Microsoft

Migrating SQL Server Databases to Azure

Microsoft Azure Essentials

Carl Rabeler

PUBLISHED BY
Microsoft Press
A division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

ISBN: 978-1-5093-0292-5

Microsoft Press books are available through booksellers and distributors worldwide. If you need support related to this book, email Microsoft Press Support at mspinput@microsoft.com. Please tell us what you think of this book at http://aka.ms/tellpress.

This book is provided "as-is" and expresses the author's views and opinions. The views, opinions and information expressed in this book, including URL and other Internet website references, may change without notice.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

Acquisitions and Developmental Editor: Devon Musgrave
Project Editor: Carol Dillingham
Editorial Production: Cohesion
Technical Reviewer: Chris Randall
Copyeditor: Ann Weaver

Visit us today at

microsoftpressstore.com

- **Hundreds of titles available** – Books, eBooks, and online resources from industry experts
- **Free U.S. shipping**
- **eBooks in multiple formats** – Read on your computer, tablet, mobile device, or e-reader
- **Print & eBook Best Value Packs**
- **eBook Deal of the Week** – Save up to 60% on featured titles
- **Newsletter and special offers** – Be the first to hear about new releases, specials, and more
- **Register your book** – Get additional benefits

Microsoft

Contents

Introduction

SQL Server is Microsoft's relational database management system (RDBMS). SQL Server can now be hosted entirely in Microsoft Azure, either in a hosted virtual machine (VM) or as a hosted service. Hosting a virtual machine in Azure is known as infrastructure as a service (IaaS), and hosting a service in Azure is known as platform as a service (PaaS). Microsoft's hosted version of SQL Server is known as Azure SQL Database or just SQL Database that is optimized for software as a service (SaaS) app development.

This ebook, *Microsoft Azure Essentials Migrating SQL Server Databases to Azure*, introduces you to SQL Server in an Azure virtual machine and to Azure SQL Database, and walks you through getting started with each approach. It takes you from creating a SQL Server instance in a virtual machine or as a platform service to migrating an on-premises database into Azure and then to securing the data and the database in Azure.

Beyond the explanatory content, each chapter includes one or more walk-throughs with extensive screenshots so you can follow along and create a trial subscription, create SQL Server in an Azure virtual machine, create an Azure SQL Database, migrate an on-premises database to each Azure environment, create users, back up and restore data, and archive data.

Who should read this ebook

This ebook exists to help existing SQL Server database users understand Microsoft's offering for SQL Server in Azure. It is designed to help you quickly start using the skillset you already have with SQL Server and to introduce you to Microsoft Azure. It is not intended as a deep dive into Azure itself or into virtual machines. Its focus is SQL Server itself and helping you understand how SQL Server in Azure is similar to SQL Server in an on-premises environment and where it is different. Its focus is also to distinguish the two Azure offerings from each other. A discussion of the Azure SQL Data Warehouse is beyond the scope of this ebook.

Assumptions

This ebook assumes you have experience with Microsoft's on-premises SQL Server offering and have little to no experience with Microsoft Azure. Furthermore, although both Transact-SQL and Azure PowerShell are used in this ebook, you do not need any significant experience with either to understand the content of this ebook or the walk-throughs at the end of each chapter.

Furthermore, this ebook assumes you have an existing SQL Server on-premises installation with a database that you can use to walk through the migration chapter. For the most recent AdventureWorks sample databases, go here: http://msftdbprodsamples.codeplex.com/.

Conventions and features in this ebook

This ebook presents information using conventions designed to make the information readable and easy to follow.

- Boxed elements with labels such as "Note" and "See Also" provide additional information or alternative methods for completing a step successfully.
- Text that you type (apart from code blocks) appears in bold.

Important Azure is evolving rapidly, as is SQL Database. Therefore, the screenshots and all information regarding the pricing, the capacity limits, and the features are accurate only as of the date this content is written.

Acknowledgments

I'd like to thank the following people: Lindsey Allen, Mark Souza, Jennifer Hubbard, Steve Stein, Sidney Higa, Rick Byham, Jeff Gollnick, Donald Gill, Devon Musgrave, Chris Randall, Carol Dillingham, Terry Monaghan, Shirley Rabeler, and Bruce Rabeler.

Errata, updates, & ebook support

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

http://aka.ms/SQLtoAzure/errata

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

If you need additional support, email Microsoft Press Book Support at mspinput@microsoft.com.

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 http://support.microsoft.com.

Free ebooks from Microsoft Press

From technical overviews to in-depth information on special topics, the free ebooks from Microsoft Press cover a wide range of topics. These ebooks are available in PDF, EPUB, and Mobi for Kindle formats, ready for you to download at:

http://aka.ms/mspressfree

Check back often to see what is new!

We want to hear from you

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this ebook at:

http://aka.ms/tellpress

We know you're busy, so we've kept it short with just a few questions. Your answers go directly to the editors at Microsoft Press. (No personal information will be requested.) Thanks in advance for your input!

Stay in touch

Let's keep the conversation going! We're on Twitter: http://twitter.com/MicrosoftPress.

CHAPTER 1

Overview of SQL Server in Microsoft Azure

In this chapter, I'll introduce you to your options for running SQL Server in Microsoft Azure, and I'll teach you enough about the Microsoft Azure platform that you'll be ready to begin using SQL Server in Microsoft Azure. After a quick overview of SQL Server in Azure, you'll learn about the Microsoft Azure platform. You'll learn how to get started in Azure and be introduced to the Azure features that support the use of SQL Server in Azure. We'll then spend a bit of time comparing and contrasting the two primary options you have when running SQL Server in Azure. This chapter will conclude with a lab that will get you started with Microsoft Azure with a trial subscription. You'll connect to the Azure portal in your browser of choice and connect to your Azure subscription using Azure PowerShell.

Overview of SQL Server in Microsoft Azure

SQL Server can be hosted entirely in Microsoft Azure, either in a hosted virtual machine (VM) or as a hosted service. Hosting a virtual machine in Azure is known as infrastructure as a service (IaaS), and hosting a service in Azure is known as platform as a service (PaaS). Microsoft's hosted version of SQL Server is known as Azure SQL Database or just SQL Database that is optimized for software as a service (SaaS) app development. These two options are compared and contrasted in more detail later in this chapter, and they are the primary focus of this ebook.

Figure 1-1 illustrates the range of environments in which SQL Server can run, from on-premises physical machines ("raw iron") and private clouds of virtual machines in private data centers to running in the cloud as either infrastructure as a service or platform as a service. The relative administration overhead for each environment appears on the *x*-axis, and the relative hardware costs for each environment appear on the *y*-axis. As virtualization increases and SQL Server is moved to the cloud, costs decrease along with administrative overhead.

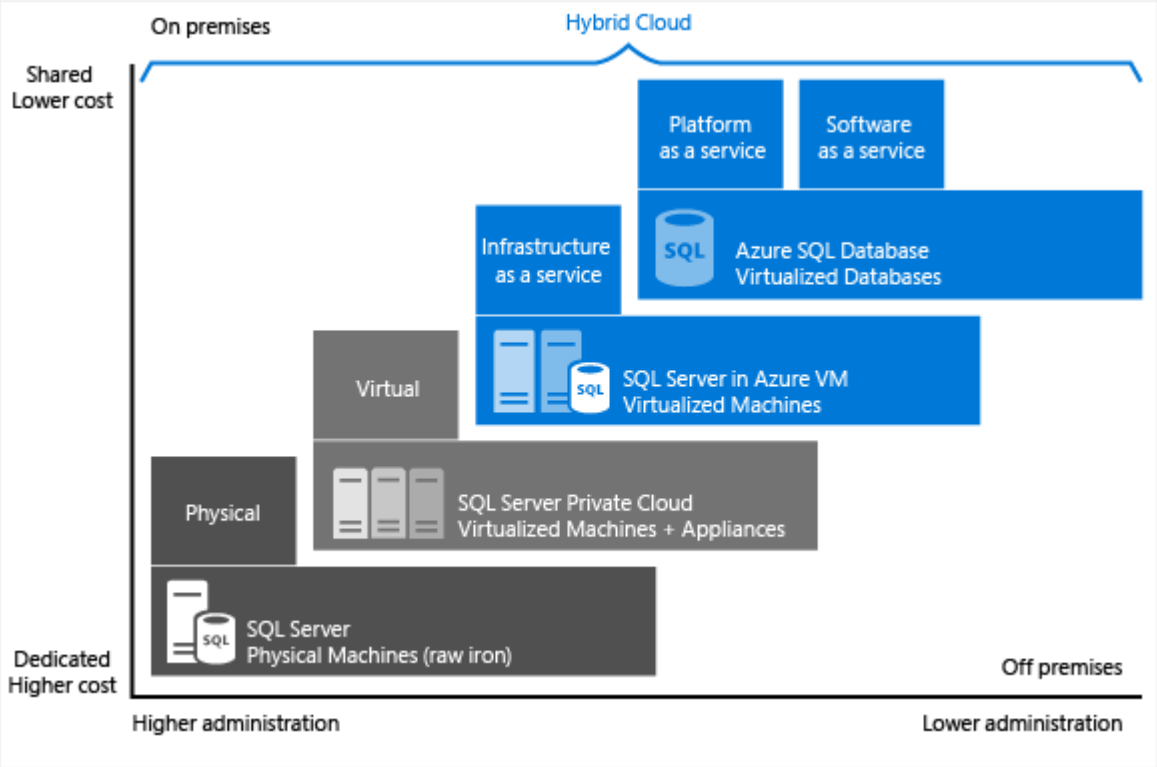


Figure 1-1: IaaS vs. PaaS vs. SaaS

SQL Server also can be deployed in a hybrid cloud scenario, extending your on-premises SQL Server environment to utilize various features of the Azure platform:

- SQL Server backup to URL: You can back up your database directly to Azure blob storage or back it up to an on-premises file store and then copy it to Azure blob storage. Using this option can save precious storage space on expensive local storage.
- SQL Server data files in Azure: You can use Azure blob storage for database files for an on-premises instance of SQL Server. Although this option primarily is used with Azure virtual machines, it has its place when developing and testing functionality in some scenarios.
- Stretch SQL Server table to Azure SQL Database: You can stretch an on-premises table to store cold and warm data (older data) in Azure SQL Database while hot or current data (more recent data) remains in the on-premises table—with all of the data being available to query. This option became available in SQL Server 2016 and is a great way to archive infrequently accessed data off your local storage. This option can save money and increase performance for online transactional processing (OLTP) operations on hot data, while the data remains available for analytical queries.
- Transactional replication to Azure SQL Database: You can use transactional replication to replicate data from an on-premises or IaaS SQL Server database to Azure SQL Database. This option is useful to replicate data close to different groups of users to improve query performance and as a prelude to migrating to Azure, enabling you to minimize downtime during migration.

- AlwaysOn Availability Group replica in IaaS: You can configure SQL Server in IaaS as an asynchronous replica of an AlwaysOn Availability Group. This option provides you with a low-cost disaster recovery scenario and can be used as a prelude to migrating to Azure, allowing you to minimize downtime.

Finally, Microsoft offers an additional PaaS service using SQL Server for data warehouse solutions, called Azure SQL Data Warehouse. Azure SQL Data Warehouse is an enterprise-class, distributed database capable of processing massive volumes of relational and non-relational data. A discussion of this service is beyond the scope of this ebook.

What do I need to know about Azure?

To get you ready to start using SQL Server in Microsoft Azure, I will introduce you to the Azure platform. This section is intended only as a quick introduction to the very basics required to get you started with SQL Server in Azure. I have included many links in this section to help you go into more detail either as you are reading or later as you have questions. For those readers who are new to the Azure platform, this section will provide enough of a foundation to begin to understand the Azure components upon which SQL Server in Azure is built and to complete the labs that appear in this ebook successfully. You will be able to complete all of the walk-throughs in this ebook using a free trial subscription (or your existing paid or MSDN subscription).

An overview of Azure services

Wikipedia describes Microsoft Azure as "a cloud computing platform and infrastructure, created by Microsoft, for building, deploying and managing applications and services through a global network of Microsoft-managed and Microsoft partner hosted datacenters." This is a pretty good general description of what Azure provides to you.

Here's Microsoft's description of Azure: "a growing collection of integrated cloud services—analytics, computing, database, mobile, networking, storage, and web—for moving faster, achieving more, and saving money."

Azure allows you to do the following:

- Use the skillsets you already possess and the technologies with which you already are familiar to develop and deploy solutions using SQL Server.
- Work with a wide range of operating systems, programming languages, databases, and devices.
- Integrate Azure with your existing IT environment, including Active Directory for single sign-on.
- Scale up and scale down your Azure services based on demand so you only pay for what you need when you need it.
- Maintain data privacy. Microsoft with Azure services was the first major cloud provider to adopt the new international cloud privacy standard, ISO 27018.
- Encrypt your SQL Server data both at rest and on the wire.
- Have enterprise-grade service-level agreements (SLAs) on services, 24/7 tech support, and round-the-clock service health monitoring.

Finally, you might be asking: Why Microsoft? A good answer is that Microsoft is the only vendor positioned as a Leader across Gartner's Magic Quadrants for Cloud Infrastructure as a Service, Application Platform as a Service, and Cloud Storage Services for the second consecutive year. Another answer is that Microsoft is the only vendor to offer SQL Server as a service and SQL Server in

IaaS. A final answer is that Microsoft is extending the capabilities of SQL Server in Azure at a faster pace than any other vendor.

Subscribing to Azure services

A subscription is required to start working with Azure services. You can start with a free trial subscription, use your MSDN subscriber benefits, or skip the Azure free trial and dive right in and buy a subscription. There are a variety of subscription plans, including low nonprofit pricing options. In the lab at the end of this chapter, you will create and begin using an Azure trial subscription.

