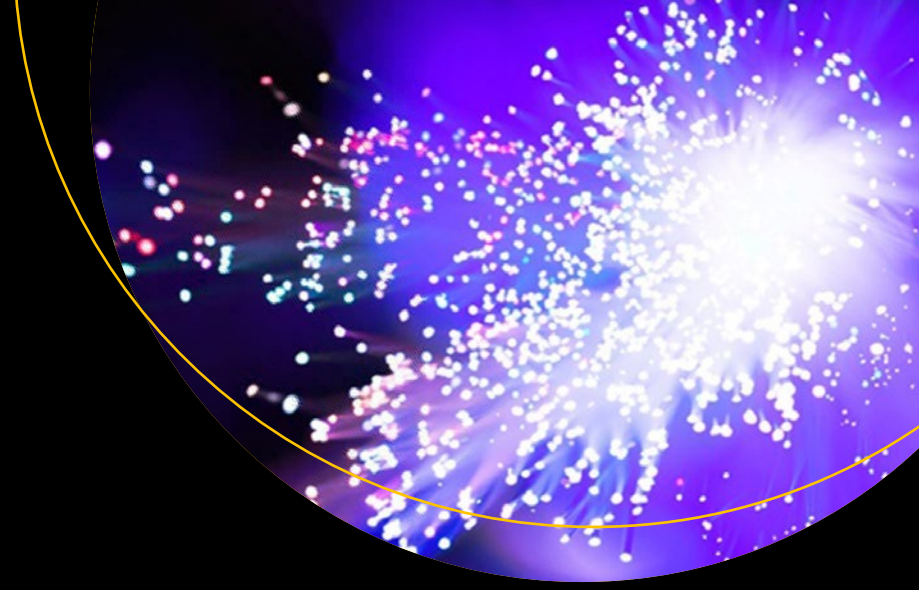

SQL Server 2019 Revealed

Including Big Data Clusters and Machine Learning

Bob Ward

Foreword by Rohan Kumar

Apress®

SQL Server 2019 Revealed

Including Big Data Clusters and Machine Learning

Bob Ward
Foreword by Rohan Kumar

Apress®

SQL Server 2019 Revealed: Including Big Data Clusters and Machine Learning

Bob Ward
North Richland Hills, Texas, USA

ISBN-13 (pbk): 978-1-4842-5418-9 ISBN-13 (electronic): 978-1-4842-5419-6
https://doi.org/10.1007/978-1-4842-5419-6

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Jonathan Gennick
Development Editor: Laura Berendson
Coordinating Editor: Jill Balzano

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484254189. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

This book is dedicated to the SQL Server Community, also known as the #sqlfamily. Without the community, this amazing product would not be what it is today.

Table of Contents

About the Author

Bob Ward is a Principal Architect for the Microsoft Azure Data SQL Server team, which owns the development for all SQL Server releases. Bob has worked for Microsoft for 26+ years on every version of SQL Server shipped from OS/2 1.1 to SQL Server 2019 including Azure. He is a well-known speaker on SQL Server, often presenting talks on new releases, internals, and performance at events such as PASS Summit, SQLBits, SQLIntersection, Red Hat Summit, Microsoft Inspire, and Microsoft Ignite. You can follow him at @bobwardms or www.linkedin.com/in/bobwardms. Bob is the author of the book *Pro SQL Server on Linux* available from Apress Media.

About the Technical Reviewer

Aaron Bertrand is a passionate technologist with over two decades of SQL Server experience. He has worked directly with several Microsoft product teams and is well known for helping improve the technical skills of the broader developer community through writing, speaking at, and moderating technical forums.

Foreword

We are truly at a unique tipping point in the history of technology, and there has never been a better time to be in the field of data, analytics, and AI. The pace of growth in data is more rapid than ever before, and digital disruption through AI and ML has created unlimited potential for companies to embrace data as a competitive advantage for their business. With the dramatic acceleration of digitization, the primary question we now face is how to take advantage of this massive volume of data to help our companies and communities transform.

We see a massive opportunity powered by the intelligent cloud and the intelligent edge. SQL Server is unparalleled in the industry in the level of consistency it provides the developers, data engineers, and administrators across the edge, on-premises, private cloud, and the public cloud. Our SQL Server community has played a very important role in this evolution, and I cannot thank them enough for their support and feedback over the last 25+ years.

SQL Server 2019 is a phenomenal release, and I am proud of what the team has delivered. SQL Server 2019 builds on the innovation that was delivered in SQL Server 2016 and SQL Server 2017. While there are several new capabilities that will serve our customers well, as is expected from every major release of SQL Server, I am most excited about the remarkable innovation that extends the skills our customers have built over multiple decades to manage and get insight from their Big Data systems. This innovation will play a critical role in driving the digital transformation for our customers.

