Microsoft

Administering Microsoft Azure SQL Solutions

Exam Ref DP-300

Craig Zacker

FREE SAMPLE CHAPTER |

Microsoft

Exam Ref DP-300 Administering Microsoft Azure SQL Solutions

Craig Zacker

Exam Ref DP-300 Administering Microsoft Azure SQL Solutions

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

Copyright © 2026 by Pearson Education, Inc.

Hoboken, New Jersey

All rights reserved. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, request forms, and the appropriate contacts within the Pearson Education Global Rights & Permissions Department, please visit www.pearson.com/global-permission-granting.html.

No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein. Please contact us with concerns about any potential bias at https://www.pearson.com/report-bias.html.

ISBN-13: 978-0-13-795656-2
ISBN-10: 0-13-795656-8

Library of Congress Control Number: 2025941870

$PrintCode

TRADEMARKS

Microsoft and the trademarks listed at http://www.microsoft.com on the "Trademarks" webpage are trademarks of the Microsoft group of companies. All other marks are property of their respective owners.

WARNING AND DISCLAIMER

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author, the publisher, and Microsoft Corporation shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the programs accompanying it.

EDITOR-IN-CHIEF
Julie Phifer

EXECUTIVE EDITOR
Loretta Yates

ASSOCIATE EDITOR
Shourav Bose

DEVELOPMENT EDITOR
Songlin Qiu

MANAGING EDITOR
Sandra Schroeder

SENIOR PROJECT EDITOR
Tracey Croom

COPY EDITOR
Charlotte Kughen

INDEXER
Timothy Wright

PROOFREADER
Barbara Mack

TECHNICAL EDITOR
Vince Averello

COVER DESIGNER
Twist Creative, Seattle

COMPOSITOR
codeMantra

Contents at a glance

	About the author	*x*
	Introduction	*xi*
CHAPTER 1	Plan and implement data platform resources	1
CHAPTER 2	Implement a secure environment	65
CHAPTER 3	Monitor, configure, and optimize database resources	111
CHAPTER 4	Configure and manage automation of tasks	157
CHAPTER 5	Plan and configure a high availability and disaster recovery (HA/DR) environment	187
CHAPTER 6	DP-300 Administering Microsoft Azure SQL Solutions exam updates	217
	Index	*223*

This page intentionally left blank

Contents

Introduction **xi**

Organization of this book *xi*

Preparing for the exam *xi*

Microsoft certifications *xii*

Access the Exam Updates chapter and online references *xii*

Errata, updates, & book support *xiii*

Stay in touch *xiii*

Chapter 1 Plan and implement data platform resources **1**

Skill 1.1: Plan and deploy Azure SQL solutions 1

Deploy database offerings on selected platforms 2

Understand automated deployment 16

Apply patches and updates for hybrid and infrastructure as a service (IaaS) deployment 18

Deploy hybrid SQL Server solutions 19

Recommend an appropriate database offering based on specific requirements 21

Evaluate the security aspects of the possible database offering 25

Recommend a table partitioning solution 26

Recommend a database sharding solution 28

Skill 1.2: Configure resources for scale and performance 30

Configure Azure SQL Database for scale and performance 31

Configure Azure SQL Managed Instance for scale and performance 33

Configure SQL Server on Azure Virtual Machines for scale and performance 35

Configure table partitioning 38

Configure data compression 39

Skill 1.3: Plan and implement a migration strategy 40

Evaluate requirements for the migration 41

Evaluate offline or online migration strategies 43

Implement an online migration strategy 47

Implement an offline migration strategy 51
Perform post-migration validations 55
Troubleshoot a migration 57
Set up SQL Data Sync for Azure 58
Implement a migration to Azure 59
Implement a migration between Azure SQL services 60
Chapter summary 62
Thought experiment 62
Thought experiment answers 63

Chapter 2 Implement a secure environment 65

Skill 2.1: Configure database authentication and authorization 65
Configure authentication by using Active Directory and Microsoft Entra ID 66
Create users from Microsoft Entra identities 69
Configure security principals 70
Configure database and object-level permissions using graphical tools 73
Apply the principle of least privilege for all securables 76
Troubleshoot authentication and authorization issues 78
Manage authentication and authorization by using T-SQL 80
Skill 2.2: Implement security for data at rest and data in transit 81
Implement transparent data encryption (TDE) 81
Implement object-level encryption 83
Configure server- and database-level firewall rules 84
Implement Always Encrypted 85
Implement Always Encrypted with VBS enclaves 88
Configure secure access 89
Configure Transport Layer Security (TLS) 91
Skill 2.3: Implement compliance controls for sensitive data 92
Apply a data classification strategy 93
Configure server and database audits 96
Implement data change tracking 98
Implement dynamic data masking 102
Manage database resources by using Azure Purview 103

Implement database ledger in Azure SQL 104
Implement row-level security 106
Configure Microsoft Defender for SQL 107
Chapter summary 108
Thought experiment 109
Thought experiment answers 109

Chapter 3 Monitor, configure, and optimize database resources 111

Skill 3.1: Monitor resource activity and performance 111
Prepare an operational performance baseline 112
Determine sources for performance metrics 115
Interpret performance metrics 116
Configure and monitor activity and performance 117
Monitor by using SQL Insights 118
Monitor by using database watcher 121
Monitor by using extended events 122
Skill 3.2: Monitor and optimize query performance 125
Configure Query Store 126
Monitor by using Query Store 128
Identify sessions that cause blocking 131
Identify performance issues using dynamic management views (DMVs) 133
Identify and implement index changes for queries 136
Recommend query construct modifications based on resource usage 137
Assess the use of query hints for query performance 138
Review execution plans 139
Monitor by using Intelligent Insights 140
Skill 3.3: Configure database solutions for optimal performance 141
Implement index maintenance tasks 142
Implement statistics maintenance tasks 144
Implement database integrity checks 146
Configure database automatic tuning 148
Configure server settings for performance 148
Configure Resource Governor for performance 149

Implement database-scoped configuration 150
Configure compute and storage resources for scaling 151
Configure intelligent query processing (IQP) 152
Chapter summary 154
Thought experiment 155
Thought experiment answers 155

Chapter 4 Configure and manage automation of tasks 157

Skill 4.1: Create and manage SQL Server Agent jobs 157
Manage schedules for regular maintenance jobs 157
Configure job alerts and notifications 161
Troubleshoot SQL Server Agent jobs 164
Skill 4.2: Automate deployment of database resources 168
Automate deployment by using Azure Resource Manager (ARM) and Bicep templates 169
Automate deployment by using Azure CLI and PowerShell 173
Monitor and troubleshoot deployments 174
Skill 4.3: Create and manage database tasks in Azure 176
Create and configure elastic jobs 176
Create and configure database tasks by using automation 177
Configure alerts and notifications on database tasks 182
Troubleshoot automated database tasks 185
Chapter summary 185
Thought experiment 186
Thought experiment answers 186

Chapter 5 Plan and configure a high availability and disaster recovery (HA/DR) environment 187

Skill 5.1: Recommend an HA/DR strategy for database solutions 187
Recommend HA/DR strategy based on Recovery Point Objective/Recovery Time Objective (RPO/RTO) requirements 188
Evaluate HA/DR for hybrid deployments 190
Evaluate Azure-specific HA/DR solutions 191

Recommend a testing procedure for an HA/DR solution 191

Skill 5.2: Plan and perform backup and restore of a database. 192

Recommend a database backup and restore strategy 192

Perform a database backup by using database tools 193

Perform a database restore by using database tools 195

Perform a database restore to a point in time 196

Configure long-term backup retention 197

Back up and restore a database by using T-SQL 198

Back up and restore to and from cloud storage 199

Skill 5.3: Configure HA/DR for database solutions 202

Configure active geo-replication 202

Configure an Always On availability group on Azure virtual machines 204

Configure failover groups 206

Configure quorum options for a Windows Server Failover Cluster 207

Configure Always On Failover Cluster Instances on Azure virtual machines 209

Configure log shipping 210

Monitor an HA/DR solution 211

Troubleshoot an HA/DR solution 212

Chapter summary 213

Thought experiment. 214

Thought experiment answers 215

Chapter 6 DP-300 Administering Microsoft Azure SQL Solutions exam updates 217

The purpose of this chapter 217

About possible exam updates 217

Impact on you and your study plan 218

Exam objective updates. 218

Updated technical content 218

Objective mapping 218

Index *223*

About the author

CRAIG ZACKER is the author or coauthor of dozens of books, manuals, articles, and websites on computer and networking topics. He has also been an English professor, a technical and copy editor, a network administrator, a webmaster, a corporate trainer, a technical support engineer, a minicomputer operator, a literature and philosophy student, a library clerk, a photographic darkroom technician, a shipping clerk, and a newspaper boy.

Introduction

This book takes a high-level approach to SQL Server administration in the Azure cloud environment, covering both the native Azure SQL implementations—Azure SQL Database and Azure SQL Managed Instance—and SQL Server installations on Azure virtual machines (VMs). SQL administrators might host their databases solely in the cloud, or they might use a hybrid environment consisting of VMs that supplement on-premises SQL servers. Azure and SQL Server provide a variety of tools that administrators can use to manage their SQL installations, ranging from the SQL Server Management Studio (SSMS) running on Windows to Azure portal utilities and services to Transact-SQL commands.

This book covers every major topic area found on the exam, but it does not cover every exam question. Only the Microsoft exam team has access to the exam questions, and Microsoft regularly adds new questions to the exam, making it impossible to cover specific questions. You should consider this book a supplement to your relevant real-world experience and other study materials. If you encounter a topic in this book that you do not feel completely comfortable with, use the "Need more review?" links you'll find in the text to find more information and take the time to research and study the topic.

Organization of this book

This book is organized by the "Skills measured" list published for the exam. The "Skills measured" list is available for each exam on the Microsoft Learn website: *microsoft.com/learn*. Each chapter in this book corresponds to a major topic area in the list, and the technical tasks in each topic area determine a chapter's organization. If an exam covers six major topic areas, for example, the book will contain six chapters.

Preparing for the exam

Microsoft certification exams are a great way to build your resume and let the world know about your level of expertise. Certification exams validate your on-the-job experience and product knowledge. Although there is no substitute for on-the-job experience, preparation through study and hands-on practice can help you prepare for the exam. This book is *not* designed to teach you new skills.

We recommend that you augment your exam preparation plan by using a combination of available study materials and courses. For example, you might use the *Exam Ref* and another study guide for your at-home preparation and take a Microsoft Official Curriculum course for

the classroom experience. Choose the combination that you think works best for you. Learn more about available classroom training, online courses, and live events at *microsoft.com/learn*.

Note that this *Exam Ref* is based on publicly available information about the exam and the author's experience. To safeguard the integrity of the exam, authors do not have access to the live exam.

Microsoft certifications

Microsoft certifications distinguish you by proving your command of a broad set of skills and experience with current Microsoft products and technologies. The exams and corresponding certifications are developed to validate your mastery of critical competencies as you design and develop, or implement and support, solutions with Microsoft products and technologies both on-premises and in the cloud. Certification brings a variety of benefits to the individual and to employers and organizations.

> ***NOTE*** **ALL MICROSOFT CERTIFICATIONS**
>
> **For information about Microsoft certifications, including a full list of available certifications, go to *microsoft.com/learn*.**

Access the Exam Updates chapter and online references

The final chapter of this book, "DP-300 Administering Microsoft Azure SQL Solutions exam updates," will be used to provide information about new content per new exam topics, content that has been removed from the exam objectives, and revised mapping of exam objectives to chapter content. The chapter will be made available from the link at the end of this section as exam updates are released.

Throughout this book are addresses to webpages that the author has recommended you visit for more information. Some of these links can be very long and painstaking to type, so we've shortened them for you to make them easier to visit. We've also compiled them into a single list that readers of the print edition can refer to while they read.

The URLs are organized by chapter and heading. Every time you come across a URL in the book, find the hyperlink in the list to go directly to the webpage.

Download the Exam Updates chapter and the URL list at *MicrosoftPressStore.com/ERDP300/downloads*

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/ERDP300/errata

If you discover an error that's 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 *support.microsoft.com*.

Stay in touch

Let's keep the conversation going! We're on X: *X.com/MicrosoftPress*.

This page intentionally left blank

CHAPTER 1

Plan and implement data platform resources

Microsoft Azure provides several means for deploying SQL databases in the cloud. These are designed to accommodate subscribers with existing on-premises SQL Server deployments, as well as those seeking to implement their SQL databases entirely in the cloud.

Skills covered in this chapter:

- 1.1: Plan and deploy Azure SQL solutions
- 1.2: Configure resources for scale and performance
- 1.3: Plan and implement a migration strategy

Skill 1.1: Plan and deploy Azure SQL solutions

There are as many SQL database configurations as there are SQL instances. When Microsoft designed its cloud-based SQL offerings, it was with the intention of creating flexible SQL implementations that could accommodate both organizations first expanding into the cloud and those already familiar with it.

This skill covers how to:

- Deploy database offerings on selected platforms
- Understand automated deployment
- Apply patches and updates for hybrid and infrastructure as a service (IaaS) deployment
- Deploy hybrid SQL Server solutions
- Recommend an appropriate database offering based on specific requirements
- Evaluate the security aspects of the possible database offering
- Recommend a table partitioning solution
- Recommend a database sharding solution

Deploy database offerings on selected platforms

Microsoft provides cloud-based SQL offerings on an Infrastructure as a Service (IaaS) or Platform as a Service (PaaS) basis. IaaS and PaaS are the service models that define where the administrative dividing line is between the cloud service provider and the client subscriber.

IaaS vs. PaaS SQL Implementations

As shown in Figure 1-1, the IaaS model limits the service provider's responsibility to the bottom four layers of the cloud infrastructure. In the PaaS model, the service provider is responsible for more: the bottom eight infrastructure layers. Therefore, more administrative responsibility falls to the subscriber in the IaaS model, while PaaS subscribers have less administration to worry about.

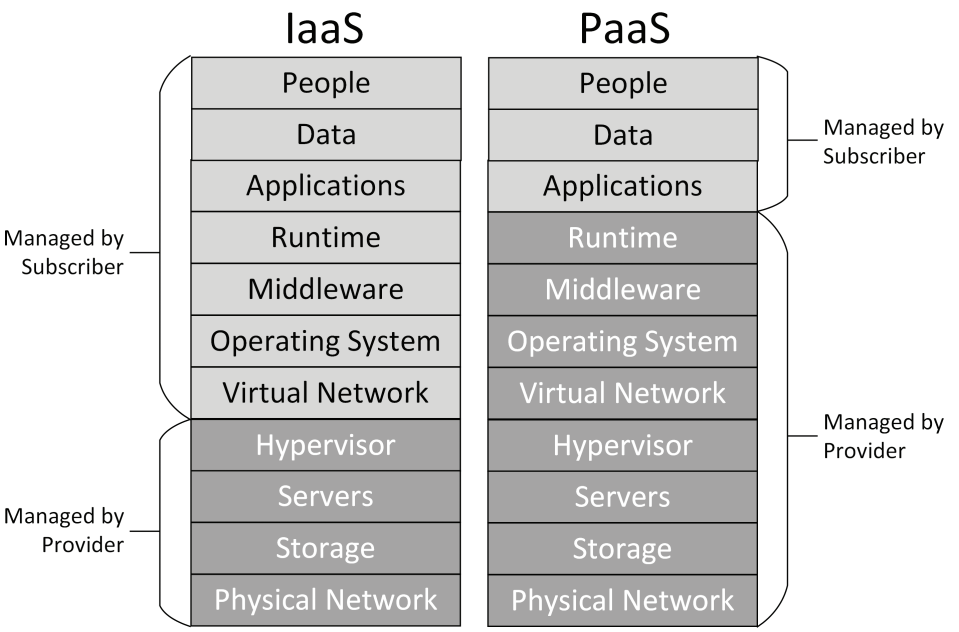


FIGURE 1-1 Shared responsibility in the IaaS and PaaS cloud service models

Thus, between its IaaS and PaaS offerings, the Microsoft Azure SQL products are as follows:

- SQL Server on an Azure virtual machine (IaaS)
- Azure SQL Database (PaaS)
- Azure SQL Managed Instance (PaaS)

All three products have advantages and disadvantages, and there are numerous deployment options for each one, as described in the following sections.

Deploying SQL Server on an Azure Virtual Machine

Microsoft Azure enables subscribers to lease access to a virtual machine in the cloud and do with it what they will. The subscriber can provision the virtual machine with whatever storage and compute resources are needed through the Azure portal and then install a SQL Server implementation on it. This is the only IaaS option Microsoft Azure offers for a SQL cloud deployment.

A SQL Server installation on an IaaS Azure VM is functionally no different from an on-premises installation. For organizations with on-premises SQL Server deployments that are considering a first expansion into the cloud, the IaaS model is a convenient option that enables the existing SQL Server administrators to continue their standard operating practices without retraining.

The IaaS model also provides other deployment advantages that the PaaS options do not, including the following:

- **Version freedom** Subscribers can install any version of any SQL implementation on an IaaS virtual machine, including older versions and open source distributions. The PaaS SQL options all use the most current Azure SQL version and are continually updated.
- **SQL services** Subscribers can install additional SQL Server tools on an IaaS VM that are not included with the PaaS SQL implementations, such as SQL Server Integration Services (SSIS), SQL Server Analysis Services (SSAS), and SQL Server Reporting Services (SSRS).
- **Application compatibility** In some instances, applications might require specific features or services to run on the same system as the SQL database instance. This might not be possible in the PaaS SQL options, but an IaaS VM can run any combination of software products possible on a physical computer.

There are also elements of the IaaS model that some subscribers would consider to be disadvantages (though others might disagree), such as the following:

- **Maintenance** On an IaaS virtual machine, the subscriber is wholly responsible for the configuration and maintenance of the operating system and applications, including SQL Server. This includes applying all patches and updates, which some administrators might consider to be an extra chore; others might appreciate the freedom to select and evaluate updates before installing them. On the PaaS SQL products, the OS and SQL are updated automatically.
- **Licensing** The IaaS subscriber is responsible for obtaining and maintaining the appropriate licenses for all software running on a virtual machine, including the operating system and SQL Server. Azure subscribers can lease a fully configured and licensed SQL Server image from the Azure Marketplace and pay a per-minute fee (Pay As You Go), or they can apply an existing SQL Server license from their on-premises installation (Bring Your Own License) using the Azure Hybrid Benefit.

NEED MORE REVIEW? **USING THE AZURE HYBRID BENEFIT**

The Azure Hybrid Benefit enables Azure subscribers with existing Windows and SQL Server licenses for their on-premises servers to apply those licenses to the same software running on an Azure VM. The existing licenses must include Software Assurance, and subscribers are required to indicate in the settings on the VM's Basics tab that the VM is operating using the Azure Hybrid Benefit. Subscribers can change the license model of a VM at any time using the Azure portal, the Azure command-line interface (CLI), or Microsoft PowerShell.

The Azure Hybrid Benefit can yield significant savings on licensing costs, but subscribers are still responsible for the compute, storage, and I/O charges incurred by the virtual machine, as well as the costs of any other services the VM runs. To determine the savings available through the Azure Hybrid Benefit, subscribers can use the Azure Hybrid Benefit Savings Calculator, located at *https://azure.microsoft.com/en-us/pricing/hybrid-benefit/*.

USING THE AZURE MARKETPLACE

Azure subscribers who are so inclined can create a SQL Server installation on an Azure virtual machine manually by creating a new VM in the Azure portal and then installing a SQL Server product on it. This might be a comfortable process for subscribers more accustomed to on-premises server installations, but Azure provides simpler methods for deploying SQL Server on a VM.

The simplest way to deploy SQL Server on an Azure virtual machine is to select one of the dozens of templates in the Azure Marketplace. The SQL-related templates provide various combinations of operating systems versions and SQL implementations, as shown in Figure 1-2.

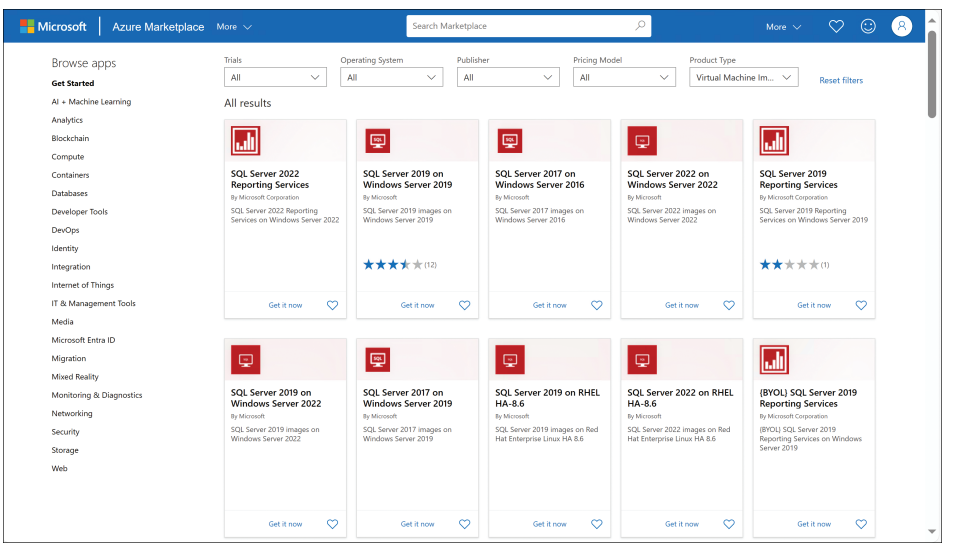


FIGURE 1-2 SQL templates in the Azure Marketplace

Selecting one of the Azure Marketplace templates displays an Overview tab like the one shown in Figure 1-3.

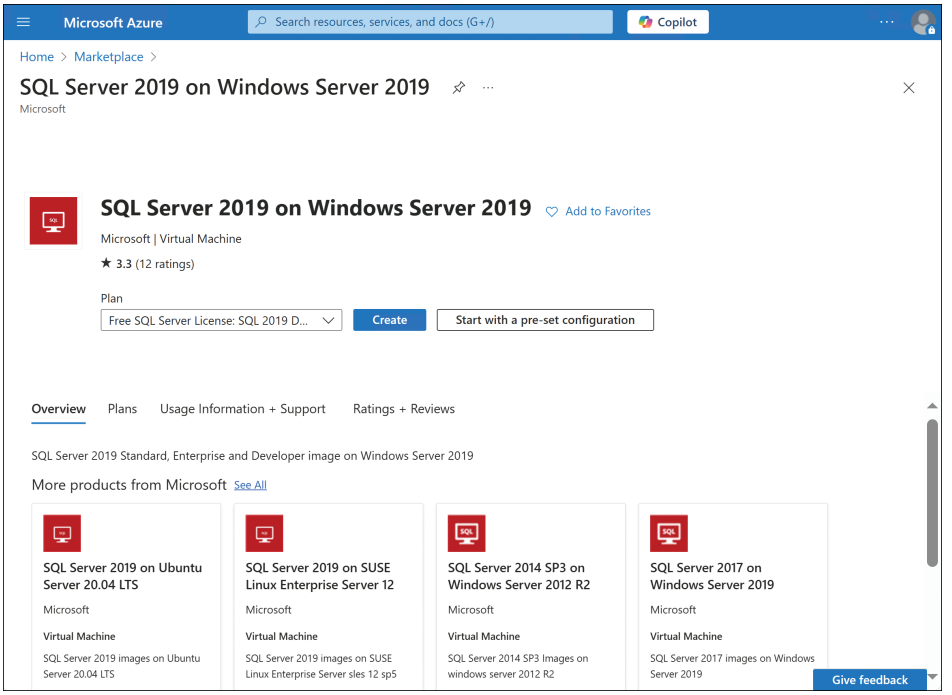


FIGURE 1-3 The Overview tab of the SQL Server 2019 on Windows Server 2019 template in the Azure Marketplace

The template's Plans tab lists the various images that the subscriber can select for installation on the virtual machine. For example, the SQL Server 2019 on Windows Server 2019 template provides six available plans, as follows:

- SQL Server 2019 Enterprise on Windows Server 2019
- SQL Server 2019 Standard on Windows Server 2019
- SQL Server 2019 Web on Windows Server 2019
- SQL Server 2019 Standard on Windows Server 2019 Database Engine Only
- SQL Server 2019 Enterprise on Windows Server 2019 Database Engine Only
- Free SQL Server License: SQL Server 2019 Developer on Windows Server 2019

The images associated with the plans can provide subscribers with the various SQL Server 2019 editions (Web, Standard, and Enterprise) and also specify what other elements of the SQL Server product should be included. The Database Engine Only plans include only the basic SQL functionality, whereas the other versions include additional services, such as SQL Server Management Studio (SSMS), SSIS, SSAS, and SSRS. The SQL Server 2019 Developer on Windows Server 2019 plan is the same as the Enterprise plan, except that the included free SQL Server license allows the VM to be used for testing and development only, not production.

Selecting a plan from the dropdown list shown in Figure 1-4 loads it into the Create a virtual machine dialog in the Microsoft Azure portal, where the subscriber can configure the plan to deploy with a preset hardware configuration for the VM.

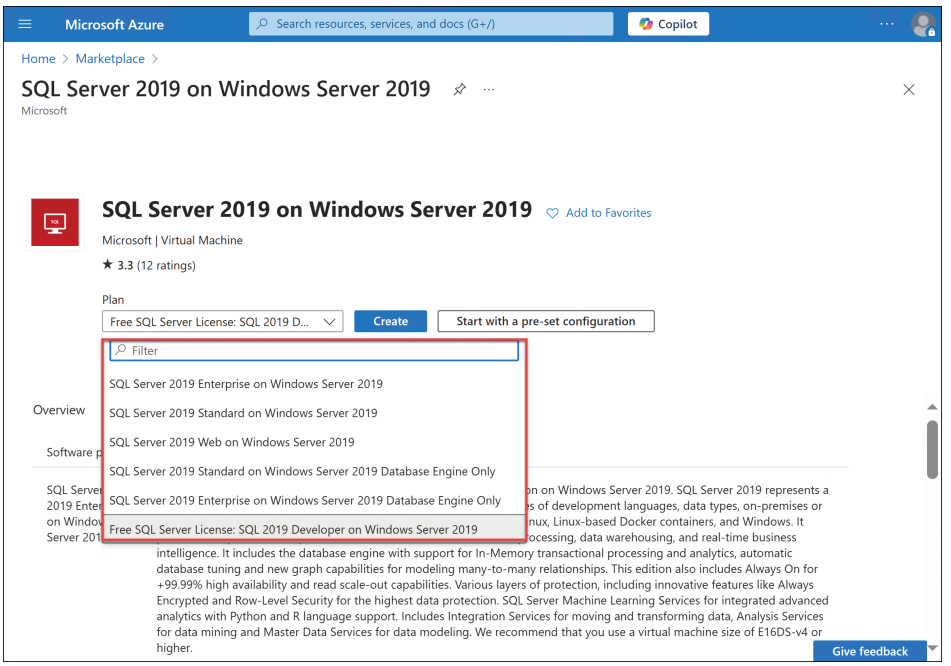


FIGURE 1-4 The SQL Server 2019 on Windows Server 2019 template plans

When the subscriber clicks the Start with a preset configuration button, the template typically displays recommendations for VM hardware based on the VM's intended workload, as shown in Figure 1-5. Subscribers can modify a virtual machine's hardware configuration at any time, but the Azure Marketplace templates can help to simplify the configuration process.

After selecting a virtual hardware configuration for the VM, a tabbed dialog appears, as shown in Figure 1-6, in which the subscriber supplies values for some basic VM settings, including names for the virtual machine, the resource group, the virtual network, and credentials for the VM's administrative account. The tabs also include a large number of other virtual machine and SQL Server configuration settings. The template supplies default values for many of the other VM settings that are typical for the selected plan.

In addition to the tabs devoted to VM configuration settings, the dialog also includes a SQL Server settings tab, as shown in Figure 1-7, containing basic configuration parameters for the SQL Server 2019 product. This tab is essentially a benefit of the IaaS Agent, which provides access to the SQL Server configuration before it's even installed.

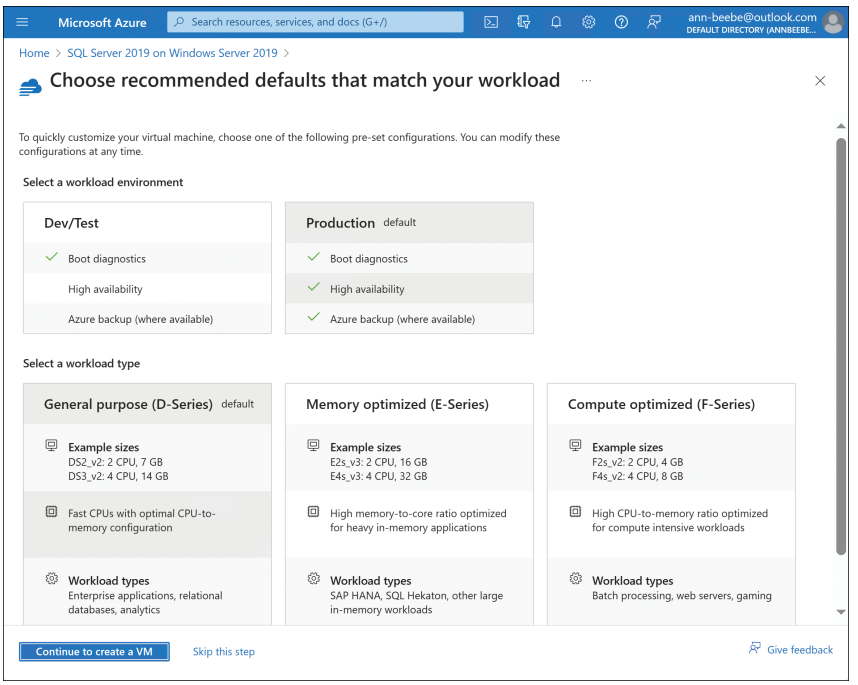


FIGURE 1-5 The Choose recommended defaults that match your workload page in the Microsoft Azure portal

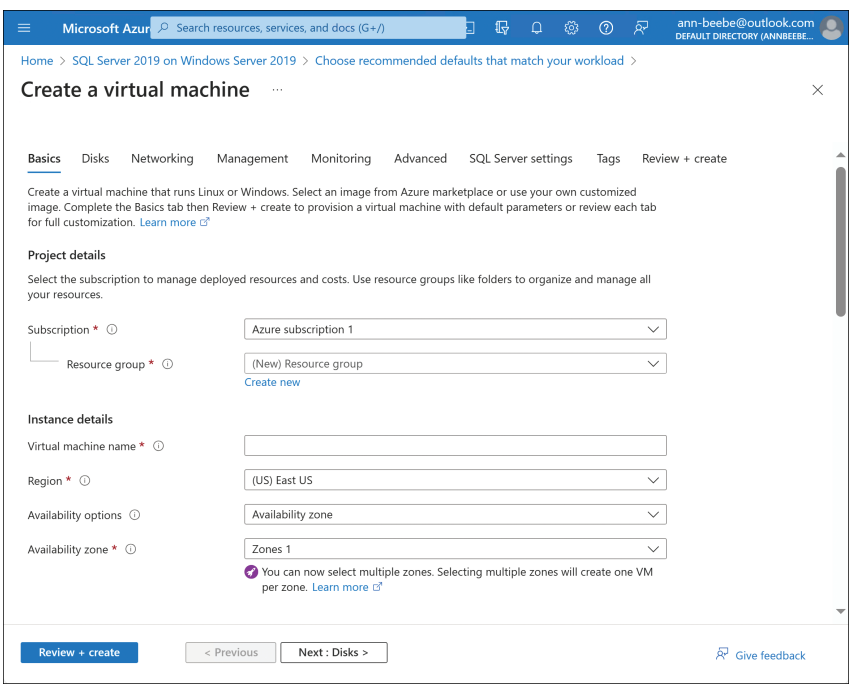


FIGURE 1-6 The Basics tab of the Create a virtual machine interface in the Microsoft Azure portal

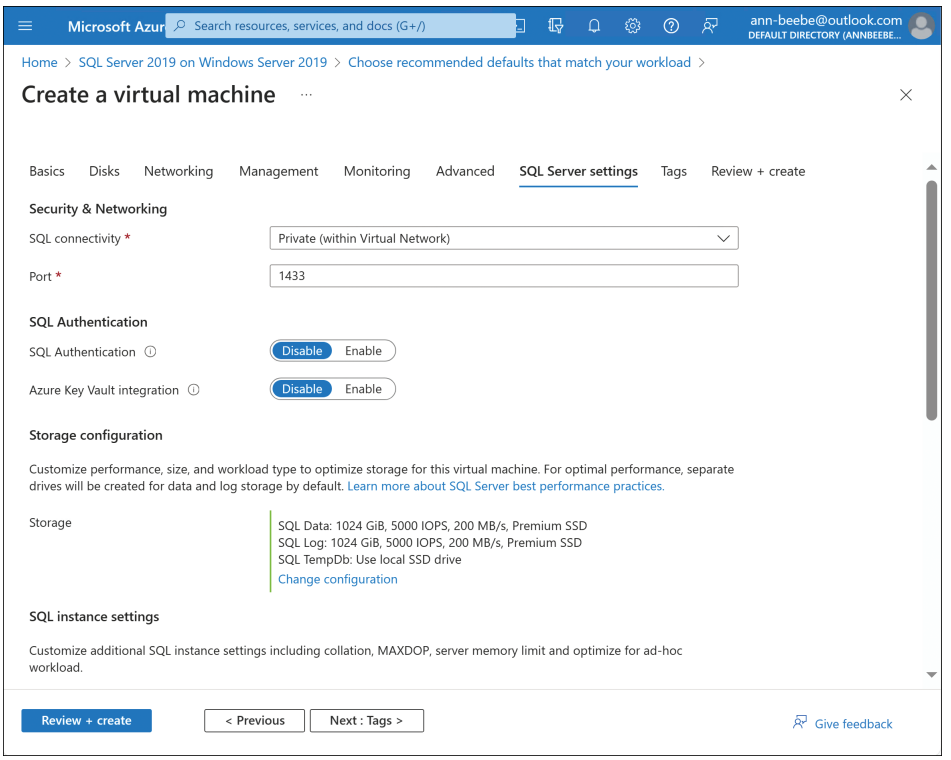


FIGURE 1-7 The SQL Server settings tab of the Create a virtual machine interface in the Microsoft Azure portal

The configuration parameters on the SQL Server Settings tab include the following:

- **SQL connectivity** Specifies whether SQL Server will be accessed locally (inside the VM), privately (inside the virtual network), or publicly (on the Internet)
- **Port** Specifies the port number of the SQL service with a default value of 1433
- **SQL Authentication** Enables SQL Server's internal authentication
- **Azure Key Vault Integration** Provides additional security for authentication keys
- **Storage configuration** Specifies the size and performance settings for SQL Server's Data, Log, and TempDb storage
- **SQL instance settings** Specifies instance level settings, including collation rules, processor utilization settings, and memory limitations
- **SQL Server License** Specifies whether the subscriber already possesses a SQL Server license
- **Automated patching** Specifies a time window during which Windows and SQL updates can be applied
- **Automated backup** Enables automated backups of the SQL databases
- **R Services (Advanced Analytics)** Enables SQL Server Machine Learning Services, providing the ability to use advanced analytics on the SQL Server

After configuring all of the required settings and any others they might need on the other tabs, the subscriber can click the Review + create button. The Review + create tab lists all of the settings that Azure will use to create the VM. Azure also validates the settings to ensure that all the required parameters have been configured appropriately, displaying warnings for any settings that might be potentially problematic.