A subscription entitles you to use all of the Azure services. When working with SQL Server in Azure, the services that you will be most interested in learning about and testing for data storage and management are:

- Azure Storage
- SQL Server on an Azure Virtual Machine
- Azure SQL Database

We will discuss each of these Azure services later in this chapter and throughout this ebook. In the walk-throughs in this ebook, you will use all three of these services.

When you are ready to build your client application(s) in Azure, you will be interested in learning a variety of services, including but not limited to:

- Azure Web Role
- Azure Cloud Services
- Azure Table Service

These services are beyond the scope of this ebook, but these links will get your started.

Paying for Azure services

Microsoft will charge you for using Azure services based on your usage of each service you choose to use. There are no upfront costs or termination costs; you only pay for the number and types of services you actually use. In general, you will pay for:

- **Compute resources**: You pay for what you use, and faster processors cost more than slower machines.
- **Storage resources**: You pay for what you use, and faster I/O subsystems cost more than slower I/O subsystems.
- **Data transfer services**: You pay for transferring data out of Azure and between Azure data centers (transfer of data into Azure is free). You pay for the amount of data transferred, and dedicated private connections cost more than routes over the public Internet.
- **Additional services**: You pay for additional services, such as:
  - SQL Server in a virtual machine
  - SQL Server as a service
  - Web services
  - Active Directory services

- Many other services

Microsoft provides a pricing calculator to help you estimate your expected monthly bill. To track your Azure usage and view your bill, go to https://account.windowsazure.com/.

It is important to remember that you only pay for what you use, so you should use only what you need when you need it. Azure provides the ability to scale up and scale down compute resources dynamically based on need/demand, to store infrequently accessed data on slower and less expensive storage resources, to turn off virtual machines when not needed, and to design applications to reduce unnecessary use of compute, storage, and data transfer resources.

What are the Azure deployment models?

Azure provides two deployment models, the classic model and the new Azure Resource Manager (ARM) stack. Underneath each model is an application programming interface (API): the Resource Manager API for ARM and the Service Management API for classic. Although developers can write code to interact with these APIs directly through the REST API, IT pros interact with these APIs indirectly using the Azure portal, using Azure PowerShell cmdlets on a Windows computer, or using the Azure Command-Line Interface (CLI) on a Windows, OS X, or Linux computer.

These two models are compatible with each other, but ARM simplifies the deployment and management of resources by managing them as a single resource group. Most newer resources support ARM, and eventually all resources will. How you create, configure, and manage Azure resources is different in these two models. The individual Azure resource features or behaviors can be different across the two models or only exist in one model or the other. Throughout this ebook, we will be creating and managing resources using ARM, and you will learn about using ARM with Azure storage, Azure virtual machines, and Azure SQL Database as we go along.

Although a deep dive into the Azure deployment models is beyond the scope of this ebook, here are some links when you are ready dive in:

- Azure Deployment Models
- Azure Resource Manager overview
- Understanding Resource Manager deployment and classic deployment
- Azure Compute, Network, and Storage Providers under the Azure Resource Manager

Generally, all new work you do should focus on ARM rather than the classic model. It is important to know that both exist because the transition to ARM is not complete, and if you are working with existing resources, they may have been built using the classic model. If so, how you interact with these objects may be somewhat different. Finally, when reading documentation, you will need to pay attention to whether it was written for ARM, classic, or both. All of the Microsoft documentation is being updated to support ARM, but you will find a lot of information on the Internet that predates ARM and as a result may be confusing if you are not aware that there are two models.

Using the Azure portal

The current Azure portal at http://ms.portal.azure.com is the portal through which you will start creating and managing Azure services, specifically SQL Server. The Azure portal includes a dashboard that you can configure to work with and monitor the resources in your environment. The Azure portal lets you administer all of your Azure platform resources in a single location. The current Azure portal uses ARM, although some classic model functionality is exposed through the new portal. The legacy or classic portal still is available for use (see tile in lower-right corner of Figure 1-2), but the new portal has been released for general availability and is the portal we will use in this ebook.

Figure 1-2 shows the Azure portal before any services have been configured.

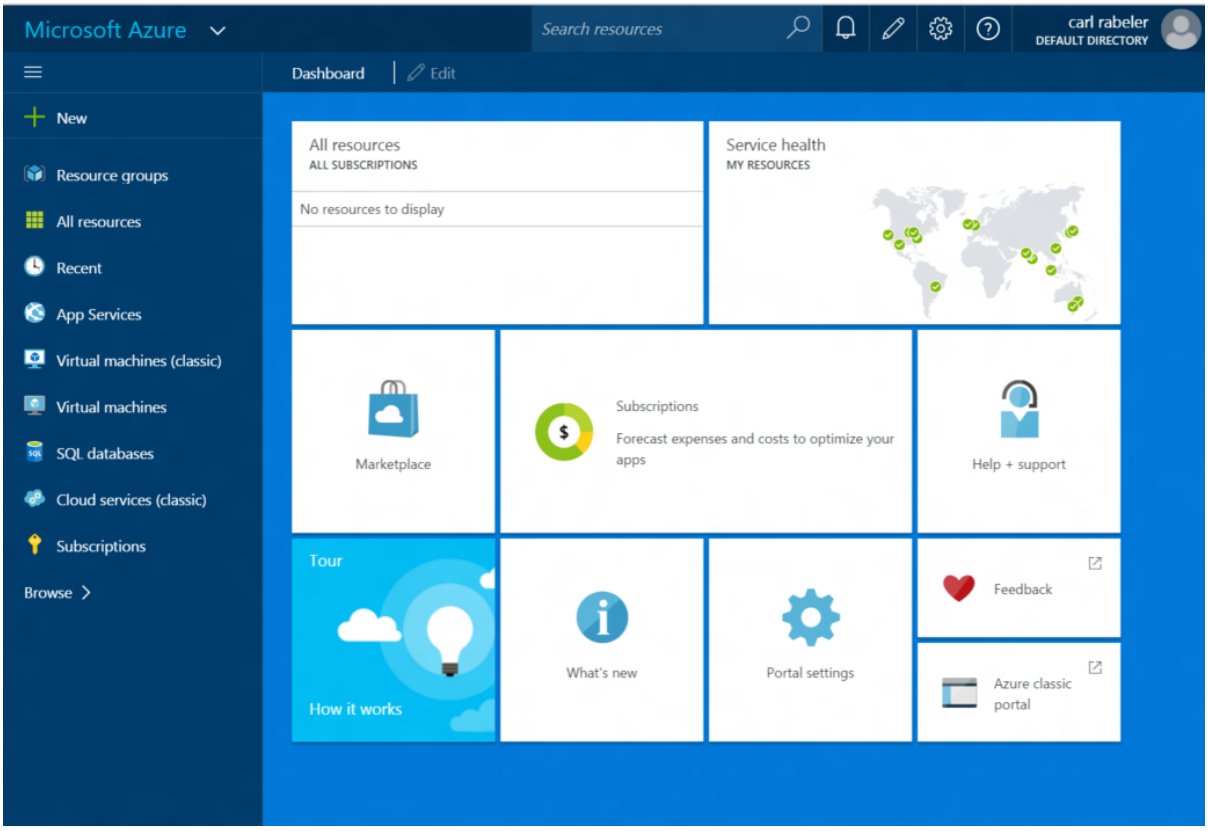


Figure 1-2: Azure portal

Using the blade on the left, you can create new Azure services or view existing services. Using the dashboard, you also can view and manage existing services. You can customize the dashboard for your environment, pinning specific services to the dashboard. The dashboard includes links to help you get started and, if necessary, to switch to the Azure classic portal.

Getting started with Azure PowerShell

In addition to using the Azure portal, you can create, configure, and manage Azure resources from a Windows computer using Azure PowerShell. Individual Azure resources have Resource Manager cmdlets, Service Management cmdlets, or both. Some resources and features only can be created and/or configured using PowerShell or the CLI. Depending on the resource, when using Resource Manager PowerShell cmdlets you may have two options for creating and configuring Azure resources:

- PowerShell cmdlets only
- PowerShell cmdlets with an Azure Resource Manager template

We will install and configure PowerShell in the lab for this chapter and use PowerShell cmdlets and JSON templates later in this ebook.

Getting started with the Azure Command-Line Interface

You also can create and configure Azure resources from Windows, OS X, or Linux computers using the Azure Command-Line Interface (Azure CLI). Read the Install the Azure CLI article to install the Azure CLI on your operating system of choice. Using Azure CLI is beyond the scope of this ebook.

What is Azure storage?

Azure provides a variety of storage options that provide great flexibility and scale, including:

- Blob storage: Used for storing all kinds of data, regardless of type or structure, such as SQL Server database and backup files
- Table storage: Used for storing structured NoSQL data (data stored using key-value pairs rather than tables and columns)
- Queue storage: Used to store messages
- File storage: SMB-based storage for existing or new applications

The primary storage that will interest us from the SQL Server in Azure perspective is blob storage. In Azure blob storage, you can store up to 500 terabytes (TB) per storage account, and you can have 50 storage accounts per subscription. Azure automatically replicates your data stored in blob storage to two additional copies within the same data center to protect against hardware failures to ensure your data is available. Optionally, you can add geo-replication to replicate your data to one or more data centers so that three additional copies are stored and are available in each remote data center(s) for additional protection against disaster. Because Azure is available in 19 regions across 4 continents and replicas can be readable, you can replicate your data to place it closer to your customers for read-only queries.

For applications with low latency and high throughput requirements, such as SQL Server, you can opt for premium storage using SSD disks that deliver up to 80,000 IOPS. Each virtual machine can have up to 64 TB of premium storage and attain up to 2,000 megabytes per second (MBps) throughput with extremely low latency for read operations.

Although not exposed as an explicit option, Azure SQL Databases also use SSD disks.

What are Azure virtual machines?

Azure virtual machines (Azure VMs) enable you to create and manage virtual machines in Azure, providing what is known as infrastructure as a service (IaaS). Microsoft runs and maintains the hardware and uses a variety of mechanisms to protect you from other users sharing the same hardware and consuming excess resources (known as noisy neighbors). These mechanisms distribute the load from the large number of virtual machines in Azure across a large number of physical machines and avoid overloading any individual machine.

Creating an Azure VM gives you the flexibility of virtualization without having to buy and maintain the physical hardware. With an Azure VM, you are responsible for configuring, patching, and maintaining the operating system and all other software that runs on the machine, although Azure does provide an option to patch SQL Server for you (more on this later). Azure VMs are available in five different series (or SKU): A, D, DS, G, and GS, with different sizes within each SKU. Each SKU and size has its own performance characteristics and price associated with it. For SQL Server, you should choose either the DS or the GS series because these have higher performance characteristics and support premium storage (that is, SSDs). Azure charges for VMs based on the VM size and the operating system, with storage priced and charged separately.

An Azure VM has an operating system (Windows Server for our purposes), local and attached storage, and networking capabilities. The Azure virtual machines gallery provides preconfigured images for your use, or you can upload your own VM image using AzCopy. A VM uses virtual hard disks (VHDs) to store its operating system and generally uses one or more virtual hard disks to store their data. A VHD is stored in an Azure blob and presented to the VM as a drive letter.

Figure 1-3 shows two Azure VMs, each with a VHD, one created from the gallery and the other uploaded by the user. The figure also shows two of the tools for creating and managing VMs (you also could use PowerShell to make the REST API calls).

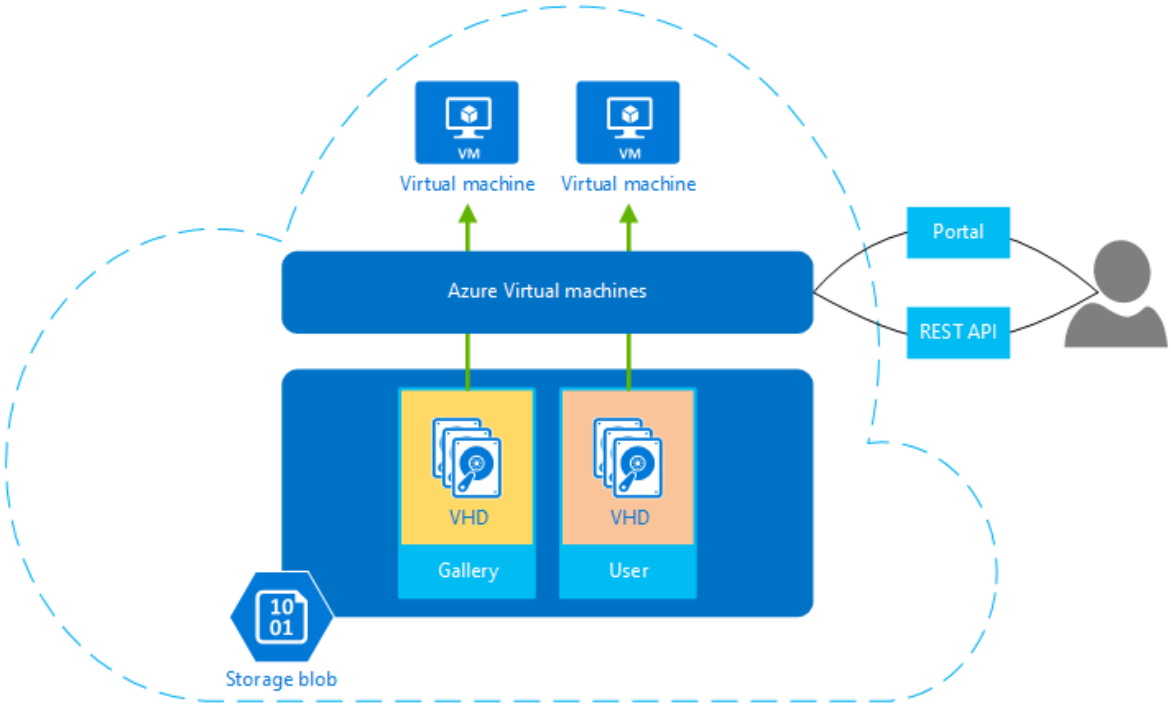


Figure 1-3: Azure virtual machines (Azure VMs)

Some applications, including SQL Server, let you store data directly in Azure blobs rather than in a VHD or in addition to in a VHD.