Bob Ward has been with the SQL Server team since the very early days and has had a notable impact on the product. There are very few who have the breadth and depth of understanding that he does, and it shows in how he manages to explain complex concepts in a simple, easy-to-understand manner in this book. I hope you enjoy reading it.

Rohan Kumar
Corporate Vice President, Azure Data at Microsoft

Acknowledgments

I have so many things in my life to be thankful for, and having the ability to write this book is one of them. I believe all of my blessings come from God, and it is through His grace I have come to know the powerful message of Jesus Christ. I must first give thanks to my beautiful and talented wife Ginger. She is my partner and my soulmate. She tirelessly heard all my complaints, watched me spend very late nights on the book, and sometimes had to be the driver so I could work on the book in the passenger seat. I don't know anyone who has stronger faith than my wife Ginger, and I am so fortunate to share a wonderful life together with her. We celebrated 30 years of marriage this year, and I still love being with her as much as possible each and every day. I also want to thank my sons Troy and Ryan. Troy these days lives in Charleston, South Carolina, where I wrote the last part of the book as Hurricane Dorian approached. Troy is someone who I admire not just for his character but his quest to make this place a better world. Ryan is in his second year at Baylor Law School (Go Bears!). He continues to amaze me with this intelligence, integrity, and ability to keep it all together with confidence yet still find time to improve his golf game. I also want to thank my mother Annette Gibaud who continues to show me the example of finding a way to be kind to someone each and every day.

There are so many people who helped contribute to this book. I want first to thank Apress Media for giving me another chance to write a book. Jonathan Gennick and Jill Balzano were there again every step of the way helping me push this book to conclusion. And this book could not have been possible and on time without my Technical Reviewer Aaron Bertrand. When I thought about writing this book, Aaron was one of the first people that came to mind for a reviewer given his incredible knowledge of SQL Server and reputation as an expert in the community. Aaron was simply superhuman in how fast he cranked out reviews of each chapter.

From Microsoft, first and foremost thank you to Rohan Kumar, Gayle Sheppard, and Asad Khan for giving me the opportunity to spread the message about SQL Server 2019 which was instrumental in my detailed learning of the product to write this book. I also want to personally thank two of my closest colleagues, Buck Woody and Anna Hoffman (Thomas). I travelled the world with Buck and Anna in 2018 and 2019 telling the story of SQL Server, Big Data Clusters, and Azure. They both made me a better storyteller and

teacher and are fun on the road. The Microsoft SQL Server Engineering team is nothing short of amazing. I am in awe to be working with such intelligent and professional people, many of whom helped me with details you find in this book. It all has to start with Slava Oks and Travis Wright who helped tell me the story of Seattle and Aris and were both instrumental in pushing through much of this release including Big Data Clusters. Conor Cunningham continues to amaze me with his deep knowledge of this product while being instrumental in delivering a quality release.

The true heroes of this book are the engineering team members who built this release and helped me with various parts of the book. In no particular order, I want to say thank you to Pedro Lopes, Pam Lahoud, Amit Banerjee, Brian Carrig, Tejas Shah, Vin Yu, Sourabh Agarwal, Mihaela Blendea, Nellie Gustafsson, Abiola Oke, James Rowland-Jones, Scott Konersmann, Stuart Padley, David Kryze, Robert Dorr, Mitchell Sternke, Ross Monster, Madeline MacDonald, Dylan Gray, Joe Sack, Shreya Verma, Jakub Szymaszek, Joachim Hammer, Raghav Kaushik, Parag Paul, Panagiotis Antonopoulos, Michael Nelson, Pranjal Gupta, Jarupat Jisarojito, Weiyun Huang, George Reynya, Umachandar Jayachandran (UC), Sahaj Saini, Mike Habben, Vaqar Pirzada, Rony Chatterjee, Vicky Harp, Alan Yu, Jack Li, Alexey Eksarevskiy, Jay Choe, Argenis Fernandez, Kevin Farlee, Arieh Bibliowicz, Alex Umansky, Matteo Taveggia, Kapil Thacker, Li Zhang, and Dong Cao.

I also want to thank members of Microsoft Marketing and CSS teams for their help including Anshul Rampal, Matthew Burrows, Marko Hotti, Debbi Lyons, Suresh Kandoth, and Pradeep M M.

