind that enterprise SMTP servers usually have an allow list of IP addresses, and you will need to add this server's IP to this list to relay email.

- **Execution Account.** You can provide this domain account optionally to be used when reports are configured to run on a schedule, to run without credentials (select the Credentials Are Not Required Option), or to connect to remote servers for external images.

  The execution account should not be the same as the SQL Server Reporting Services service account.

  This account should have minimal read-only access to any data sources that will require it. You also can give it EXECUTE permissions for data sources that use stored procedures, but you should never give it any 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 SQL Server Reporting Services databases have been created, you should back up this instance's encryption keys. This key is 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, or rotate the key by using the *Change* operation on this page to replace the key, 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.

- **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 use this file share credential, which can be used on this page in a central location.

  This account should be different from the SQL Server Reporting Services execution account to serve its purpose appropriately.

- **Scale-Out Deployment.** Visit this page on multiple SQL Server Reporting Services instances to join them together. By using the same SQL Server Reporting Services databases for multiple SQL Server Reporting Services 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 a Network Load Balancing Cluster, 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 SQL Server Reporting Services 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 SQL Server Reporting Services 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://docs.microsoft.com/sql/reporting-services/install-windows/configure-a-native-mode-report-server-scale-out-deployment*.

- **PowerBI Integration.** Use this page to associate the SQL Server Reporting Services instance to a Microsoft Power BI account, specifically to an account in Azure Active Directory. The administrator joining the Power BI instance to the SQL Server Reporting Services instance must be:
  - A member of the Azure AD
  - A member of the system administrator role of the SQL Server Reporting Services instance
  - A sysadmin on the SQL Server instance that hosts the SQL Server Reporting Services databases.

➤ **For the latest information on Power BI/SQL Server Reporting Services integration and the latest Azure authentication features, visit *https://docs.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 SQL Server Analysis Services instance.

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

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

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

Index