What is SQL Server in an Azure virtual machine (IaaS)?

Running SQL Server in an Azure VM is similar to running SQL Server in a virtualized environment in your own data center or in a traditional hosting environment, except that Microsoft provides the hardware on which the virtual machine runs and provides an availability SLA of 99.95 percent. You are in charge of, responsible for, and have either some or complete control of:

- **Choosing the amount of compute resources**: Many virtual machine size options
- **Choosing and configuring storage**: Many storage options
- **Choosing an operating system**: Many versions, including Windows (SQL Server only runs on Windows)
- **Choosing the version of SQL Server**: Any version of SQL Server
- **Choosing SQL Server license model**: Choose preinstalled version with per-minute licensing or bring your own license with SA
- **Managing the virtual machine**: Manage the virtual machine itself via Remote Desktop Protocol (RDP), PowerShell, or CLI

- **Managing SQL Server**: Manage SQL Server using SQL Server Management Studio or any other SQL tool you wish
- **Securing the virtual machine**: Manage firewalls and access using standard Windows methods, including choosing to join a domain (directly or using Azure Active Directory)
- **Securing SQL Server**: Manage authentication and access using standard SQL Server methods
- **Optimizing virtual machine performance**: Tune the virtual machine using standard virtual machine tuning methodologies
- **Optimizing SQL Server performance**: Tune the SQL Server instance using standard SQL Server tuning methodologies
- **Managing costs**: Dynamically scale up and scale down the virtual machine size as computer power is needed, and add or remove disks as storage and throughput is needed
- **Patching the operating system**: Your responsibility
- **Patching SQL Server**: Your responsibility, but Microsoft provides tooling to assist
- **Backing up SQL Server databases**: Your responsibility, but Microsoft provides tooling to assist
- **Managing availability**: Microsoft ensures the availability of the VM itself within a single data center, but all additional availability is your responsibility with Microsoft providing tooling to assist (see AlwaysOn Availability Groups).

As you can see, with SQL Server on an Azure VM, you have great ability to configure and manage SQL Server in a VM in a manner similar to how you currently manage IT resources in your on-premises environment. In Chapter 2, "Getting started with SQL Server in an Azure virtual machine" and Chapter 5, "Authentication, authorization, and data resiliency" of this ebook, we will dive into these choices in detail, building and managing SQL Server in an Azure VM.

What is Azure SQL Database (PaaS)?

Azure SQL Database is a service at the database level that delivers predictable performance, scalability with no downtime, business continuity, and data protection. Azure SQL Database is designed to deliver predictable database performance with very low levels of administration for performance at a variety of predictable levels. Microsoft automatically configures, patches, and upgrades the database for you. Microsoft provides an availability SLA of 99.99 percent.

In contrast to Azure virtual machines, with SQL Database you are guaranteed a certain level of performance, regardless of usage by other users. Predictable performance in Azure SQL Database is delivered based on service tiers, from Basic to Premium, with different levels of performance and capabilities both within and across tiers to support lightweight to heavyweight database workloads. The amount of performance you get on each tier is represented as a number of DTUs. A DTU is a database transaction unit and represents a combination of compute, database I/O, and memory resources. A certain amount of these resources is guaranteed at each service tier level. Furthermore, there are maximum limits for each performance level for sessions, concurrent logins, concurrent requests, and in-memory OLTP (premium tier feature only). The guaranteed DTU resources, features, and limits per service tier and performance level are shown in Figure 1-4.

	Basic	Standard				Premium				
		S0	S1	S2	S3	P1	P2	P4	P6/P3	P11
DTUs	5	10	20	50	100	125	250	500	1,000	1,750
Max storage (GB)	2	250				500				1,000
Max In-memory OLTP storage (GB)	N/A	N/A	N/A	N/A	N/A	1	2	4	8	14
Max concurrent workers	30	60	90	120	200	200	400	800	1,600	2,400
Max concurrent logins	30	60	90	120	200	200	400	800	1,600	2,400
Max concurrent sessions	300	600	900	1,200	2,400	2,400	4,800	9,600	19,200	32,000
Point-in-time restore	Any point last 7 days	Any point last 14 days				Any point last 35 days				
Disaster recovery	Active Geo-Replication, up to 4 offline or online (readable) secondary backups									

Figure 1-4: Service tiers, performance levels, and limits

You can build your first app on a small database for a few bucks a month, then change the service tier and performance level manually or programmatically at any time as your app requires resources, with minimal downtime to your app or your customers. You are billed at an hourly fixed rate for outgoing Internet traffic only (not by query) based on the service tier and performance level you choose (more on these in Chapter 3, "Getting started with an Azure SQL Database"). The first 5 GB of network traffic per month is free.

Azure SQL Database V12 is based on SQL Server 2016, delivering close compatibility with SQL Server 2016. There is a limited set of features in SQL Server 2016 (and in earlier versions) that are not yet supported in Azure SQL Database. This set of features is shrinking. At the same time, Microsoft has adopted a cloud-first approach to new features in SQL Server—delivering many new features first in Azure SQL Database (in both private and public preview mode) before releasing them to SQL Server 2016 (and to rolling upgrades to SQL Server 2016 that will be coming).

Microsoft combines the power of machine learning and automation with Azure SQL Database to deliver a number of features that are available only in Azure SQL Database. These include:

- Built-in backup: Reduces administration costs when there are large numbers of databases and supports point in time restore, geo-restore, standard geo-replication, and active geo-replication
- Auditing and threat detection: Tracks database events to maintain regulatory compliance, understand database activity, and gain insight into discrepancies and anomalies that could indicate business concerns or security violations
- Index advisor: Recommends and/or automatically adds indexes based on actual query performance and removes added indexes that provide no value
- Query performance insight: Provides insight into resource usage by top CPU consuming queries and the ability to drill down into query details of problematic queries
- Elastic database pools: Enables you to pay for a pool of resources and share them across databases to efficiently spend more for resources as needed

Furthermore, you can manage many aspects of Azure SQL Database, including monitoring its health, directly through the Azure portal dashboard. The following figure shows an Azure SQL Database in the Azure portal.

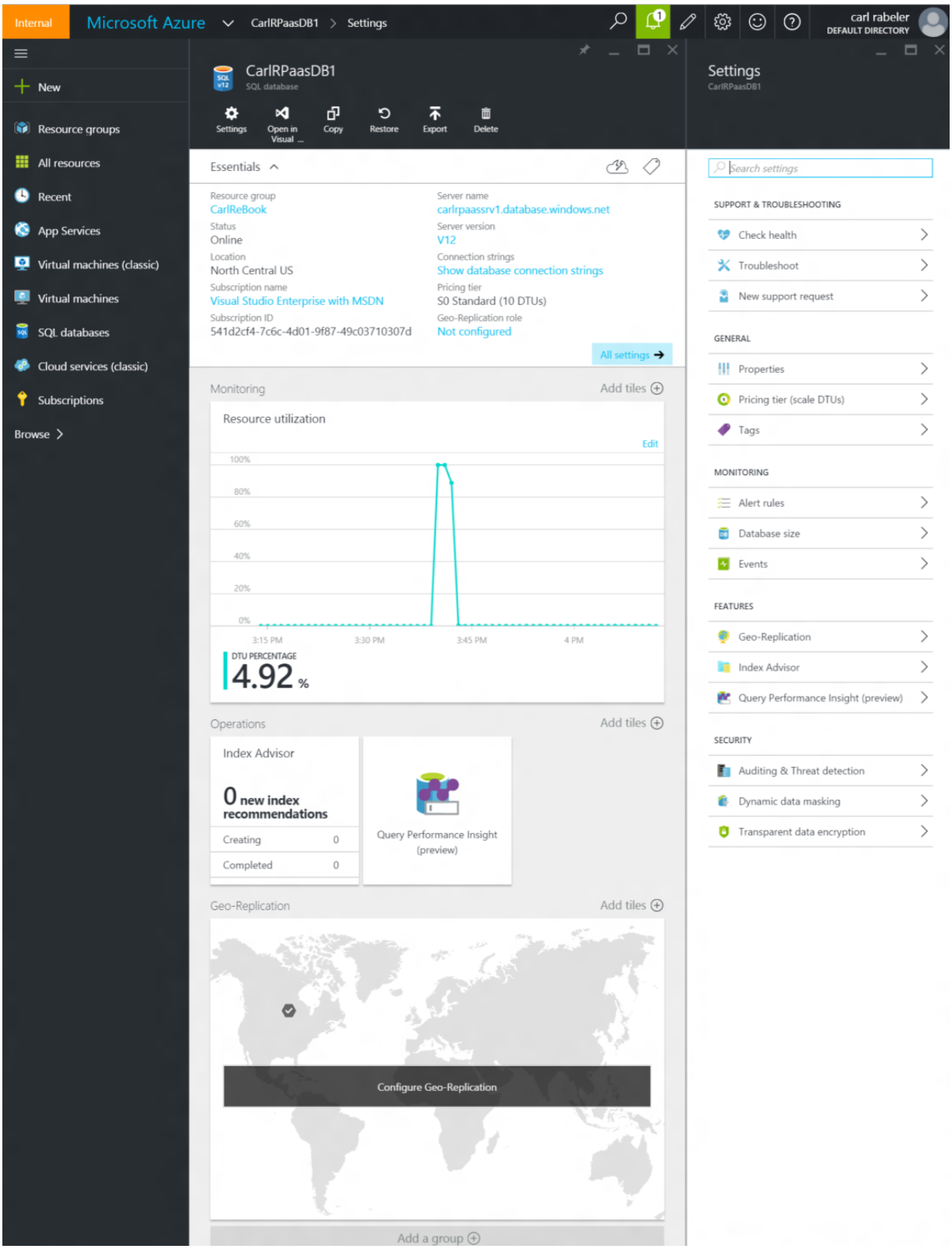


Figure 1-5: SQL Database in Azure portal

When to use SQL Server in a VM (IaaS) and when to use Azure SQL Database (PaaS)

The following table summarizes the main characteristics of SQL Server on an Azure VM and Azure SQL Database:

	SQL on an Azure VM	Azure SQL Database
Best for	Running existing applications with minimal changes.	New cloud-designed applications for which developer productivity and rapid time to market is important.
	You need a customized IT environment with full administrative rights.	You need built-in high availability, disaster recovery, and upgrade mechanisms.
	Rapid development and test scenarios.	Development by department teams.
	Disaster recovery for on-premises solutions (AlwaysOn replica).	Applications using scale-out patterns.
	Databases larger than 1 TB.	Building SaaS applications.
	Using features not supported in SQL Database.	Use newest cloud-born SQL Database features.
Resources	You have IT resources to manage.	You do not have or do not want to use IT.
Total Cost of Ownership	Eliminates hardware costs (CapEx).	Eliminates hardware costs (CapEx) and reduces administrative costs (OpEx).
Business Continuity	You can build high availability and disaster recovery solutions.	Built-in fault tolerance, high availability, and disaster recovery.
Hybrid Cloud	Extend Active Directory Domain, AlwaysOn Availability Groups, transactional replication.	Extend Active Directory Domain, StretchDB, transactional replication.

Lab: Create an Azure subscription and connect to Azure portal

In this lab, you will create an Azure trial subscription, connect to the Azure portal using the browser of your choice, install and configure PowerShell, and connect to your Azure subscription using PowerShell.

Create an Azure trial subscription

1. Open the browser of your choice, connect to https://azure.microsoft.com/en-us/pricing/free-trial/, and click Try It Now to start the signup for a one-month trial, as shown in Figure 1-6.

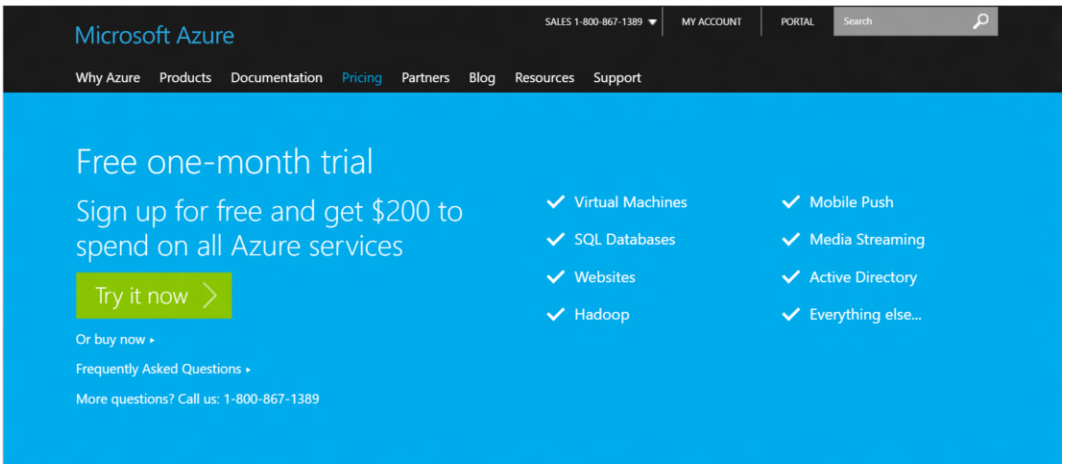


Figure 1-6: Free one-month trial subscription

2. Provide your Microsoft account information (your email or phone number and password for an account that does not have a Microsoft Azure subscription already, or sign in with your Azure subscription account and skip to the next section) and click Sign In, as shown in Figure 1-7. If you have not registered an email or a phone number previously, you must do so to continue.