This book and my work would not have been possible without partners such as HPE, DELL, and Red Hat who allowed me to tell the story of SQL Server 2019 to their customers. Thanks to Wendy Harms, Bill Dunmire, Urs Renggli, Robert Sonders, Louis Imershein, and Stephane Bureau (my video guy). Special thanks also to David DeWitt for your insights into the history of Polybase, Brendan Burns for your insights and my foundation for Kubernetes knowledge, and Anthony Nocentino for your great knowledge of Linux and Containers.

Finally, thank you to the SQL Server Community across the world. We now pour releases of SQL Server at you faster than ever before, yet you still exhibit immense enthusiasm and appreciation every time I present on SQL Server.

Introduction

Like my first book *Pro SQL Server on Linux*, the pages you are about to read have seen some mileage. I've travelled more in the year 2019 than any in my lifetime. That meant I needed to be prepared to write wherever and whenever I could. This includes flights, hotels, trains, and car rides across cities like Seattle, London, Manchester (UK), Nashville, Las Vegas (multiple times), San Antonio, Austin, Houston, Orlando, St. Lucia (that was on vacation), Genesee (Colorado), Charleston, Boston, Dubai, Johannesburg (South Africa), Greenville (SC), and Indianapolis and late nights in my office at my home in North Richland Hills, Texas.

I thought after finishing my first book I would not be ready to write another one, but I couldn't resist the chance to tell the story of SQL Server 2019. This book really does represent that famous saying "A labor of love." I've put my heart and soul into learning, teaching, complaining about, breaking, documenting, testing, and using SQL Server 2019. This book represents all of that and more.

I wrote this book for data professionals and developers who have a fundamental knowledge of SQL Server but want a comprehensive look at SQL Server 2019 in one book. This book has plenty of examples, figures, and references to guide you along the way. I wrote this book so it would not only be a complete understanding of SQL Server 2019 but also as a reference you can come back to at any time.

While each chapter is independent, I highly recommend you start with Chapter 1 as it gives you the history and background of the release. I also set the stage for all the key capabilities of SQL Server 2019 and why I think it is a compelling product. From there, you can go through the book in chapter sequence or skip around some. One thing is for sure, in order to get the most out of Chapter 10 on Big Data Clusters, you must read Chapters 6, 7, 8, and 9 first.

The book is essentially broken down into these major sections:

- Chapter 1 to introduce the history and the overall SQL Server 2019 release.
- Chapters 2, 3, and 4 to cover performance, security, and availability. There is a lot in these chapters alone to get you excited about SQL Server 2019.

- Chapter 5 stands on its own for developers.
- Chapters 6, 7, and 8 are all about Linux, Containers, and Kubernetes.
- Chapter 9 introduces you to Data Virtualization with Polybase.
- Chapter 10 is a big chapter for a big topic: Big Data Clusters.
- Chapter 11 concludes the book by talking about other new features and migration.

I love "learning by example" so I've included many examples for almost every chapter in the book (and in some cases, I explain how to use an example already created). You can find all the examples for this book on GitHub using the link for the book's reference at `www.apress.com/9781484254189` or on my GitHub repo at `https://aka.ms/bobsqldemos` (`https://github.com/microsoft/bobsql`).

I also recommend you take a look at free training resources our team has built at `https://aka.ms/sqlworkshops`. This includes free hands-only lab training with SQL Server 2019!

For this book, I spent a lot of time thinking for each chapter "what would a reader want" on a particular topic or example. I hope you can see and feel that as you read the book. If you have any questions or issues with the book, I really want to hear about them. Please e-mail me directly at bobward@microsoft.com.

Bob Ward
North Richland Hills, Texas
September 2019

CHAPTER 1

Why SQL Server 2019?

In July of 2017, I made one of my regular visits to Redmond, Washington, as a member of the SQL Server engineering team. I live in North Richland Hills, Texas, and modern technology allows me to do much of my job remote from most of the SQL Engineering team. But I'm still a bit of an "old-school" person, and, in some cases, nothing beats working with people face to face. By July of 2017, I had been in the SQL Engineering team for over a year, focused mostly on SQL Server 2016 (see an example of my work on SQL Server 2016 on the Web at https://channel9.msdn.com/Events/Ignite/2016/BRK3043-TS).