For example, in the Review + create tab shown in Figure 1-8, the settings for the proposed VM have passed the validation test, indicating that the essential parameters have been configured correctly. However, Azure still displays a warning that the VM, as configured, will have the Remote Desktop Protocol (RDP) port 3389 left open to the Internet, which can be a security hazard. The subscriber can still make configuration changes in response to any warnings before Azure actually creates the VM.

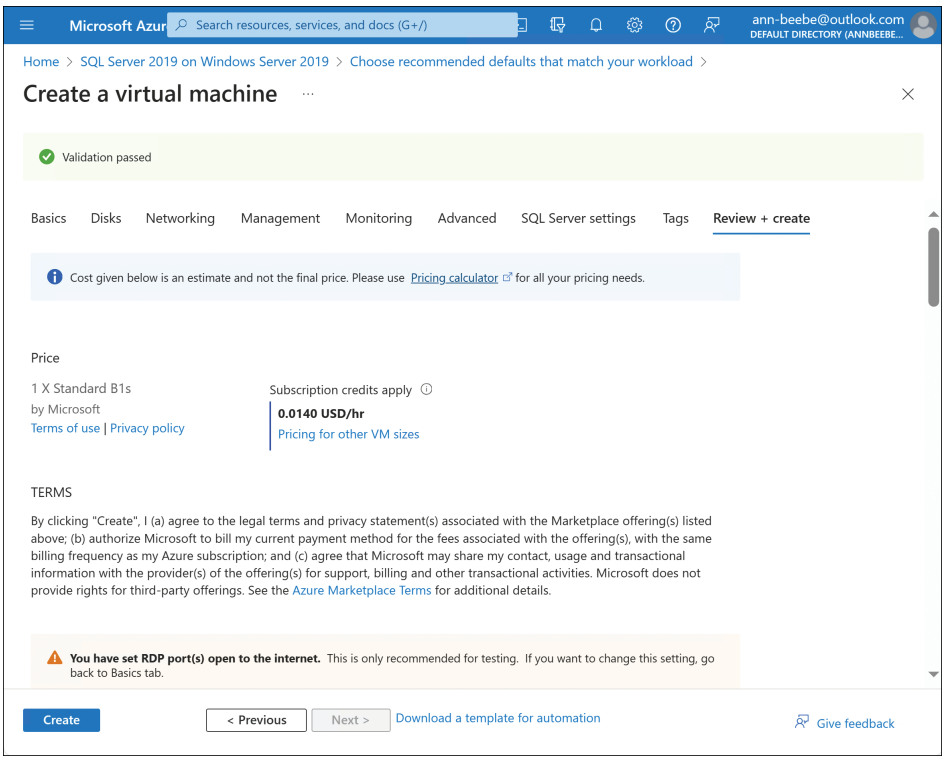


FIGURE 1-8 The Review + create tab of the Create a virtual machine interface in the Microsoft Azure portal

USING THE SQL SERVER IAAS AGENT EXTENSION

The SQL Server implementation that an IaaS subscriber runs on a virtual machine can be the same one they run on their on-premises servers. Administrators can manage SQL Server directly on the VM, but it is also possible to integrate SQL Server administration functions into the Azure portal by installing the SQL Server IaaS Agent Extension, a free component that,

in addition to integrating SQL Server management functions into the Azure portal, provides additional benefits, including the following:

When the SQL Server IaaS Agent Extension is first registered, Azure copies the installation files for all of the agent's features to the virtual machine, but the agent is not installed until the subscriber enables a SQL IaaS Agent extension that calls for it. The basic registration functionality provides the following features:

- **Portal-based management** Enables management of all the subscriber's SQL Server VMs through the Azure portal.
- **Flexible licensing** Enables the subscriber to switch the VM's licensing between the Bring Your Own License model using the Azure Hybrid Benefit and the pay-as-you-go model.
- **Flexible version or edition** Subscribers who change the edition or version of the SQL Server installation on a VM should reregister the agent and can modify the version and edition properties in the VM metadata using the Azure portal, the Azure CLI, or PowerShell.

The SQL Server IaaS Agent Extension includes many other features, but it's not until the subscriber enables these features that Azure installs the necessary files on the VM. These features include the following:

- **Automated backup** Configures SQL Server Managed Backup to Microsoft Azure to back up a single (default or named) instance using Azure Blob storage.
- **Automated patching** Creates a maintenance window during which automated updates can be applied to the OS and SQL Server on the VM. Automated patching can't install cumulative updates for SQL Server, however.
- **Azure Key Vault integration** Azure Key Vault is a highly secure service for the storage of encryption keys. The agent enables SQL Server to use the vault to store its various encryption keys, such as those for transparent data encryption (TDE), column level encryption (CLE), and backup encryption.
- **Azure Update Manager integration** Azure Update Manager is a service (in preview, as of this writing) designed to manage all updates for virtual machines and SQL Server instances. Unlike the automated patching feature, Azure Update Manager can install cumulative updates for SQL Server.

NOTE **AZURE UPDATE MANAGER AND AUTOMATED PATCHING**

Azure Update Manager and automated patching are both features that can automate the process of applying operating system and SQL Server updates, but the two are not compatible with each other. Subscribers should enable only one of these services or risk unintended conflicts.

- **Temporary storage configuration** Subscribers can modify the configuration of SQL Server's temporary storage (tempdb) by specifying the number of files, their initial size, their location, and their autogrowth size.

- **View disk utilization** Provides a graphical display of the SQL data files' disk utilization in the Azure portal
- **Defender for Cloud portal integration** Subscribers enabling database protection in the Microsoft Defender for Cloud configuration settings can monitor their SQL Server status and view security recommendations using the Azure portal.
- **SQL best practices assessment** Once the best practices feature is enabled, the SQL VM Management page in the Azure portal includes assessments of the SQL Servers running on VMs and recommendations for things like indexes, deprecated features, enabled or missing trace flags, and statistics. The Assessment results section of the page lists all of the recent assessment runs, enabling subscribers to compare the results from different times.

Selecting an Azure SQL Deployment Option

Microsoft Azure provides several ways for subscribers to create new SQL databases. Clicking the Azure SQL icon on Azure's Home page opens the Azure SQL page, as shown in Figure 1-9. This page lists all of the subscriber's existing Azure SQL databases, if any.

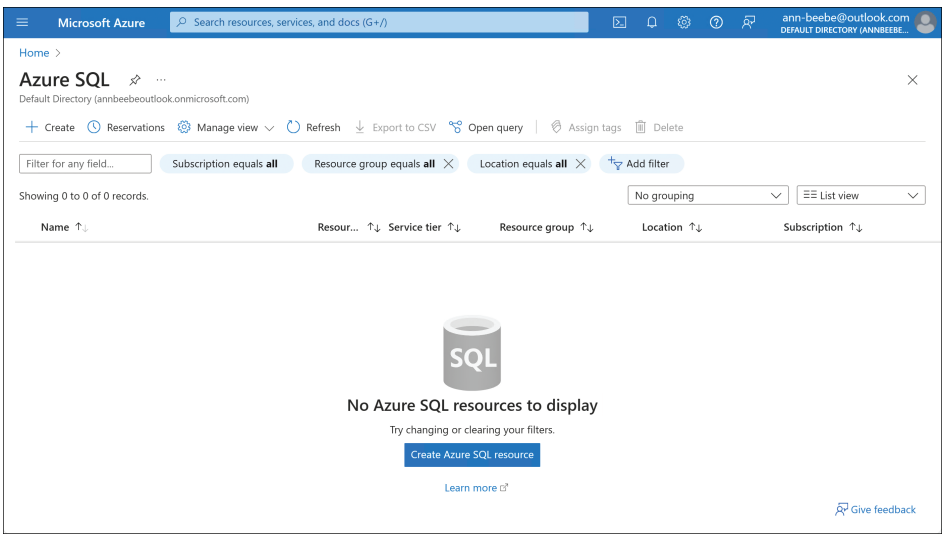


FIGURE 1-9 The SQL databases page in the Microsoft Azure portal

Clicking the +Create button at the top of the Azure SQL page, or the Create Azure SQL resource button at the bottom of the page if there are no Azure SQL resources deployed, opens a Select SQL deployment option page, as shown in Figure 1-10. This page enables subscribers to create a new Azure SQL installation using any one of the three SQL database options: SQL databases, SQL managed instances, or SQL virtual machines.

Each of the Azure SQL options on the page has a dropdown that enables the subscriber to configure the basic installation by selecting, for example, an elastic pool, an Azure Arc instance, or a virtual machine image. Then, clicking the Create button initiates the creation of the selected SQL resource.

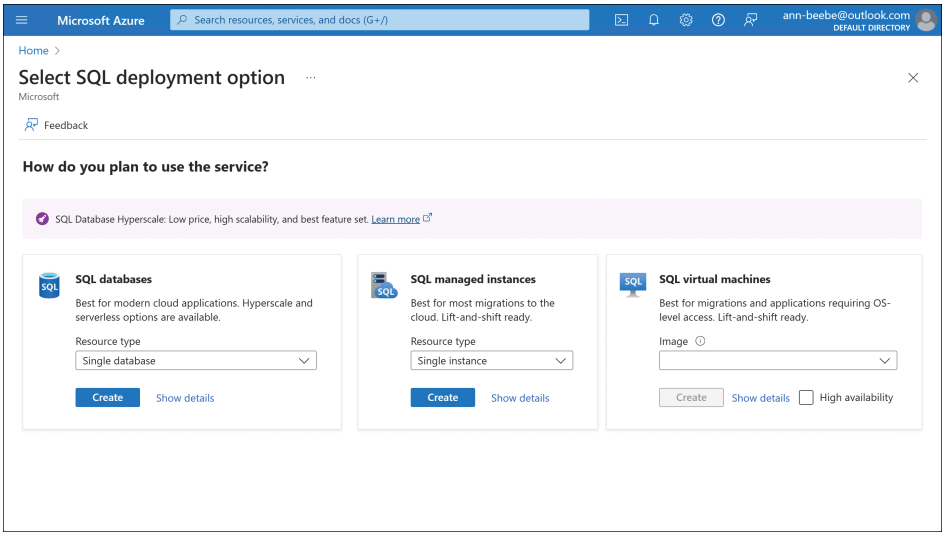


FIGURE 1-10 The Select SQL deployment option page in the Microsoft Azure portal

Creating an Azure SQL Database

On the Select SQL deployment option page, clicking Create for the SQL databases option opens the Create SQL Database tabbed dialog. This same dialog is also accessible from the Azure portal home page by selecting SQL Databases and clicking +Create.

The Create SQL Database dialog opens to the Basics tab shown in Figure 1-11.

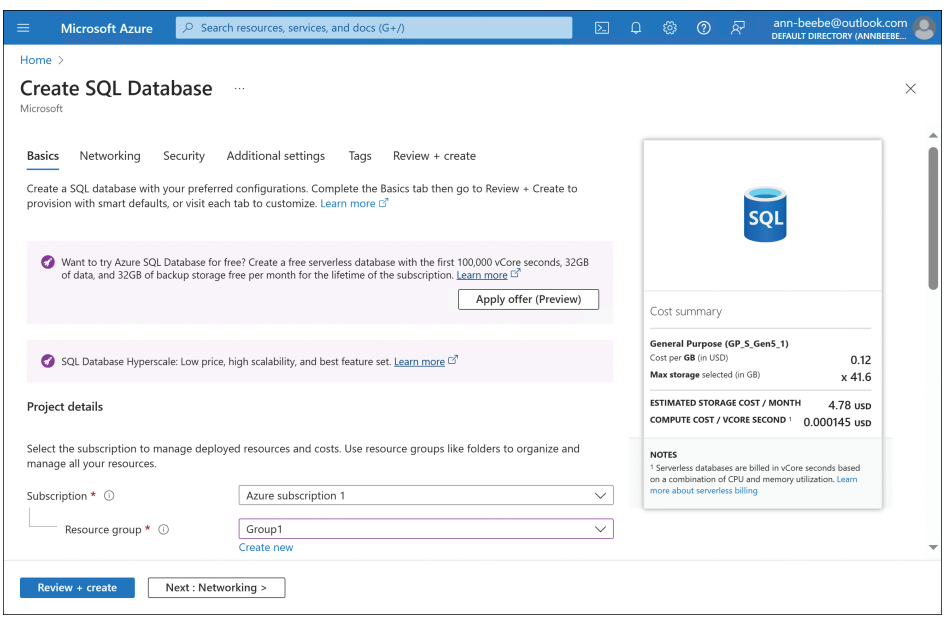


FIGURE 1-11 The Select SQL deployment option page in the Microsoft Azure portal

While the dialog provides access to many configuration parameters, there are relatively few settings that the subscriber must configure before creating the database, including the following:

- **Subscription** Specifies the Azure subscription that will host and be billed for the SQL database.
- **Resource group** Specifies the name of a container for Azure associated resources that share the same permissions and policies as the database.
- **Database name** Specifies a name for the SQL database.
- **Server** Selects or creates a logical server that hosts the SQL database. The Create SQL Database Server page, shown in Figure 1-12, specifies a unique name for the server, which is added to the database.windows.net domain.
- **Want to use SQL elastic pool?** Enables subscribers to create a pool of storage resources and eDTUs that multiple databases can share.
- **Compute + storage** Specifies the service tier (from the DTU and vCore options), the Compute tier (Provisioned or Serverless), and the virtual hardware configuration settings for the database server. By default, the new database receives a general purpose serverless vCore configuration with up to two vCores available, as shown in Figure 1-13.
- **Backup storage redundancy** Specifies whether the database backups should use locally redundant, zone-redundant, or geo-redundant storage.

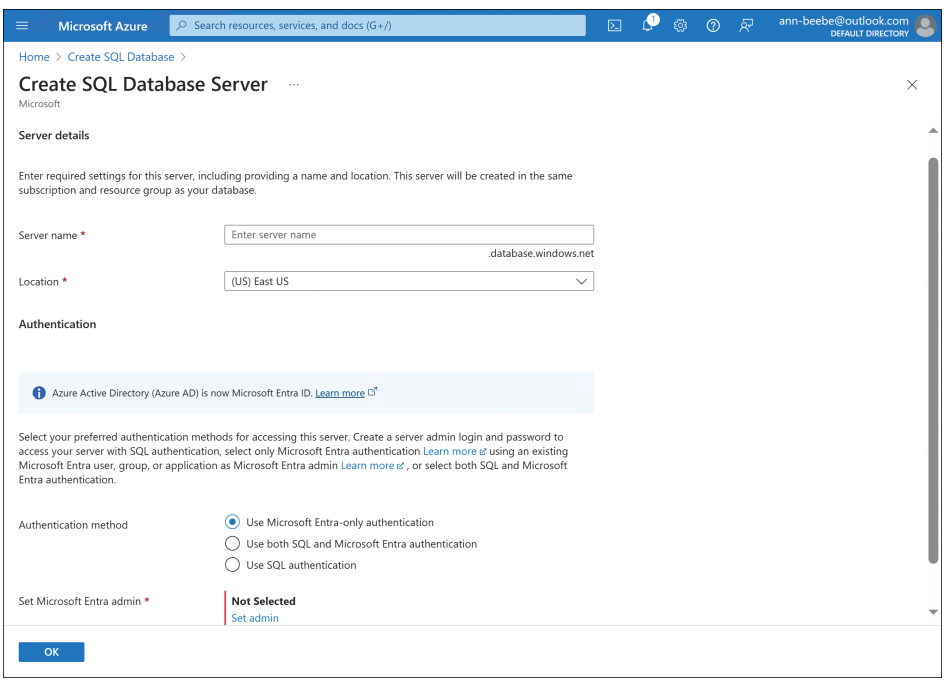


FIGURE 1-12 The Create SQL Database Server page in the Microsoft Azure portal



EXAM TIP

DP-300 exam candidates must be conscious of the differences between an actual server installed on an IaaS VM and the logical database server created during an Azure SQL Database deployment. The Server setting in the Create SQL database dialog creates a logical server to host the SQL database and control access to it; the setting does not create a subscriber-managed VM. The only options to configure when creating a logical server are its unique name, its regional location, the authentication method it will use, and the identity of the database administrator and/or the Microsoft Entra administrator.

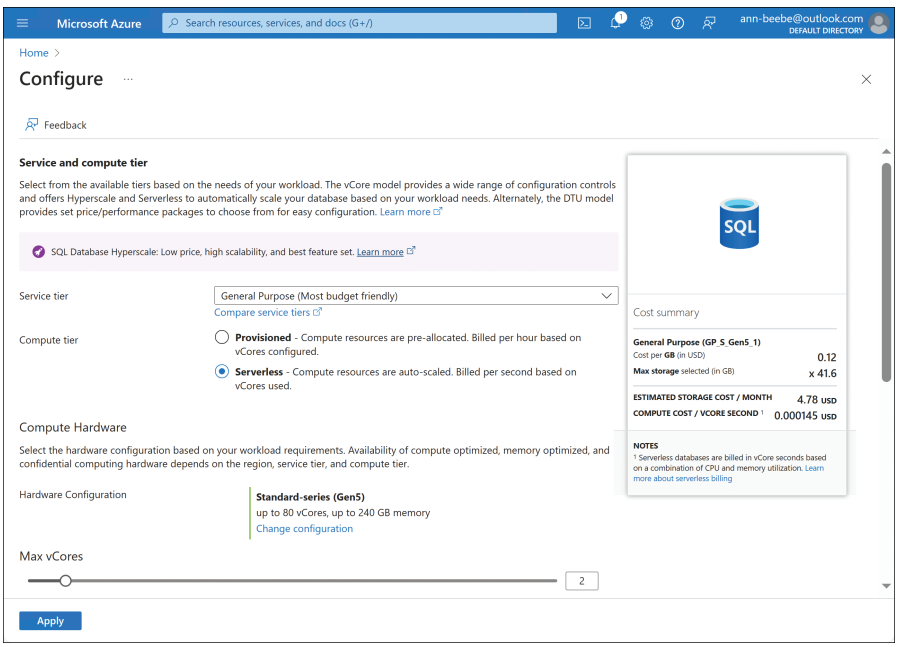


FIGURE 1-13 The Configure page for the service and compute tiers in the Microsoft Azure portal

In addition to the Basics tab, the other tabs in the Create SQL Database dialog enable the subscriber to modify parameters in the following areas:

- **Networking** Specifies network connectivity settings, connection policies for communication with the database server, and encryption options
- **Security** Specifies encryption settings for database protection, including a free trial of Microsoft Defender for SQL
- **Additional settings** Specifies a maintenance window and how to populate the new database
- **Tags** Enables subscribers to create tags to identify specific resources for billing purposes
- **Review + create** Summarizes all of the database creation and configuration settings, as well as the currently configured cost for the database

Creating an Azure SQL Managed Instance

When deploying SQL Server on-premises, the user creates a server (physical or virtual) and installs the SQL Server product, which creates a SQL instance. Within that instance, it's possible to create multiple databases. When an Azure subscriber creates a new SQL Database installation, they get direct access only to the database, not the instance in which it's located. To obtain full access to a SQL instance in Azure, a subscriber must create an Azure SQL Managed Instance installation.

An Azure SQL Managed Instance is the closest thing to an on-premises SQL Server installation that is available in Microsoft Azure. For subscribers seeking to migrate their on-premises SQL databases to the cloud, a Managed Instance provides nearly 100 percent compatibility with the on-premises SQL Server product.

As with the Azure SQL Database product, a SQL Managed Instance is automatically patched, updated, backed up, and made highly available as part of the PaaS platform. A Managed Instance installation also enables subscribers to access instance-scoped features such as Service Broker and SQL Server Agent that are not accessible in a SQL Database installation.

The process of creating an Azure SQL Managed Instance installation is similar to that of creating a SQL Database. Selecting the Create button in the SQL managed instances box on the Select SQL deployment option page opens the Create Azure SQL Managed Instance dialog, as shown in Figure 1-14.

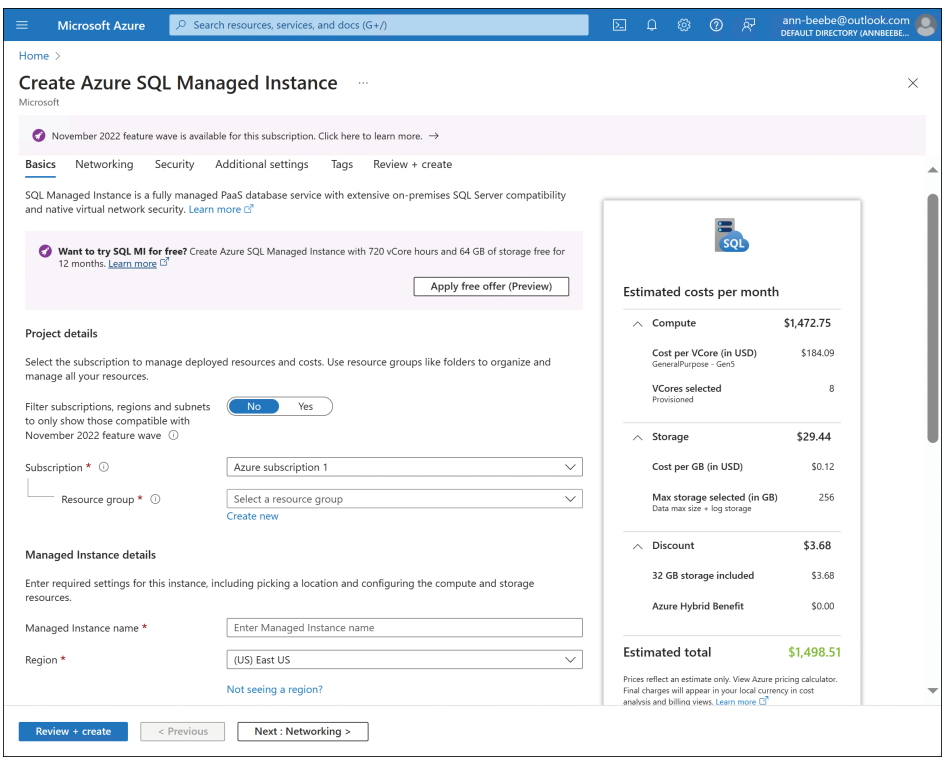


FIGURE 1-14 The Basics tab in the Create Azure SQL Managed Instance dialog in the Microsoft Azure portal

On the Networking tab, shown Figure 1-15, the subscriber creates a virtual network/subnet for the managed instance, as opposed to the virtual server creation in a SQL Database installation. It is also possible to create a public endpoint that provides access to the managed instance from the Internet and to specify the minimum TLS version required for the encryption of inbound database connections. Virtually all of the other settings in the dialog are the same as those for a SQL Database installation.

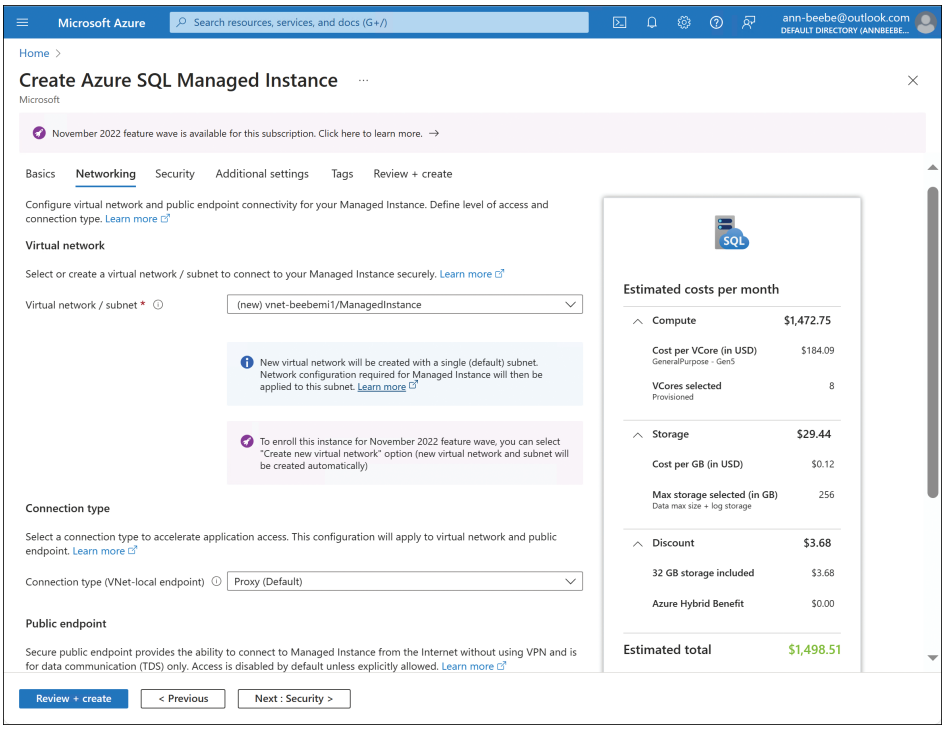


FIGURE 1-15 The Networking tab of the Create Azure SQL Managed Instance dialog in the Microsoft Azure portal

Understand automated deployment

The Azure portal provides subscribers with several paths to the dialogs for creating SQL Databases, SQL Managed Instances, and virtual machines. These deployment processes are relatively simple and quick, but they can create only one SQL installation at a time. The deployments are also essentially manual; subscribers must configure the settings for each database, instance, or VM individually.

In an environment with many SQL databases, instances, and servers, creating new ones individually on a regular basis can be time-consuming. In addition, manual deployments are subject to user errors that can lead to inconsistent configuration settings. In the case of a migration to the cloud, it might be necessary to create many new Azure SQL installations in

rapid succession. To address these issues, Azure provides other deployment methods that can help subscribers to automate the process of creating SQL installations in the cloud and ensure that all of the deployments are consistent in their settings. These methods include Azure Resource Manager templates, the Azure CLI, and PowerShell commands.

Azure Resource Manager

Obviously, Azure must authenticate and authorize subscribers before it allows them to create and deploy the components of a new SQL installation. *Azure Resource Manager (ARM)* is the service that performs those authentications and authorizations, whichever deployment method a subscriber elects to use. Thus, whatever means a subscriber uses to create a new SQL Database, whether it be a portal dialog, an ARM template, a CLI or PowerShell script, or a representational state transfer (REST) API, the request goes through ARM, which evaluates whether the subscriber has the permissions needed to deploy the requested compute, storage, and networking components.

In addition to authentication and authorization, ARM is also responsible for grouping resources and maintaining the dependencies between them so that the resources are always deployed consistently and in the correct order, a process called *orchestration*. Because of these dependencies, ARM is able to use a *declarative model* for deploying resources. In the declarative model, a template specifies the resources to be installed, as shown in Figure 1-16, and ARM is responsible for orchestrating their deployment. The alternative to the declarative model is the *imperative model*, which defines a set of tasks that must be executed in order, as in a script or batch file.

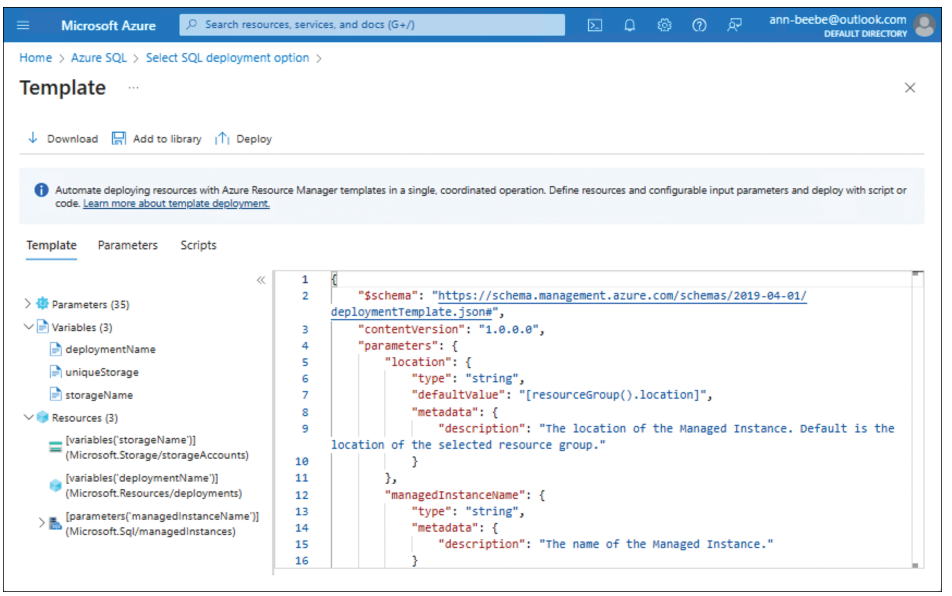


FIGURE 1-16 The Template page in the Microsoft Azure portal

Apply patches and updates for hybrid and infrastructure as a service (IaaS) deployment

The Azure SQL PaaS products are all automatically patched and updated; they require no administration from the subscriber in that respect. However, subscribers running SQL Server on an Azure virtual machine in an IaaS deployment must apply patches and updates themselves or automate the process. In the case of a hybrid deployment, administrators must continue to patch and update their on-premises SQL servers, while making sure that the SQL Server and operating system versions on-premises are synchronized with those in the cloud.

Azure VMs support Automated patching for both operating system and SQL Server updates, but the subscriber must enable this feature in the VM configuration, either during VM's initial deployment or afterward. In the Create a virtual machine dialog, the SQL Server settings page provides access to the SQL Automated Patching interface, as shown in Figure 1-17.