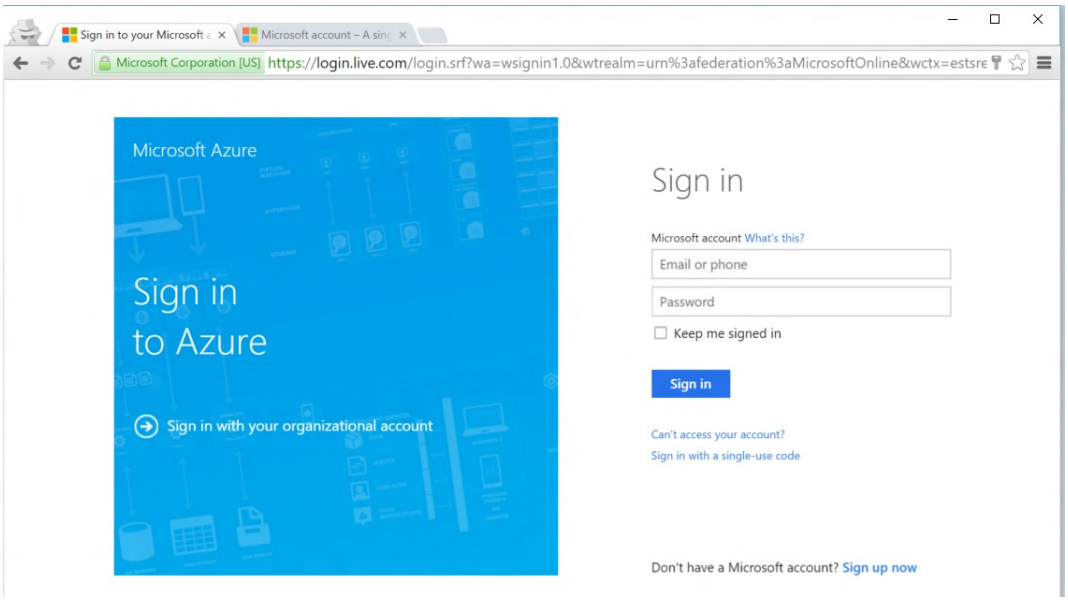


Figure 1-7: Sign in with Microsoft account

3. Complete the form, providing the requested information, as shown in Figure 1-8. You will need to confirm your identity by phone and by credit card (you will not be charged). Select the check box to agree to the free trial and click Sign Up.

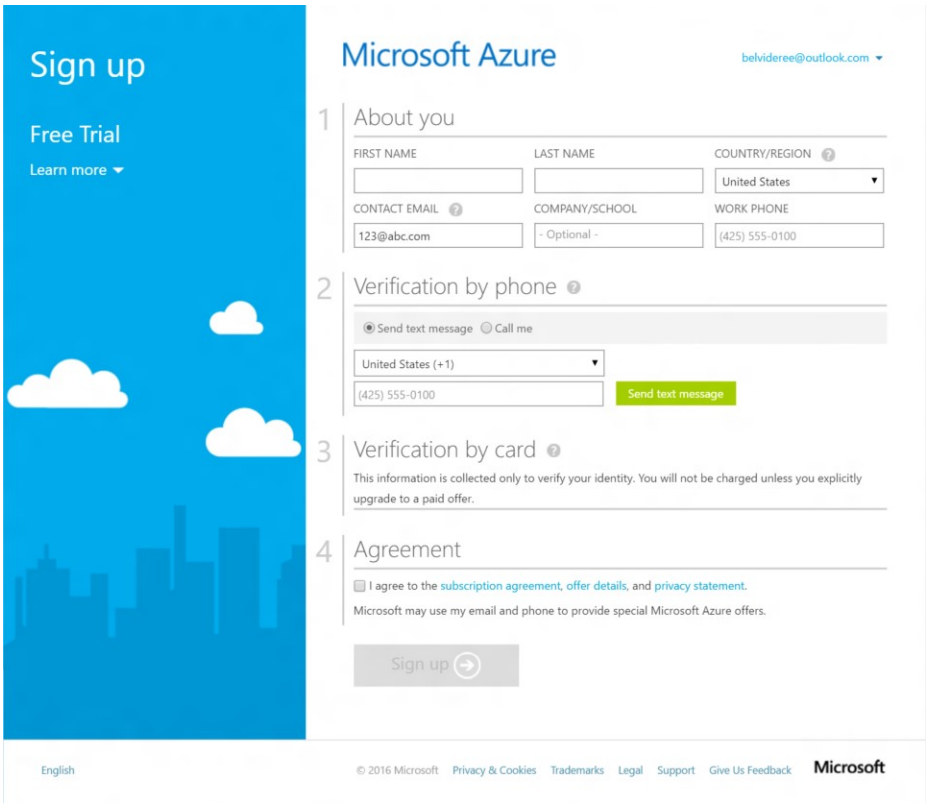


Figure 1-8: Sign up and verify identity for free trial

4. Wait while your subscription is being created, as shown in Figure 1-9.

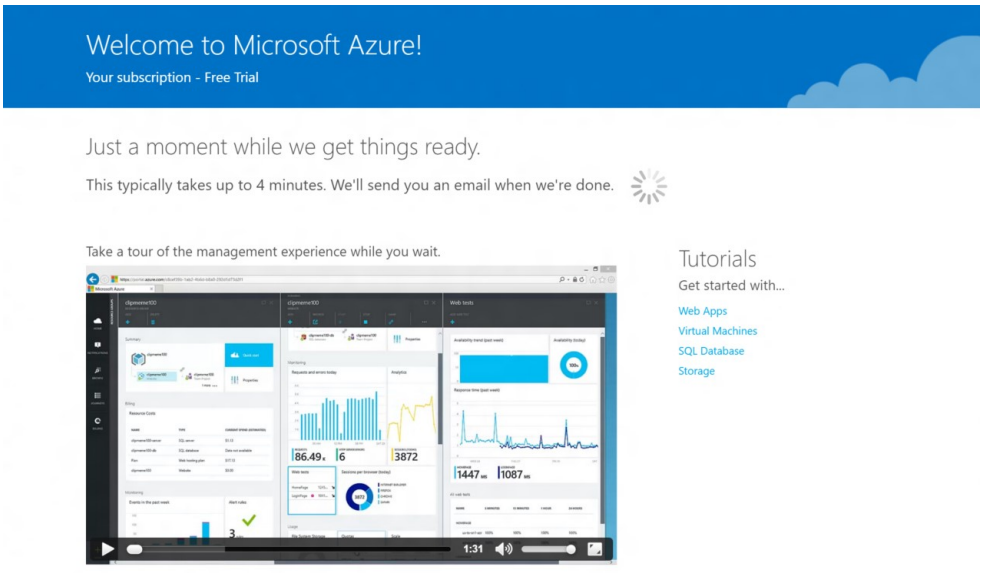


Figure 1-9: Wait while your subscription is being created

5. After you see the screen shown in Figure 1-10, continue to the next procedure.

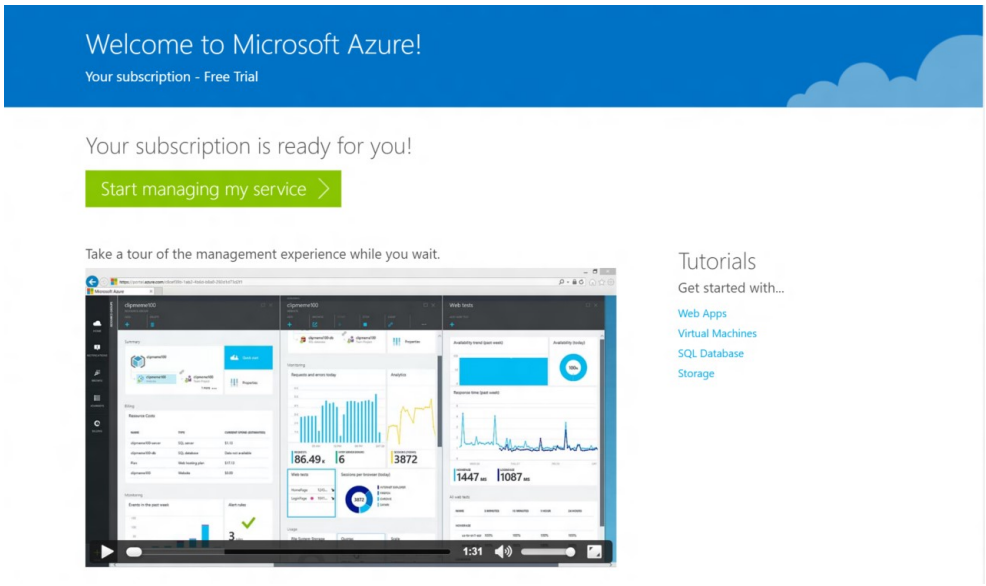


Figure 1-10: Subscription complete

Connect to the Azure portal with your Azure trial subscription using your browser

1. In the browser, click Start Managing My Service or open the browser of your choice, connect to https://account.windowsazure.com/Subscriptions, and log in using your Microsoft account, as shown in Figure 1-11.

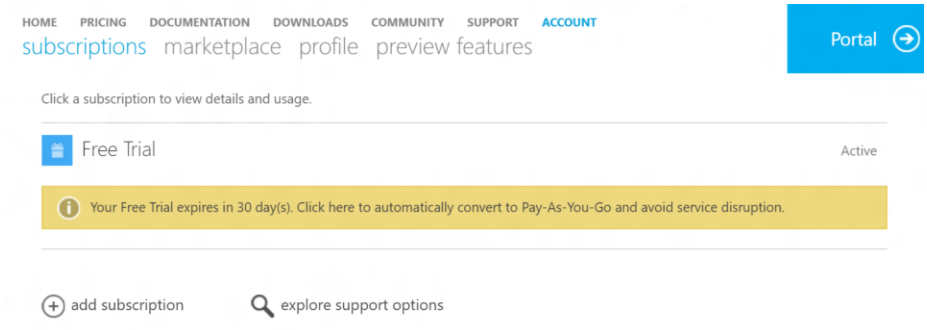


Figure 1-11: Azure subscriptions

2. Review this page and then click Portal, as shown in Figure 1-12.

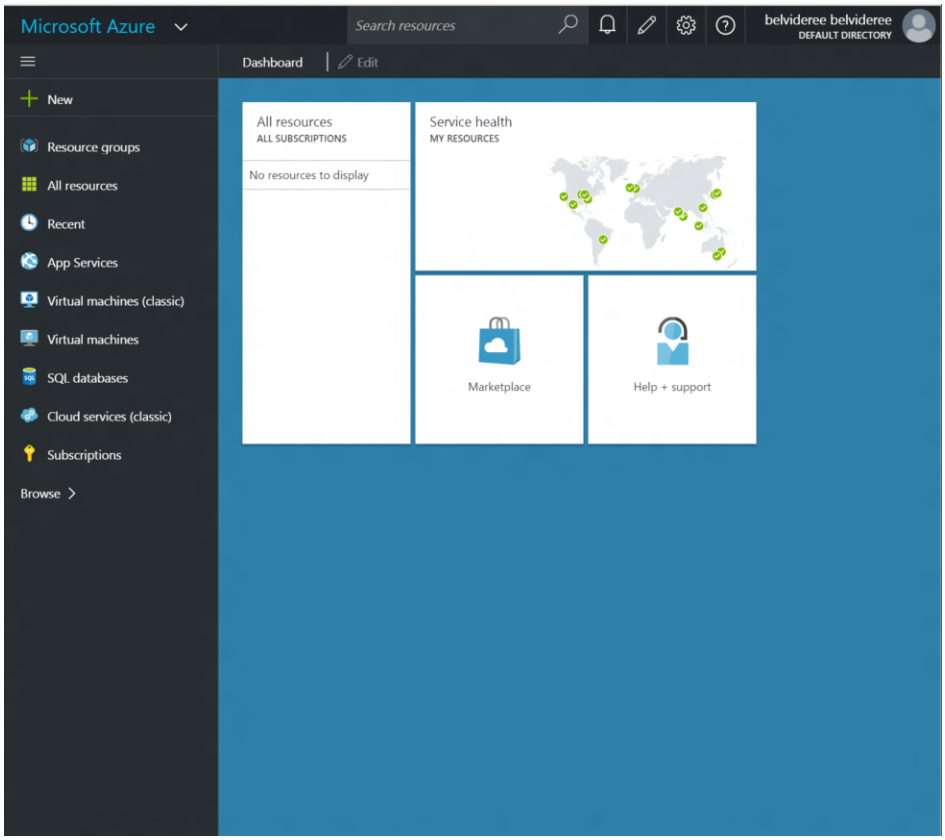


Figure 1-12: Azure portal

3. Review this page. You currently have no resources to display. Play around. Have fun. In the next chapters, we will create a variety of SQL Server resources. But first, let's get Azure PowerShell installed and connected to your Azure subscription.

Install Azure PowerShell 1.0

1. Download Azure PowerShell from http://aka.ms/webpi-azps and click Run to start the install of WindowsAzurePowershellGet.3f.3fnew.exe.
2. Click Yes in the User Access Control window.
3. Click Install in the Microsoft Azure PowerShell window, as shown in Figure 1-13.