Up until this time, I was a member of the famous Tiger Team, but, as part of my visit in 2017, I was asked to take on new tasks to focus specifically on the upcoming SQL Server 2017 release. This included SQL Server on Linux, which ultimately led to me authoring my first book, *Pro SQL Server on Linux* (www.apress.com/us/book/9781484241271). So on my visit, I started meeting and talking to various members of the team about SQL Server 2017 – performance enhancements, the overall set of new features, and the details behind SQL Server on Linux and Containers. One of the people I spoke with that week was Slava Oks. Slava is the lead development manager for SQL Server and one of the inventors of SQL Server on Linux. He wrote the foreword of *Pro SQL Server on Linux*, and Chapter 1 of that book talks about the history of his involvement in the project. At that time, Slava liked to come in early to the office; when I'm in Redmond I, too, try to work "Texas time" – which means I also come in very early. So we would often meet for coffee before most others were in the office, in Building 16, though now our team works in Building 43. One morning, as Slava and I talked about SQL Server 2017, he said to me, "Hey have I told you about our plans for the next version of SQL Server, the one after SQL Server 2017?" I of course pretended to know – "Sure, Slava, I've heard of it, but don't know the details." He then invited me to come to a meeting the next day where he would explain to many of our engineering team the plan for the project. I had just spent a year focusing on SQL Server 2016, was now assigned to dive into SQL Server 2017 and Linux, and here Slava wanted me to learn about the

B. Ward, *SQL Server 2019 Revealed*, https://doi.org/10.1007/978-1-4842-5419-6_1

release after the release that had not been shipped yet? Of course, I was not going to turn him down, because, well, it's Slava Oks. This may make it sound like Slava is some type of intimidating person, but he is one of the nicest people I've ever known at Microsoft. So while I was starting to pack my brain on the details of SQL Server 2017, I started down the path to learn about what we were doing for the future version of SQL Server, code named Project SQL Server *Seattle*.

Project Seattle

In the meeting the next day with Slava, I quickly learned in the span of a few hours we were embarking on one of the most ambitious enhancements to SQL Server I had ever seen in my career. I'm saying this with the knowledge already that we were bringing to market SQL Server on Linux, which nobody had previously thought was possible.

Slava and the team chose the code name "Seattle" because the team had used Helsinki for the code name for SQL Server 2017 and were looking for a new "city" name. Ironically, no one at Microsoft had used the name Seattle before, so it quickly stuck. I asked Slava when he first started planning Project Seattle. I was amazed to hear all the way back in January of 2017. The fact that folks like Slava, Conor Cunningham, and Travis Wright were planning Project Seattle while working on building the final pieces of SQL Server 2017 and Linux was a testament to both their dedication to the team and also their desire to keep SQL Server leading innovation in the database industry.

It was hard to believe we could so quickly plan something bigger after having delivered so many compelling and innovative features in SQL Server 2016 and SQL Server 2017.

In SQL Server 2016, we brought new performance diagnostic capabilities with Query Store. We included new features for developers such as temporal tables and JSON integration. We upped our game on security with Always Encrypted, dynamic data masking, and row-level security. And we introduced two new innovations outside the "normal" type of features for a relational database system. One of these was integration of the R language for Machine Learning models. The second was integration with Hadoop systems with a feature called *Polybase* (which will lead to something bigger in 2019, but I'm getting ahead of myself). Building features to enable new scenarios like Machine Learning and Big Data led myself and others at Microsoft to start pitching the idea that SQL Server was no longer just a relational database engine but a *data platform*.

However, to be modern and a complete data platform, we needed to be able to empower applications on systems other than just Windows Server. This led to our release of SQL Server 2017 with support for Linux and Docker Containers. Running on Linux and Containers was a very big move for Microsoft, but SQL Server 2017 also included other capabilities such as Adaptive Query Processing, automatic tuning, graph database, *clusterless* Availability Groups, and Python integration to complement R language support for Machine Learning Services.

With all of this innovation in mind, how could we in a short period of time plan and build something new, different, and exciting than SQL Server 2016 and 2017? I asked myself this question as I intently listened in my first Project Seattle meeting. In the first few minutes, I would be introduced to an idea that, when later announced to the public, would be considered quite radical. And that innovation started as the "big rock" of the Seattle project, which has a project name of its own: *Aris*.

Project Aris

In January of 2017, Slava and the leadership of the SQL Server engineering team were given direction by Rohan Kumar, Corporate Vice President of Azure Data, to look into how to integrate SQL Server with *Big Data*. Big Data is a term loosely used in the industry related to a data system that can handle *large* amounts of data, usually through a distributed, scalable computing platform. I personally like my colleague Buck Woody's definition of Big Data as, "Any data that you can't process in the time you want with the technology you have." And for many years, the preferred choice for a Big Data system has been Hadoop. So, for several months in the spring and summer of 2017, the team looked to Travis Wright for ideas on how to make the vision of Big Data integration a reality. During the summer of 2017, our Azure Data team had several projects underway with code names like Polaris, Socrates, and Plato. I asked Slava how did you decide on the name Aris? The answer: Socrates was the tutor of the famous Greek philosopher Plato, and Plato's pupil was Aristotle. Given that the word Aris is also part of the name Polaris, the name resonated with everyone on the team and our leadership.