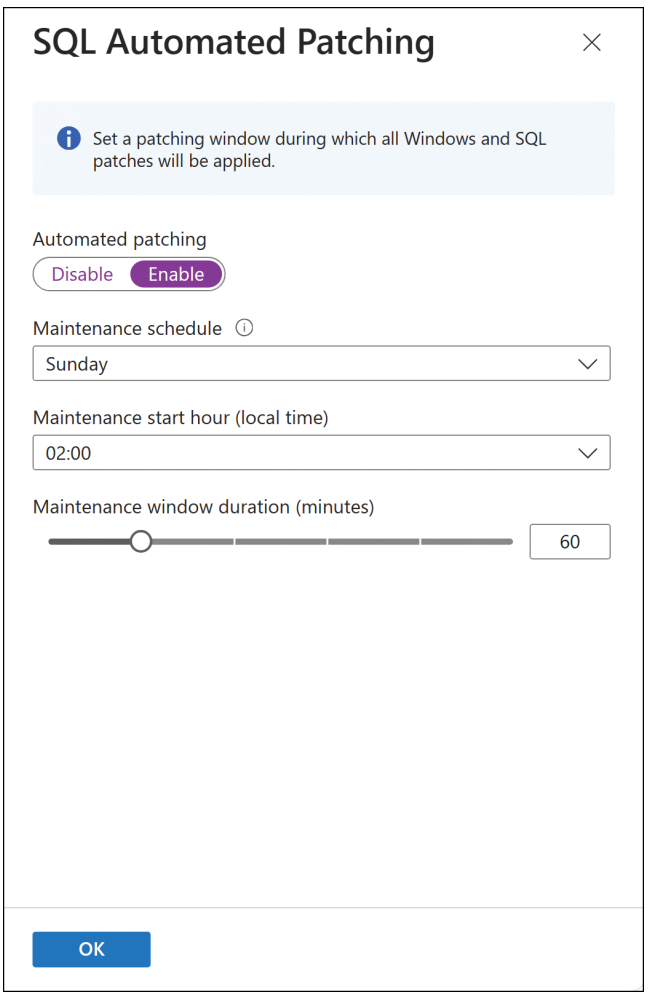


FIGURE 1-17 The SQL Automated Patching page in the Create a virtual machine dialog

Enabling Automated patching on a SQL VM requires that the virtual machine have the SQL Server IaaS Agent installed. The controls allow the subscriber to select a maintenance window, which is the only time of day during which Azure can install patches and updates.

Subscribers can also activate Automatic patching for an existing VM from the Patching page of the virtual machine's configuration window in the Azure portal. The controls for configuring the maintenance window are the same as those in the Create a virtual machine dialog.

It's also possible to configure Automated patching from PowerShell. To do this, use the *New-AzVMSqlServerAutoPatchingConfig* cmdlet to specify the maintenance window and the *Set-AzVMSqlServerExtension* cmdlet to apply the maintenance window to a virtual machine.

Deploy hybrid SQL Server solutions

As noted earlier, a hybrid SQL Server solution bridges the gap between an on-premises SQL Server installation and the Azure SQL products available in the cloud. The Azure SQL products are nearly 100 percent compatible with the current on-premises SQL Server products. Because the products can interact, there are various scenarios in which administrators might want to expand their existing on-premises SQL Server installations into the cloud.

An on-premises SQL Server installation can consist of physical servers, local virtual machines, or a combination of both. Adding SQL resources in Azure can be the opening steps in a full migration from on-premises servers to the cloud (as covered later in this chapter). However, an expansion into the cloud can also be a permanent solution that provides the on-premises SQL installation with additional fault tolerance, high availability, backups, and/or bandwidth on demand.

Adding bandwidth

Expanding an on-premises network can be an expensive proposition. Adding virtual machines to existing host servers is easy, but there will eventually need to be additional hardware installed to achieve any substantive expansion. There is also the issue of whether the need for additional bandwidth is a permanent one that's worth the investment in additional server hardware.

Expanding a SQL installation by adding Azure cloud resources requires no hardware investment. Azure subscribers can add as many SQL resources as they need. The most common hybrid scenario is the addition of Azure VMs in the cloud running the same SQL Server version as the on-premises servers. Once the Azure elements are in place, the cloud side of the installation is completely flexible.

For example, if a business regularly experiences a predictable busy season, a hybrid scenario, once established, can allow the subscriber to add more VMs during that time or upgrade the virtual hardware in the existing VMs. When the busy season ends, the subscriber can scale the cloud resources back.

Adding hybrid connectivity

For a hybrid SQL installation to function properly, the connection between the on-premises datacenter and Azure must be reliable and secure. This connection is typically either a site-to-site virtual private network (VPN) link or an ExpressRoute tunnel.

The VPN option is secure and less expensive, but it's also reliant on the datacenter's Internet connection. By contrast, an ExpressRoute tunnel is a dedicated circuit connecting the subscriber's site to a Microsoft datacenter. Apart from the difference in expense, Azure subscribers should also consider the latency requirements of their SQL applications.

Adding fault tolerance

An on-premises SQL environment can have local fault tolerance in the form of multiple servers—either physical or virtual—that can provide a failover in the event of a server or drive failure. Larger organizations might even maintain multiple datacenters in the same general area to provide failovers if a fire or other localized disaster should affect an entire datacenter.

However, in the event of a major disaster that threatens an entire region, such as an earthquake or a war, the entire SQL installation could be a total loss unless there are failover servers located in other regions. The cost of building and maintaining a redundant datacenter in another region or country or continent might be too much even for a large organization. However, an Azure hybrid infrastructure makes it possible to deploy failover SQL VMs in multiple regions around the globe easily and inexpensively, as shown in Figure 1-18.

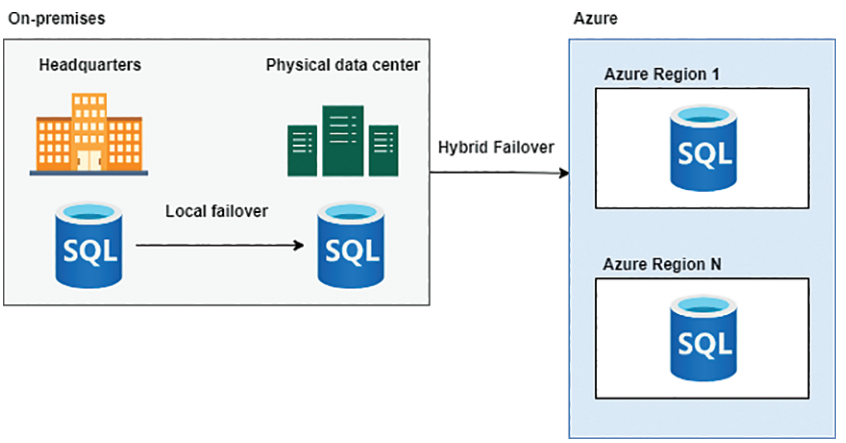


FIGURE 1-18 SQL hybrid infrastructure with regional failovers

Adding off-site backups

In addition to using a hybrid SQL infrastructure for fault tolerance and disaster recovery, subscribers can also perform offsite backups to Azure. On-premises datacenters often have a local backup mechanism, and the Azure Backup service supports SQL Server on VMs.

However, it's also possible to back up on-premises SQL data directly to Azure Storage, using a URL or an Azure SMB file share. This way, if the on-premises backup fails, the SQL data is also accessible from the cloud, and the subscriber can even import the data into a new Azure SQL installation, if necessary.

Using Azure Arc

In a hybrid SQL Server deployment, the databases in the on-premises servers and those in the Azure cloud might be identical, but the servers are administered separately by default. Administrators familiar with SQL Server might be comfortable working with the on-premises servers but experience a learning curve with Azure administration. The opposite might also be true.

Azure Arc is a Microsoft product that addresses this issue by enabling on-premises servers to extend into the Azure environment. Azure administrators can therefore monitor and maintain the on-premises servers using the Azure administrative interface, as shown in Figure 1-19.

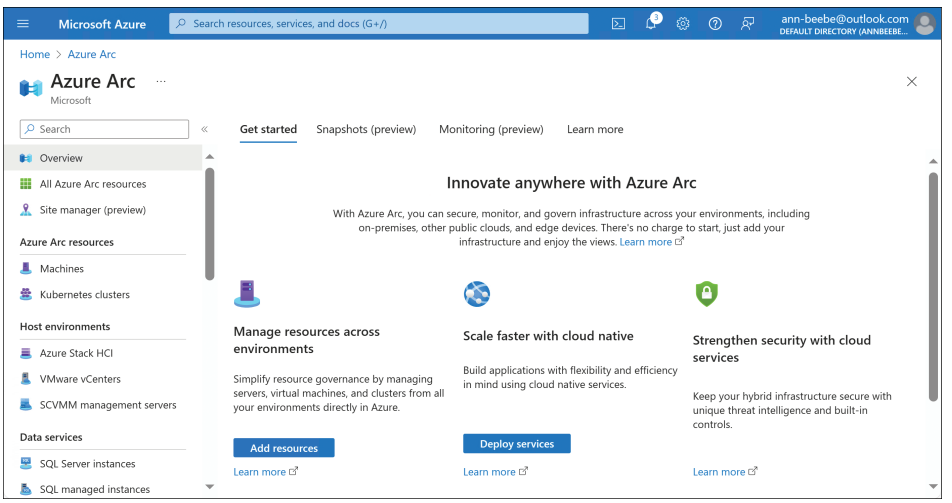


FIGURE 1-19 The Azure Arc page in the Azure portal

Recommend an appropriate database offering based on specific requirements

As noted earlier, Azure offers a variety of SQL products, and subscribers might be confused by the multiplicity of cloud infrastructures, purchasing models, and service tiers. For a basic database, any of the options would be acceptable, but selecting a database offering is always a tradeoff between performance and price.

IaaS vs. PaaS

What the choice between IaaS and PaaS means to the potential cloud service subscriber is that the IaaS model provides the physical computing elements, including the network, the storage subsystem, the physical servers, and the hypervisor used to create virtual machines on the servers. The subscriber contracts to lease a virtual machine from the cloud service provider, on which they can install a server operating system and any applications they need, including a SQL Server implementation.

In the IaaS model, the subscriber has full administrative control over the operating system installed on the leased virtual machine (VM), as well as any installed applications. This provides the subscriber with the freedom to configure the OS and applications as though they were running on a physical computer, but it also requires the subscriber to maintain the software by installing required patches and updates as needed.

The alternative to the IaaS model is Platform as a Service (PaaS), and for cloud service subscribers, this means far less administrative overhead than an IaaS solution. In Microsoft's PaaS SQL offerings, subscribers are leasing a SQL database or an entire SQL instance and only that. The PaaS subscriber does not have direct access to the virtual machine hosting the SQL databases or any of the infrastructure elements below that. Some administrators might see this as a drawback, but lack of access also means lack of responsibility for the underlying infrastructure; Azure takes care of that. The subscriber can, however, configure and manage their SQL database or instance.

PaaS SQL options

The IaaS option for deploying SQL Server in the cloud is intended for specific types of workloads and licensing considerations. The PaaS SQL Server options provide far simpler deployment processes and eliminate the need for the subscriber to maintain and configure the virtual machines hosting the SQL databases.

The PaaS deployment options for SQL database implementations are as follows:

- **Azure SQL Database** A single virtualized SQL database with low maintenance, great flexibility and scalability, and a granular deployment process
- **Azure SQL Managed Instance** A fully managed, virtualized SQL Server instance suitable for migration from on-premises SQL Server installations
- **Azure SQL Edge** A database specifically designed for edge deployments and Internet of Things (IoT) devices

For subscribers with relatively simple needs, such as a single database used relatively infrequently, Azure SQL Database can be an ideal solution. However, subscribers who want to duplicate the on-premises SQL Server experience in the cloud—or migrate from on-premises servers to the cloud—would likely be better served by Azure SQL Managed Instance.

Selecting the appropriate Azure SQL offering for a particular application can be complicated, but one of the best things about using Azure SQL databases is that, because the infrastructure is virtual, subscribers can change it at will. For example, subscribers can scale out an installation by creating additional VMs and/or databases, or they can scale up by adding storage, upgrading the compute resources, or changing the service tiers of existing installations. It's even possible to migrate from a SQL Database to a SQL Managed Instance, should the need arise. All of these are simple tasks that subscribers can perform in the Azure portal or from an Azure CLI or PowerShell command prompt.

PaaS purchasing models

The PaaS SQL options in Azure support two basic purchasing models: one based on data transaction units (DTUs) and another called vCore.

DTU

For Azure SQL Database subscribers, the DTU-based purchasing model provides a relatively simple option with preconfigured service tiers and virtual hardware resource options. A database transaction unit (DTU) is a blended measurement of CPU cycles, memory resources, I/O reads, and I/O writes that are allocated to a single SQL database as a unit. Subscribers can also create elastic resource pools that are shared by multiple databases; these pools use elastic data transaction units (eDTUs).

When selecting the DTU purchasing model for a SQL Database, the subscriber chooses from three service tiers: basic, standard, or premium. The tiers provide varying compute levels, based on an adjustable number of DTUs, an adjustable maximum database size, and increasing periods of backup retention. When creating a new Azure SQL Database, a subscriber selecting a DTU service tier sees an interface like the one shown in Figure 1-20, with sliders to select the number of DTUs and the maximum database size. The Cost summary box on the right self-adjusts to reflect the price per DTU of the SQL Database installation as currently configured.

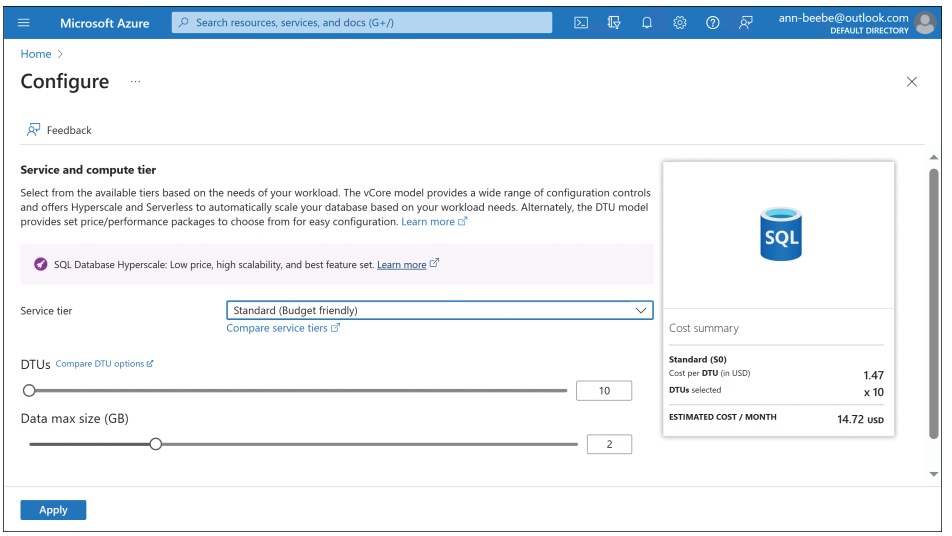


FIGURE 1-20 The Configure interface for a Standard DTU installation in the Create SQL Database process

VCORE

Supported by both Azure SQL Database and Azure SQL Managed Instance, the virtual core (vCore) purchasing model provides subscribers with added flexibility and scalability by allowing them to scale the compute, memory, and storage resources allocated to the SQL product independently. The vCore model also provides higher limits for virtual hardware resources and a wider variety of hardware configurations to suit various workloads.

A vCore (or virtual core) is a logical representation of a particular CPU, in combination with memory and storage limitations. Subscribers can configure a SQL Database or SQL Managed Instance with as many vCores as their workload requires and change the vCore allocation at any time to suit their needs.

Like the DTU model, the vCore purchasing model provides subscribers with three service tiers, but the tiers are different ones, as follows:

- **General purpose** Provides virtual hardware configurations to support typical workloads at budget prices
- **Business critical** Provides higher I/O performance with local SSD storage and additional fault tolerance and high availability with multiple secondary replicas, including a free read-only secondary replica
- **Hyperscale** Provides support for workloads requiring highly scalable and independently scalable compute and storage resources with much higher maximum limits than the other tiers, as well as subscriber scalable fault tolerance and high availability options

The pricing for a vCore installation is a combination of the selected service tier, the virtual hardware configuration, the number of vCores, the amount of memory, the reserved database storage, and the actual backup storage in use. When configuring the Compute + Storage options for a new SQL Database installation, as shown in Figure 1-21, the subscriber selects a service tier and then one of the following Compute tier options:

- **Provisioned** Azure allocates the selected compute capacity to the database continuously, regardless of the database's workload or activity level. Provisioned databases incur an hourly compute charge based on the selected hardware options and the number of vCores selected.
- **Serverless** Azure automatically allocates compute capacity to the database as needed, based on its activity and workload. When there is no activity, Azure pauses the database, and there are no compute charges until the next connection attempt (although storage charges continue). Serverless databases incur compute charges on a per-second basis for each vCore in use. Subscribers select the minimum and maximum number of vCores Azure can allocate to the database.

The box on the right of the page dynamically reflects the price of the SQL Database based on the subscriber's selections. Provisioned installations display an estimated cost per month, while serverless installations display a storage cost per month and a compute cost per vCore second.

A serverless database can be more economical than the provisioned alternative because compute capacity is only allocated as needed, but it's not suitable for all applications. In some cases, an application might require the database to be running at all times. Applications should also include retry logic that enables them to cope with failed connection attempts to a paused database.

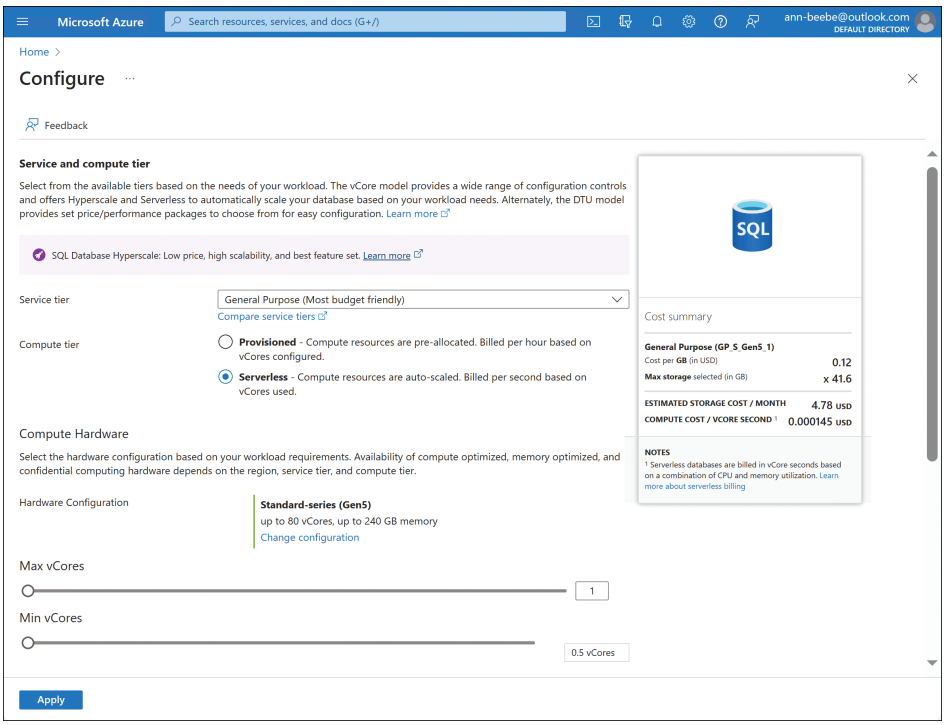


FIGURE 1-21 The Configure interface for a General Purpose vCore installation in the Create SQL Database process



EXAM TIP

Exam candidates should be aware that the term "serverless" is a misnomer in this case because the SQL database clearly must be running on a server somewhere. The term actually refers to the fact that the variable compute levels possible in a serverless SQL Database installation mean that the database is not permanently associated with a particular server but may be moved to different servers as needed to suit the workload.

Evaluate the security aspects of the possible database offering

Security for the various Azure SQL offerings depends largely on the scope of the installation and how much of the underlying infrastructure that scope makes accessible to the subscriber. For example, Azure SQL Database subscribers have virtually no access to the underlying server and operating system, so their security concerns are centered solely on the database itself. By contrast, Azure SQL Managed Instance provides the subscriber with some server access, while an Azure VM with SQL Server installed provides full administrative access to the VM and operating system. Both of these products require additional security maintenance and monitoring from the subscriber.

Some of the security concerns that subscribers should consider when evaluating the Azure SQL products include the following:

- **Auditing** Auditing enables administrators to track the occurrence of specific activities and whether they're successfully completed. All three of the Azure SQL products support auditing—Azure SQL MI and Azure VMs at the server level and Azure SQL Database at the database level (because there's no server access). SQL DB and SQL MI both store the audit log files in Azure Blob storage, whereas the operating system on an Azure VM is responsible for storing the audit logs.
- **Protection** All Azure SQL databases, elastic pools, and managed instances can add security through Microsoft Defender for SQL, which includes SQL Advanced Threat Protection and vulnerability assessment. The additional cost (as of this writing) for Azure-connected databases is $0.021 per instance per hour. For SQL databases outside of Azure, the cost is $0.015 per instance per vCore per hour. In addition, Microsoft Defender for Cloud (formerly known as Azure Security Center) can provide more generalized protection for all cloud-based applications.
- **Encryption** Azure supports three types of data encryption: Transport Layer Security (TLS) for data-in-motion, Transparent Data Encryption (TDS) for data-at-rest, and Always Encrypted, which is designed to protect specific data stored in SQL databases, such as credit card and Social Security numbers, and not reveal it to the database administrators.
- **Authentication** SQL Server includes its own username/password-based authentication mechanism that subscribers can use with any of the Azure SQL products. For SQL DB and SQL MI, subscribers can also use Azure Active Directory authentication. For Azure VMs running Windows, subscribers can also use Windows authentication.

Recommend a table partitioning solution

As the tables in databases grow, they can conceivably get so large that they test the limits of their environment, often causing query performance to be negatively affected. A table can grow to exceed its contracted storage limits, or overwhelm its compute capacity, or require more bandwidth than the network can provide. There are two basic ways to address this issue, by scaling up the server hardware or by scaling up the data itself.

In hardware scaling—also known as scaling up, as shown in Figure 1-22—subscribers can add storage, compute, and network resources to a database, managed instance, or VM to accommodate the table's increased size. Azure makes scaling up a simple process, and its SQL products have options offering extensive upgrade paths. This is a good thing because the subscriber will likely be repeating the process as the table continues to grow. For example, the storage limit for most of the service tiers in the Azure SQL products is 4 terabytes (TB). If a table grows to exceed that size, then the only way to scale farther up is the Hyperscale service tier, which supports up to 100 TB of storage.

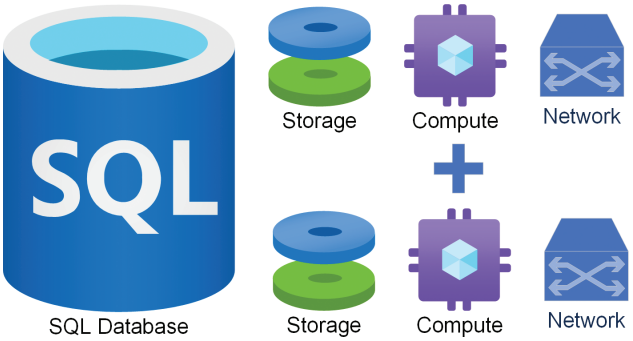


FIGURE 1-22 Vertical scaling is the addition of storage, compute, and/or network resources to an existing installation to enhance its performance.

Hardware scaling has its limits, and it's usually a temporary solution at best. The other option is vertical partitioning, in which the large table is divided into separate partitions using a key column value. Based on that key column value, SQL divides the table into subsets of rows and creates a partition for each subset. The partitions are stored elsewhere in the same database instance, and the index for the entire table directs queries to the proper partitions, as shown in Figure 1-23.

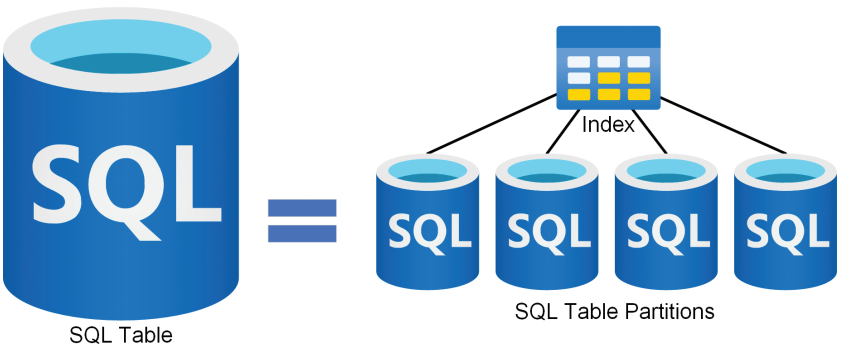


FIGURE 1-23 Vertical partitioning in Azure SQL is the division of a table into multiple partitions stored in the same database instance.

How a subscriber elects to partition a table is usually based on the nature of the data it contains. The typical process is to choose one column from the table and use the contents of that column to create partitions. One of the most common methods is to choose a column containing dates and use it to create separate partitions for specific days, months, or years. However, it's also possible to partition a table based on products, departments, or any other criteria. Once the partitions are in place, their relatively small size will cause queries to execute faster, and if a query should require access to multiple partitions, the queries will execute simultaneously.

It's also possible to partition a table such that different columns are stored in different partitions. This is called horizontal partitioning. Separating columns in this way can allow the database administrators to apply different levels of security and performance on the individual partitions.

In addition to improved query performance, other SQL functions should speed up as well because they're addressing smaller tables. Backing up 10 relatively small table partitions, for example, should be significantly faster than backing up a single large one. This is because the individual partition backups can occur simultaneously.

Partitioning tables can also enable Azure subscribers to save money by catering the resources of the individual databases to the sensitivity of the data and the frequency at which the data is needed. For example, if an order/entry database is partitioned by years, the partitions with the older data are probably accessed less frequently than the recent ones, so they might not need the same level of storage and compute resources.

Recommend a database sharding solution

Scaling up is the process of adding resources to a server to enhance its performance; scaling out is the process of adding more servers to share the load. *Sharding* is another method for dividing a table or database into partitions to improve its performance. However, while vertical table partitions are all stored in the same database instance (scaling up), shards are stored in different database instances (scaling out), as shown in Figure 1-24.

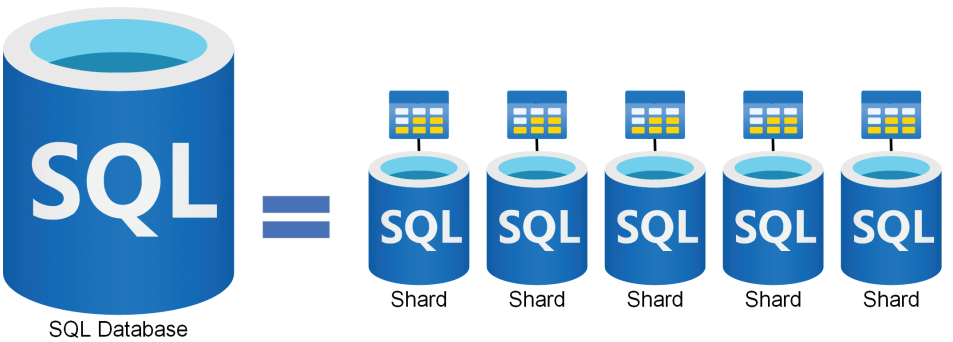


FIGURE 1-24 Sharding in Azure SQL is the division of a database into multiple partitions stored in separate database instances.

Subscribers typically create shards on Azure SQL Database installations, but it's possible to create them on SQL Managed Instance or Azure VMs running SQL Server. Scaling out through sharding provides the database with almost unlimited room for expansion and can improve SQL performance and fault tolerance as well.

The decision to shard a database is not one to be taken lightly because the process of creating and managing the shards can be difficult. In many cases, scaling up a single-instance database by adding virtual hardware is a perfectly adequate and completely reversible solution. For databases that are primarily read, rather than written to, replication of the entire database might be effective and far easier than sharding.

As with single-instance table partitioning, the division of the database among the shards must depend on the nature of the data stored in the database. One partitioning strategy calls for shards to be divided by rows, for example, with each shard containing a subset of complete records conforming to a specific date, location, or other attribute. However, it's also possible to divide a database by columns, so that each shard contains a portion of the records stored there. For example, the columns containing customers' credit card numbers can be stored in a shard on a server with enhanced security.

Three of the most commonly used strategies for dividing a database into shards are as follows:

- **Range-based sharding** The administrator selects a column in the database and creates shards containing rows based on ranges of values for that column. The ranges can be dates, product numbers, or any other logical division. This is a relatively simple partitioning strategy because all the shards use the same schema, but it does nothing to balance the amount of data in each shard. A shard that contains more data and is therefore accessed more frequently than the others is called a *hotspot*. Hotspots can be an administrative problem as the shards grow or shrink unevenly.
- **Hash-based (or key-based) sharding** The administrator selects a column in the database to use as a shard key. SQL performs a hash calculation on the shard key value for each row and uses the result to store the row in a particular shard. One benefit of this strategy is that it divides the data evenly among the shards, eliminating hotspots. However, scaling out further by adding more shards can be problematic because the data might have to be redistributed among the shards and some rows migrated between shards.
- **Lookup-based (or directory-based) sharding** The administrator selects a column in the database to use as a shard key and creates a lookup table that specifies a shard identifier for each value of the shard key. This method, which is better suited for use with shard keys that have relatively few values, has the advantage of providing the administrator with complete freedom to assign key values to any shard. This simplifies the process of scaling out to additional shards as well.

Sharding a database has both advantages and disadvantages, as discussed in the following sections.

Advantages of sharding

By placing the shards on separate servers, incoming queries can be directed to the specific server containing the shard with the required data. The shard is only a relatively small piece of the entire database, so there is relatively little to search, and the query processes faster.

Sharding also provides fault tolerance because each shard is a separate entity that can operate independently from the other shards. This independent functionality is due to the creation of a duplicate instance of the schema for each shard. If a server containing a shard should fail, only the part of the SQL database in that shard is offline; the other shards continue to function normally. This way, sharding can enable a SQL installation to avoid a total service outage.

When a database stored on a single server grows extremely large, its storage configuration can become unwieldy. There are only so many disks that can be added to a single server. Splitting the database into shards that are stored in different instances enables administrators to control the storage requirements for each shard individually. If some shards contain more sensitive data than others, administrators can enhance the security for only the servers containing those shards. In the same way, if some shards contain data that is accessed more frequently than that of others, additional compute resources can make the servers containing those shards more responsive.

Disadvantages of sharding

Sharding is a complicated process, and administrators should be aware of its potential disadvantages. By definition, sharding requires additional servers, which administrators must maintain. Every administrative task typically performed on a single-instance database is multiplied by the number of shards, plus any service nodes that might be needed.

Querying a sharded database can also be problematic. The installation must have a service or mechanism for routing queries to the appropriate shards, which can add a degree of latency to the database response. This routing capability is often built into the application making use of the database, but it can be implemented on the server as well. Another possible source of latency occurs when a query needs data stored in multiple shards. In that case, SQL must route the query to multiple servers and combine the replies to form the response.

Adding shards obviously incurs additional Azure costs for the servers and/or instances that will host the shards. It's up to the subscriber to compare the costs of scaling up the virtual hardware of a single-instance database with the cost of scaling out by creating new shards in additional instances.

Skill 1.2: Configure resources for scale and performance

One of the primary benefits of the Azure SQL implementations is its ability to configure its many options on the fly, with immediate results. These modifications can result in changes in the scale of the installation and its performance, as well as its ongoing cost.

This skill covers how to:

- Configure Azure SQL Database for scale and performance
- Configure Azure SQL Managed Instance for scale and performance
- Configure SQL Server on Azure Virtual Machines for scale and performance
- Configure table partitioning
- Configure data compression

Configure Azure SQL Database for scale and performance

Azure SQL Database provides subscribers with options that encompass a wide range of scalar and performance values, which are of course tied into variations in the subscription fees for the installation. Subscribers can configure most of the SQL DB options either during the deployment of the installation or afterward at any time.

When a subscriber creates a new SQL DB installation, the default Compute + storage settings are as follows:

Service tier: vCore General Purpose

Compute tier: Serverless

Compute hardware: Standard series (Gen5)

Each of these settings affects the scale, the performance, and the cost of the SQL Database. With few exceptions, subscribers can modify the settings as needed to scale the installation up or down or enhance its performance.