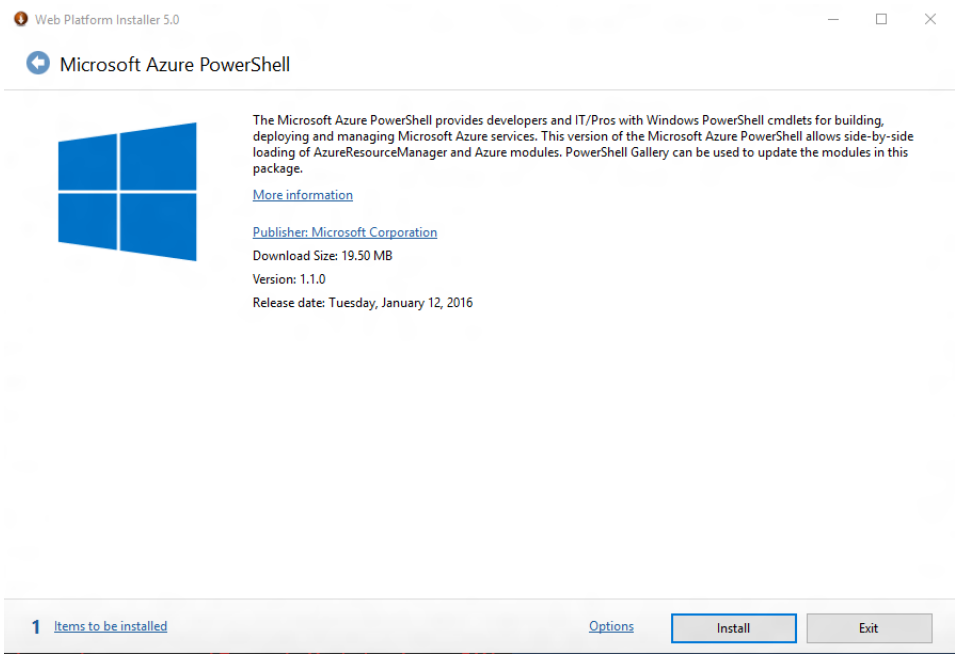


Figure 1-13: Windows Azure PowerShell Web Platform Installer 5.0

4. In the licensing window, shown in Figure 1-14, click I Accept.

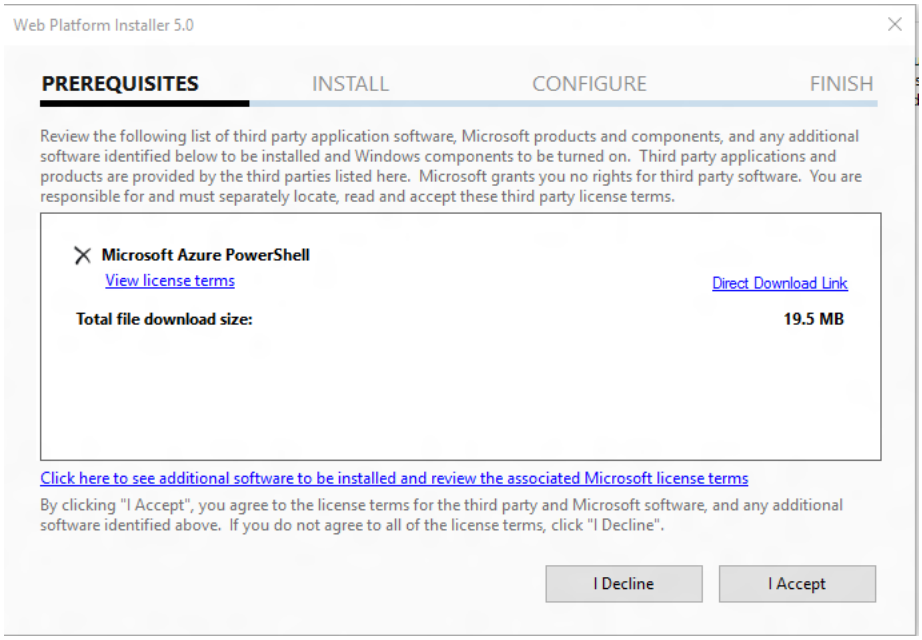


Figure 1-14: Web Platform Installer 5.0 licensing

5. When installation completes, click Finish, as shown in Figure 1-15.

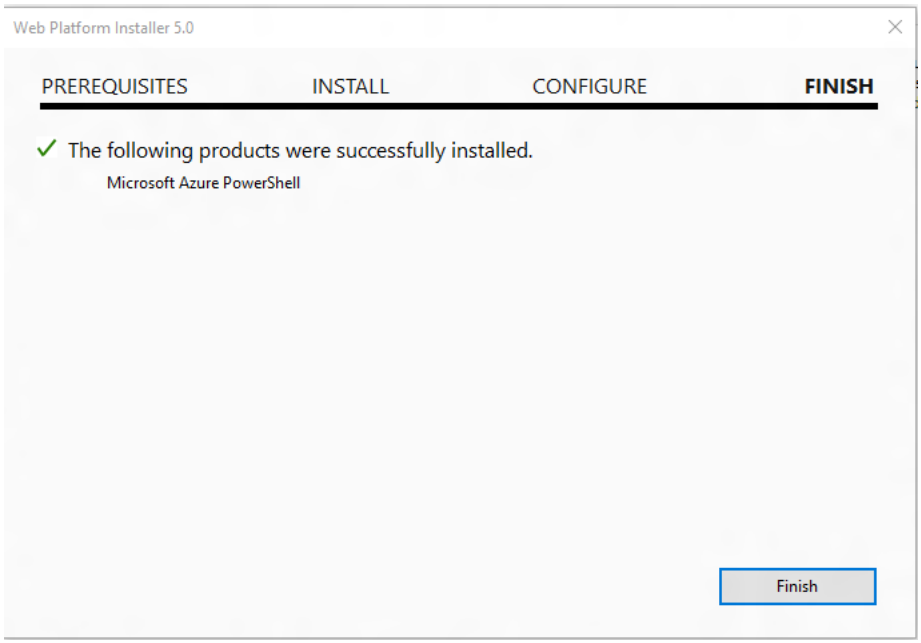


Figure 1-15: Microsoft Azure PowerShell installed

6. In the Web Platform Installer 5.0 window, shown in Figure 1-16, notice that Microsoft Azure PowerShell is installed and then click Exit to close this window.

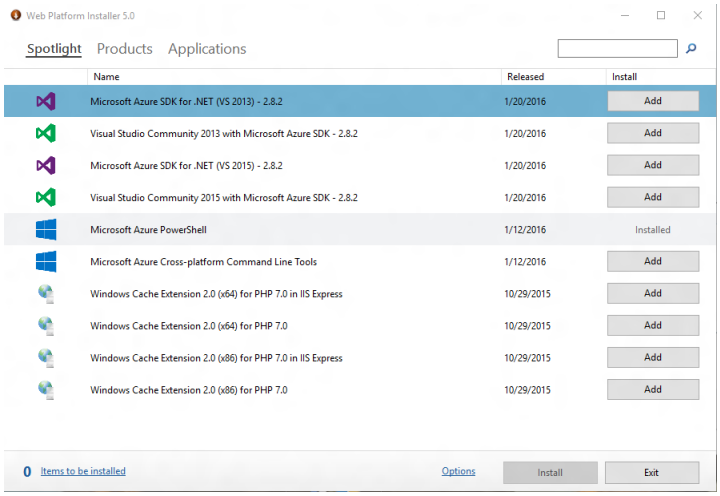


Figure 1-16: Web Platform Installer 5.0 window

Connect to Azure using PowerShell

1. Open Microsoft PowerShell, as shown in Figure 1-17.

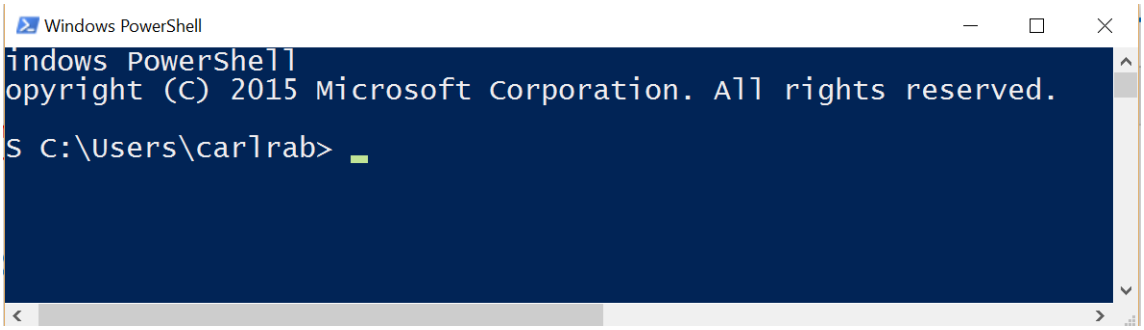


Figure 1-17: Windows PowerShell command prompt window

2. Run the following command to list available Azure PowerShell modules. The results are shown in Figure 1-18.

```
# To make sure the Azure PowerShell module is available after you install
Get-Module –ListAvailable
```

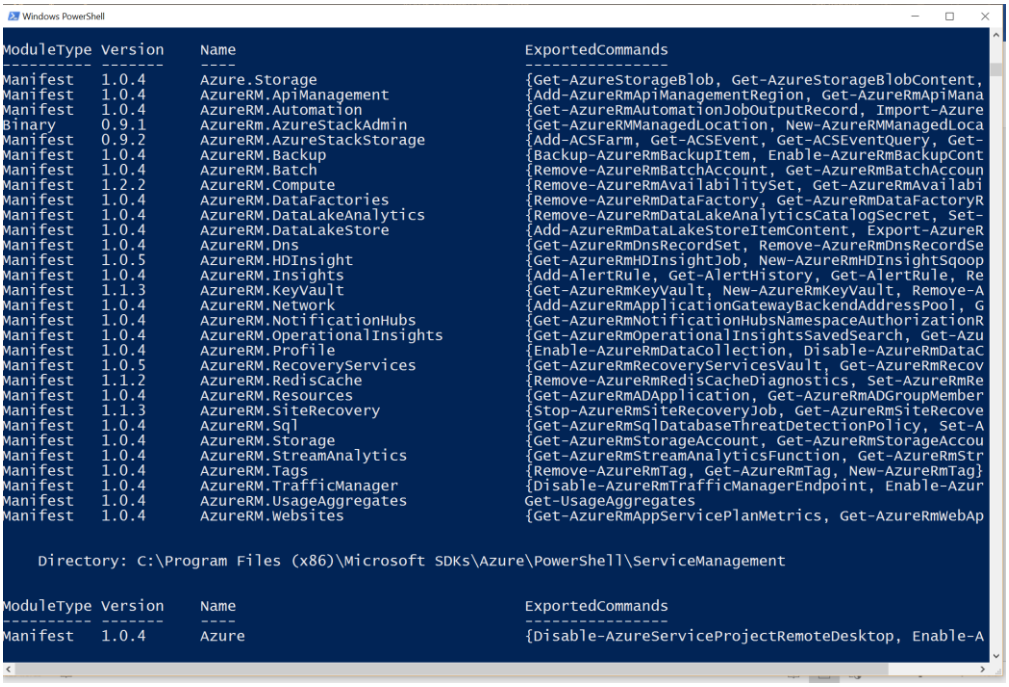


Figure 1-18: List of available modules

3. If the Azure PowerShell module does not appear, run the following command to import it and then verify the Azure module appears (as shown in Figure 1-19).

```
# If the Azure PowerShell module is not listed when you run Get-Module, you may need to import it
Import-Module Azure
# To make sure the Azure PowerShell module is available after you install
Get-Module –ListAvailable
```

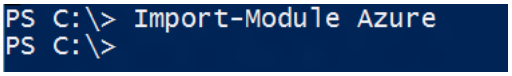


Figure 1-19: Import-Module Azure

4. If it still does not appear, you will need to either reboot Windows or add the Azure PowerShell path to the **$env:PSModulePath**. This is due to a known bug. See Install and configure PowerShell.

5. Start your login to Azure by running the following Azure PowerShell command and, when prompted, type either Y or N to enable or disable data collection to help Microsoft improve PowerShell cmdlets, as shown in Figure 1-20.

> **Note** If the email account for your trial subscription is your work or school account and you are logged in using that account, you can retrieve and use that credential. Again, see Install and configure PowerShell.

```
# To login to Azure Resource Manager
Login-AzureRmAccount
```

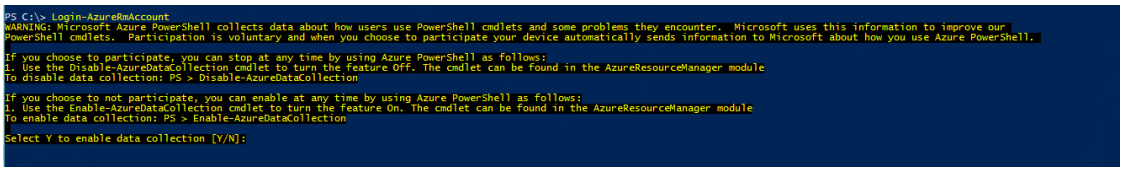


Figure 1-20: Data collection participation opt-in prompt

6. In the Sign In To Your Account window, shown in Figure 1-21, provide the name and password you used for your trial account and click Sign In.