Since integration for Big Data implied *something* to do with Hadoop, Travis spent several meetings with the team that brought Polybase to SQL Server 2016 and Azure Data Warehouse. The vision of Polybase was to allow SQL Server users to query (and ingest) data from a Hadoop system all through the T-SQL language so familiar to our existing customers. Furthermore, instead of just building a simple data extract system, Polybase

could use the power of distributed computing that exists with Azure Data Warehouse and Analytics Platform System (formerly known as Parallel Data Warehouse) to *push down* computations and partition query processing to achieve scalable performance against large datasets in the target Hadoop system. I never really saw Polybase take off in SQL Server 2016 and 2017, since integrating Big Data Hadoop systems with relational systems like SQL Server was not easy. Polybase requires a significant amount of installation and configuration, and security models differ from Hadoop systems and SQL Server. In addition, the pushdown computation implementation relied on a concept called MapReduce, requiring Java to be installed on the same computer as SQL Server and Polybase services. Still, the architecture and the concepts for integrated SQL Server and Big Data systems were available to build something bigger (including a T-SQL extension called EXTERNAL TABLE). If we could simplify the deployment and configuration story for Polybase, and add in more data source support, it might become more adopted in the industry. Furthermore, Travis came to learn very quickly that, if you wanted to be taken seriously in the Big Data world of data processing, you needed to consider another technology called *Spark*.

Armed with this knowledge, Slava, Travis, and a core set of members of the team that built SQL Server on Linux had a goal to build a prototype of SQL Server integration with Big Data including Spark. They embarked on a multi-day huddle in a big conference room and dubbed it the "Aris Hackathon." Those team members were Slava Oks, Travis Wright, Scott Konersmann, Stuart Padley, Michael Nelson, Pranjal Gupta, Jarupat Jisarojito, Weiyun Huang, George Reynya, David Kryze, Umachandar Jayachandran (UC), and Sahaj Saini. By the time they were done, they had a working *cluster* that combined the existing Polybase functionality of SQL Server with Spark. Figure 1-1 shows a rough diagram of the cluster the team built.

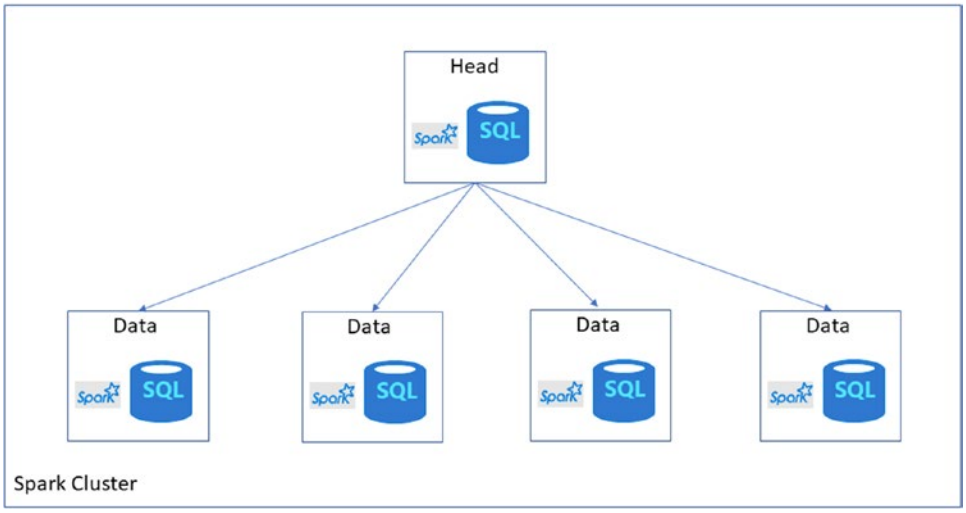

Figure 1-1. *The first Aris cluster*

In the prototype, they built a Hadoop cluster including components for Apache Spark and HDFS, but also combined with SQL Server Polybase. They used Spark to stream data into the *Data* nodes and then used Polybase to join data in the Head node in SQL Server with the data ingested with Spark into HDFS. The idea behind the prototype was to prove they could integrate Spark, Hadoop, and SQL Server together.

Around this same time, Travis had been talking to engineers who had joined the team from a company Microsoft had acquired, called Metanautix. As part of this acquisition, our team had technology to connect to a range of data sources, through ODBC, including ORACLE, SQL Server, Teradata, and MongoDB. The team thought that if we could integrate this technology with the Aris project, we could build a pretty compelling story for *Data Virtualization*. SQL Server could now be a hub for accessing data in different data platforms and systems without having to move the data to SQL Server (with techniques like Extract, Transform, and Load (ETL)).