Using elastic pools

One of the first options that appears when you create an Azure SQL Database installation is the ability to use an elastic pool for its storage. An elastic pool, as the name implies, is a collection of storage resources that are shared among multiple SQL databases. If a subscriber plans to create multiple SQL DBs with similar requirements, an elastic pool can simplify the storage management process.

Once you create an elastic pool for your first single SQL DB, which is simply a matter of assigning a name to the pool, you can then create additional databases that share those same pooled resources. When necessary, subscribers can scale up by adding more storage to the pool.

Choosing a service tier

As noted in "PaaS purchasing models," earlier in this chapter, Azure SQL Database supports two purchasing models: database transaction units (DTUs) and vCore, each with its own service tiers. The DTU service tiers use a single metric to define the CPU, memory, and I/O resources allotted to the SQL DB, while vCore provides more granular ability to scale the installation's compute, memory, and storage resources.

For subscribers seeking a relatively simple, preconfigured solution, the DTU service tiers (Basic, Standard, and Premium) provide compute levels based on an adjustable number of DTUs, an adjustable maximum database size, and increasing periods of backup retention. The cost per DTU varies for each of the tiers, and subscribers can adjust the number of DTUs allotted to the installation.

The default service tier for a newly created SQL Database installation is the General Purpose vCore option. Subscribers can scale any of the vCore service tier installations by specifying a number of vCores from 2 to 128.

Subscribers requiring storage with lower levels of latency can enhance the performance of a vCore database by switching to the Business Critical service tier, which provides better I/O performance and fault tolerance. The price per vCore in the Business Critical tier is approximately 2.7 times that of the General Purpose tier, largely due to the three additional database replicas included, and the storage cost per GB reflects the added performance of the local SSD storage.

The third vCore service tier option is Hyperscale, which is designed to support databases with extremely large storage needs. Compared to the 4 TB maximum database size for the other service tiers, Hyperscale supports databases of up to 100 TB and up to 327,680 I/O operations per second (IOPS). Hyperscale can also have 10.2 GB of memory per vCore, which is twice that of the other service tiers.

Hyperscale is one of the relatively few Azure options that the subscriber cannot reverse. Once configured to use the Hyperscale service tier, the only way to change a SQL Database to another service tier is to redeploy it and migrate the data.

Choosing a Compute tier

In the vCore purchasing model, there is a separate Compute tier that enables subscribers to choose between Provisioned and Serverless options. The pricing for a vCore installation is a combination of the service tier, the virtual hardware configuration, the number of vCores selected, the amount of memory, the reserved database storage, and the actual backup storage in use.

The Provisioned tier preallocates Compute resources and charges the subscriber an hourly rate based on the number of vCores selected, regardless of the database's workload. Choosing the Serverless tier causes Azure to allocate compute capacity to the SQL DB installation as needed for its workload and charge the subscriber a Compute cost per vCore second plus the additional charge for storage.

When using the serverless tier, Azure can dynamically scale up the SQL DB to as many as 80 vCores, using as much as 240 GB of memory. The memory-to-vCore ratio can scale as well, supporting up to 24 GB of memory per vCore. When the database is not in use, Azure pauses it, and there are no Compute charges until it's reactivated (although storage charges continue).

Choosing Compute hardware

The vCore purchasing model enables subscribers to select a hardware configuration on which the SQL DB will run from a list like that shown in Figure 1-25.

The SQL hardware configuration consists of the following elements:

- **Max vCores** The maximum number of vCores supported by the installation
- **Max memory** The maximum amount of memory supported by the installation
- **Max storage** The maximum amount of storage permitted to the installation

Based on these values, Azure calculates a Compute cost per vCore per second. Subscribers can then scale the installation's compute power by specifying a maximum and minimum number of vCores in the hardware configuration.

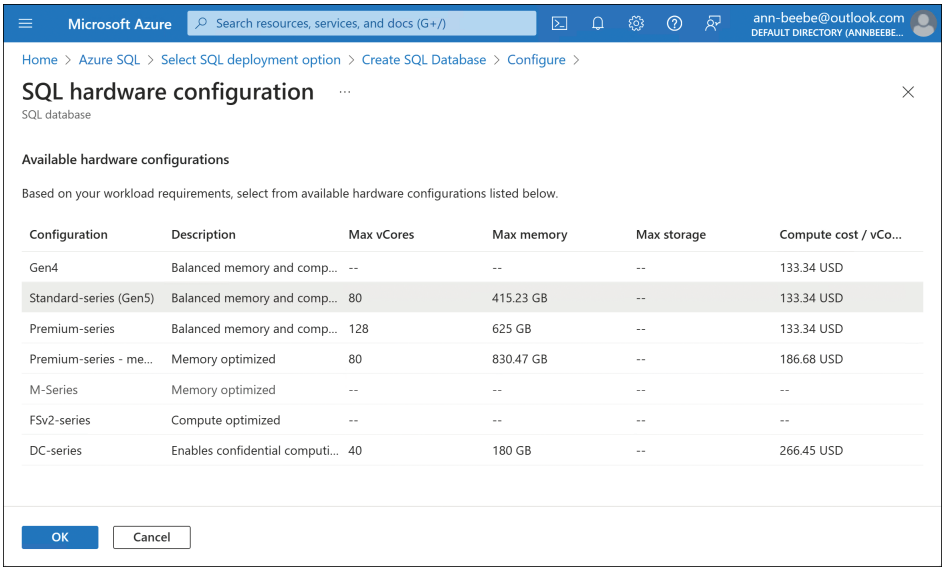


FIGURE 1-25 The SQL hardware configuration page

Configure Azure SQL Managed Instance for scale and performance

Like Azure SQL Database, Azure SQL Managed Instance is a PaaS offering that's similar in its deployment and scalability. When a subscriber creates a new SQL MI installation, the hardware and settings they select apply to all databases created within the managed instance.

When selecting a Compute + storage option for a SQL MI installation, only the vCore purchasing option is available; there are no DTU-based service tiers, and there is no Hyperscale option.

Selecting a service tier

The Service tier options for SQL MI are as follows:

- **General purpose** Recommended for most workloads, this tier separates the compute and storage resources and provides latency levels of 5-10 ms.
- **Next-gen General Purpose (currently in preview)** Enhanced version of the General Purpose tier with improved performance and reliability at a similar price, including increased maximum storage size and maximum number of databases.
- **Business critical** Recommended for workloads requiring lower (1–2 ms) latency, three additional replicas, higher availability, and faster recovery from failures. In this tier, compute and storage resources are integrated. The price is approximately 2.7 times that of the General Purpose tier.

The basic limitations of the SQL MI service tiers are shown in Table 1-1.

TABLE 1-1 Azure SQL Managed Instance service tier limits

	General Purpose	Next-gen General Purpose	Business Critical
Max vCores	80	128	128
Max Instance Storage	16 TB	32 TB	16 TB
Max databases	100	500	100
Read-only replicas	0	0	1
Availability replicas	Standby nodes	Standby nodes	4, with one read-scale replica

Selecting Compute hardware

After selecting a service tier, the subscriber must then select one of the following Compute hardware options:

- Standard series (Gen5)
- Premium series
- Premium series – memory optimized (Not available in all regions)

The memory for each hardware option is based on the numbers of vCores selected by the subscriber. To scale the memory and storage provisioned to the instance up or down, the subscriber can use the vCores and Storage in GB sliders when deploying or managing a SQL MI, as shown in Figure 1-26.

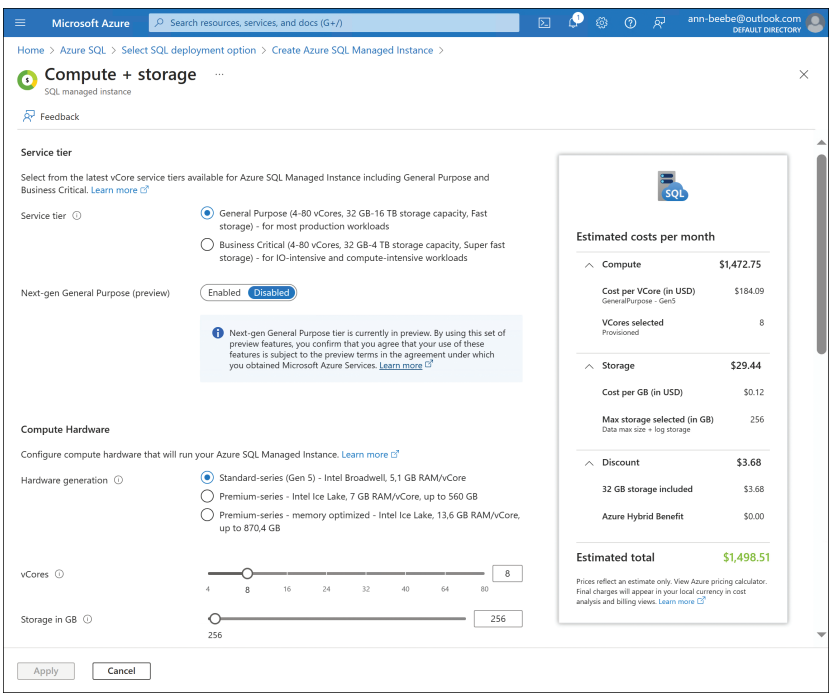


FIGURE 1-26 The SQL hardware configuration page

The basic properties of the SQL MI Compute hardware options are shown in Table 1-2.

TABLE 1-2 Azure SQL Managed Instance Compute hardware properties

	Standard series (Gen5)	Premium series	Premium series – memory optimized
CPU	Intel Broadwell (2.3 GHz), Skylake (2.5 GHz), or Cascade Lake (2.5 GHz)	Intel Ice Lake (2.8 GHz)	Intel Ice Lake (2.8 GHz)
Number of vCores	2 to 80	2 to 128	4 to 128
Memory per vCore	5.1 GB	7 GB	13.6 GB
Maximum memory	408 GB	560 GB	870.4
Maximum reserved storage per instance	16 TB (General Purpose) 4 TB (Business Critical	16 TB (General Purpose) 5.5 TB (Business Critical	16 TB (General Purpose) 16 TB (Business Critical

Configure SQL Server on Azure Virtual Machines for scale and performance

The PaaS Azure SQL products are designed to be easy to deploy and configure. The scaling and performance options that Azure provides are deliberately limited to those best suited to most applications. However, for subscribers who have extensive or special Azure SQL needs, IaaS is the usual solution, in the form of SQL Server running on an Azure virtual machine.

The IaaS Azure SQL model provides a multitude of compute and storage options that subscribers can select while creating the virtual machine to scale it to nearly any size or performance level.

As noted earlier in this chapter, the Select SQL deployment option page has as one of those options a SQL virtual machines box with a dropdown containing dozens of images from which to select, as shown in Figure 1-27. The images include various Windows and Linux operating system versions combined with SQL Server versions going back to 2012.

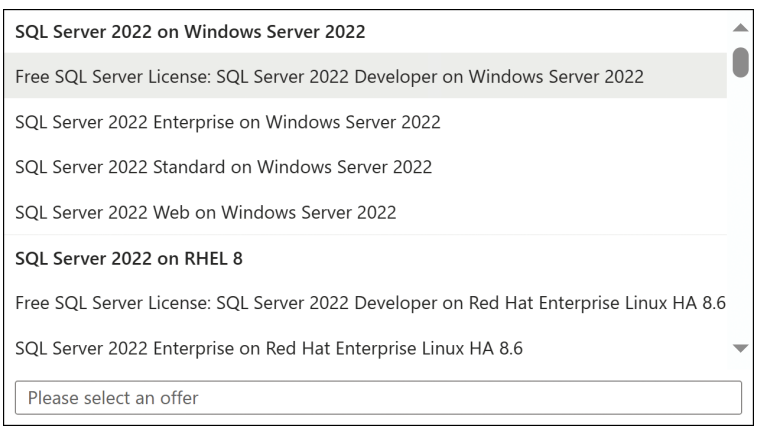


FIGURE 1-27 The Image dropdown on the Select SQL deployment option page

In the Create a virtual machine dialog, subscribers can scale the virtual machine by selecting from a large collection of virtual machine sizes. The contents of the Size dropdown depends on the operating system platform and version selected earlier in the Image dropdown. The Size list contains Azure's most popular sizes, but clicking the See all sizes link opens a page displaying the entire list of hundreds of virtual machine configurations, a few of which are shown in Figure 1-28.

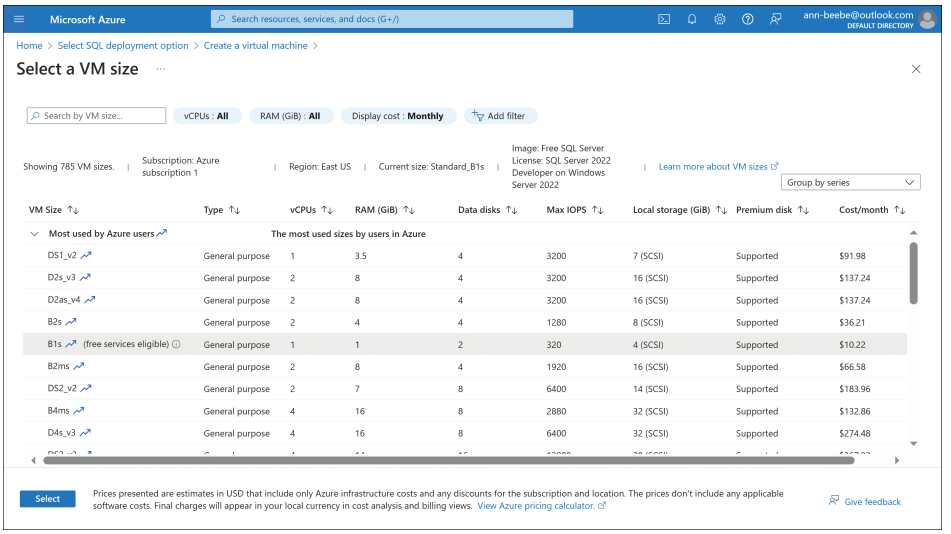


FIGURE 1-28 The Select a VM size page in the Create a virtual machine dialog

The Azure virtual machine hardware configurations are grouped into six types that describe various ratios between the CPU, memory, and storage resources they include. Each type has configurations of various sizes. Selecting a size from one of the type families enables subscribers to scale the performance of the VM to suit their workload. The six Azure VM hardware configuration types are as follows:

- **General purpose** Balanced CPU-to-memory ratio best suited for development and small databases
- **Compute optimized** Higher CPU-to-memory ratio, for medium traffic web and application servers
- **Memory optimized** Higher memory-to-CPU ratio, good for relational database servers
- **Storage optimized** Higher I/O performance, suitable for large SQL databases
- **GPU** Enhanced graphical performance intended for image rendering and video editing
- **High performance compute** Highest performance CPUs available; intended for extreme workloads, such including fluid dynamics, weather simulation, and financial analysis

Obviously, the prices for the various configurations go up as they provide more resources. For help selecting an appropriate VM configuration, Microsoft provides a Virtual machines selector tool at *https://azure.microsoft.com/en-us/pricing/vm-selector*.

In addition to the compute options, storage is also important on a virtual machine running SQL Server. On the Disks tab of the Create a virtual machine dialog, shown in Figure 1-29, the subscriber can scale the VM's storage performance by selecting an OS disk size and OS disk type.

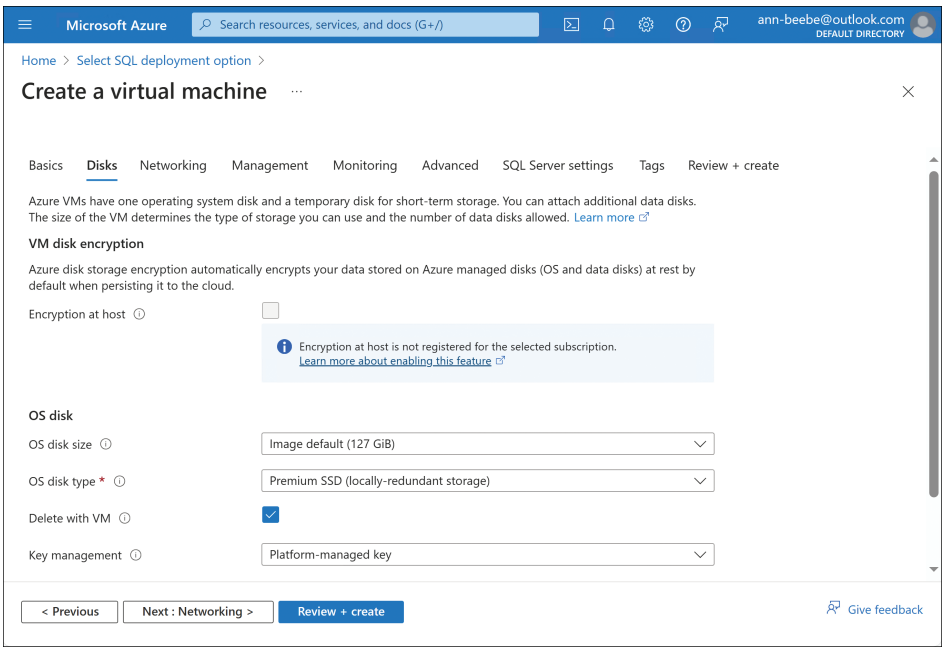


FIGURE 1-29 The Disks tab in the Create a virtual machine dialog

The OS disk type dropdown enables the subscriber to choose from the following disk types:

- **Standard SSD** Solid-state disk recommended for development and testing or light workloads; available with local or zone redundancy
- **Premium SSD** Solid-state disk recommended for SQL Server workloads; available with local or zone redundancy
- **Standard HDD** Hard disk drive recommended for backups and occasional access; available with local redundancy only

On the SQL Server settings tab, the subscriber can scale the VM's data, log, and tempdb storage by clicking the Change configuration link to display the interface shown in Figure 1-30. Here, the subscriber can choose between Premium SSD and Ultra SSD. Ultra SSDs provide enhanced I/O performance and lower latency for data-intensive workloads.

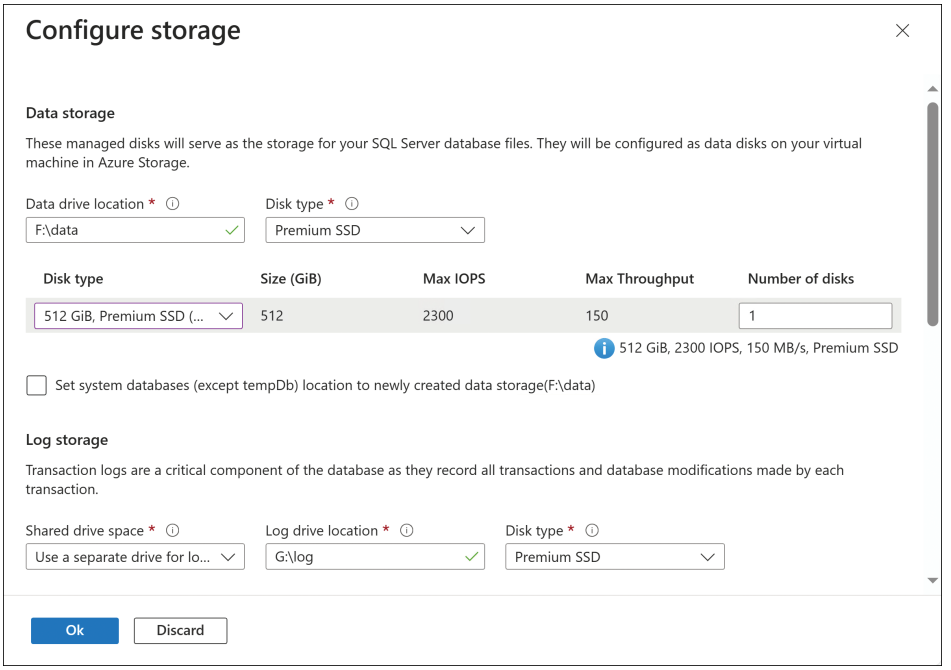


FIGURE 1-30 The Configure storage page in the Create a virtual machine dialog

Configure table partitioning

As noted earlier in this chapter, it's possible to divide a table into partitions within a single instance to improve query performance. The process of partitioning a table in a SQL database consists of the following steps:

1. Create filegroups (optional).
2. Create data files.
3. Create partition function.
4. Create partition schema.
5. Create clustered index.

Creating table partitions using the `CREATE PARTITION` command in SQL Server Management Studio (SSMS) automatically creates the file groups and files for the individual partitions. SQL also unifies the index for the partitions so that the same queries that worked with the large single partition will now function equally well with the smaller partitions. This means that no changes are needed to the application making use of the database as a result of the partitioning; the queries the app generates will still function normally, or possibly even faster than before.

Configure data compression

Data compression eliminates redundancy at the bit level to make files smaller. Smaller files take less time to query and can improve I/O performance. The tradeoff in data compression is that to achieve better storage performance, additional compute resources are required. The algorithm needed to compress files for storage and decompress them again for access adds to the server's CPU burden. In most cases, the advantages of data compression outweigh the cost of additional CPU cycles.

The compression ratio for a table is the percentage of storage space saved by the compression process. The storage savings realized by any form of data compression depends on the nature of the data being compressed. Some data formats can compress to half of their original size, while others do not compress at all.

When applying data compression to entire tables, table partitions, or indexes, SQL supports four types of compression, as follows:

- **Row compression** Requires minimal additional compute resources and uses variable-length formatting for columns to eliminate extra spaces.
- **Page compression** Requires three separate compression passes—row compression, prefix compression, and dictionary compression—to eliminate redundant characters and achieve higher compression ratios at the cost of greater compute resources.
- **Columnstore compression** Columnstore is a table that stores data in a columnar format, rather than the standard rowstore format. Columnstore indexes provide a greater degree of compression and query performance and are intended for high-volume data warehousing and analytics. All columnstore tables and indexes use this compression automatically.
- **Columnstore_archive compression** Additional compression capability for columnstore tables and indexes that are less frequently accessed and for which the additional time and compute resources needed to implement the compression and decompression are acceptable.

To apply compression to a table or other element using SQL Server Management Studio (SSMS), right-clicking a table and selecting Storage > Manage compression from the context menu opens the Data Compression Wizard, as shown in Figure 1-31.

Subscribers can also modify the compression status of a table or index using the `ALTER TABLE` command with the `DATA_COMPRESSION` parameter, as in the following example:

```
ALTER TABLE dbo.CompressionTest REBUILD PARTITION = ALL
WITH
(DATA_COMPRESSION = PAGE)
GO
```

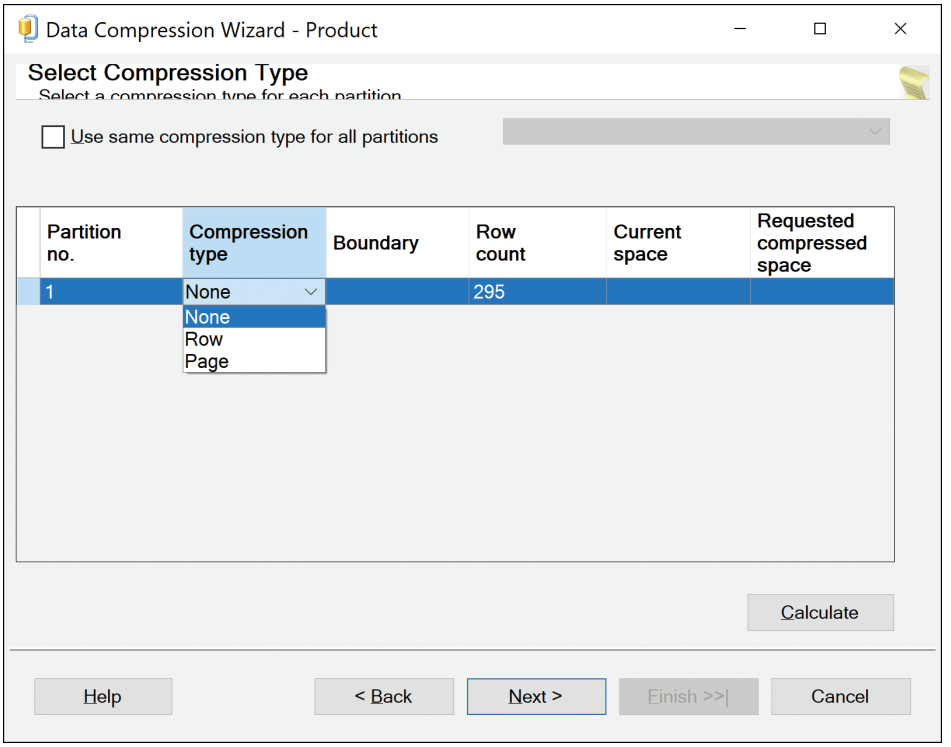


FIGURE 1-31 The Data Compression Wizard in SQL Server Management Studio dialog

The `DATA_COMPRESSION` parameter can have any of the following values:

- NONE
- ROW
- PAGE
- COLUMNSTORE
- COLUMNSTORE_ARCHIVE

Skill 1.3: Plan and implement a migration strategy

Many organizations maintain a hybrid SQL infrastructure that incudes both on-premises SQL servers—either physical or virtual—and cloud-based installations. Azure's SQL products can support a hybrid infrastructure of almost any size, with data duplicated in multiple locations, including the cloud. However, for some organizations that rely on SQL, the hybrid infrastructure is just a temporary measure to facilitate the permanent migration of the company's data to the cloud.

A data migration is an inherently dangerous undertaking because the original data sources will presumably be taken offline once the migration is completed. Administrators must see to

it that all data is successfully moved to the cloud location and that all updates and queries are directed to the database's new location.

This skill covers how to:

- Evaluate requirements for the migration
- Evaluate offline or online migration strategies
- Implement an online migration strategy
- Implement an offline migration strategy
- Perform post-migration validations
- Troubleshoot a migration
- Set up SQL Data Sync for Azure
- Implement a migration to Azure
- Implement a migration between Azure SQL services

Evaluate requirements for the migration

Planning is a critical part of the SQL data migration process. Subscribers must familiarize themselves with the Azure SQL products and how they compare with their on-premises SQL installations. For a time, the Azure product and the on-premises servers will have to be connected together in a hybrid infrastructure. Eventually, after the data migration is completed, the on-premises servers will presumably be taken offline.

Assessing compatibility levels

One of the first elements for subscribers to consider in a migration is the compatibility of their on-premises SQL Server databases and the Azure SQL products. The Azure PaaS options—Azure SQL Database and Azure SQL Managed Instance—always run the latest confirmed version of SQL Server and are continually updated. To run any other version of SQL Server in the Azure cloud, subscribers must use the IaaS alternative and install a virtual machine with the SQL Server application running on it.

Comparing SQL Server on-premises versions with those in the cloud can be a problem. The version numbers for Azure SQL DB and Azure SQL MI do not correspond with the build numbers of the Microsoft SQL Server application. However, the Azure products do use the same compatibility levels as SQL Server.

A compatibility level is a setting for an individual database from 80 to 150 that specifies its compatibility with other databases in areas such as Transact-SQL syntax. Administrators can configure different compatibility level settings for each of the databases in a SQL instance.

To check the compatibility level of a SQL database, use the SELECT command, as follows:

```
SELECT name, compatibility_level FROM sys.databases;
```

Figure 1-32 shows the command executed in SSMS. As long as a subscriber's on-premises databases have the same compatibility level value as an Azure database, the migration can proceed.

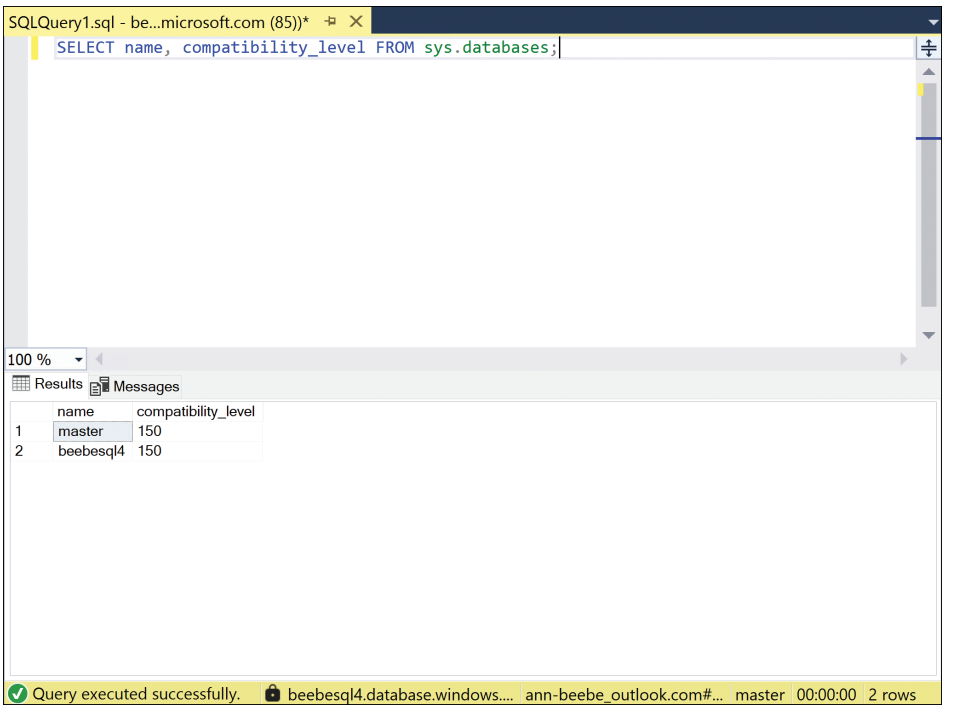


FIGURE 1-32 Checking compatibility level with the SELECT command

Assessing resource requirements

When planning a data migration to the Azure cloud, on-premises SQL Server administrators must consider first which Azure SQL product they will use as the target of their data migration. Networks running a compatible version of SQL Server can migrate to one of the PaaS options: SQL Database or SQL Managed Instance.

However, sometimes applications require a database running on a specific SQL Server version, which might in turn have specific operating system version requirements. If an on-premises installation requires a previous version of SQL Server or a specific operating system version, then the administrator seeking to migrate to the Azure cloud will have to use the IaaS model and create an Azure virtual machine that is compatible with the on-premises servers.

By creating a VM in Azure, the subscriber can choose from many versions of SQL Server and many operating system versions to create a suitable environment for the migrated data.

Assessing downtime

Another important consideration when planning a data migration is whether the databases to be migrated can tolerate sufficient downtime for the migration to occur. Depending on the

amount of data involved and the criticality of the applications making use of the databases, taking the databases offline might not be feasible.

There are two types of data migrations: offline, in which the source database servers are taken down while the data migration occurs, and online, in which the databases remain accessible to queries during the migration. Online migrations can complicate the process enormously, especially when there are new updates applied to the databases while the migration is occurring.

Assessing security requirements

Whatever security requirements are currently applicable to an on-premises SQL installation must also apply to an Azure installation in the cloud. Therefore, subscribers considering a data migration must ascertain whether Azure can provide the same security as the on-premises installation.

Security requirements can be the result of government regulation or individual contractual agreements. For example, there might be contractual restrictions on where or how the data in a database is stored. This could require subscribers to modify their Azure storage strategy before the migration; it might even prevent the migration from taking place entirely if the contract precludes storing data in the cloud.

Evaluate offline or online migration strategies

Azure can support migrations of on-premises SQL Server data, schema, and objects to the cloud or migrations of other database platforms, such as MySQL, PostgreSQL, and MariaDB. There are a variety of Azure tools that facilitate the different types of migrations, both online and offline.

The only downtime during an online migration is the brief interval during which the source database cuts over to the new target database, usually only a few seconds. An offline migration can take substantially longer, depending on the amount of data involved and the throughput of the connection to the cloud.

The downtime necessary for an offline migration is something database administrators have to assess before they commit to a specific migration tool or strategy. Some administrators perform offline migrations on a test platform, to determine just how much downtime would be needed in the production environment. If the application using the SQL databases cannot tolerate the downtime needed for an offline migration, then an online one might be the only alternative.

Using Azure Migrate

Azure Migrate is an Azure service that coordinates migrations between on-premises SQL Server databases and Azure SQL products, including Azure SQL Database, Azure SQL Managed Instance, and Azure SQL virtual machines running SQL Server. In some cases, such as a migration from an on-premises SQL Server installation to an Azure SQL MI, Azure Migrate can perform the migration online.