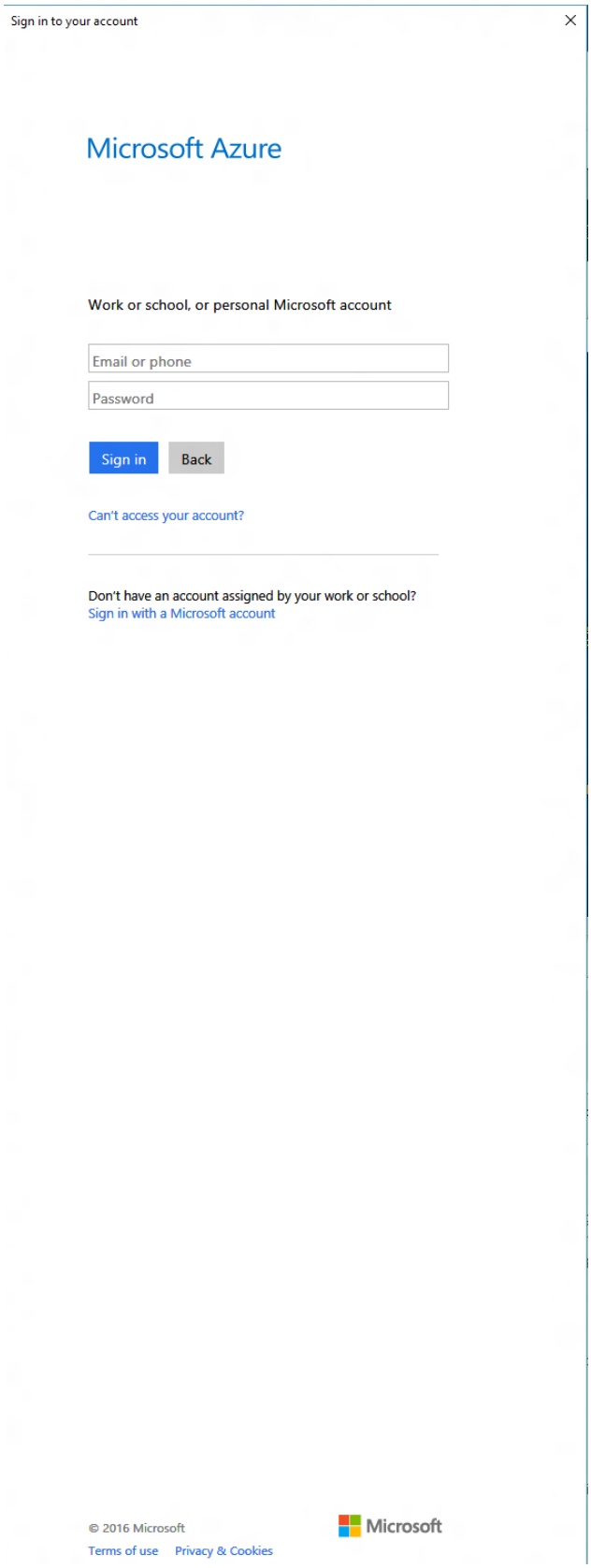


Figure 1-21: Sign In window

7. You are connected to Azure, and information about your Environment, Account, TenantId, SubscriptionID, and CurrentStorageAccount is returned, as shown in Figure 1-22.

   Notice that you do not have a storage account configured yet.

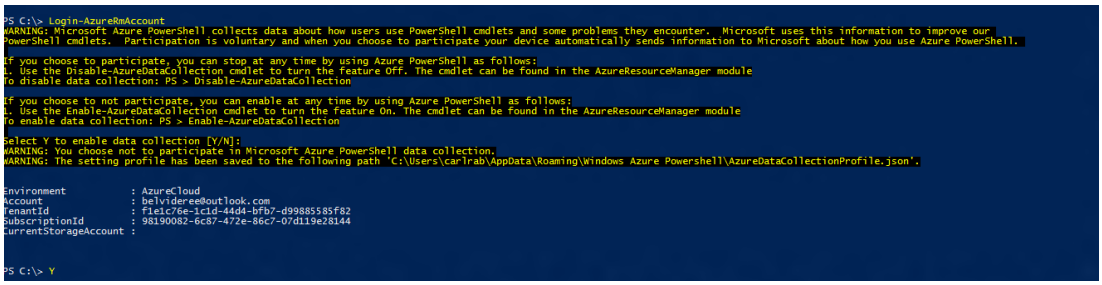


Figure 1-22: Connected to Azure using Azure PowerShell

8. Run the following command to return information about your subscription, shown in Figure 1-23.

```
# To return information about your subscription
Get-AzureRmSubscription
```

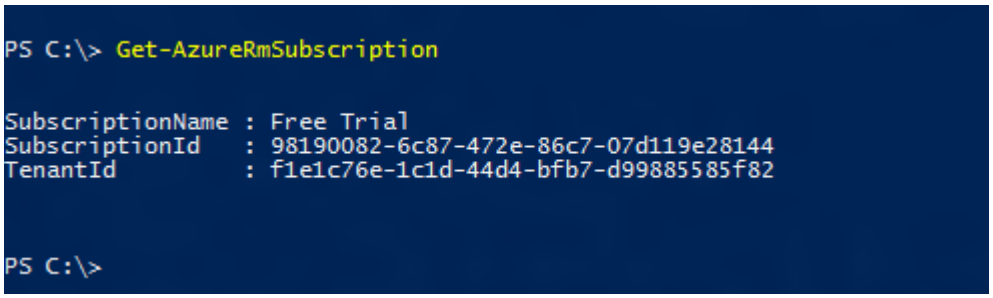


Figure 1-23: Data collection participation prompt

In the next chapter, we'll work with Azure VMs, storage accounts, and SQL Server in a VM. Let's go!

CHAPTER 2

Getting started with SQL Server in an Azure virtual machine

In this chapter, I will discuss the Microsoft Azure resources you will configure for your SQL Server Azure virtual machine and your options for applying a virtual machine image to your virtual machine. In particular, I will discuss your image choices and the SQL Server licensing implications of those choices. I also will discuss optimizing storage for SQL Server performance in your virtual machine and connectivity requirements. I will finish the chapter with a guided walk-through of the creation of an Azure SQL Server virtual machine using an image in the SQL Gallery that concludes by connecting to the virtual machine using Windows Remote Desktop Protocol (RDP) and SQL Server Management Studio.

Overview of an Azure virtual machine

An Azure virtual machine (Azure VM) consists of the following resources:

- Compute resources – the virtual machine object, starting with the virtual machine size. Azure virtual machines come in a variety of sizes, beginning with the A-series with 1 to 8 cores up through the G and GS series virtual machines with up to 32 cores. Each machine size within each series has different limits for the amount of memory, number of NICs, maximum number of data disks, size of cache, maximum IOPS and bandwidth, and maximum network bandwidth.
- Storage resources – the type, amount, and configuration of Azure blob storage for the virtual machine operating system and other software, such as SQL Server binaries and database files. The type and amount of storage is dependent on the machine size selected.
- Network resources – the network interfaces, virtual networks, public and/or private IP addresses, subnets within virtual networks, and other network components for the virtual machine. The network bandwidth is dependent on the machine size selected.

You have numerous decisions to make regarding how to configure these resources within Azure. In addition to these virtual machine components, you have a virtual machine image that is attached to the virtual machine containing, in our case, Windows and SQL Server. The image contains the operating system for the virtual machine you are configuring.

Selecting a deployment model

In Chapter 1, "Overview of SQL Server in Microsoft Azure," we discussed the fact that Azure supports two deployment models: the Azure Resource Manager (ARM) deployment model and the classic or Service Management deployment model. Although many resources operate without issue in both models, a few resource providers offer two versions of the resource due to architectural differences between the two models. The three resources that differ between the two models and interact with other Azure resources differently are the following resources, which are required resources in a virtual machine:

- Compute resources
- Storage resources
- Network resources

With the classic model, these virtual machine resources are handled individually. Each is deployed, managed, and monitored independently. As a result, this is a very complex environment.

With ARM, all of these resources are deployed and contained in a single resource group, enabling you to deploy, manage, and monitor all of these services as a group. You also can define dependencies between resources so they are deployed in the right order and apply access control to all services in the resource group. Additionally, ARM supports the use of declarative templates to define a deployment. Using ARM is recommended for simplicity and manageability.

> **Note** You cannot convert a virtual machine created using the classic model to ARM, even though you can add it to a resource group. You must continue to manage and monitor it through classic operations.

In the remainder of this chapter, I will discuss using ARM, and all examples in this chapter will use the Resource Manager deployment model. For a full discussion of these two models in the context of virtual machines, see Understanding Resource Manager deployment and classic deployment and Azure Compute, Network, and Storage Providers under the Azure Resource Manager.

Important When searching for information regarding Azure virtual machines on the Internet, be sure you are reading information for the appropriate deployment model or you might become confused.

Defining a resource group within an Azure region

With the Resource Manager deployment model, the first object you define when creating the deployment script for your virtual machine is the resource group. All virtual machine resources will be contained in that resource group. A resource group exists within a specific data center, and its resources almost always will reside within that data center.

Select a region that is geographically close to your application and your users. Minimizing network latency across your users, your application, and the virtual machine will improve the response time for your application and ultimately improve your users' satisfaction with their cloud experience.

Service availability varies by region. New services and resources are deployed in some regions before others. Before selecting a region, verify the services you wish to use are available in the region you plan to use. For example, for best performance for SQL Server on your virtual machine, you will want to use premium storage—which, at the time of this writing, is not yet available in all regions.

The resource group name must contain only letters, numbers, dashes, underscores, periods (except at the end), left parenthesis, and right parenthesis. It cannot have more than 64 characters.

Important Moving resources and data between data centers can be time-consuming and cost you data transfer fees. For a list of services by location, see Services by region.

Defining storage resources

The virtual machine requires storage resources for the operating system disk and for the SQL Server data and log files. You must decide whether to use standard or premium storage. Azure premium storage provides the highest performance for SQL Server in Azure VMs, providing up to 80,000 input/output operations per second (IOPS) and 2,000 MB per second disk throughput by using solid state drives (SSDs). You can choose to use premium storage for the operating system and all data disks. You also can choose to use premium storage for the SQL Server log files and the hot data only and use standard disks for the operating system and warm and cold data within SQL Server.

These storage resources are associated with a storage account, which is either a premium or a standard storage account. The storage account type and the storage must be of the same type.

You must provide either the name for a new storage account or the name of an existing storage account of the appropriate type for the storage disk types you will be using. The storage account name can include only lowercase letters and numbers and must be between 3 and 24 characters.

Defining network resources

The virtual machine requires a number of network resources for network connectivity. These include:

- One or more network interfaces (NICs)
- One or more virtual networks
- An internal address space

- One or more subnets
- A gateway

Virtual networks are logically isolated from one another in Azure. In a virtual network, you configure the IP address ranges, subnets, route tables, gateways, and security settings in a manner similar to a traditional network in your local on-premises data center. Virtual machines within the same virtual network can, by default, access one another. Connectivity from outside the virtual network, such as from within Azure or from the Internet, to a virtual machine requires a private or a public IP address. Network security groups (NSGs) enable you to allow or deny network traffic to a virtual machine using Access Control List (ACL) rules.

- Each virtual machine requires at least one network interface. You must provide an external public IP address name and configure its IP address assignment to be either dynamic or static (dynamic is the default). This name must be provided if you wish to communicate with the virtual machine from outside the virtual network. You optionally can provide a DNS domain name for a public IP address. This public IP address will be used either by clients to connect to the virtual machine from the Internet or for a gateway with a site-to-site VPN connection, either between virtual networks or to your on-premises network. Without a public IP address, the virtual machine is accessible only by other resources on the virtual network.
- Each virtual machine requires a virtual network. You must provide either the name for a new virtual network or the name of an existing virtual network. The name must begin with a letter or a number; end with a letter, number, or underscore; and contain only letters, numbers, underscores, periods, or hyphens. Also, the name must be between 1 and 80 characters.

  You also must provide the virtual networks internal address space. By default, Windows Azure allows you to configure an address space in the ranges of 10.0.0.0, 172.16.0.0, and 192.168.0.0. The default address space for a virtual machine configured through the Azure portal is 10.0.0.0/16.
- A virtual network must have at least one subnet defined. The default address space for the default subnet for a virtual machine configured through the Azure portal is 10.0.0.0/24, which provides 256 addresses within the subnet (choose a number between 16 and 32 for as many as 65,536 addresses and as few as 1 address within the subnet).

For more information on configuring Azure virtual networks, see Virtual Network documentation.

Defining compute resources

When defining compute resources, you start by selecting a virtual machine name, which is limited to 15 characters and cannot contain special characters, and then selecting a virtual machine size. Azure virtual machines are available in five different series (or SKUs): A, D, DS, G, and GS, with different sizes within each SKU. Each SKU and size has its own performance characteristics and price. For SQL Server deployments, you should choose either the DS or the GS series because these have the higher performance characteristics and support premium storage (SSDs).

The DS series supports up to:

- 16 cores
- 112 GB of memory
- 8 NICs
- 224 GB SSD for local storage
- 32 attached disks, standard or premium

- 576 GB cache
- 50,000 IOPS
- 512 MB per second bandwidth

The GS series supports up to:

- 16 cores
- 448 GB of memory
- 8 NICs
- 896 GB SSD for local storage
- 64 attached disks, standard or premium
- 4,224 GB cache
- 80,000 IOPS
- 2,000 MB per second bandwidth

For a SQL Server production environment, you should start with a DS11 or higher. For testing purposes and for just kicking the tires, I would recommend starting with at least a DS2. The higher the SKU, the more capabilities are available and the higher the cost is per month for the compute resources.

Because you will not know precisely what size you will need until you begin using the virtual machine with a real workload, it is important to understand that you can dynamically change the size of your virtual machine either using the Azure portal or from the command line. There are a few limits to be aware of when changing virtual machine sizes:

- You can change the size of a virtual machine within the same SKU without stopping the virtual machine. Azure performs a live migration to accomplish this, provided the size is available within the Azure region of the existing virtual machine.
- To perform cross-SKU migration for a virtual machine within a resource group, you must stop the virtual machine (and all virtual machines in the same availability set) before all SKUs become available to select for the migration.
- Changing the size or SKU of a virtual machine will cause the loss of content on the temporary drive. This has implications for storing TempDB and Buffer Pool Extensions within your virtual machine (see below for a complete discussion).