Before we could deliver software that customers could use and try, we needed to decide on a platform to run all of these components. We needed a platform that would allow for easy deployment of all the software, including Polybase, Hadoop, and Spark; provide manageability and security; and enable elastic scale and high availability. Containers seemed like a logical choice given the nature of how easy they are to deploy, and, with SQL Server 2017, we had delivered on supporting SQL Server with containers. The next natural choice for the team was to select Kubernetes as a platform to build out a cluster running these containers. Kubernetes was quickly gaining momentum as a

platform for distributed computing and scalable performance. Our learnings had taught us that Linux was the preferred OS to run Kubernetes and Hadoop systems, and, since SQL Server was already supported on Linux, it was a good fit to build on.

And so, in late 2017, our team embarked on the journey of building out an *Aris* cluster that would enable the vision of Data Virtualization, but integrate with Big Data technologies such as Spark and HDFS. From the very beginning, our team decided that all of this needed to "ship in the box." That is, if you bought SQL Server, we would install all of these components as part of the license (not knowing whether this would be a new edition, but all of this would be included with SQL Server). The final product as you see now with SQL Server 2019 and what we call *Big Data Clusters* has much more than the early Aris prototypes, but the vision and concepts are the same: provide an easy-to-deploy Data Virtualization platform with built-in scalable performance, security, and manageability.

Seattle Becomes SQL Server 2019

While the concept of Aris and Big Data clusters was huge, innovative, and, quite frankly, a bit scary, every major release of SQL Server includes enhancements across several areas of the platform. This includes performance, security, and availability, the three areas Conor Cunningham often refers to as "the meat and potatoes of SQL Server." Our team had also launched SQL Server on Linux with SQL Server 2017. As amazing as that product has been, there were a few features that ship with SQL Server on Windows that needed to also be added to Linux. We also knew that containers are big, and I mean big in the sense that they are a future direction to deploy and run applications, including SQL Server. So there was some work there we know we needed to do, including exploring new scenarios with Kubernetes clusters (not just the Big Data Cluster solution).

So many teams contribute to the amazing product that is SQL Server. Our Enterprise team (aka the Tiger Team) had a pile of new features they wanted in the new release with true customer value (because that is what they do!). Our friends who build new features for performance, availability, and security for Azure SQL Database wanted to see their work in Project Seattle, since the engines that run the Azure service and SQL Server are the same. As I saw this play out in 2017, I could see the momentum for a historic release.

As the calendar year of 2017 ended, we were all set up for the next release of SQL Server, SQL Server 2018. This all made sense to me. We shipped two major versions of SQL Server in back to back years, SQL Server 2016 and SQL Server 2017, so why not SQL Server 2018?

Conor Cunningham, our product and release architect, has told me that, with our agile engineering capabilities, we could ship SQL Server every month if we wanted to. And we can do it with quality. Of course, we don't do this, because we want to ship SQL Server releases that have both quality and major value for our customers. As we started moving forward into the early months of calendar year 2018, we had to decide if we wanted to ship a major new version in that year. When we looked at the landscape of capabilities that we could put into this release, including Big Data Clusters, we made the decision in the spring of 2018 that we would ship our first preview of SQL Server vNext in calendar year 2018. (When we don't know an official name to call the next release, even if we have a project name like Seattle, we call it "vNext.") And you may have noticed we often try to make announcements for major new releases at big events. Looking at the calendar, one of the biggest global customer events for Microsoft has become Microsoft Ignite (it is now in Orlando, with ~30,000 people). So in the summer of 2018, our leadership decided to launch the preview of SQL Server vNext at Microsoft Ignite and call it SQL Server 2019, meaning that we would make this release GA (which means General Availability) sometime in calendar year 2019.

This made sense to everyone on the team. It gave us more runway to land Big Data Clusters, plus more capabilities with the "core" of SQL Server all based on customer feedback and experience. My task? Take the work I had done to evangelize and showcase SQL Server 2016 and 2017 and show our customers, the industry, and community that we have truly built a *Modern Data Platform* with SQL Server 2019.

Modernizing Your Database with SQL Server 2019

Figure 1-2 is my main "pitch" diagram when I talk about SQL Server 2019. Built by one of my colleagues in Microsoft marketing, Debbi Lyons (you may have seen myself and Debbi sometimes appearing together talking SQL Server), it represents a full picture of the new Modern Data Platform of SQL Server 2019.