Azure Migrate supports several different migration scenarios, as shown in Figure 1-33. The most basic form of SQL migration is known as a "lift and shift," in which an on-premises SQL architecture is duplicated in an Azure VM running SQL Server, after which the data and workload are migrated from the datacenter to the cloud.

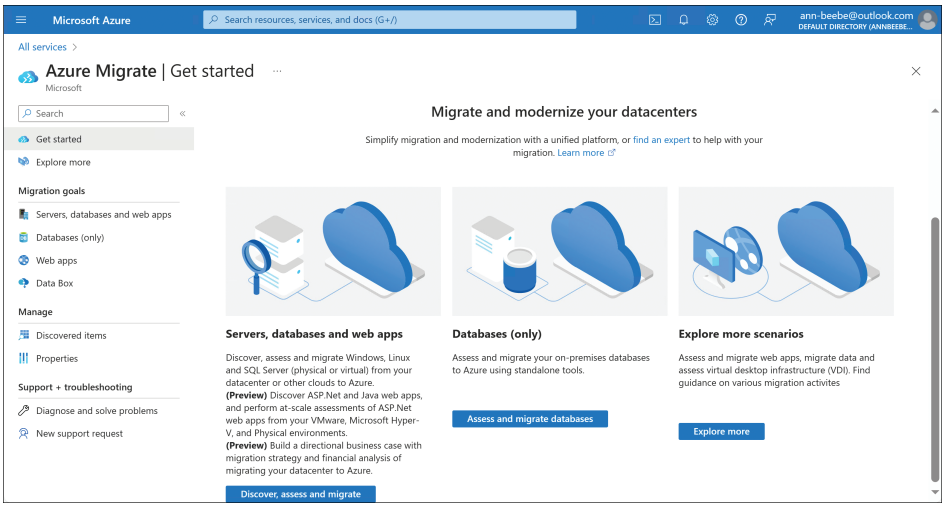


FIGURE 1-33 The Azure Migrate Get started page

Lift and shift might appear to be the easiest migration solution for the database administrator, but Azure can provide alternatives that might save them time and money, such as Azure SQL DB and SQL MI.

Azure Migrate is a cloud service, so to coordinate a data migration, it must have contact with the on-premises servers. This contact requires a software agent running on-premises called the Azure Data Migration Assistant (DMA). Once downloaded and installed on-premises, the DMA discovers the SQL Server (either physical or virtual) and allows the subscriber to select a database and scan it for feature parity and compatibility issues with a particular Azure SQL product, as shown in Figure 1-34.

When the scan is completed, the DMA can upload the result to Azure Migrate in the cloud, and Azure Migrate displays the database assessment, as shown in Figure 1-35. From the Azure portal, the subscriber can then use the assessment information to perform the migration using Azure Data Migration Service or another tool.

While Azure Migrate and DMA are effective migration assessment tools, there are others, including the following:

- **Database Experimentation Assistant** Evaluates the suitability of specific SQL Server versions for a given workload
- **SQL Server Migration Assistant (SSMA)** Automates the migration of Microsoft Access, DB2, MySQL, Oracle, and SAP ASE databases to SQL Server, Azure SQL Database, or Azure SQL Managed Instance

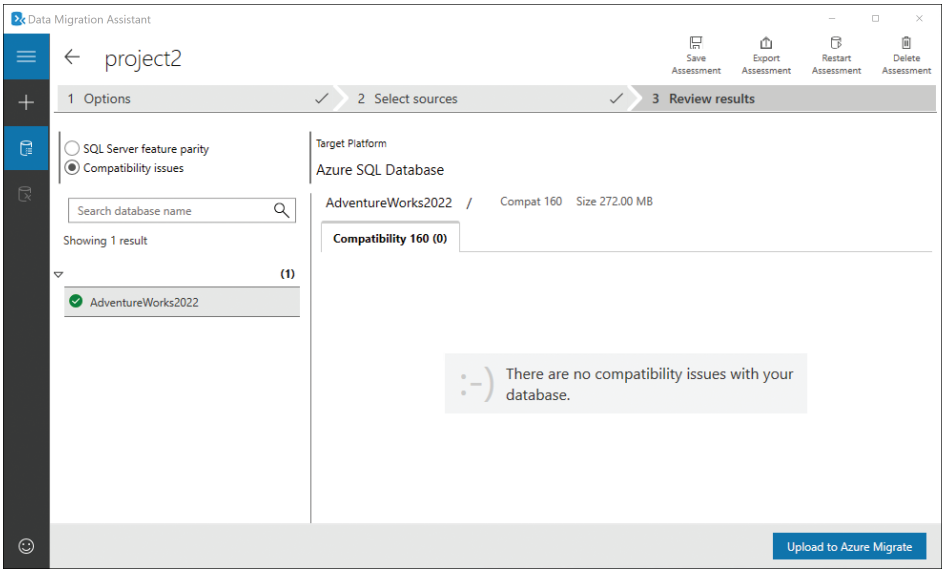


FIGURE 1-34 The Azure Data Migration Assistant

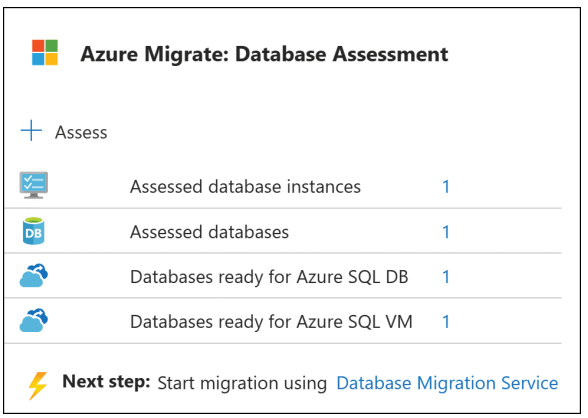


FIGURE 1-35 The Azure Migrate: Database Assessment display

Using Azure Database Migration Service (DMS)

The current primary tool for migrating on-premises databases to Azure SQL is the Azure Database Migration Service (DMS). DMS is a cloud service that is designed to provide online or offline migrations of on-premises databases to the Azure SQL products.

While DMS emphasizes online migrations, it can support both online and offline migrations for some combinations of source and target databases, but not all of them. Table 1-3 lists the supported source and target databases for online and offline migrations in Azure DMS.

TABLE 1-3 Azure DMS online and offline support

Source	Target	Online migration supported	Offline migration supported
SQL Server	Azure SQL Database	No	Yes
SQL Server	Azure SQL Managed Instance	Yes	Yes
SQL Server	Azure SQL VM	Yes	Yes
MongoDB	Azure Cosmos DB	Yes	Yes
MySQL	Azure Database for MySQL	Yes	Yes
PostgreSQL	Azure Database for PostgreSQL	Yes	No

At a high level, the steps to perform a typical online migration using DMS are as follows:

1. Provision an Azure SQL Database, Managed Instance, or VM SQL installation and copy the on-premises databases to it.
2. Configure the on-premises installation to continuously synchronize all new database transactions with the Azure SQL installation.
3. When ready, cut over from the on-premises SQL servers to the Azure SQL installation by changing the connection strings in the applications accessing the databases.
4. Stop the replication process and take the on-premises servers offline.

Using Azure Data Studio

Azure Data Studio is a cross-platform data analytics tool that, with the Azure SQL migration extension installed, as shown in Figure 1-36, can perform an analysis of an on-premises SQL installation, recommend an Azure SQL product as the target for migration, and perform the actual migration using Azure Data Migration Service.

Other migration tools

In addition to the tools listed in the previous sections, there are several other ways to migrate data from an on-premises SQL installation to the Azure cloud, including the following:

- **Transactional replication** After duplicating the source database on the target installation, administrators can configure the source server to publish all new SQL transactions to the target on a subscription basis in near or real time.
- **Import/Export Service** BACPAC is a file format for the exportation of a SQL database's data and schema that administrators can use to transfer data from an on-premises SQL Server installation to an Azure SQL product in the cloud.
- **Bulk copy** The bulk copy program (bcp) is a cross-platform tool that exports data from a SQL Server installation to a data file.
- **SQL Data Sync** A bidirectional data synchronization service included in Azure SQL Database that enables subscribers to create sync groups using a hub-and-spoke topology.

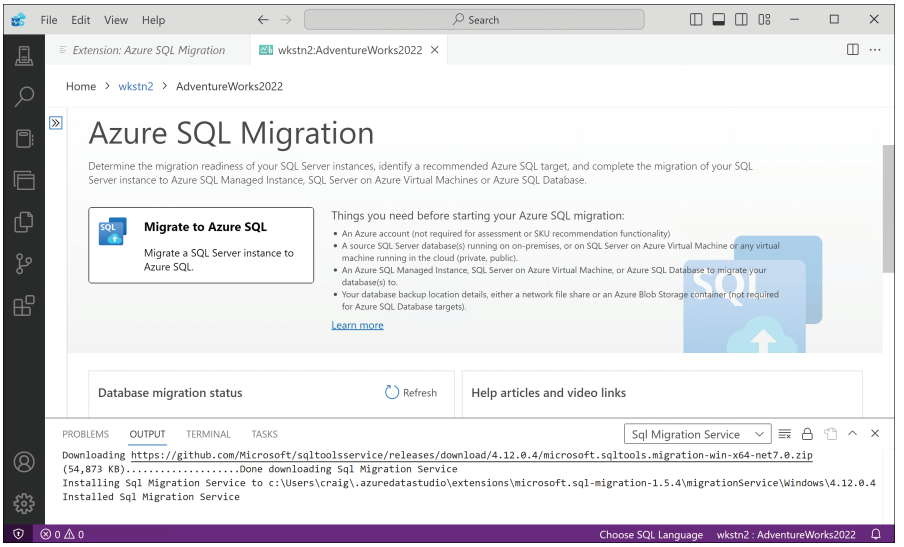


FIGURE 1-36 Azure Data Studio with the Azure SQL migration extension installed

Implement an online migration strategy

To perform an online migration using the Azure Data Migration Service, the subscriber first creates a DMS instance by selecting Create on the Azure Database Migration Service page. This opens the Select migration scenario and Database Migration Service page, as shown in Figure 1-37.

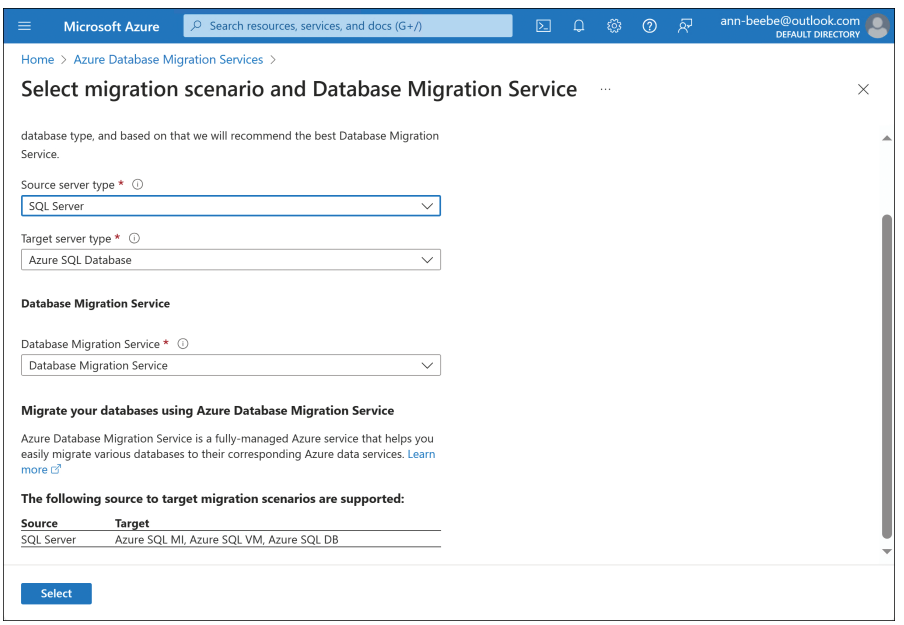


FIGURE 1-37 The Select migration scenario and Database Migration Service page

On this page, the subscriber defines the nature of the migration by selecting a Source server type and a Target server type. This enables the subscriber to configure a migration from a variety of (Microsoft and non-Microsoft) database products to any of the three Azure SQL products.

After creating an instance of the Data Migration Service, selecting the instance opens its page, as shown in Figure 1-38. From here, the subscriber can initiate the migration or monitor ongoing ones.

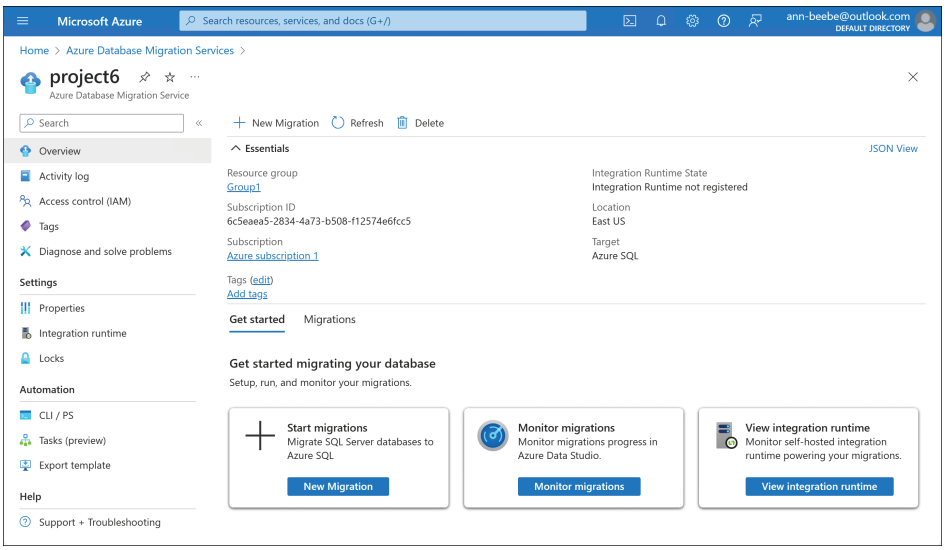


FIGURE 1-38 An Azure DMS instance page

Clicking the New Migration button opens the Select new migration scenario page, as shown in Figure 1-39. Here, the subscriber specifies the Source server type and the Target server type, including the specific Azure SQL product. The other options that appear on the page depend on those source and target server type selections. Below the selections, Azure DMS displays a context-sensitive list of prerequisites the subscriber must complete before the migration can begin.

In this example, migrating an on-premises SQL Server to an Azure SQL Managed Instance using blob storage enables the subscriber to select either an Online or Offline Migration mode.

> ***NOTE*** **ONLINE AND OFFLINE MIGRATION OPTIONS**
>
> **As noted in Table 1-3, Azure DMS does not support online migrations from SQL Server to Azure SQL Database, so selecting that option from the Target server type dropdown would disable the Online Migration mode option.**

Next in this example, the DMS launches the Azure SQL Managed Instance Online Blob Migration Wizard, which takes the subscriber through the process of selecting the migration target, as shown in Figure 1-40.

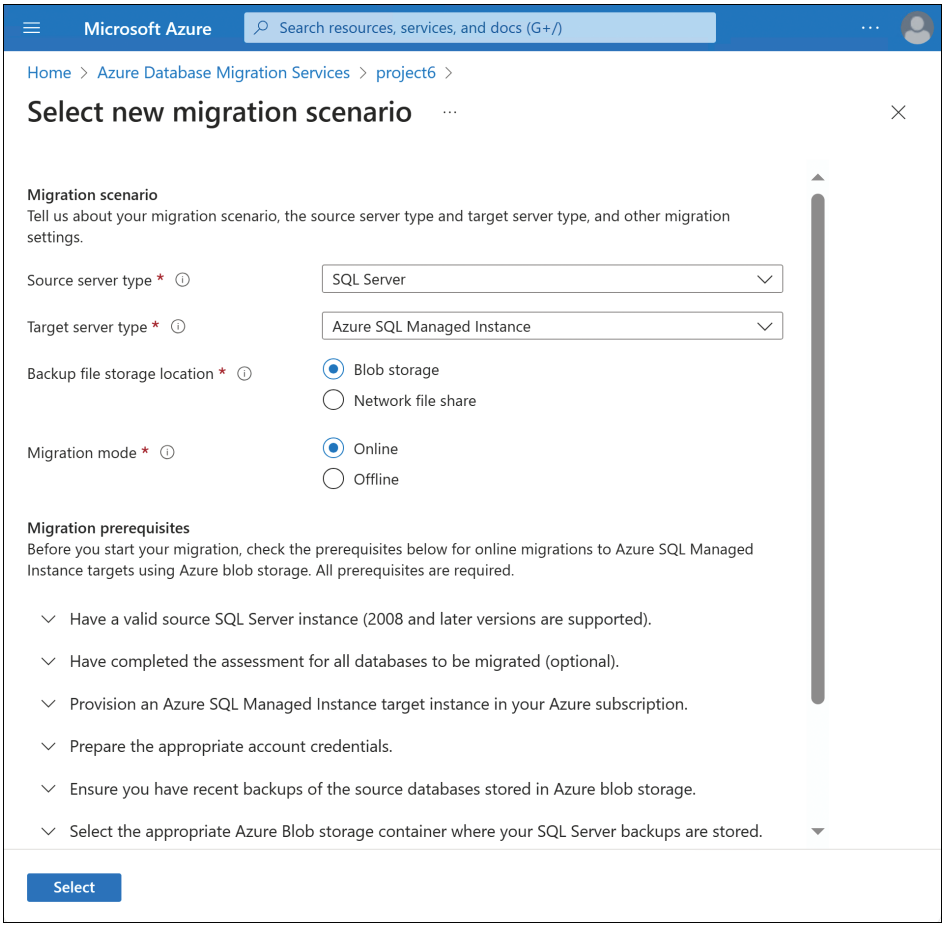


FIGURE 1-39 The Azure DMS instance page

Included in the prerequisites on the Select new migration scenario page is the need for the subscriber to create a storage account and upload backups of the source SQL Server databases to blob storage in the cloud. The Data source configuration page in the Azure SQL Managed Instance Online Blob Migration Wizard calls for the subscriber to identify the location of those database backups in blob storage, as shown in Figure 1-41.

After the subscriber reviews the Database migration summary page and starts the migration, the DMS begins restoring the database from the backup in blob storage to the target managed instance, as shown in Figure 1-42, and then initiates the replication of new transactions from the source to the target.

Once the backup data is written to the target, the Migration status indicator changes to Ready for cutover, indicating that the database administrators can redirect their applications to point to the Azure SQL database in the cloud. This brief cutover interval is the only downtime for the databases during an online migration.

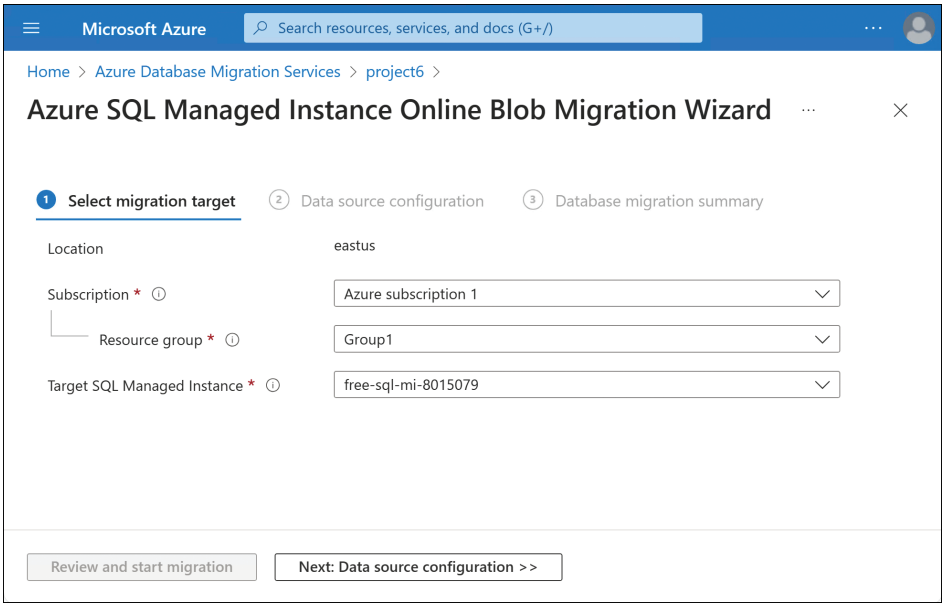


FIGURE 1-40 The Select migration target page in the Azure SQL Managed Instance Online Blob Migration Wizard

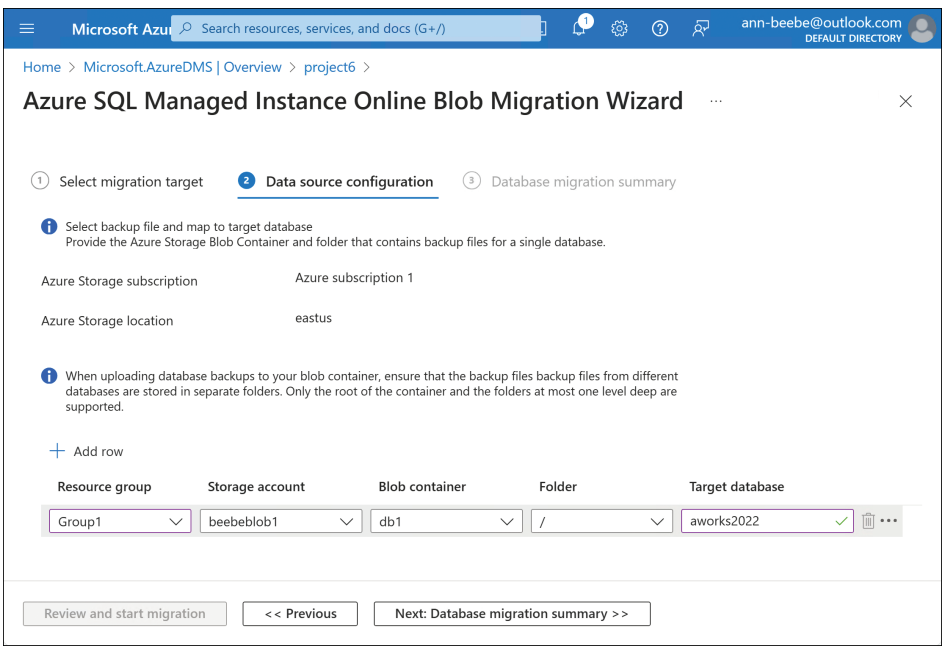


FIGURE 1-41 The Data source configuration page in the Azure SQL Managed Instance Online Blob Migration Wizard

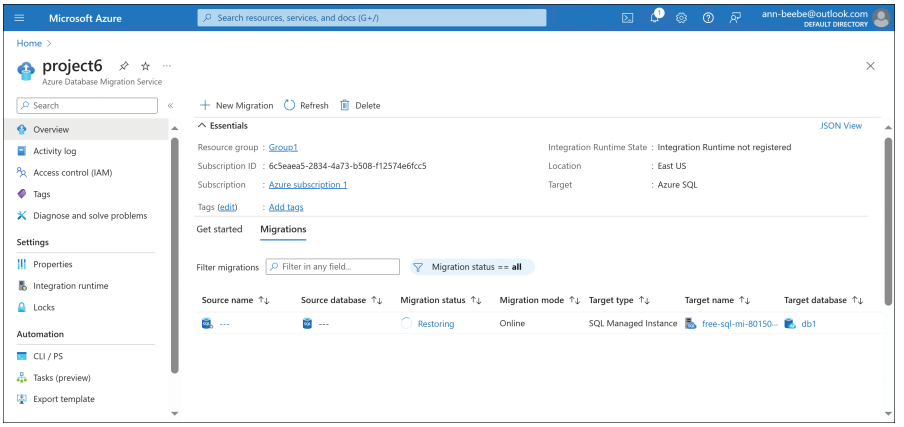


FIGURE 1-42 The Azure DMS instance page with a migration in progress

Implement an offline migration strategy

Azure Data Studio, with the Azure SQL migration extension installed, can also perform both online and offline SQL migrations. Unlike Azure DMS, which is a cloud service, Azure Data Studio is an application that runs on the source network.

When performing a migration, Azure Data Studio walks the subscriber through the steps involved, as shown in Figure 1-43. First, the subscriber selects the on-premises database to analyze. Azure Data Studio then examines the database and prompts the subscriber to start the data collection process that the tool uses to assess the database traffic and recommend an Azure virtual hardware configuration.

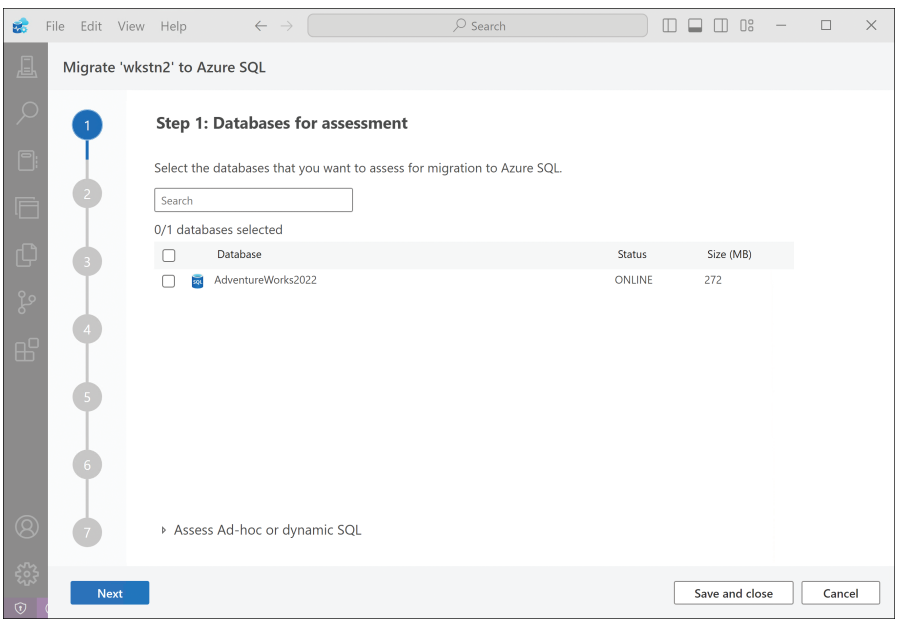


FIGURE 1-43 Step 1 of a data migration in Azure Data Studio with the Azure SQL migration extension installed

When Azure Data Studio completes its assessment of the source database, it displays a summary of potential migration targets, as shown in Figure 1-44. Unlike Azure Migrate, Azure Data Migration Assistant, and other assessment tools, Azure Data Studio includes virtual hardware recommendations for the various Azure SQL products based on the source database's actual traffic.

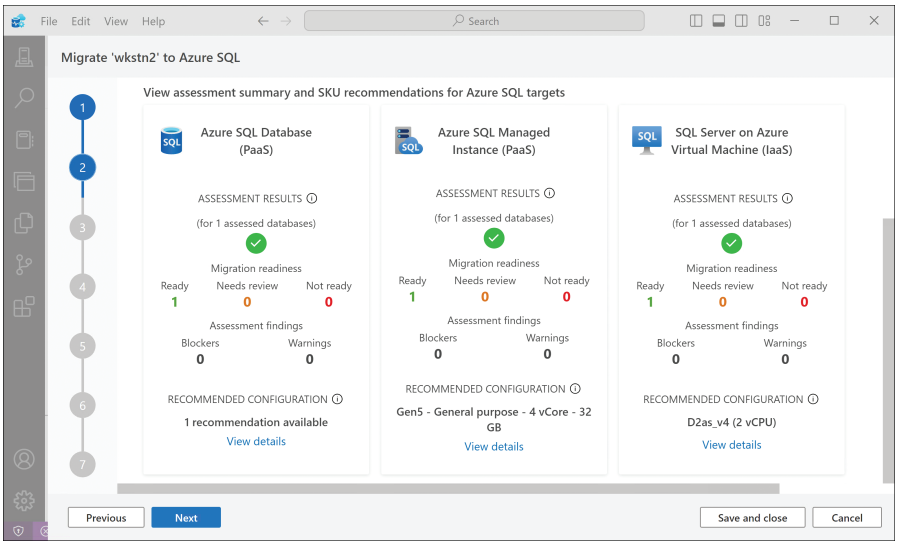


FIGURE 1-44 Assessment summary in Azure Data Studio with the Azure SQL migration extension installed

In Step 3, the subscriber selects one of the recommended options in the Select target type dropdown. This displays the assessment summary, as shown in Figure 1-45, and allows the subscriber to select the source databases to be migrated.

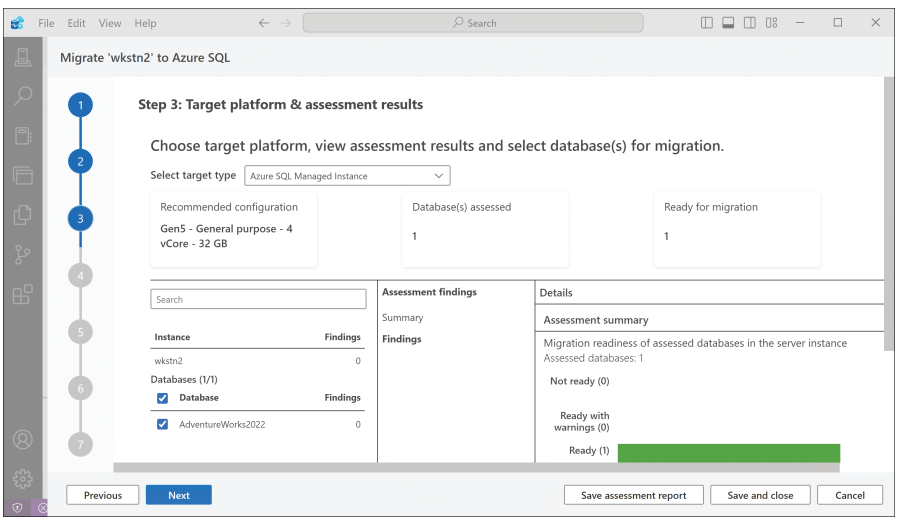


FIGURE 1-45 Step 3 of a data migration in Azure Data Studio with the Azure SQL migration extension installed

In Step 4, as shown in Figure 1-46, Azure Data Studio prompts the user for an Azure account and the location of the target SQL managed instance.

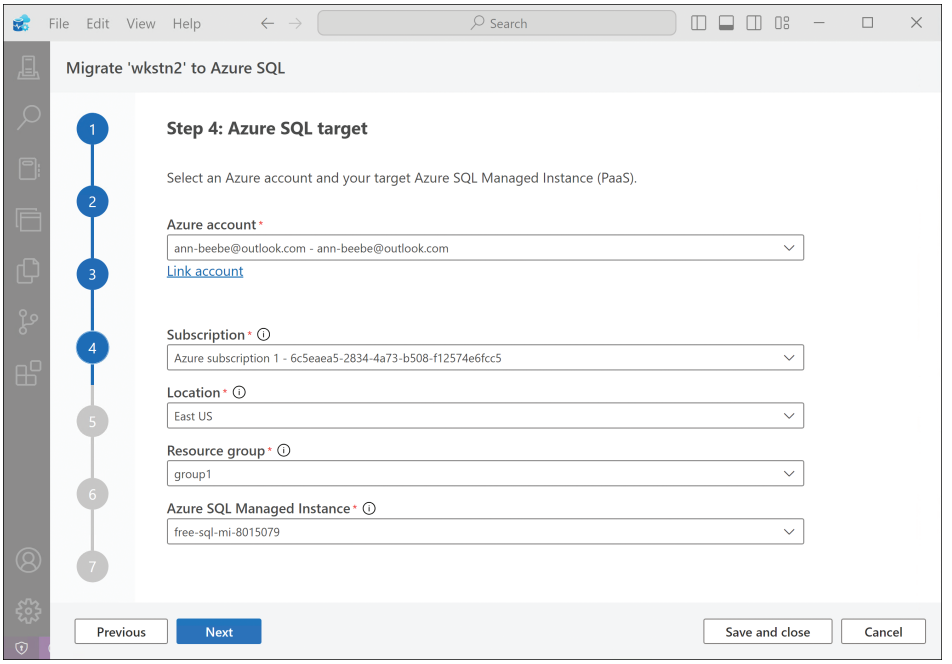


FIGURE 1-46 Step 4 of a data migration in Azure Data Studio with the Azure SQL migration extension installed

In Step 5, after having identified the source and the target for the migration, the subscriber can specify whether to perform an online or offline migration and whether the database backup files are located in blob storage or on a network share, as shown in Figure 1-47. Finally, the subscriber selects an instance of the Azure Data Migration Service (or creates a new one). Azure Data Studio relies on the DMS to handle the cloud end of the migration.