Warning When you create your initial virtual machine, you should validate that the range of machines in the series you plan to use is available in your data center. The highest-end SKUs are not always available in all data centers when they initially are released. Changing the size of a virtual machine within a data center generally is easy; changing between data centers is not. You can determine the range of machines in a data center using the Azure portal or using either PowerShell (Get-AzureLocation cmdlet) or the REST API.

Additional configuration options

Azure virtual machines provide a number of additional configuration options.

Configuring diagnostics

Optionally, you can configure your Azure virtual machine to record metrics for every minute for your virtual machine. These metrics are stored in a diagnostics storage account, and you can create alerts based on these metrics. For more information, see Azure Diagnostics for Azure Virtual Machines.

Configuring an availability set

Microsoft Azure periodically performs updates around the globe to improve the reliability, performance, and security of the host infrastructure that underlies virtual machines. Some updates do not require a reboot, but merely a 30-second pause. Some updates do require a reboot of virtual machines to apply the required updates to the infrastructure. The virtual machines are shut down while the infrastructure is patched, and the virtual machines are restarted. If your virtual machine is restarted, SQL Server will be restarted, client connections will need to be reestablished, and transactions in process will need to be retried (applications need to be designed to handle this scenario—think retry logic).

To protect against downtime due to planned or unplanned maintenance, Azure virtual machines support the concept of availability sets for virtual machines. With availability sets, you configure multiple machines in a single availability set with an optional load balancer (used for application tier scenarios) to ensure that at least one virtual machine is running at all times to ensure availability. For more on this, see Manage the availability of virtual machines and Planned maintenance for Azure virtual machines. To achieve SQL Server high availability with Azure virtual machines, use SQL Server AlwaysOn Availability Groups. For more information, see the following resources:

- SQL Server AlwaysOn Offering in Microsoft Azure Portal Gallery
- Configure AlwaysOn Availability Groups in Azure (GUI)
- Configure an ILB listener for AlwaysOn Availability Groups in Azure
- Deploy SQL Server AlwaysOn with an Azure Resource Manager template
- Extend on-premises AlwaysOn Availability Groups to Azure
- High availability and disaster recovery for SQL Server in Azure Virtual Machines

A detailed discussion of this topic is beyond the scope of this ebook.

> **Note** An availability set is required to meet the Azure SLA of 99.95 percent. The availability set of a virtual machine cannot be changed after it is created.

Provisioning the virtual machine with the Windows operating system and SQL Server

Now that we have discussed the virtual machine resources, let's talk about attaching a virtual machine image to the virtual machine object. To make this decision, you need to understand the two licensing model choices and how that impacts your choice of virtual machine image. There are two approaches:

- **Per-minute licensing**: You can pay for SQL Server by the minute. This is the licensing model you get when you select a preexisting virtual machine image preinstalled with a version of SQL Server from the Azure Virtual Machines Marketplace (also called the SQL Gallery). When using an image from the SQL Gallery, you cannot change the SQL Server edition without migrating to another

virtual machine running another edition. Those images include licensing of SQL Server in the pricing for the virtual machine.

At the time of this writing, preconfigured images for the Enterprise, Standard, and Web editions of the following versions of SQL Server and Windows are available in the SQL Gallery:

- SQL Server 2008 R2 SP3 running on Windows Server 2008 R2
- SQL Server 2012 SP2 running on Windows Server 2012
- SQL Server 2012 SP2 running on Windows Server 2012 R2
- SQL Server 2014 running on Windows Server 2012 R2
- SQL Server 2014 SP1 running on Windows Server 2012 R2
- SQL Server 2016 running on Windows Server 2012 R2
- The gallery also includes an image for a SQL Server 2014 Availability Group using AlwaysOn.

Note New images for Windows Server 2016 will be available soon after the release of Windows Server 2016.

- **Existing software assurance license**: You can use your own license to install SQL Server on a virtual machine created from a Windows image in the gallery, or you can upload a virtual machine to Azure with Windows and SQL Server already installed and attach it to your virtual machine. For details on bringing your own license, see License Mobility through Software Assurance on Azure.

  Using your own license, you can use other versions of the Windows operating system and other versions of SQL Server.

  - **Operating system versions**: To run SQL Server in an Azure virtual machine in a supported configuration, you must run Windows Server 2008 or newer. For more support details, see Microsoft server software support for Microsoft Azure virtual machines and Support policy for Microsoft SQL Server products that are running in a hardware virtualization environment.

    Generally, you should run Windows Server 2012 R2 or newer unless you have a specific compatibility reason for running an older version of the Windows operating system. Microsoft does not support an upgrade of the operating system of a Microsoft Azure virtual machine. Instead, you should create a new Azure virtual machine that is running the supported version of the operating system that is required and then migrate the workload.

Note Azure virtual machines include a license for using Windows Server in the Microsoft Azure environment.

- **SQL Server versions**: To run SQL Server in an Azure virtual machine in a supported configuration, you must run SQL Server 2008 or newer on Windows Server 2008 or newer.

Note Extended support for SQL Server 2005 ended on April 12, 2016. If you are migrating to Azure from SQL Server 2005, you can migrate and upgrade at the same time. We will discuss this scenario in Chapter 4, "Migrating a database to Azure."

Important You can domain join an Azure virtual machine to an Azure virtual machine domain controller using the same virtual network or to an on-premises domain controller by establishing a gateway to your on-premises network. For more information, see Virtual Network Documentation.

Configuring SQL Server in the virtual machine

Configuring SQL Server in the virtual machine consists of configuring and optimizing storage for database files and configuring the authentication mode.

Configuring SQL Server storage

Your virtual machine will come with the operating system installed on drive C and a temporary drive as drive D. Drive C will be configured for read/write caching. When using images from the SQL Gallery, the system databases also are configured to run on drive C. You will want to add storage for your data and log files, and you also may want to move the system databases to a non–operating system disk.

- For best performance for workloads requiring fast I/O, you should choose premium storage for all storage.
- For less demanding workloads, you can choose standard storage for the operating system and for some of the database files (such as for cold and warm data).
- With virtual machines, you generally will attach virtual hard disk (VHD) drives to your virtual machine. However, you have the option to store database files directly in Azure storage as blobs.

Configuring data and log disk storage

For data and log disks, use premium storage (SSDs) for the best performance. It is recommended that you use a minimum of two P30 disks (one for log files; one for data files and TempDB). Add drives as needed for additional performance and/or storage capacity. These additional drives can be on premium storage or standard storage or can be stored directly in Azure blobs. Premium storage will give you the best performance at the highest cost. Azure blob storage will give you virtually unlimited storage at a lower cost but also at lower performance.

> **Important** When configuring multiple data disks for SQL Server through the Azure portal, the settings discussed in this section on configuring data and log disks are set for you. Currently, all system files are on the C drive. These are all advanced options that generally are beyond the scope of this ebook. They are discussed in some depth to make you aware of them and encourage you to configure your high-performance virtual machines using the Azure portal until you fully understand these options. In general, these options only can be set manually using PowerShell or the REST API.

For maximum throughput and bandwidth with premium or standard storage, use disk striping.

- For Windows Server 2012 or later, use Storage Spaces. Set the stripe size to 64 KB for OLTP workloads and 256 KB for data warehousing workloads to avoid performance impact due to partition misalignment. In addition, set column count = number of physical disks. To configure a Storage Space with more than eight disks, you must use PowerShell (not Server Manager UI) to explicitly set the number of columns to match the number of disks. For more information on how to configure Storage Spaces, see Storage Spaces Cmdlets in Windows PowerShell.
- For Windows Server 2008 R2 or earlier, you can use dynamic disks (OS striped volumes), and the stripe size always is 64 KB. Note that this option is deprecated as of Windows Server 2012. For more information, see the support statement at Virtual Disk Service is transitioning to Windows Storage Management API.

If your workload is not log intensive and does not need dedicated IOPS, you can configure just one storage pool. Otherwise, create two storage pools, one for the log file(s) and another for the data file(s) and TempDB. Determine the number of disks associated with each storage pool based on your

load expectations. Keep in mind that different virtual machine sizes allow different numbers of attached data disks. For more information, see Sizes for Virtual Machines.

As stated above, when creating your virtual machine in the Azure portal using premium storage, storage pools are created for you automatically based on your I/O and throughput requirements. In the walk-through later in this chapter, you will see how this configuration is accomplished using the Azure portal (see Figures 2-23 and 2-24). A detailed discussion of configuring storage spaces using PowerShell or the REST API is beyond the scope of this ebook, but for more information see Premium Storage: High-Performance Storage for Azure Virtual Machine Workloads.

With multiple data drives in a storage pool, set the following additional performance optimization settings:

- Enable instant file initialization
- Enable lock pages in memory
- Specify simple recovery
- Set the number of columns = number of disks

Enable read caching on the data disks hosting your data files and TempDB only. If you are not using premium storage, do not enable any caching on any data disks. Again, when creating your virtual machine in the Azure portal using premium storage, these advanced performance optimization settings are configured for you (see Figure 2-26). To configure them yourself, you must use either PowerShell (see Set-AzureRMVMDataDisk and Set-AzureRmVMOSDisk) or the REST API.

> **Note** When provisioning your SQL Server virtual machine using an image from the SQL Gallery, you have the option to optimize the storage for transaction latency (using trace flags 1117 and 1118) or for query processing, analysis, and throughput (trace flags 610 and 1117). By default, the storage is not optimized for any particular workload. We will walk through configuring these options in the walk-through later in this chapter.

Configuring TempDB and Buffer Pool Extensions

For best performance, it is common to store TempDB and/or Buffer Pool Extensions on SSDs, particularly on a striped set of SSDs. The former improves the performance of workloads that use temporary objects heavily (for example, queries handling large recordsets, index rebuilds, row versioning isolation levels, temp tables, and triggers). The latter improves the performance of read workloads that working set doesn't fit in memory.

If you are not using premium storage, you should consider storing TempDB and/or Buffer Pool Extensions on SSDs. An Azure virtual machine includes temporary storage, which appears as the D drive on a Windows virtual machine. On the DS and GS series virtual machines, this temporary drive is an SSD. This drive should not be used to store data that you are not willing to lose.

> **Important** In general, you should not use this drive for TempDB or Buffer Pool Extensions because you will lose all data (all files and folders) on the temporary drive if you change the size or SKU of the virtual machine or if Microsoft moves the virtual machine for host machine maintenance purposes.

If you store TempDB and Buffer Pool Extensions in the temporary drive, SQL Server will not start when it is moved to a new virtual machine until you re-create the necessary folder structure. If you are not using premium storage, you can use the temporary drive if you handle the scenario in which the virtual machine moves to a different host. For detailed steps, see Using SSDs in Azure VMs to store SQL Server TempDB and Buffer Pool Extensions.

Configuring authentication for SQL Server

By default, when you install SQL Server from a SQL Gallery image, it is configured for Windows Authentication. If your virtual machine is not domain joined, you will want to configure it for SQL Server Authentication and configure SQL Server login accounts for administrators and for users to connect to your SQL Server instance.

> **Note** When provisioning your SQL Server virtual machine using the Resource Manager deployment model, you can choose to have this step done for you as part of the provisioning of SQL Server on the virtual machine.

Configuring connectivity to your Azure virtual machine

By default, when your virtual machine is configured from a gallery image, the only port that will be open on the firewall will be port 3389 to support Windows Remote Desktop connections. To enable SQL Server traffic to pass through the Windows firewall, you will need to open port 1433 for SQL Server traffic.

> **Note** When provisioning your SQL Server virtual machine using the Resource Manager deployment model, this step is done for you.

Selecting a provisioning method

You have several methods for provisioning the virtual machine, virtual network, and virtual machine image.

- Use the Azure portal: This is the easiest way to deploy a SQL Server virtual machine in Azure. As I have pointed out throughout the discussion above, configuring your first virtual machine through the Azure portal is by far the easiest method.
- Use Azure PowerShell: You can use Azure PowerShell to deploy a SQL Server virtual machine in Azure. However, although you can configure your entire virtual machine using Azure PowerShell cmdlets, not all tasks can be performed using cmdlets.
- Use PowerShell with a JSON template: You can use Azure PowerShell with a JSON template to deploy a SQL Server virtual machine in Azure with all of the features and configuration options supported by the Azure portal. The Azure Quickstart Templates site provides many examples to create virtual machines using PowerShell and JSON templates.

Walk-through: Getting started with SQL Server in an Azure virtual machine

In this walk-through, please follow along as we provision a SQL Server virtual machine using the Azure portal and then connect to this virtual machine using Windows Remote Desktop and SQL Server Management Studio.

Provisioning a SQL Server virtual machine using the Azure portal

Let's start by provisioning a SQL Server virtual machine using the Azure portal based on an image in the SQL Gallery.

1. If you are not still connected to the Azure portal, connect to the Azure portal now.
2. To create an Azure virtual machine from a gallery image using the Azure portal, click Virtual Machines**.** (You actually can click either Virtual Machines or Virtual Machines (classic) because we will select the deployment model after we select the image, but this blade will show only virtual machines deployed using the classic model.)

> **Note** You can either start by clicking Virtual Machines in the default blade or start with Marketplace in the dashboard. I am going to start from Virtual Machines in the default blade to limit the list of items in the marketplace from which to select virtual machine images.