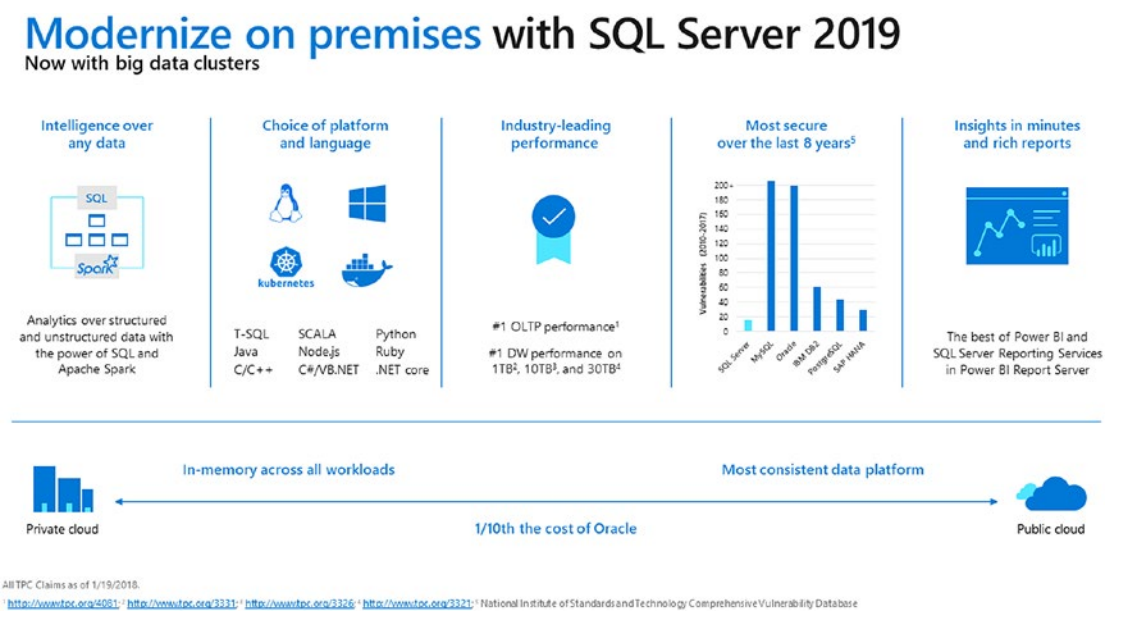

Figure 1-2. *Modernize with SQL Server 2019*

If you have ever seen me talk about SQL Server 2016 or 2017, you will notice the slide looks a bit similar, but with key differences:

- An integrated Data Virtualization solution integrating Spark, HDFS, and SQL Server in a new and innovative way (basically SQL Server "meets Big Data")
- New capabilities to continue the platform of choice value to our customers across Windows, Linux, Containers, and Kubernetes

SQL Server continues to lead the database industry in performance and is the least vulnerable data platform over the last decade. With a SQL Server license, customers have access to Business Intelligence services, such as Power BI Report Server. In addition, with the new Azure SQL Database Managed Instance service, functionality is virtually the same from SQL Server in your private cloud and Azure in the public cloud. The consistency message doesn't stop there. Your skills in T-SQL apply across SQL Server and Azure, and our tools continue to work seamlessly across SQL Server and Azure Data services.

Another set of capabilities that seems to get lost in the conversation of new features is that SQL Server (and Azure) provides in-memory features that allow you to maximize your computing resources, including In-Memory OLTP and Columnstore Indexes. All of this comes with the SQL Server 2019. Figure 1-3 is a more detailed picture of major new key functionality unique to SQL Server 2019.