In Step 6, shown in Figure 1-48, the subscriber specifies the location of the backup previously uploaded to blob or network storage.

Step 7, shown in Figure 1-49, is the Summary page for the subscriber to review before initiating the migration. In an offline migration like this one, the downtime for the database begins as soon as the migration starts and continues until the cutover is completed.

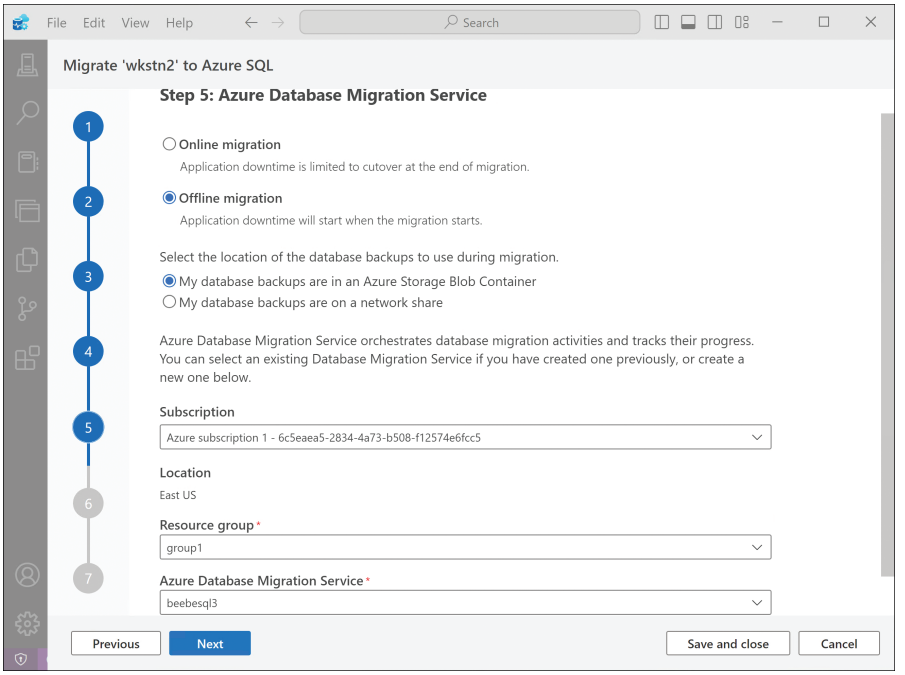


FIGURE 1-47 Step 5 of a data migration in Azure Data Studio with the Azure SQL migration extension installed

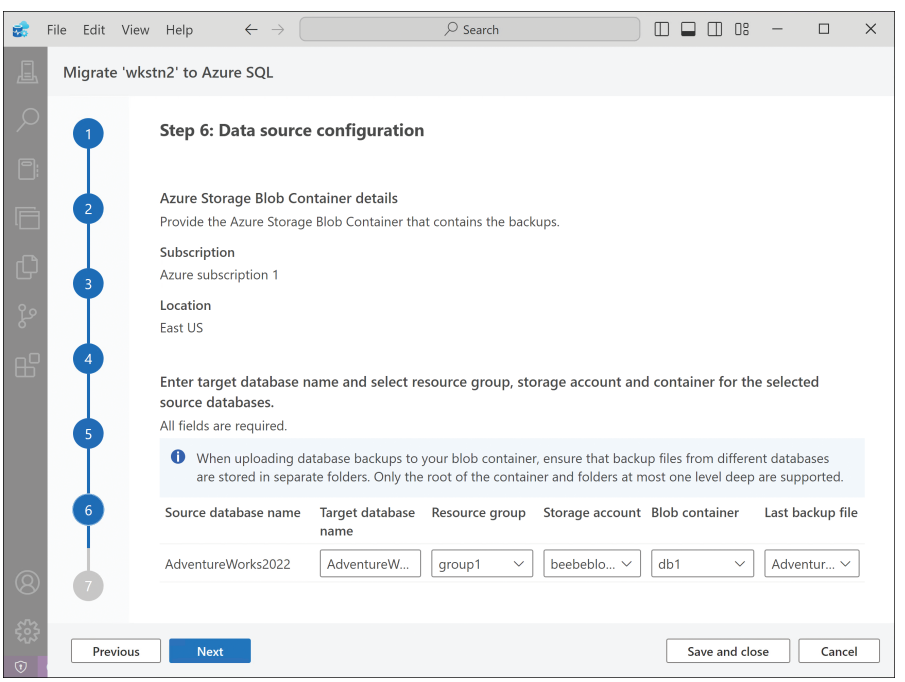


FIGURE 1-48 Step 6 of a data migration in Azure Data Studio with the Azure SQL migration extension installed

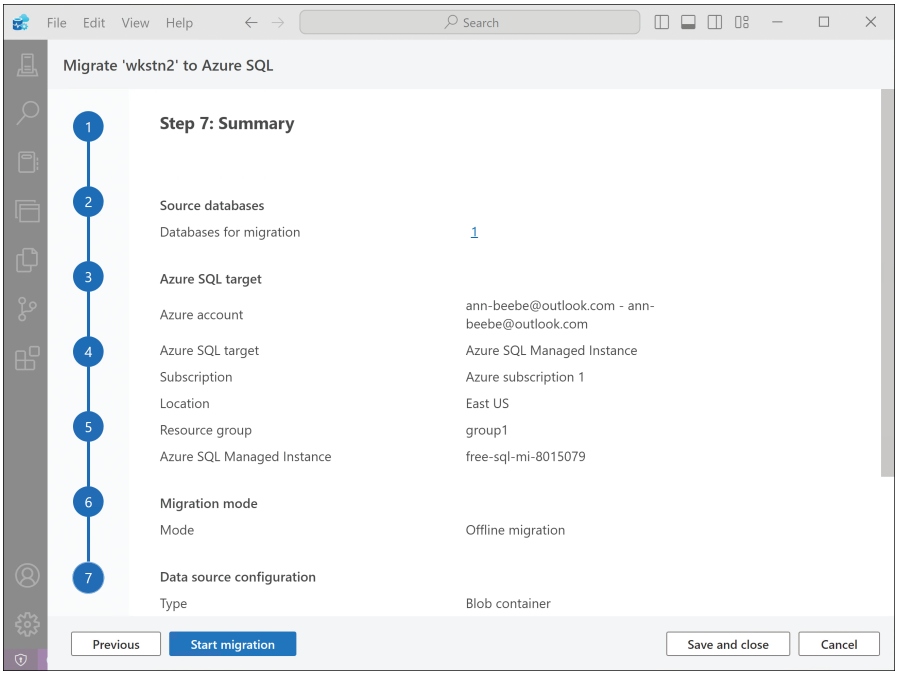


FIGURE 1-49 Step 7 of a data migration in Azure Data Studio with the Azure SQL migration extension installed

Perform post-migration validations

Migrating SQL databases to Azure cloud services is a major undertaking for SQL administrators and for the organization they are supporting. However, the job is not over, even after the actual data migration is completed. Following are some of the additional tasks administrators might have to perform once the data is migrated to the cloud.

Testing database performance

Once the on-premises data is copied to the Azure installation and before the applications are cut over to the cloud, the SQL database administrators should perform a series of identical query tests against both the source database and the newly created target to confirm that both databases return the same result.

Another consideration is the performance level of the new cloud installation when compared to that of the on-premises servers. If administrators have properly assessed the source databases using Azure Data Studio with the Azure SQL migration extension, the recommended virtual hardware configuration for the cloud installation should yield a similar level of performance.

However, administrators should definitely test those performance levels. If the cloud installation is performing substantially less well than the on-premises servers, then the administrators might want to consider a cloud hardware upgrade before taking the new databases live.

Administrators should also make sure that all of the applications that utilize the SQL databases can access them successfully in the cloud. Issues such as permissions and cloud

connectivity are often not predictable or measurable by an assessment tool, so administrators should do extensive testing before committing the organization to the new databases.

Migrating SQL logins

One of the many decisions SQL administrators should make before migrating their databases to the Azure cloud is when they should create the necessary SQL logins, user roles, and permissions on the target installation, before the migration or after. Except in the simplest circumstances, migrating logins after the database migration is preferable for several reasons, including the following:

- Migrating a complex permission structure after the database migration enables Azure to apply the permissions to the post-migration databases in their final form.
- Migrating the logins before the databases might cause a slowdown of the post-migration testing process.
- If administrators are making any changes to the database table structures during the data migration process, migrating the logins after the data enables them to accommodate the final SQL structures.

Administrators can migrate logins to Azure SQL Managed Instance or an Azure VM running SQL Server using the Azure Data Studio tool with the Azure SQL migration extension. The administrator identifies the Azure SQL target on the Step 1 page, as shown in Figure 1-50, and then specifies the logins to migrate in Step 2. It is possible to migrate logins at any time, but performing a login migration while a database migration is in process could cause the mappings between logins and user roles to be confused.

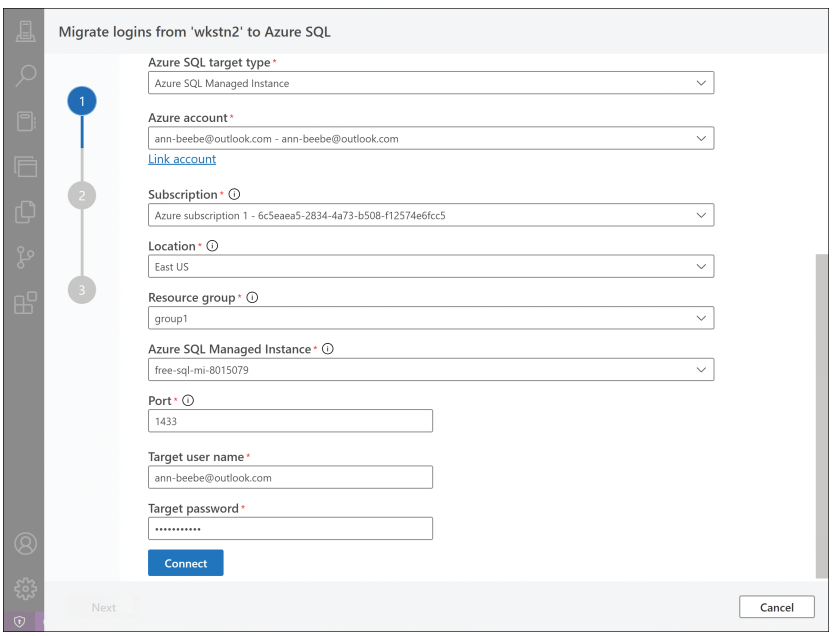


FIGURE 1-50 Step 1 of a login migration in Azure Data Studio with the Azure SQL migration extension installed

Configuring SQL security

Security is as important for an Azure SQL database installation as it is for on-premises servers. Once a database migration has been completed, administrators should confirm that the databases in the cloud are protected, using tools such as the following:

- **Firewall rules** Azure SQL Database supports two levels of firewall rules: server rules that specify the IP addresses permitted to access the server and database rules that control access to specific databases on the server.
- **Microsoft Defender for Azure SQL** A comprehensive SQL security product that supports all of the Azure SQL products (at an additional cost). Administrators can apply Defender at the subscription level, to protect all of the SQL databases in the tenancy, or at the individual resource level.
- **SQL vulnerability assessment** An Azure tool that uses Microsoft's best practices knowledge base to examine a SQL installation for security issues, possibly caused by misconfigured parameters or permissions, and assess their severity.

Troubleshoot a migration

SQL data migrations to the Azure cloud are complicated efforts, often involving a variety of tasks and tools. Problems can occur at any point, typically generating error messages that administrators might not be able to act on while the migration is underway.

As noted earlier, when the Azure Data Migration Service is performing a migration, the DMS instance page in the Azure portal displays the current migration status in real time. In addition to the normal Restoring status, there are other status indicators that can indicate potential problems with the migration.

For example, a migration status of Running with errors indicates that DMS has encountered one or more errors, has retried them, and is continuing the migration. Administrators should check the migration logs afterward to determine the exact nature of the errors and whether any action is needed. A migration status of Failed indicates that the migration must be repeated once the administrator investigates the error and resolves the problem.

Other error message types that might appear during a SQL data migration include the following:

- **Version number errors** Error messages that cite discrepancies between the version numbers of the source and target databases might require adjustments to compatibility level settings.
- **Database already exists** Errors of this type indicate that a database with a duplicate name already exists on the target installation. The solution might just be a matter of specifying a different target database name.
- **Permission errors** Errors citing denied permissions on the DMS account typically indicate that administrators must grant the specified permissions to the DMS service account.

Set up SQL Data Sync for Azure

SQL Data Sync is an Azure service that enables administrators to create bidirectional synchronization relationships between SQL Database and SQL Server installations. The synchronization uses a hub-and-spoke topology in which two or more SQL databases are part of a sync group. One SQL database is designated as the hub database, while one or more others are member databases. Synchronization traffic occurs only between the hub and member databases. In addition to the databases participating in the sync group, there is also a sync metadata database, which contains the metadata and logs for the Data Sync process.

Using SQL Data Sync, administrators can support hybrid applications by synchronizing an on-premises SQL Server database with an Azure SQL Database installation. Another possible application for Data Sync is to create synchronized copies of a database to split their operations: One copy can handle incoming queries while administrators use the other copy for data analytics.

To set up SQL Data Sync, an administrator opens an existing SQL Database installation in the Azure portal, chooses a database, and then selects Sync to other databases in the Data management menu. On the Sync to other databases page that appears, the administrator can then click +New sync group to open the Create Data Sync Group page, as shown in Figure 1-51.

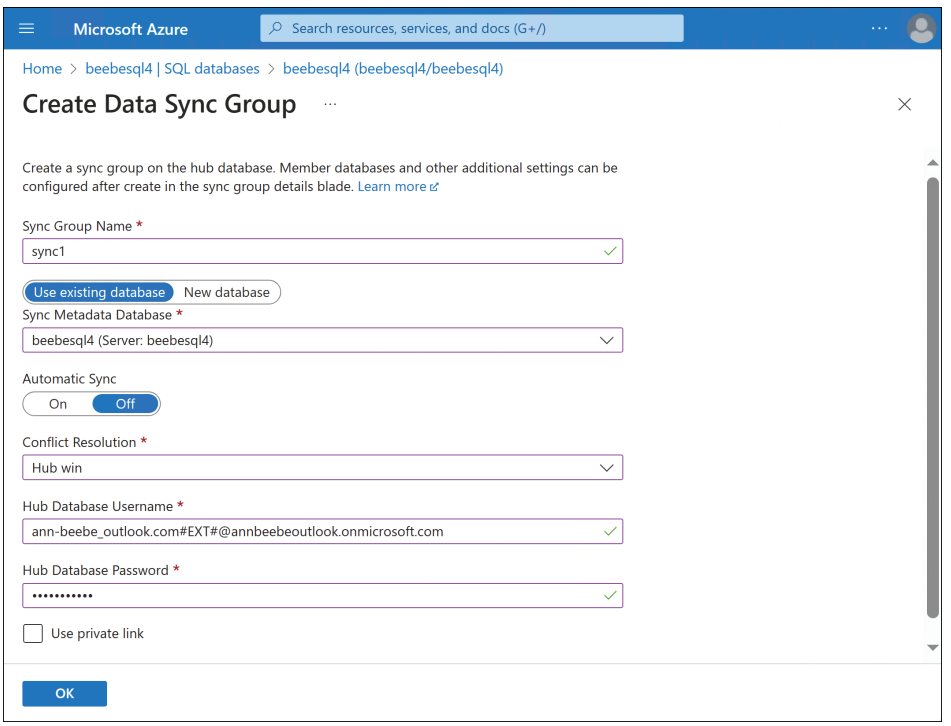


FIGURE 1-51 The Create Data Sync Group page in the Azure portal

After creating the sync group, the administrator can add member databases to it. The databases from other Azure SQL Database installations can be members of the sync group, as can

on-premises databases running on SQL Server. However, before a database from an on-premises SQL Server can be a member of the sync group, administrators must first install the Azure SQL Data Sync Agent on the server. This agent connects to the hub database in the cloud, enabling the administrator to add the on-premises databases to the sync group as members.

> ***NOTE*** **SQL DATA SYNC GROUPS**
>
> **SQL Data Sync only supports databases running on Azure SQL Database and SQL Server as members of a sync group. Azure SQL Managed Instance is not supported.**

Once the members are added to the sync group, administrators can select specific tables for synchronization on the sync group's page. When the SQL Data Sync setup is completed, the administrator can manually trigger the synchronization from the sync group page or schedule it to occur at a specific time.

Implement a migration to Azure

As this chapter has shown, there are several tools that administrators can use to migrate on-premises SQL databases to the Azure cloud. There are also several ways to launch and use those tools. When an Azure subscriber connects to the Azure portal, the Home page has a Create a resource icon that provides access to a long list of Azure services and Marketplace products.

On the Create a resource page, select the Migration category, as shown in Figure 1-52. This provides access to the Azure Data Migration Service and Azure Migrate tools covered earlier, as well as other tools that can aid in the migration process.

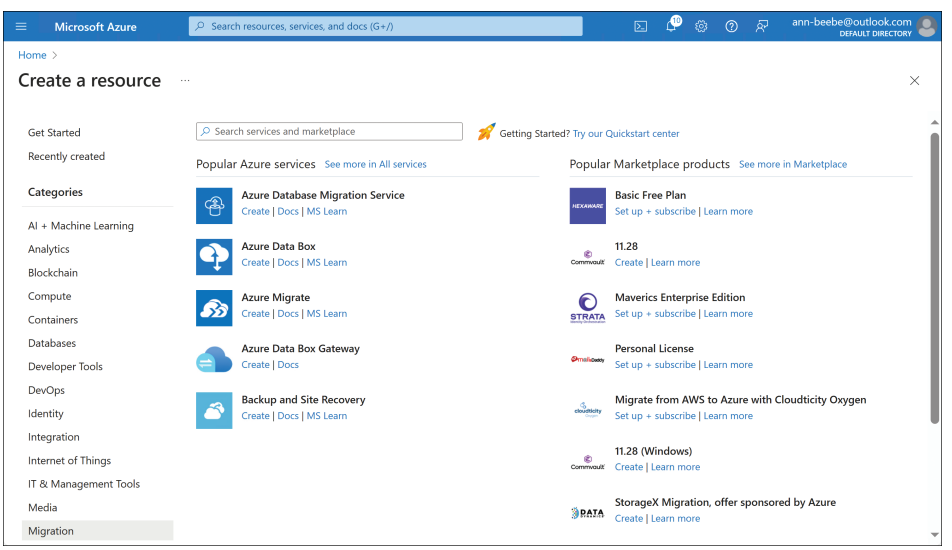


FIGURE 1-52 The Migration category in the Create a resource page in the Azure portal

Each icon includes links to the documentation for the tool, as well as a Create link that provides access to the tool itself, as follows:

- **Azure Database Migration Service** Provides access to the Select migration scenario and Database Migration Service page, as described in "Implement an online migration strategy," earlier in this chapter.
- **Azure Data Box** Provides offline transfers of large amounts of data to or from Azure storage. Data Box Disk supports up to a 35 TB USB transfer, Data Box up to an 80 TB SMB or NFS transfer, and Data Box Heavy up to a 1 petabyte (PB) transfer in two separate nodes using SMB or NFS. The Create link opens the Get Started with Data Box page, as shown in Figure 1-53.
- **Azure Migrate** Provides access to the Azure Migrate Get started page, as described in "Using Azure Migrate," earlier in this chapter.
- **Azure Data Box Gateway** A virtual appliance running on-premises that performs large online data transfers to or from Azure. The Create link opens the Create a Gateway resource page, on which the administrator specifies the Azure subscription, resource group, and region where the transferred data will be stored, and the instance name to be assigned to it.
- **Backup and Site Recovery** The Azure Site Recovery service provides disaster recovery and failover services for Azure virtual machines. The Create link opens the Create Recovery Services vault page.

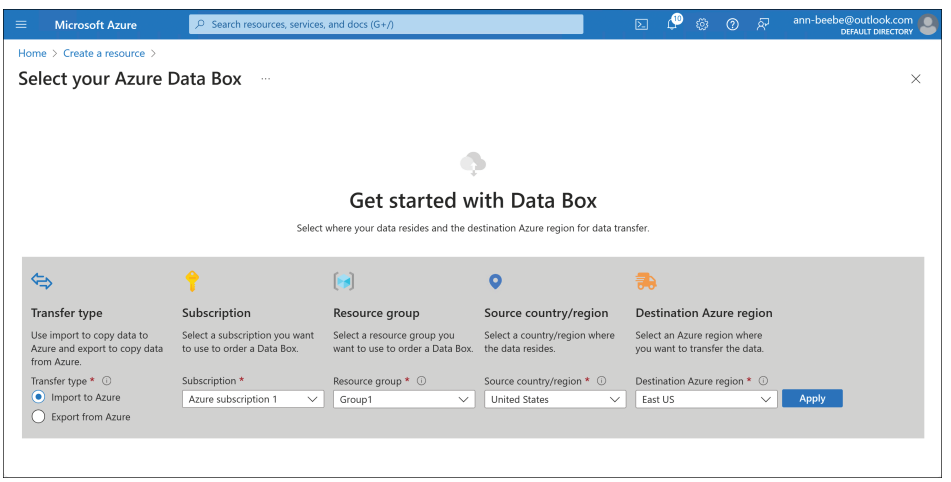


FIGURE 1-53 The Select your Azure Data Box page

Implement a migration between Azure SQL services

The most typical migration scenario is one in which an organization with on-premises SQL Server databases wants to migrate them to the Azure cloud, either as a hybrid solution or a

permanent one. However, there are also situations in which administrators want to migrate databases between Azure services. There are several ways to do that, depending on which services and how much data are involved.

Migrating a VM running SQL Server to a PaaS installation

From a migration standpoint, an Azure virtual machine running SQL Server is no different from an on-premises SQL Server installation, whether on a physical or virtual machine. Therefore, the process of migrating the SQL databases from the Azure VM to a SQL Database or SQL Managed Instance installation can use any of the tools discussed earlier for an on-premises to cloud migration, such as Azure Database Migration Services or Azure Migrate.

Migrating Azure to Azure

To migrate databases from one Azure SQL Database installation, such as to create failover databases in other geographic regions, for example, administrators can open a database in SQL Server Management Studio and export the data to another SQL Database installation using the SQL Server Import and Export Wizard, as shown in Figure 1-54.

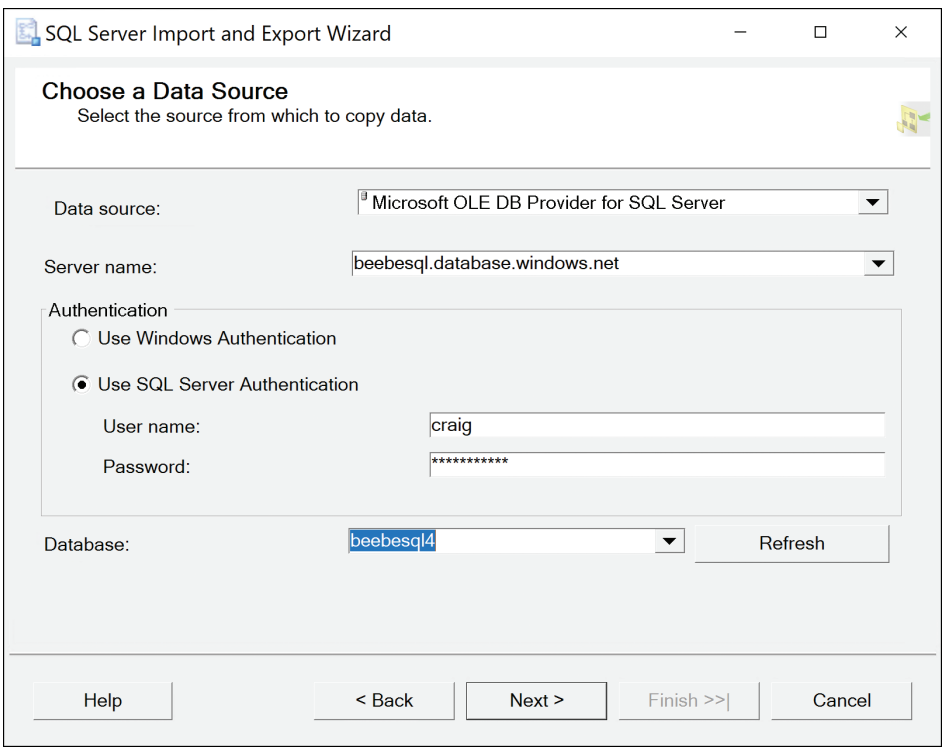


FIGURE 1-54 The SQL Server Import and Export wizard

Chapter summary

- Azure supports SQL database deployments with a variety of products, including Azure SQL Database, Azure SQL Managed Instance, and standard Azure virtual machines running SQL Server on them.
- Azure Resource Manager (ARM) supports the use of templates to deploy SQL databases in the Azure cloud. Administrators can also use ARM templates to automate SQL deployments using Azure CLI or PowerShell scripts.
- A hybrid SQL database solution creates a connection between an on-premises SQL Server installation (physical or virtual) and the Azure SQL products available in the cloud.
- Partitioning and sharding are techniques for splitting tables and storing the pieces in different databases to enhance query performance.
- Azure SQL Database supports service tiers using two purchasing models: database transaction unit (DTU) and vCore. There is also a separate compute tier with Provisioned and Serverless options. The Provisioned tier allocates compute resources and charges an hourly rate per vCore. The Serverless tier causes Azure to allocate compute capacity as needed and charge a compute cost per vCore second.
- Azure SQL Managed Instance supports three service tier options: General purpose; Next-gen General Purpose, with improved performance; and Business critical, with much better performance and three added replicas.
- When configuring an Azure VM running SQL Server, there is a dropdown containing dozens of images, which include various Windows and Linux operating system versions combined with SQL Server versions going back to 2012.
- When preparing for a database migration from on-premises servers to the Azure cloud, administrators should consider the compatibility levels of the various SQL products involved, the resources the databases will require, and whether the organization can tolerate the downtime needed for an offline migration. If not, an online migration might be necessary.
- Azure provides a variety of database assessment and migration tools, including Azure Migrate, Azure Database Migration Services (DMS), and Azure Data Studio with the Azure SQL Migration Extension installed.

Thought experiment

In this thought experiment, demonstrate your skills and knowledge of the topics covered in this chapter. You can find the answers in the section that follows.

Ralph is the SQL database administrator for an organization with multiple on-premises SQL Server installations. The company is expanding, but the datacenter is currently running at its physical capacity and cannot support any additional servers. It has been suggested that the

company expand its SQL implementation by creating a new database in the Azure cloud, but Ralph is concerned that he has no experience with cloud technology. With this in mind, consider the following questions about Ralph's possible SQL expansion to the cloud.

1. Which of the following initial Azure SQL deployment options should Ralph consider to expand the company's database implementation, while remaining as close as possible to his SQL administrative experience?
   A. Use Azure SQL Database to deploy a new database.
   B. Create an Azure virtual machine and install SQL Server on it.
   C. Create an Azure SQL Managed Instance.
2. Which of the following platforms would provide Ralph with the greatest freedom to configure a cloud server's operating system, IaaS or PaaS?
3. After deploying a database instance in the cloud, which of the following techniques can Ralph use to expand its capacity without creating additional servers?
   A. Scaling up
   B. Sharding
4. Ralph is considering a SQL database migration from an on-premises server to a cloud-based implementation. Which of the following tools actually performs the migration?
   A. Azure Migrate
   B. Azure Data Studio
   C. Azure Database Migration Service

Thought experiment answers

This section contains the solution to the thought experiment. Each answer explains why the answer choice is correct.

1. **B.** Creating an Azure virtual machine and installing SQL Server on it would be the closest to Ralph's experience with on-premises SQL servers.
2. The Infrastructure as a Service (IaaS) platform is the only one of the Azure SQL deployment options that provides the subscriber with full control over the operating system.
3. **A.** Scaling up is the process of adding resources to an existing server to enhance its performance. Sharding requires the storage of database partitions in different database instances (also known as scaling out).
4. **C.** Azure Migrate and Azure Data Studio are tools that can analyze and coordinate SQL migrations, but both of these tools use Azure Data Migration Service to perform the actual migration.

This page intentionally left blank

CHAPTER 2

Implement a secure environment

Some SQL database administrators are hesitant to store their data in the cloud because they believe it to be inherently insecure. Microsoft Azure provides subscribers with multiple options for deploying SQL in the cloud. One of the primary reasons for doing this is to provide differing approaches to security.

Skills covered in this chapter:

- 2.1: Configure database authentication and authorization
- 2.2: Implement security for data at rest and data in transit
- 2.3: Implement compliance controls for sensitive data

Skill 2.1: Configure database authentication and authorization

Databases can contain sensitive information of any type, so it is essential that the database management application and the environment in which it runs be securable. The first steps in implementing any sort of security infrastructure are verifying the identities of the users accessing the data and granting each user an appropriate degree of access. These processes are called authentication and authorization.

This skill covers how to:

- Configure authentication by using Active Directory and Microsoft Entra ID
- Create users from Microsoft Entra identities
- Configure security principals
- Configure database and object-level permissions using graphical tools
- Apply the principle of least privilege for all securables
- Troubleshoot authentication and authorization issues
- Manage authentication and authorization by using T-SQL

Configure authentication by using Active Directory and Microsoft Entra ID

On-premises SQL Server installations typically rely on Active Directory Directory Services (AD DS) for authentication and authorization. AD DS uses servers called domain controllers to provide identity and access management to SQL Server and other Windows services and applications.

Azure SQL Database and Azure SQL Managed Instance use Microsoft Entra ID for authentication and authorization. At one time, Entra ID was known as Azure Active Directory, but the fundamental differences between the two security infrastructures justified the name change. Entra ID, for example, supports conditional access and multifactor authentication, which AD DS does not.

When an on-premises SQL Server installation is configured to use Active Directory for authentication, the users' initial login to Windows also grants them their SQL database access. In the same way, Azure users' Entra ID logins can grant them access to a SQL Database or SQL Managed Instance installation.

SQL Server authentication

In addition to Active Directory on-premises and Azure Entra ID, all of the Azure SQL products support the native SQL Server authentication found in the Microsoft SQL Server product. This is an independent authentication mechanism with separate usernames and passwords that are stored in the SQL server's master database or in its user databases. Administrators decide which authentication model to use during the SQL installation.

The native SQL Server authentication is inherently less secure than either Active Directory or Entra ID. Both of the latter use complex infrastructures and specialized security protocols to provide authentication and authorization services without transmitting passwords and other sensitive traffic over the network in a readable form.

Active Directory uses a security protocol called Kerberos and the Lightweight Directory Access Protocol (LDAP) for authentication and authorization traffic, while Azure Entra ID uses HTTPS protocols such as Security Assertion Markup Language (SAML) and OpenID Connect. SQL Server authentication, by contrast, uses no security protocols and transmits usernames and passwords over the network in plain text, which leaves them open to compromise.

Configuring authentication

When a subscriber creates a new Azure SQL Database installation, they configure an authentication method for the new server by opting to use Entra ID authentication by itself, SQL authentication by itself, or both Entra and SQL authentication, as shown in Figure 2-1.

In addition, administrators can configure existing SQL Database and SQL Managed Instance installations to use Entra ID authentication only, thus disabling SQL authentication, by using the interface shown in Figure 2-2.