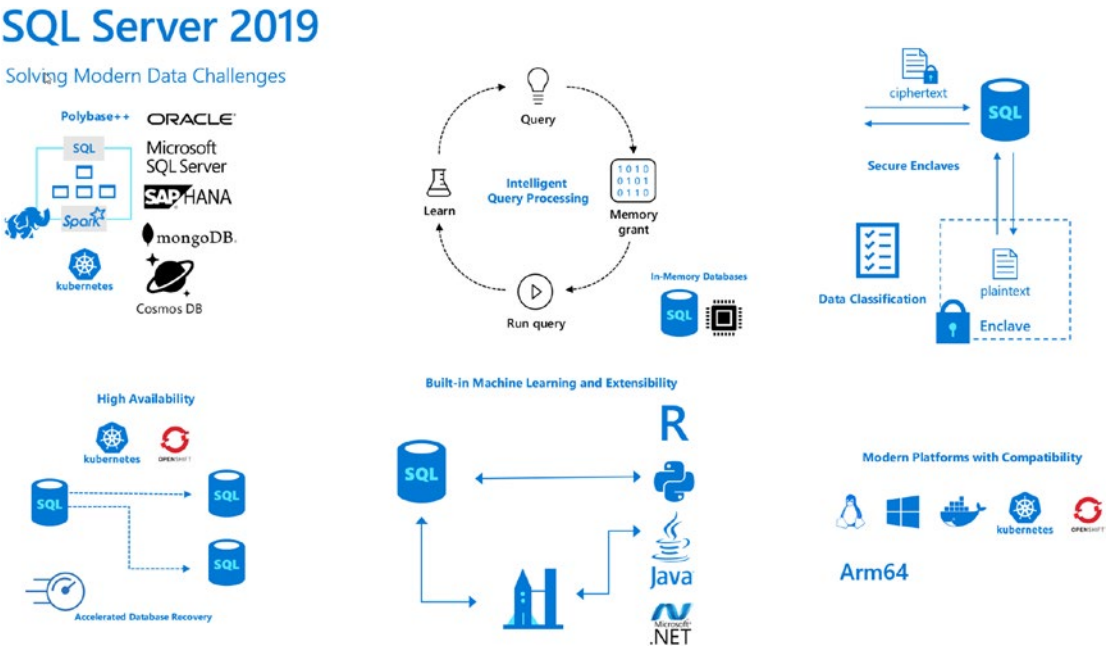

Figure 1-3. *SQL Server 2019 key functionality*

I'm going to use this diagram (going left to right, starting in the upper left-hand corner) to sketch out for you the major new features of SQL Server 2019, which will be like a blueprint for your reading for the remainder of the book. As you read through these new capabilities, keep in mind that **SQL Server powers Azure SQL Database**, which means many of the capabilities you see in this book work the same in Azure SQL Database. Furthermore, everything you see in this book can be done in Azure whether it is SQL Server in Azure Virtual Machine or containers and Kubernetes in the cloud.

Data Virtualization

Previously in this chapter, I've discussed the origins of Data Virtualization with Project Aris. SQL Server 2019 is the realization of that vision with two specific capabilities:

- **Polybase in SQL Server 2019**

  I call this Polybase++ because we have extended the functionality of Polybase that shipped with SQL Server 2016 (for more info on Polybase, see https://docs.microsoft.com/en-us/sql/relational-databases/polybase/polybase-guide?view=sql-server-2017) to provide different data source connectors including Oracle, SQL Server, MongoDB (CosmosDB), and Teradata. And you can connect to these data sources without installing any client software; SQL Server has what you need built-in. In addition, you can connect to other sources such as SAP HANA by installing your own ODBC driver. I'll cover the new Polybase in SQL Server 2019 in Chapter 9.

- **Big Data Clusters**

  As I described our vision for Project Aris earlier in the chapter, we decided to build a complete solution that deploys SQL Server with the new Polybase functionality, HDFS, Spark, and other components for management, security, and availability. There is so much more to this than I can describe here, so read more on Big Data Clusters in Chapter 10.

Note I originally wanted to come right out in the second and third chapters of this book on these topics. However, I later decided that if you need some more information about containers and Kubernetes, it would help to put those chapters ahead of this topic. So, instead, I'll "go out with a bang" with this new innovation in the book. If you can't help yourself, dive right into Chapter 9.

Performance

We always work on performance in any SQL Server release. Always. However, just making your queries run fast is not enough. We need to keep making the SQL Server engine smarter and more intelligent, adapting to your workload, hardware investments, and complex query patterns. Chapter 2 has a complete look at performance capabilities of SQL Server 2019 including but not limited to

- *Intelligent Query Processing*, which is an extension to Adaptive Query Processing introduced in SQL Server 2017.
- Query plan *insights anywhere and anytime* you need it with Lightweight Query Profiling, Last Execution Plan, and Query Store enhancements.
- A family of capabilities to provide a true *in-memory database* including enlightened I/O and Hybrid Buffer Pool for persistent memory and memory-optimized tempdb schema. Combining these technologies with our built-in Columnstore Indexes and In-Memory OLTP provides a compelling in-memory database solution.

Security

SQL Server is not only the least vulnerable database product in the industry over the last decade, but includes a wide range of features and tools to meet the modern security needs of any business. This includes the following enhancements for SQL Server 2019:

- **Always Encrypted with Secure Enclaves**

  SQL Server 2016 introduced a new end-to-end security system for data applications called *Always Encrypted*. While this system provides for encryption at rest, in-memory, and across the network, there were a few limitations, most importantly *rich computing*. In Chapter 3, I'll talk about how Always Encrypted, using a concept called *Secure Enclaves*, enables rich computing and other interesting security scenarios.