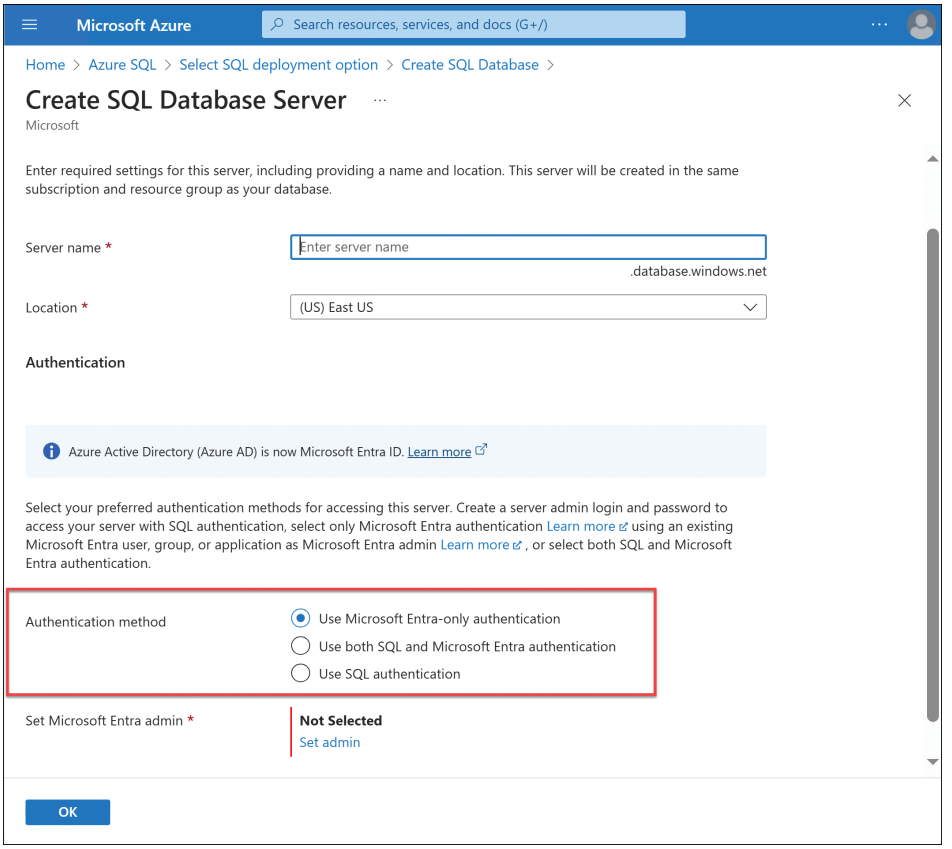


FIGURE 2-1 The Create SQL Database Server page in the Azure portal

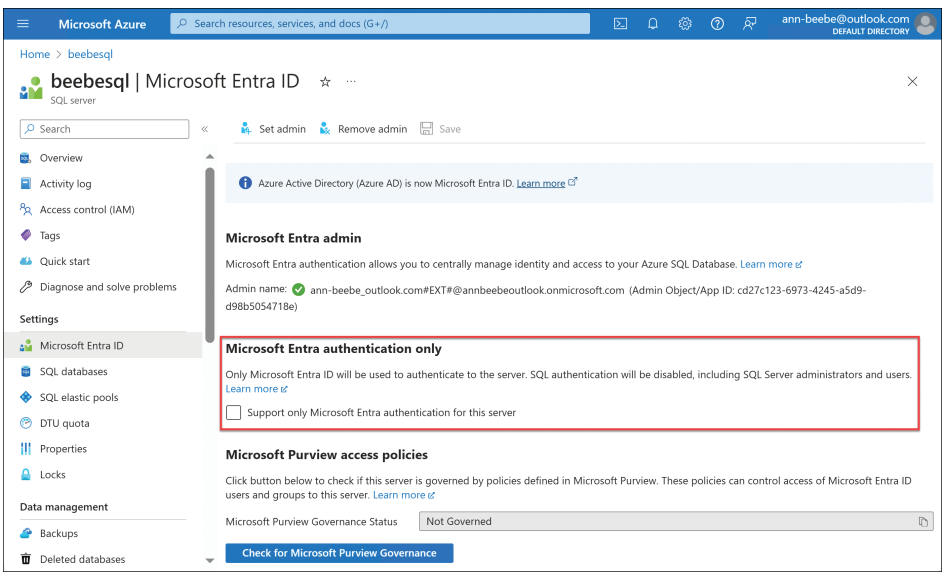


FIGURE 2-2 The Microsoft Entra ID page for a SQL Database server in the Azure portal

Entra identities are required to access Microsoft Azure, so subscribers already have them when they create a new SQL Database or SQL Managed Instance installation. Azure administrators can create and manage Entra identities through the Azure portal or by using the *az ad user* commands in Azure command-line interface (CLI) or the *AzureADUser* cmdlets in PowerShell.

For example, a CLI command to create a new Entra user appears like the following:

```
az ad user create --display-name ann-beebe --password Pa$$w0rd1234 --user-principal-name
ann-beebe@outlook.com
```

In a SQL Database installation, the Azure subscriber specifies an Entra identity to use as the administrator of the server, as shown in Figure 2-3. This account is granted administrative access to the server and all of the databases on that server. Using a group identity for the administrative account is a recommended practice because it eliminates the need to modify the account to accommodate personnel changes.

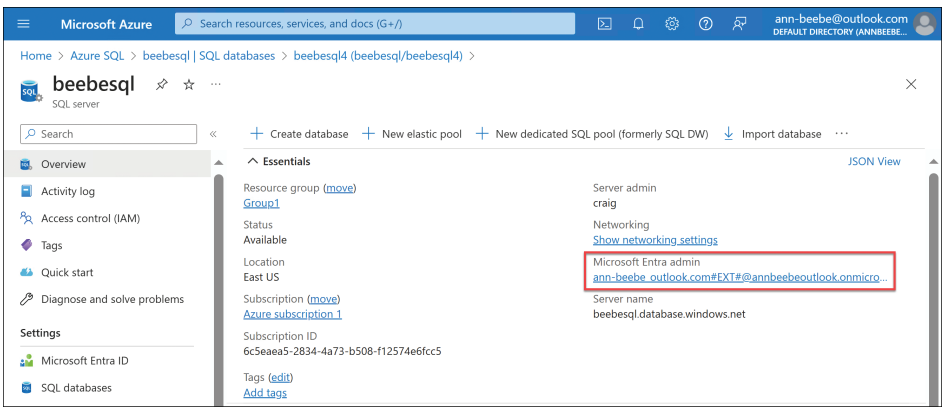


FIGURE 2-3 The Microsoft Entra admin setting on an Azure SQL Database Overview page in the Azure portal

In an on-premises SQL Server installation that uses Active Directory for authentication, the administrators must create the Active Directory infrastructure first by installing a Windows Server and assigning it the Active Directory Domain Services role to make it a domain controller. Domain controllers store user account information and provide authentication and authorization services for Windows operating systems and applications, including SQL Server.

When installing SQL Server, the Database Engine Configuration page of the Setup Wizard provides the options to use Windows authentication mode or Mixed Mode, as shown in Figure 2-4. Selecting Windows authentication mode disables the native SQL Server authentication mechanism, while Mixed Mode supports both Windows authentication and SQL Server authentication. There is no option to use SQL Server authentication only because users must always log in to Windows, even if they don't need access to SQL Server and its databases.

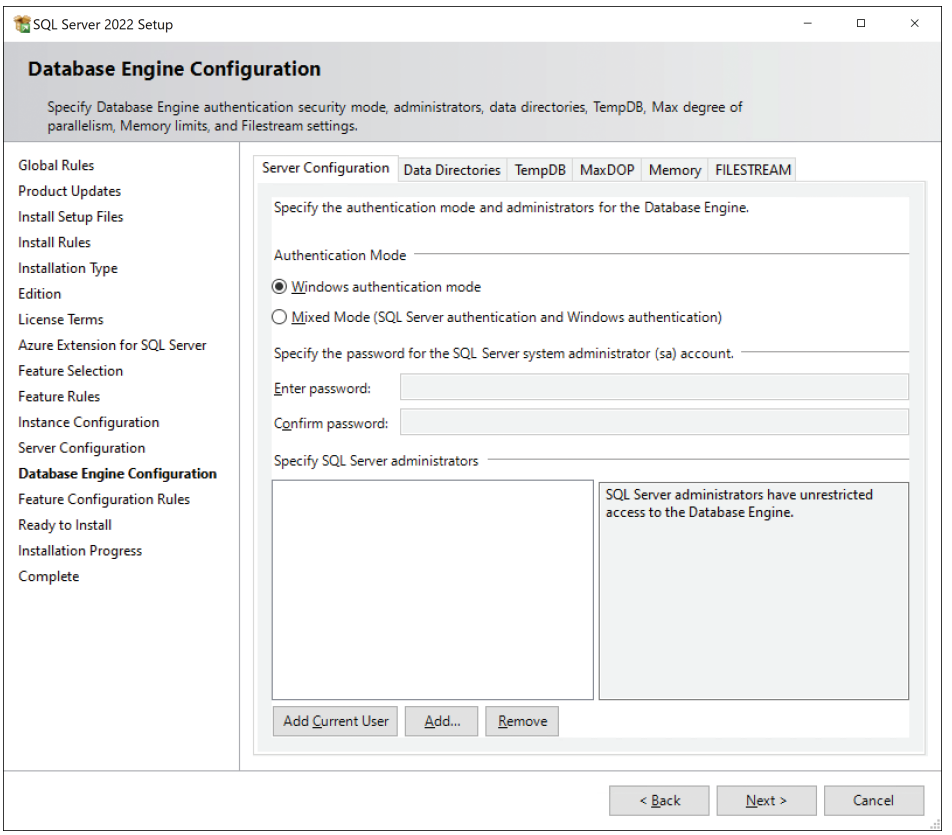


FIGURE 2-4 The Database Engine Configuration page in the SQL Server 2022 Setup Wizard

> ***NOTE*** **WINDOWS AUTHENTICATION AND ACTIVE DIRECTORY**
>
> **A Windows network is neither required to use Active Directory for authentication, nor does SQL Server require its use. Although it's not a recommended practice, it's possible to create user accounts on an individual Windows computer instead of on a central domain controller and authenticate Windows users locally. This is why the SQL Server setup and configuration interfaces refer to Windows authentication and not Active Directory. If SQL Server is installed on a Windows computer that is joined to an Active Directory domain, then adding a SQL Server administrator will prompt the administrator to choose an Active Directory user.**

Create users from Microsoft Entra identities

Microsoft Entra identities enable users to log on to Azure and access its services, but to create a login for SQL Database or SQL Managed Instance from an Entra identity, an administrator must

access the master database and use the T-SQL CREATE LOGIN command, with the following syntax:

```
USE master
GO
CREATE LOGIN [username@contoso.com] FROM EXTERNAL PROVIDER;
GO
```

The FROM EXTERNAL PROVIDER clause indicates that the command will create a SQL login from the credentials associated with the Entra identity *username* in the *contoso.com* domain.

To create a SQL login that does not directly match the Entra display name, the administrator can add the WITH OBJECT_ID clause to specify the object identifier for the Entra identity, as in the following example:

```
CREATE LOGIN [username@contoso.com] FROM EXTERNAL PROVIDER
  WITH OBJECT_ID='4466e2f8-0fea-4c61-a470-xxxxxxxxxxxx';
```

Configure security principals

In SQL, a *security principal* is any entity that can request access to SQL server resources and to which administrators can grant the permissions they need to do so. The resources accessed by the security principals are known as securables, and SQL divides the securables into three separate scopes: server, database, and schema. There are security principals in SQL at the server level and the database level.

Some of the terms applied to SQL security principals are used differently from those in other computing contexts. For example, in SQL, the terms "login" and "user" are not interchangeable. Both are security principals, but a *login* is an identity at the server level, whereas a *user* is a database-level identity. These are the two most important security principals in a SQL installation.

Creating logins

As shown in the previous section, administrators can use the T-SQL command CREATE LOGIN to create new logins at the server level. In addition to creating logins from Entra identities (using the FROM EXTERNAL PROVIDER argument), administrators can also create logins for SQL server authentication, as shown in Figure 2-5, by specifying a username and password, as in the following example:

```
CREATE LOGIN testuser WITH PASSWORD = 'Pa$$w0rd';
```

Creating users

At the database level, the user is the main security principal. A login grants an individual access to the server, but to actually work with a database on that server, the individual must have a database-level user identity, and the login must be mapped to the user.

In T-SQL, the CREATE USER command allows administrators to create database-level user identities, but it must be executed from the context of the database, as shown in Figure 2-6. As with creating logins, administrators can use CREATE USER to create user identities for Entra users by including the FROM EXTERNAL PROVIDER argument.

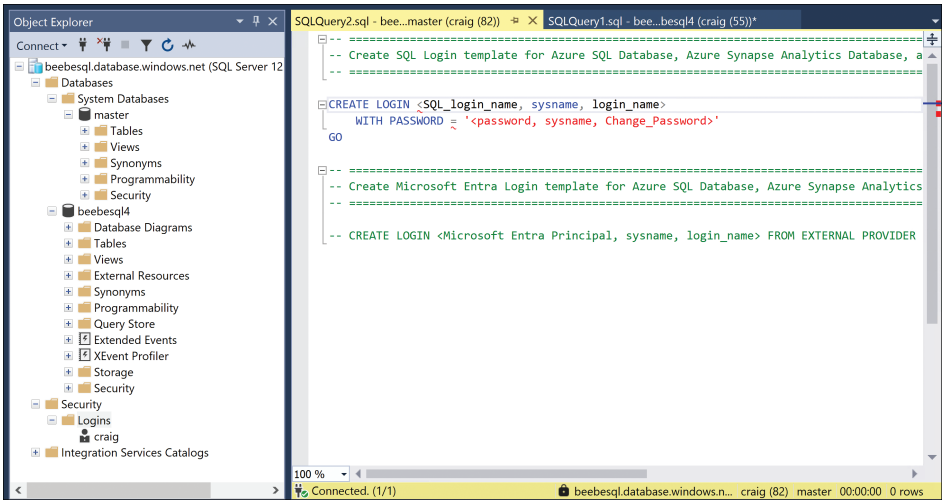


FIGURE 2-5 Creating logins in SQL Server Management Studio (SSMS)

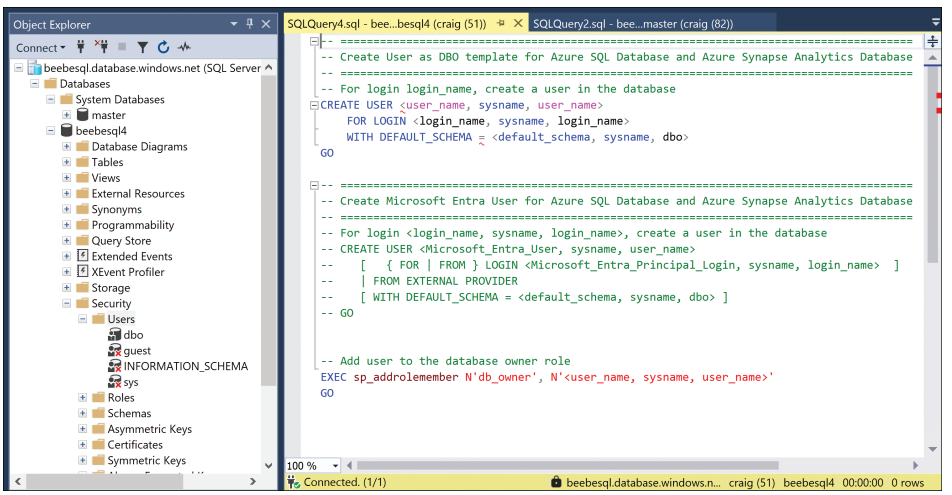


FIGURE 2-6 Creating users in SSMS

For an identity that already exists as a login, administrators can use the CREATE USER command with the FROM LOGIN argument to create a new user identity and map it to the specified login:

```
CREATE USER testuser FROM LOGIN testuser
```

Using roles

Role-based security is common in many network environments. Administrators create identities called roles and assign permissions to them. Functionally, roles operate like security groups. Assigning users to a role causes them to inherit all of the permissions granted to that role. This

simplifies the administrators' task of managing permissions; instead of having to assign permissions to many individual users, they can assign the permissions to a role once and then add users to or remove them from the role as needed.

Administrators can create SQL roles at the server or the database level, depending on which SQL product they are running. All of the Azure SQL products support database roles, but only SQL Server and Azure SQL Managed Instance support server roles. To grant access to objects in a database, administrators must use database roles.

The SQL products contain predefined roles that administrators cannot change, but they also support the creation of custom roles. Administrators can create a custom role by connecting to a SQL database and executing the CREATE ROLE command in T-SQL, as shown in Figure 2-7. Then, the GRANT command allocates permissions to the new role and the ALTER ROLE command adds an existing database user as a member of the role. A typical command sequence might appear as follows:

```
CREATE ROLE [dbusers]
GO
GRANT SELECT, EXECUTE ON SCHEMA::Production TO [dbusers]
GO
ALTER ROLE [dbusers] ADD MEMBER [testuser]
GO
```

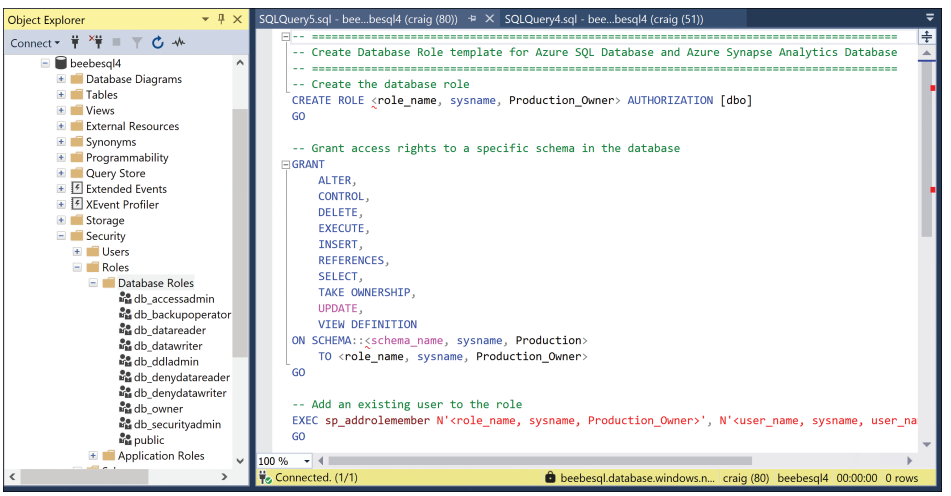


FIGURE 2-7 Creating database roles in SSMS

The predefined roles in a SQL database contain sets of permissions that allow administrators to delegate standard tasks to specific users. The roles built in to a SQL database (on any platform) are as follows:

- **db_accessadmin** Allows the creation of new users in the database
- **db_backupoperator** Allows users to execute database backups in SQL Server or SQL MI, but not SQL Database
- **db_datareader** Allows users to read all tables and views in the database

- **db_datawriter** Allows users to write to all tables and views in the database
- **db_ddladmin** Allows users to create or modify objects in the database
- **db_denydatareader** Denies users with other granted permissions the ability to read from the database
- **db_denydatawriter** Denies users with other granted permissions the ability to write to the database
- **db_securityadmin** Allows users to grant other users permissions for the database
- **db_owner** Provides users with administrative access to the database
- **public** Default role—initially with no permissions—to which all database users are automatically granted membership

The predefined server roles included in SQL Managed Instance and SQL Server (but not SQL Database) are as follows:

- **sysadmin** Allows full access to any activity on the server
- **serveradmin** Provides access to server-wide configuration settings
- **securityadmin** Provides administrative access to server logins and their properties, as well as server and database permissions
- **processadmin** Allows members to kill SQL Server processes
- **setupadmin** Allows members to add and remove linked servers using T-SQL
- **bulkadmin** Allows members to execute the BULK INSERT command
- **diskadmin** Allows members to manage backup devices in SQL Server
- **dbcreator** Allows members to create and alter any database
- **public** Default role with no permissions to which all server logins are automatically granted membership

Both servers and databases have a public role to which SQL adds logins and users automatically. The role has no permissions by default, but unlike the other predefined roles, it's possible for administrators to add and remove permissions from the public roles.

Azure SQL Database doesn't have the same built-in server roles as SQL Server or SQL Managed Instance because the SQL Database product does not have a server in the same sense. SQL Database has only a logical server, which exists mainly because certain applications expect to find it there. There are two server roles in SQL Database, however, as follows:

- **dbmanager** Allows members to create new databases
- **loginmanager** Allows members to create new logins at the server level

Configure database and object-level permissions using graphical tools

Users require permissions to access SQL resources, just as they need permissions for file systems and other protected elements. SQL supports a variety of tools for setting permissions, but

the data manipulation language (DML) for all of the SQL platforms, whether on-premises or in Azure, uses four basic permissions for the securable elements in a database, as follows:

- **SELECT** Allows the user to view the data in the securable element
- **INSERT** Allows the user to add data to the securable element
- **UPDATE** Allows the user to modify existing data in the securable element
- **DELETE** Allows the user to delete data from the securable element

There are other permissions that are specific to certain platforms. For example, Azure SQL Database and SQL Server also support the following permissions in addition to the basic four:

- **CONTROL** Grants the principal all rights to the securable
- **REFERENCES** Allows the principal to view the securable's foreign keys
- **TAKE OWNERSHIP** Allows the principal to assume ownership of the securable
- **VIEW CHANGE TRACKING** Allows the principal to view the securable's change tracking setting
- **VIEW DEFINITION** Allows the principal to view the definition of the securable

In addition, functions and procedures support the following permissions:

- **ALTER** Allows the principal to modify the definition of the securable
- **CONTROL** Grants the principal all rights to the securable
- **EXECUTE** Allows the principal to execute the securable
- **VIEW CHANGE TRACKING** Allows the principal to view the securable's change tracking setting
- **VIEW DEFINITION** Allows the principal to view the definition of the securable

Administrators can grant, revoke, or deny these permissions to logins, users, roles, and other security principals. Granting a permission allows the principal access to the securable, while denying a permission explicitly prevents that access. In the case of permission conflicts, such as when a user inherits permissions from multiple sources, deny permissions always supersede the granted ones. Permissions are also cumulative, so that if a user receives permissions from multiple sources, the user's effective permissions reflect the combination of all those sources.

Configuring permissions with T-SQL

The primary tools that administrators use to configure permissions for SQL databases are T-SQL commands and SQL Server Management Studio (SSMS). The basic syntax for the T-SQL permission commands is as follows:

```
AUTHORIZATION PERMISSION ON SECURABLE::NAME TO PRINCIPAL;
```

- **AUTHORIZATION** Specifies the verb GRANT, REVOKE, or DENY
- **PERMISSION** Specifies the permission to be granted, revoked, or denied
- **SECURABLE::NAME** Specifies the type and name of the securable to which the permissions will be applied
- **PRINCIPAL** Specifies the name of the security principal to which the permissions will be applied

For example, the following command grants the select and update permissions for a database called testdb to a user called testuser.

```
GRANT SELECT, UPDATE ON DATABASE:testdb TO testuser
```

The PERMISSION argument can be any of the permissions listed earlier, with multiple permissions separated by commas. The SECURABLE can be a server or database or any object on a server or in a database, such as a table or a view. When the context of a T-SQL script is unambiguous, the SECURABLE argument can be omitted and the target object referenced by name only. The PRINCIPAL can be any login or user, but many administrators prefer to assign permissions to roles instead and then assign the roles to logins or users.

Administrators should also understand the difference between revoking permissions and denying them. Revoking a permission is essentially removing it, so that it has no further influence on the securable. For example, the following command revokes all permissions to the testdb database from the user testuser:

```
REVOKE SELECT, INSERT, UPDATE, DELETE ON DATABASE::testdb TO testuser
```

As a result of the command, there are no permissions to testdb granted to the testuser user account. However, if testuser receives permissions from a role assignment, those permissions would still be applicable.

Alternatively, using the DENY command instead of REVOKE explicitly forbids the user from working with the securable, as in the following example.

```
DENY SELECT, INSERT, UPDATE, DELETE ON DATABASE::testdb TO testuser
```

While the REVOKE command removes the existing permission assignments, the DENY command creates new permission assignments of a different type. As a result of this command, testuser is expressly forbidden from accessing the testdb database. Even if testuser receives granted permissions from another source, these deny permissions supersede them.

Configuring permissions with SSMS

SSMS provides a graphical interface that administrators can use to configure permissions for databases and objects. For example, administrators can connect to a SQL server, browse to a database object, such as a table, open its Properties dialog, and select the Permissions page, to display the interface shown in Figure 2-8.

In this SSMS interface, administrators can add security principals to the Users or roles box and configure their permissions using checkboxes that represent the following authorizations:

- **GRANT** Indicates that the security principal will receive the selected permission
- **WITH GRANT** Indicates that the security principal will receive the selected permission and the ability to grant that permission to other principals
- **DENY** Indicates that the security principal will be prevented from exercising the selected permission, even if they are granted the same permission by other means

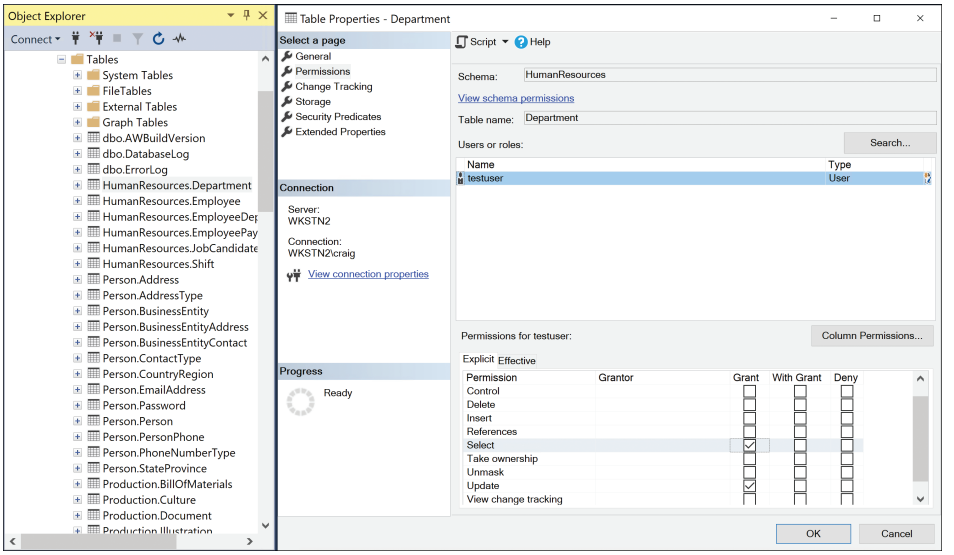


FIGURE 2-8 Configuring table permissions by user in SSMS

Administrators can also approach this task from the other direction by selecting a user in a database's Security\Users folder, opening its Properties, adding entries to its Securables box, such as a table, and configuring their permissions, as shown in Figure 2-9.

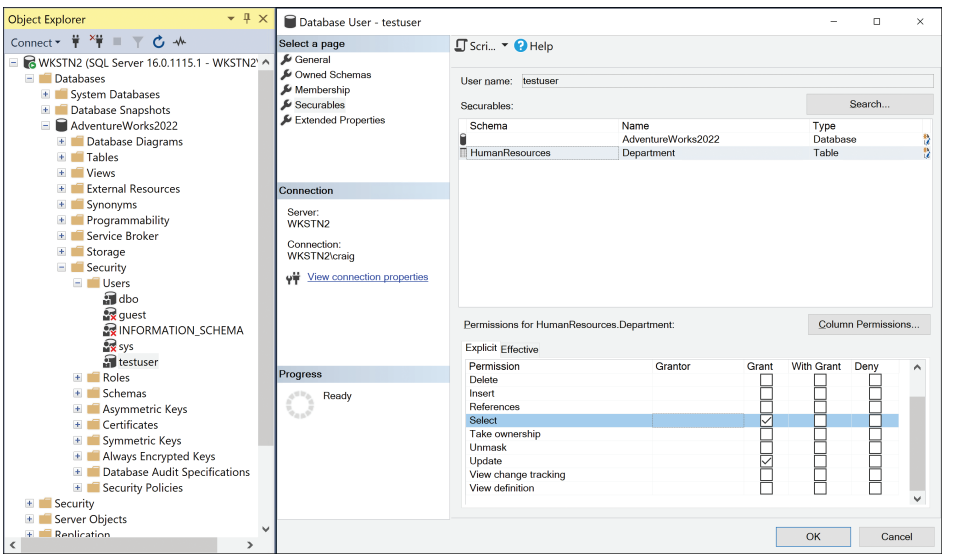


FIGURE 2-9 Configuring user permissions by securable in SSMS

Apply the principle of least privilege for all securables

In SQL security, as in all network security, the principle of least privilege states that users, applications, and other processes should receive only the privileges to securables that they need to

perform their allotted tasks, and no more. The objective is for individuals to receive the access they need without unnecessarily endangering securables.

Obviously, the simplest (and least secure) security solution to implement is to give everyone access to everything, and the most secure solution is to give no one access to anything. The perfect solution lies somewhere between these two extremes.

Role-based access control is one way for administrators to implement the principle of least privilege easily. Granting permissions to individual users is not only more work for administrators than using roles but administrators also tend to lose track of the permission assignments they have made to specific user accounts. Using roles, administrators can assign permissions more consistently.

The predefined roles in the Azure SQL products enable administrators to assess the needs of their users and apply privileges appropriate to their tasks. Azure even warns administrators when applying highly privileged roles. Adding the Owner role to a user, for example, causes Azure to display a message questioning whether a less-privileged role would be adequate, as shown in Figure 2-10.

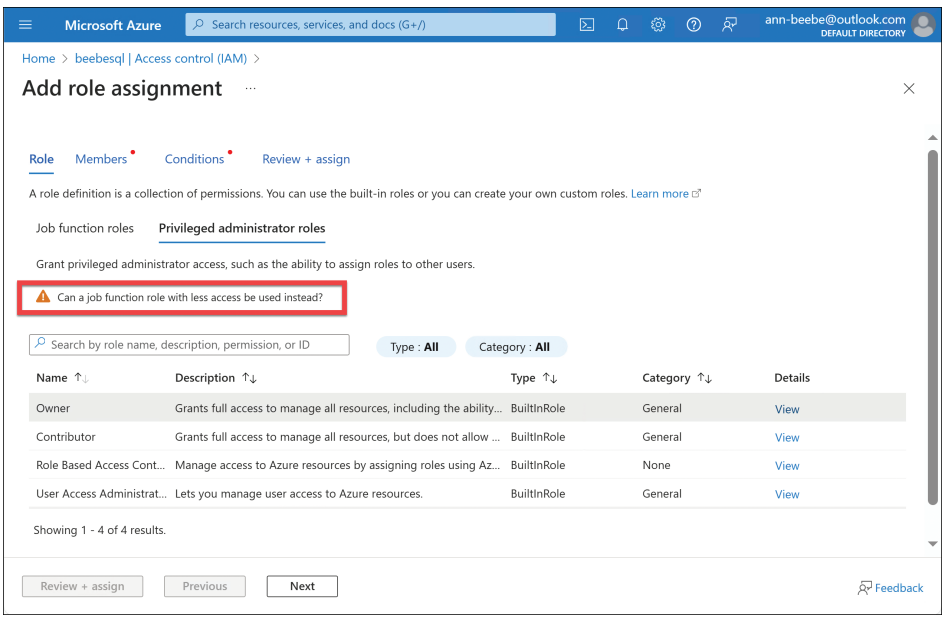


FIGURE 2-10 Adding privileged administrator role assignments generates a warning in Azure

Another factor to consider when applying least privilege is selecting the correct securables to which users need permission to perform a specific task. For example, in the case of an application that relies on stored procedures, users only need the EXECUTE permission for those procedures; they do not need permissions for the underlying tables accessed by the procedures.

Troubleshoot authentication and authorization issues

Authentication and authorization are processes that run frequently on many SQL installations, and there are many possible reasons why those processes can fail. Usually, the most common causes of authentication failures are administrative, such as incorrect, lost, or forgotten passwords. For installations that use multifactor authentication (MFA), the administrative issues can be compounded as users grow accustomed to the new procedures. However, when administrative factors can be eliminated as a potential cause of authentication failures, it's time to troubleshoot other possible causes.

Transient issues in communication and Azure processing can affect authentication and authorization, usually delaying them for only a few seconds. Communication interruptions, such as momentary Internet outages, are always a possibility that can occur at any point between the user and the Azure cloud service.

When a "login failed" error message occurs repeatedly, however, administrators can begin troubleshooting the problem by confirming that the login is not disabled. To do this, administrators can open the *sys.sql_logins* catalog view in the master database and check the value of the *is_disabled* column, as shown in Figure 2-11. A value of False indicates that the login is enabled.

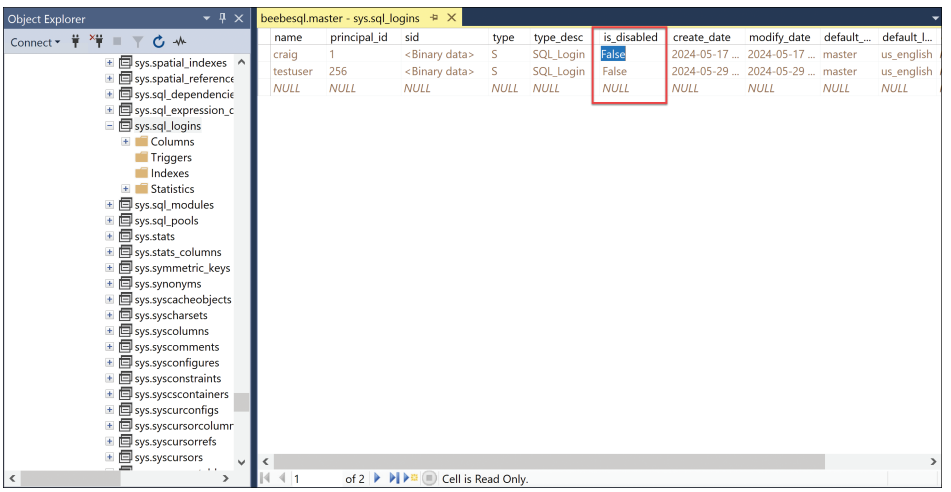


FIGURE 2-11 Examining the *sys.sql_logins* view in the master database in SSMS

If the value of the *is_disabled* column is True, indicating that the login is disabled, the administrator can enable it using a T-SQL command like the following:

```
ALTER LOGIN testuser ENABLE;
```

If the login does not exist in the *sys.sql_logins* view at all, the administrator can create it using the CREATE LOGIN command and then create a corresponding database user with the CREATE USER command. If the user already exists but lacks the necessary permissions, administrators can assign a role to the user with the ALTER ROLE command or use the GRANT command to assign permissions to the individual user.

The Azure services themselves can also incur delays. For example, the PaaS products, such as Azure SQL Database and Azure SQL Managed Instance, can scale according to changes in their workloads. Processes such as server reconfiguration or virtual machine migration can cause authentication processes to fail or be delayed temporarily, generating errors such as the following:

- Cannot open database "%.*ls" requested by the login. The login failed.
- The service is currently busy. Retry the request after 10 seconds. Incident ID: %ls. Code: %d.
- Cannot process request. Too many operations in progress for subscription "%ld".

Another possible cause of authentication failures is that the database might have reached the limit of its Azure resources. Both storage and compute resources are subject to limitations, and the service can stop when those limits are exceeded, resulting in error messages such as the following:

> Cannot process request. Not enough resources to process request. The service is currently busy. Please retry the request later.

User error is also a possible source of authentication problems, due to the use of an incorrect connection string. This is most common in a newly created or updated database that requires users to alter their connection process. To confirm that the connection string is correct, each PaaS SQL database has a Connection strings page that displays the connections for the various Microsoft Entra and SQL authentication mechanisms, as shown in Figure 2-12.

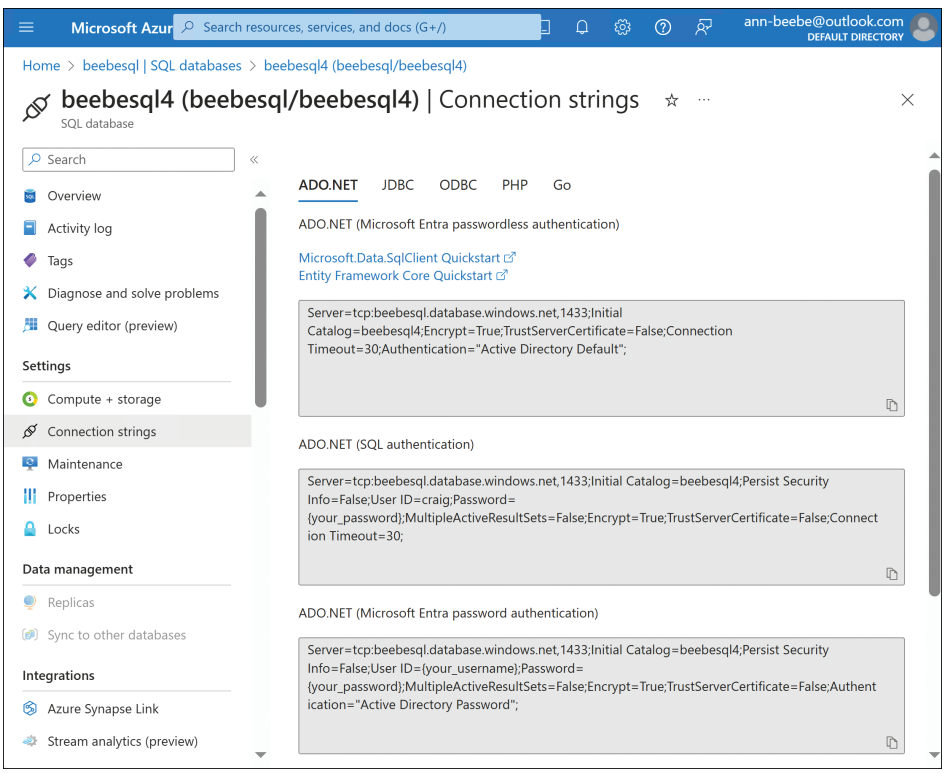


FIGURE 2-12 The Connection strings page for a SQL database in the Azure portal

Manage authentication and authorization by using T-SQL

As noted earlier in this chapter, administrators are required to select an authentication mode when installing a SQL product in Azure. This setting determines whether SQL will use its own internal (and less secure) SQL authentication or Microsoft Entra.

Administrators can change the authentication mode at any time using the graphical interface on the SQL server's Microsoft Entra ID page in the Azure portal. In SSMS, administrators can open a SQL server's Properties dialog and select the Security page to display the Server authentication settings shown in Figure 2-13. Administrators must restart the SQL server after modifying the settings.

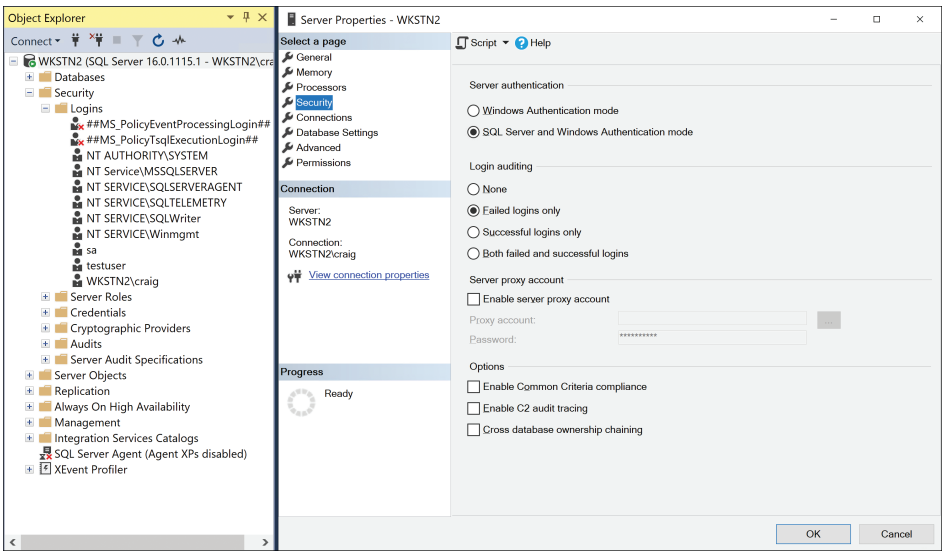


FIGURE 2-13 The Security page in the Server Properties dialog in SSMS

Finally, it's possible to manage the SQL authentication and authorization settings using T-SQL commands. Enabling SQL server authentication during installation creates and activates a login called *sa*. Selecting Microsoft Entra or Windows authentication during the installation disables the *sa* login. If an administrator later decides to change the authentication mode to SQL authentication or mixed mode, the *sa* login remains disabled.

To enable the *sa* login and set a password for it, connect to the master database and use the following T-SQL commands:

```
ALTER LOGIN sa ENABLE;
ALTER LOGIN sa WITH PASSWORD = 'Pa$$w0rd';
```

To change an installation from SQL authentication only to Windows authentication or mixed mode, it is necessary to modify a Windows registry setting. To do this, use a T-SQL command like the following:

```
EXEC xp_instance_regwrite 'HKEY_LOCAL_MACHINE',
     'Software\Microsoft\MSSQLServer\MSSQLServer',
     'LoginMode', REG_DWORD, 1;
```

This command changes the authentication mode to Windows authentication only. To change to mixed mode, use the following command:

```
EXEC xp_instance_regwrite 'HKEY_LOCAL_MACHINE',
     'Software\Microsoft\MSSQLServer\MSSQLServer',
     'LoginMode', REG_DWORD, 2;
```

After changing the mode, the administrator might want to disable the *sa* login, which is a potential avenue of attack. The command to do this is as follows:

```
ALTER LOGIN sa DISABLE;
```

Skill 2.2: Implement security for data at rest and data in transit

The process of securing an organization's data must acknowledge the three possible states of data because they present administrators with different security challenges. As the names imply, *data at rest* refers to the state of data while it's motionless, stored safely away on an internal or external device. *Data in transit* (or data in motion) refers to the state of data while it's being transmitted from one location to another, either on a private network or the Internet. The third data state, *data in use*, refers to data that's currently loaded into RAM or CPU cache memory. Securing data typically means encrypting it, but these states have differing levels of vulnerability. Therefore, SQL provides different means of encrypting sensitive data in its various states.

This skill covers how to:

- Implement transparent data encryption (TDE)
- Implement object-level encryption
- Configure server- and database-level firewall rules
- Implement Always Encrypted
- Implement Always Encrypted with VBS enclaves
- Configure secure access
- Configure Transport Layer Security (TLS)

Implement transparent data encryption (TDE)

Transparent data encryption (TDE) is the default encryption method for data at rest in SQL Server, Azure SQL Database, and Azure SQL Managed Instance. TDE is enabled by default in all newly created Azure SQL and SQL Server databases.

Index

A

access
  principle of least privilege, 76–77
  secure, 89–90
Active Directory, 66. *See also* authentication
  Kerberos, 66
  SQL Server authentication, 68
active geo-replication, 202
Add Counters dialog, 112–113
Add masking rule dialog, 102
Advanced Threat Protection, 107
AG (Availability Group), 189
AI (artificial intelligence), Intelligent Insights, 140–141
alerts, 182
  automation account, 182–184
  job, 161, 162–164
ALTER DATABASE command, 101, 126, 148, 152
ALTER LOGIN command, 78
ALTER ROLE command, 72
Always Encrypted, 84
  implementing object-level encryption, 85–88
  implementing with VBS enclaves, 88–89
ARM (Azure Resource Manager), 17, 169–171, 172–173
auditing, 26, 96–98
authentication, 65
  Azure SQL Database, Entra ID, 66–68
  managing with T-SQL, 80–81
  SQL Server, 26
    Active Directory, 68
    Mixed Mode, 68–69
    native, 66
    Windows, 68–69
  troubleshooting, 78–79
authorization, 65
  managing with T-SQL, 80–81
  troubleshooting, 78–79
automated deployment, 16–17
automated patching, 10–11, 18–19
Automatic Tuning, 136–137, 148
automating SQL deployments, 169
  runbooks, 177–181, 185
  using ARM templates, 169–171
  using Azure CLI and PowerShell, 173–174
  using Bicep files, 171–173
automation account
  alerts, 182–184
  creating, 178
availability groups, 204
availability sets, 191
availability zones, 191
az deployment group create command, 173
az sql server audit-policy update command, 97
az ts create command, 170
az vm extension set command, 119
AzSqlElasticJobTargetGroup cmdlet, 177
Azure Arc, 21
Azure Automation, runbooks, 177–181
Azure CLI
  az deployment group create command, 173
  az sql command group, 174
  az sql server audit-policy update command, 97
  az ts create command, 170
Azure Data Migration Assistant (DMA). *See* DMA (Azure Data Migration Assistant)
Azure Data Studio, 46, 51–53
Azure Hybrid Benefit, 4
Azure Marketplace, SQL templates, 4–9
Azure Migrate, 43–44
Azure Purview, 103–104
Azure site recovery, 191
Azure SQL, 17
  auditing, 96–98
  automated deployment, 16–17
  compatibility with on-premises equipment, 41–42

Azure SQL, *continued*
creating a database, 12–14
creating a Managed Instance, 15–16
extended events, 122–124
Ledger feature, 104–106
selecting a database offering, 21
database sharding solutions, 28–30
IaaS vs. PaaS, 21–22
PaaS purchasing models, 22–24
PaaS SQL options, 22
security considerations, 25–26
table partitioning solutions, 26–28
selecting a deployment option, 11
Azure SQL Database
authentication, Entra ID, 66–68
Automatic Tuning, 136–137, 148
Compute, 32
configuring for scale and performance, 31
data classification, 93–96
elastic pools, 31
Intelligent Insights, 140–141
monitoring, 113–114
Performance overview page, 117–119
purchasing model
DTU, 31
vCore, 31–32
security, 57
server roles, 73
service tier, 31–32
SQL Data Sync, 100
troubleshooting blockages, 131–132
Azure SQL Managed Instance, 33
monitoring, 113–114
selecting a service tier, 33–34
selecting compute hardware, 34–35
SQL Server Agent. *See* SQL Server Agent
Azure Update Manager, 10–11
Azure VM
always on failover cluster instances, 209–210
configuring SQL Server, 35
hardware configurations, 36
hybrid cloud deployment, 19
scaling, 36
SQL Server Agent. *See* SQL Server Agent
SQL Server deployment, 3
SQL Server on Windows 2019 template, 5–9
using the SQL Server IaaS Agent Extension, 9–10
SQL Server, monitoring, 112–113

B

backup and restore, 192–193
from cloud storage, 199–201
database backup, 193–194
database restore, 195–196
database restore to a point in time, 196–198
log shipping, 210–212
long-term backup retention, 197–199
using T-SQL, 198–199
BACPAC, 46
bandwidth, hybrid SQL Server, 19
baseline, 112, 115
Bicep files, 171–173
blockages, troubleshooting
in Azure SQL Database, 131–132
in SQL Server, 132–133
bulk copy, 46

C

CDC (change data capture), enabling, 98–99
change tracking, 100–101
checklist testing, 191
classification, data, 93–96
CLI commands. *See* Azure CLI
cloud. *See also* IaaS (Infrastructure as a Service); PaaS (Platform as a Service)
backup and restore, 199–201
IaaS (Infrastructure as a Service), 2
applying patches and updates, 18–19
maintenance, 3
SQL Server deployment, 3
SQL Server IaaS Agent Extension, 9–10
PaaS (Platform as a Service), 2
resources, orchestration, 17
cmdlets
AzSqlElasticJobTargetGroup, 177
New-SqlColumnEncryptionSettings, 87–88
Set-AzVMExtension, 119
code
Bicep, 173
infrastructure as, 169
JSON, 172–173
modules, 178
routines, 120–121
Columnstore compression, 39
Columnstore_archive compression, 39

command/s
Azure CLI
az deployment group create, 173
az sql command group, 174
az sql server audit-policy update, 97
az ts create, 170
az vm extension set, 119
PowerShell
New-AzResourceGroupDeployment, 173–174
New-AzSqlServer, 174
T-SQL
ALTER DATABASE, 101, 126, 148, 152
ALTER LOGIN, 78
ALTER ROLE, 72
backup and restore, 198–199
CDC (change data capture), 98–99
CREATE LOGIN, 69–70
CREATE PARTITION, 38
CREATE USER, 70–71
DBCC CHECKDB, 146–147
DENY, 75
permission, 74–75
REVOKE, 75
compatibility, on-premises and Azure SQL products, 41–42
compression ratio, 39
compute, 151–152
hardware selection
Azure SQL Database, 32
Azure SQL Managed Instance, 34–35
scaling, 151–152
Compute tier, 24, 32
Configure ledger dialog, 105
Configure Overall Resource Consumption dialog, 128–130
connection strings, 120–121
connectivity, hybrid SQL Server, 19–20
counters, performance, 113, 115
Create a virtual machine dialog, 6–9
Create an alert rule dialog, 117–118
Create Azure SQL Managed Instance dialog, 15–16
Create Data Sync Group dialog, 100
Create new profile dialog, 120
CREATE PARTITION command, 38
Create SQL Database dialog, 12–14, 89
CREATE USER command, 70–71
Create virtual machine dialog, 18–19, 36–37
creating
automation account, 178
Azure SQL database, 12–14
Azure SQL Managed Instance, 15–16
elastic jobs, 176–177
job schedule, 159
SQL login, 69–71
SQL user, 70–71

D

data at rest, 81
data classification, 93–96
data compression, 39–40
data in transit, 81
data migration, 40
offline, 51–53
requirements, 41
assessing compatibility levels, 41–42
assessing downtime, 42–43
resource, 42
security, 43
troubleshooting, 57
database
Automatic Tuning, 148
backup, 193–194
integrity checks, 146–147
-level firewall rules, 84–85
restore, 195–198
-scoped configuration, 150–151
storage, 30
table
compression ratio, 39
data compression, 39–40
partitioning, 38
Database Properties dialog, 152–154
database watcher, 121
DBCC CHECKDB command, 146–147
declarative model, 17, 169
DEK (data encryption key), 83
denying, permissions, 75
dependencies, resource, 17
deployment/s
automated, 16–17
Azure SQL, 11
errors, 174–175
hybrid, patching, 18–19
hybrid SQL Server, 19

deployment/s, *continued*
adding bandwidth, 19
adding connectivity, 19–20
adding off-site backups, 20
Azure Arc, 21
fault tolerance, 20
orchestration, 17
SQL Server on Azure VM, 3
SQL Server on Windows 2019 template, 5–9
using the SQL Server IaaS Agent Extension, 9–10
deterministic encryption, 86
dialog
Add Counters, 112–113
Add masking rule, 102
Configure ledger, 105
Configure Overall Resource Consumption, 128–130
Create an alert rule, 117–118
Create Azure SQL Managed Instance, 15–16
Create Data Sync Group, 100
Create new profile, 120
Create SQL Database, 12–14, 89
Create virtual machine, 6–9, 18–19, 36–37
Database Properties, 152–154
Manage Schedules, 159
New Credential, 179
New Statistics, 145
Properties, 75, 80, 82, 100–101
Reorganize Indexes, 142–144
Restore, 197
Select Columns, 146
SQL Server Agent Properties, 158
SQL Server Management Studio, 39–40
differential backup, 194
DMA (Azure Data Migration Assistant), 44, 45–46
DMF (dynamic management function), 132
DMO (dynamic management object), 132
DMS (Azure Data Migration Service), implementing an online migration, 47–49
DMVs (dynamic management views), 132, 133–136
domain controllers, 66
downtime
data migration, 42–43
RTO (Recovery Time Objective), 188–189
DP-300 exam
objectives, 219–221
updates, 217–218
DR (disaster recovery), 187. *See also* backup and restore; HA (high availability)
Azure-specific solutions, 191
for hybrid deployments, 190–191
hybrid SQL infrastructure, 20
monitoring, 211–212
PaaS (Platform as a Service), 190
RPO (Recovery Point Objective), 188–189
RTO (Recovery Time Objective), 188–189
testing procedure, 191–192
troubleshooting, 212–213
DTU-based purchasing model, 23, 31
dynamic data masking, 102

E

elastic jobs, creating, 176–177
elastic pool, 31, 135
encryption, 26
deterministic, 86
object-level, 83–84, 85–88
randomized, 86
transparent data, 81–83
Entra, 68–70
Entra ID, Azure SQL Database authentication, 66–68
errors, deployment, 174–175
execution plans, 139–140
ExpressRoute tunnel, 19–20
extended events, 122–124

F-G

failover groups, 206–207
fault tolerance, 20, 29
FCI (Failover Cluster Instance), 189, 209–210
firewall, rules, 84–85
fragmentation, 142–144
full backup, 194
full interruption testing, 192

H

HA (high availability), 187. *See also* backup and restore
active geo-replication, 202
availability groups, 204
Azure-specific solutions, 191
failover groups, 206–207
for hybrid deployments, 190–191
monitoring, 211–212

PaaS (Platform as a Service), 190
SQL Server, 189–190
testing procedure, 191–192
troubleshooting, 212–213
hardware
configurations, Azure VM, 36
scaling, 26–27
hash-based sharding, 29
hints, query, 138
horizontal partitioning, 28
hybrid deployment
HA/DR solutions, 190–191
patching, 18–19
SQL Server, 19
adding bandwidth, 19
adding connectivity, 19–20
adding off-site backups, 20
Azure Arc, 21
fault tolerance, 20
Hyperscale, 32

I

IaaS (Infrastructure as a Service), 2
applying patches and updates, 18–19
maintenance, 3
versus PaaS, 21–22
SQL Server deployment, 3
SQL Server IaaS Agent Extension, 9–10
identities, Entra, 69–70
imperative model, 17
Import/Export Service, 46
index
maintenance, 142–144
performance, 136–138
infrastructure as code, 169
installation, vCore, 24
integrity checks, database, 146–147
Intelligent Insights, 140–141
interpreting, metrics, 116
IQP (intelligent query processing), 152. *See also* query/ies

J-K

job/s
alerts, 161, 162–164
elastic, 176–177
notifications, 161
runbooks, 177–181
schedule, creating, 159
troubleshooting, 164–168
JSON (JavaScript Object Notation), 172–173
Kerberos, 66

L

LDAP (Lightweight Directory Access Protocol), 66
Ledger, 104–106
licensing, SQL Server, 3
"lift and shift", 43
log shipping, 190, 210–212
login, 70
migration, 56
sa, 80–81
SQL, creating, 69–71
troubleshooting, 78–79
long-term backup retention, 197–199
lookup-based sharding, 29

M

maintenance. *See also* troubleshooting
IaaS VM, 3
index, 142–144
statistics, 144–146
Manage Schedules dialog, 159
Managed Instance, Azure SQL, creating, 15–16
masking data, 102
metrics
interpreting, 116
operational performance baseline, 112
sources, 115
Microsoft Defender for Cloud, SQL Server protection, 11
Microsoft Defender for SQL, 26, 107
Microsoft Entra. *See* Entra
Microsoft Entra ID. *See* Entra ID
migration. *See also* data migration
to Azure, 59–60
between Azure SQL services, 60–61
DMS (Azure Data Migration Service), 45–46
"lift and shift", 43
login, 56
offline, 51–53
online, 45–46, 47–49

migration, *continued*
performing validations, 55–56
troubleshooting, 57
using Azure Migrate, 43–44
Mixed Mode authentication, SQL Server, 68–69
modules, 178
monitoring resources
Azure SQL Database, 113–114
Azure SQL Managed Instance, 113–114
SQL Server on an Azure VM, 112–113
using database watcher, 121
using DMVs, 133–136
using extended events, 122–124
using Intelligent Insights, 140–141
using Query Store, 128–130
using SQL Insights, 118
adding SQL Server connection strings, 120–121
adding the monitoring VM to the profile, 120
creating a monitoring profile, 119–120
creating a monitoring VM, 119

N

native authentication, SQL Server, 66
New Credential dialog, 179
New Statistics dialog, 145
New-AzResourceGroupDeployment command, 173–174
New-AzSqlServer command, 174
New-SqlColumnEncryptionSettings cmdlet, 87–88
node
cluster, 208
SQL Server Agent, 158–159
notifications, 161. *See also* alerts

O

objectives, DP-300 exam, 219–221
object-level encryption, 83–84, 85–88
offline migration, 51–53
off-site backups, hybrid SQL infrastructure, 20
online data migration, 42–43
online migration, 45–46, 47–49
OpenID Connect, 66
operating system, automated patching, 10–11
operational performance baseline, 112. *See also* performance
orchestration, 17
outages. *See* DR (disaster recovery)

P

PaaS (Platform as a Service), 2
HA/DR solutions, 190
versus IaaS, 21–22
purchasing models, 22
DTU, 23, 31
vCore, 23–24, 31–32
SQL options, 22
page compression, 39
parallel testing, 192
partitions/partitioning, 38
compression ratio, 39
data compression, 39–40
horizontal, 28
sharding, 28–30
vertical, 27–28
patching
automated, 10–11
hybrid and IaaS deployment, 18–19
PEC (private endpoint connection), 90
performance. *See also* monitoring resources
Azure SQL Database, service tier, 31–32
baseline, 112, 115
counters, 113, 115
database, testing, 55–56
index, 136–137
metrics, 112
interpreting, 116
sources, 115
monitoring, using DMVs, 133–136
query
assessing the use of hints, 138
execution plans, 139–140
index changes, 136–137
recommending construct modifications based on resource usage, 137–138
SQL Server
Resource Governor, 149–150
settings, 148–149
Performance Monitor, 112
permissions, 73–74
configuring with SSMS, 75–76
denying, 75

principle of least privilege, 76–77
revoking, 75
policy
backup retention, 198–199
security, 107
PowerShell
AzSqlElasticJobTargetGroup cmdlet, 177
New-AzResourceGroupDeployment command, 173–174
New-AzSqlServer command, 174
on-premises installation, SQL Server, 19
pricing, vCore installation, 24
principle of least privilege, 76–77
properties, Query Store, 126
Properties dialog, 75, 80, 82, 100–101, 158
purchasing model, PaaS (Platform as a Service)
DTU-based, 23, 31
vCore, 23–24, 31–32

Q

Query Store, 139
configuring, 126
configuring properties, 126
enabling, 126
performance monitoring, 128–130
query/ies
execution plans, 139–140
hints, 138
index changes, 136–137
recommending construct modifications based on resource usage, 137–138
quorum options, WSFC (Windows Server Failover Cluster), 207–209

R

randomized encryption, 86
range-based sharding, 29
Reorganize Indexes dialog, 142–144
requirements, data migration, 41
assessing compatibility levels, 41–42
assessing downtime, 42–43
resources, 42
security, 43
Resource Governor, 149–150
resources
compute
hardware selection, 32, 34–35
scaling, 151–152
data migration requirements, 42
dependencies, 17
managing with Azure Purview, 103–104
monitoring
using database watcher, 121
using DMVs, 133–136
using extended events, 122–124
using Query Store, 128–130
using SQL Insights, 118–121
scaling up, 26–27
securables, 70
Restore dialog, 197
revoking permissions, 75
role-based access control, 77
role-based security, 71–73
routines, 120–121
row compression, 39
row-level security, 106–107
RPO (Recovery Point Objective), 188–189
RTO (Recovery Time Objective), 188–189
rules
alert, 182
firewall, 84–85
runbooks, 177–181, 185

S

sa login, 80–81
SAML (Security Assertion Markup Language), 66
scaling
Azure VM, 36
compute and storage resources, 151–152
up, 26–27
schedule, job, creating, 159
scripts
elastic jobs, 176–177
troubleshooting blocked transactions, 132
securables, 70, 76–77. *See also* permissions
Secure Enclaves, 88–89
security. *See also* authentication; encryption
Azure SQL, 25–26
Azure SQL Database, 57
data migration, 43

security, *continued*
firewall, rules, 84–85
policy, 107
role-based, 71–73
row-level, 106–107
Transport Layer. *See* TLS (Transport Layer Security)
security principals, 70
logins, creating, 70–71
user, creating, 70–71
Select Columns dialog, 146
selecting a database offering, 21
database sharding solutions, 26–28
PaaS
versus IaaS, 21–22
purchasing models, 22–24
SQL options, 22
security considerations, 25–26
table partitioning solutions, 26–28
server-level firewall rules, 84–85
"serverless", 25
service tier, 24
Azure SQL Database, 31–32
Azure SQL Managed Instance, 33–34
Set-AzVMExtension cmdlet, 119
settings
database-scoped configuration, 150–151
SQL Server, 148–149
sharding, 26–28
simulation testing, 192
sp_set_database_firewall_rule stored procedure, 85
split-brain situation, 207
SQL
Always Encrypted, 84
implementing object-level encryption, 85–88
implementing with VBS enclaves, 88–89
auditing, 96–98
roles, 71–73
Secure Enclaves, 88–89
security principals, 70
login, creating, 69--71
user, creating, 70–71
SQL Data Sync, 46, 58–59, 100
SQL database
blockages, troubleshooting, 131–132
predefined roles, 72–73
Query Store
configuring, 126
configuring properties, 126–128
enabling, 126
secure access, 89–90
wait statistics, 116
SQL Insights, 118
adding SQL Server connection strings, 120–121
adding the monitoring VM to the profile, 120
creating a monitoring profile, 119–120
creating a monitoring VM, 118
SQL Managed Instance, predefined roles, 73
SQL Server
authentication, 66
Active Directory, 68
Mixed Mode, 68–69
native, 66
Windows, 68–69
Azure Hybrid Benefit, 4
Azure Update Manager, 10–11
on an Azure VM, monitoring, 112–113
best practices assessment, 11
configure settings for performance, 148–149
configuring on Azure VMs, 35–37
deploying on an Azure VM, 3
SQL Server on Windows 2019 template, 5–9
templates, 4–5
using the SQL Server IaaS Agent Extension, 9–10
HA/DR solutions, 189–190
hybrid deployment, 19
adding bandwidth, 19
adding connectivity, 19–20
adding off-site backups, 20
Azure Arc, 21
fault tolerance, 20
IQP (intelligent query processing), 152
licensing, 3
predefined roles, 73
on-premises installation, 19
Resource Governor, 149–150
temporary storage configuration, 10
troubleshooting blockages, 132–133
versions, 81–82
SQL Server Agent, 157
job/s
alerts, 161, 162–164
notifications, 161
schedule, creating, 159
troubleshooting, 164–168
Manage Schedules dialog, 159

node, 158–159
Properties dialog, 158
starting and stopping, 158
SQL Server Management Studio dialog, 39–40
SQL Server on Windows 2019 template
plans, 5–6
Review + create tab, 8–9
SQL Server settings, 6–8
VM configuration and settings, 6–7
SSMS (SQL Server Management Studio)
applying data compression to a table or other element, 39–40
configuring permissions, 75–76
data classification, 95
extended events, 122–124
starting, SQL Server Agent, 158
statistics, maintenance, 144–146
storage
Azure SQL Database, elastic pools, 31
cloud, backup and restore, 199–201
database, 30
SQL Server, 10
VM (virtual machine), 37
stored procedures, sp_set_database_firewall_rule, 85
sync group, 100
syntax, T-SQL permission command, 74–75

T

tables
change tracking, 100–101
data compression, 39–40
Ledger, 104
masking data, 102
partitioning, 26–28, 38
row-level security, 106–107
statistics, maintenance, 144–146
tabletop testing, 192
TDE (transparent data encryption), 81–83, 91
templates, 8–9
ARM, 17, 169–171
SQL, 4–5
SQL Server on Windows 2019, 5–9
plans, 5–6
Review + create tab, 8–9
SQL Server settings, 6–8
VM configuration and settings, 6–7
testing
database performance, 55–56
HA/DR solutions, 191–192
TLS (Transport Layer Security), 91–92
tools, migration, 46
transactional replication, 46
troubleshooting
authentication and authorization issues, 78–79
automated database tasks, 185
blockages
in Azure SQL Database, 131–132
in SQL Server, 132–133
data migration, 57
deployment errors, 174–175
HA/DR solutions, 212–213
login, 78–79
SQL Server Agent jobs, 164–168
T-SQL commands
ALTER DATABASE, 101, 126, 148, 152
ALTER LOGIN, 78
ALTER ROLE, 72
authentication and authorization management, 80–81
backup and restore, 198–199
CDC (change data capture), 98–99
CREATE LOGIN, 69–70
CREATE USER, 70–71
DBCC CHECKDB, 146–147
DENY, 75
permission, syntax, 74–75
REVOKE, 75
stored procedures. *See* stored procedures

U

updates
automated patching, 10–11
DP-300 exam, 217–218
hybrid and IaaS deployment, 18–19
user, 70. *See also* login
permissions, 73–74
configuring with SSMS, 75–76
denying, 75
revoking, 75
SQL, creating, 70–71

V

validation, migration, 55–56
VBS enclaves, implementing Always Encrypted, 88–89
vCore
  Compute tier, 32
  purchasing model, 23–24, 31–32
versions, SQL Server, 81–82
vertical partitioning, 27–28
vertical scaling, 26–27
Virtual Machines Selector tool, 36
VM (virtual machine). *See also* Azure VM
  Azure Hybrid Benefit, 4
  performance monitoring, 115
  SQL Insights monitoring agent, 119
  SQL Server deployment, 3
    SQL Server on Windows 2019, 5–9
    templates, 4–5
  storage, 37
VPN (virtual private network), 19–20
Vulnerability Assessment, 107

W-X-Y-Z

wait statistics, 116
Windows authentication, 68–69
WSFC (Windows Server Failover Cluster), quorum options, 207–209