3. In the Virtual Machines blade, click Add to open the Compute blade. See Figure 2-1.

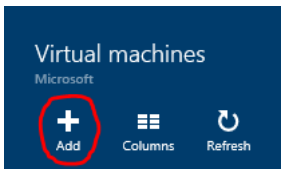


Figure 2-1: Add new virtual machine

4. On the Computer blade, scroll down to locate the Database Servers collection. See Figure 2-2.

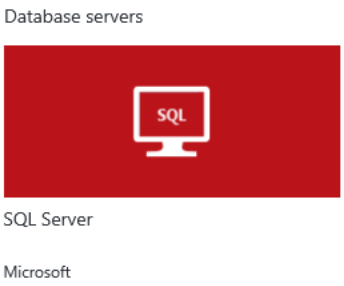


Figure 2-2: SQL Server image collection

5. Click SQL Server to open the SQL Server blade containing the list of available images for SQL Server in the SQL Server blade and click your selected image. For this ebook, I will pick the most recent SQL Server 2016 image, which is CTP 3. By the time you read this ebook, you will be able to select the released version of SQL Server 2016 or any other image you wish to use. See Figure 2-3.

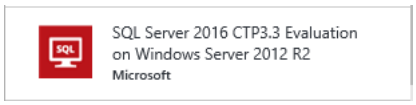


Figure 2-3: SQL Server 2016 image

> **Note** In this portal, we start creating our virtual machine with the virtual machine image and proceed to define the virtual machine components and where they will be created in subsequent steps in the wizard. This creation order enables the portal to assist us in selecting the appropriate resources for the virtual machine image selected.

6. Review the information about the image at the top of the page related to CEIP, the legal terms, and the Microsoft privacy statement. See Figure 2-4.

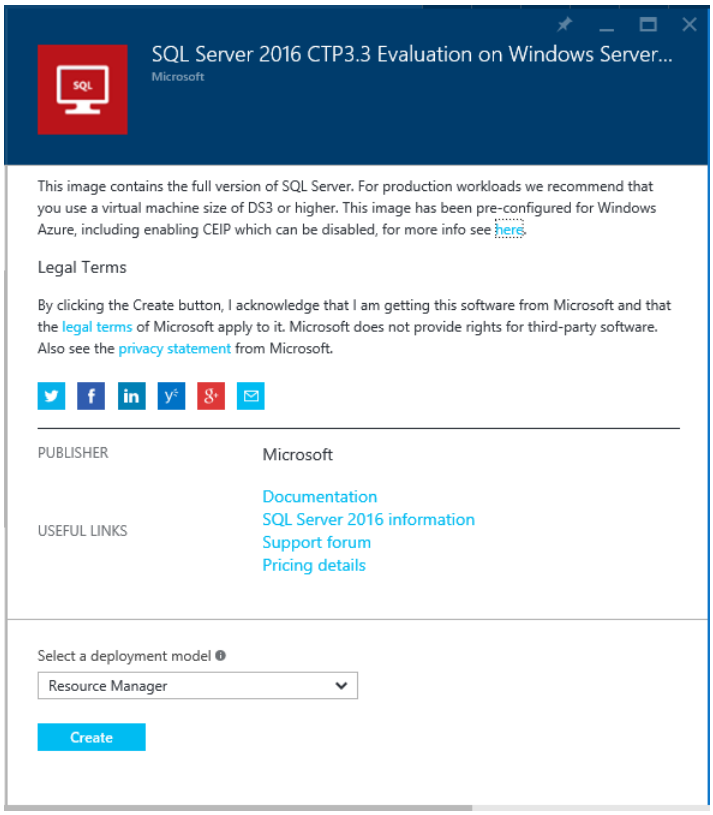


Figure 2-4: CEIP, Legal Terms, and privacy statement

7. Verify that Resource Manager is selected as the deployment model and then click Create to launch the Create Virtual Machine blade. See Figure 2-5.

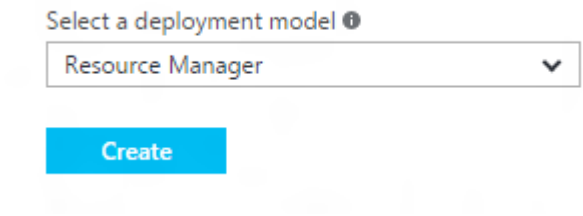


Figure 2-5: Selecting a deployment model

8. In the Basics blade, provide the requested configuration values for your subscription, resource group, resource group location, virtual machine name, and local administrator information. See Figure 2-6.

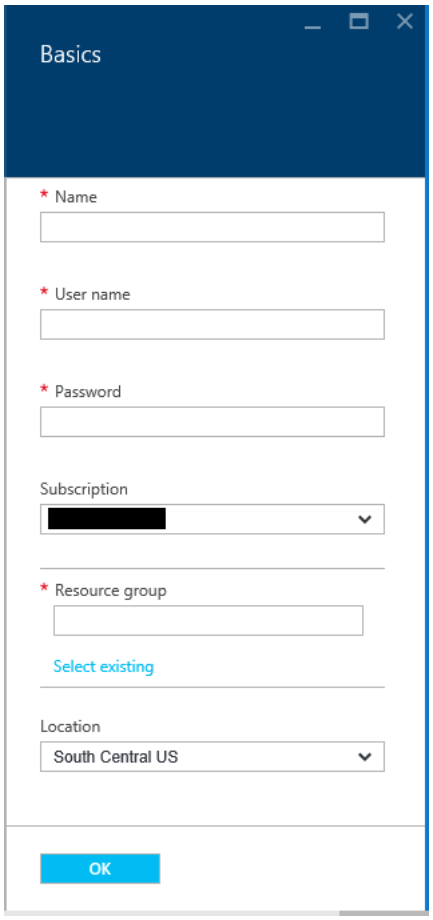


Figure 2-6: Basics blade

a. **Name**: Provide a name for your virtual machine. I will call mine carliaas.

b. **User name**: Provide a name for the local administrator account you will use to RDP into your virtual machine.

c. **Password**: Provide the password for this account.

Tip If you forget this password, Azure provides a method for you to reset this password on the blade for your virtual machine.

d. **Subscription**: Provide the subscription you will use for this virtual machine and all of its resources. The subscription with which you logged in will be selected. Use this subscription unless you have multiple subscriptions from which to choose.

e. **Resource group**: Provide the name of the resource group in which all of the virtual machine resources will be created. This can be the name for a new group, or you can select an existing group (if one exists) if you wish to manage multiple virtual machines (and other resources) in a single group. I will call mine carlrg1.

f. **Location**: Provide the Azure region for your resource group and its resources. If you select a region and discover as you walk through configuring the input values for creating this virtual machine that the size you wish to use is not available in the region you selected, you can return to this blade and choose another location.

See Figure 2-7.

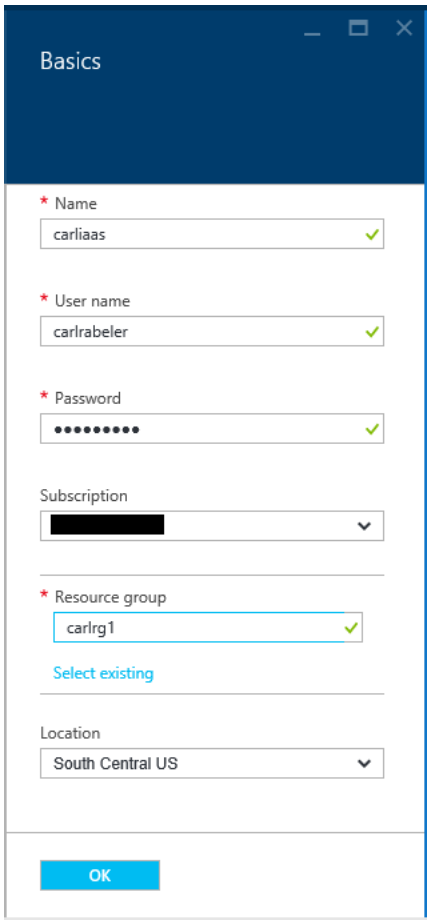


Figure 2-7: Basics blade completed

9. When complete, click OK to continue and choose your virtual machine size.

10. On the Size blade, select a virtual machine size. Review the recommended virtual machine sizes for your selected image that are available in your region (some regions will have newer hardware than others). See Figure 2-8.

    Notice the estimated retail prices per month for both the Azure machine and the SQL Server software. These prices will vary depending upon the type of subscription you have. You will see different prices for your trial subscription, but you will be paying nothing during the period of your trial.

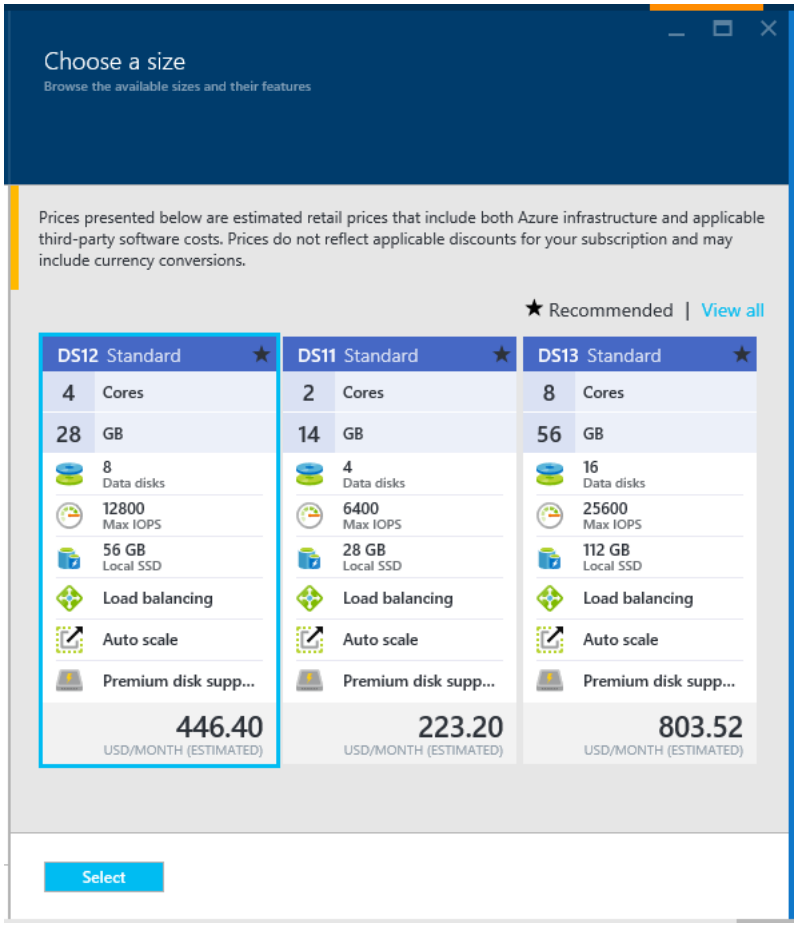


Figure 2-8: Size blade

11. Click View All to see all of the available virtual machines and their estimated monthly costs.

    A large number of virtual machine sizes are available. I am seeing 42 different sizes from which to choose.

12. Click Recommended to restrict your choices to recommended virtual machine sizes.

13. Select a size for your virtual machine. Notice in Figure 2-8 that I am selecting a DS12.

    Remember that you will incur no charges during your trial period and that if you wish to convert your trial subscription to a paid subscription, you can downsize later to save money until you need more hardware resources. Also, if you are using your MSDN subscription, remember that you have monetary limits and need to monitor your usage at https://account.windowsazure.com/subscriptions.

> **Tip** Shut down your virtual machines when you are not using them to save money.

14. Click Select to continue and configure the remaining storage and network resource settings on the Settings blade.

15. On the Settings blade, review the default settings that are recommended for the workload and for the virtual machine size you have selected. See Figure 2-9.

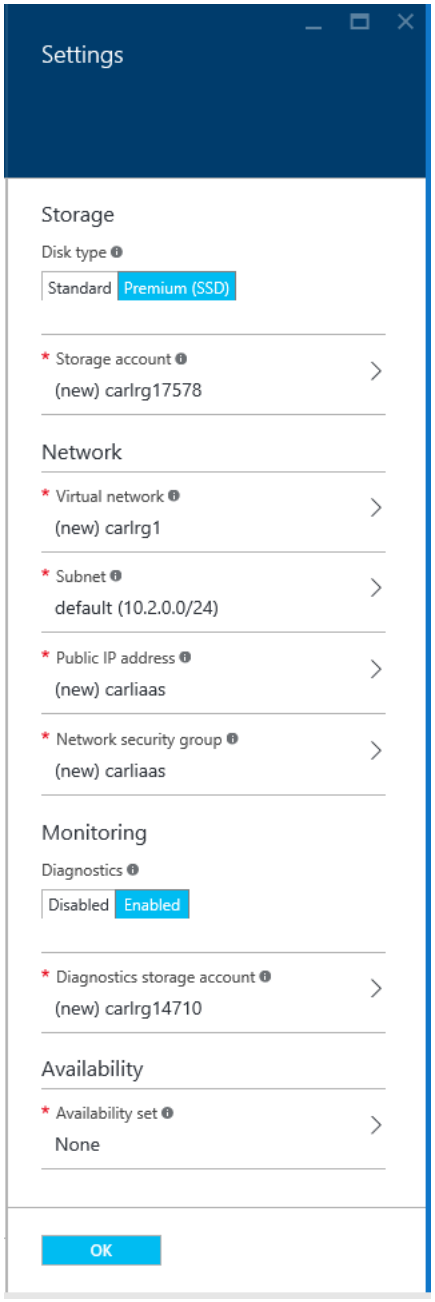


Figure 2-9: Storage Settings blade

You could click OK on this blade to continue without changing any of the defaults. However, I will walk you through some of the nuances and decision points that are useful to understand.

16. Define the storage type and storage account name for the operating system drive for your virtual machine. Before continuing, notice the default values for storage type, storage account, and storage account name.

    a. **Storage – Disk Type**: Premium storage is recommended for SQL Server workloads, and you will notice that this is the default selection. Your choice here will determine the disk type for the operating system disk.

Important If you select Premium storage, you will be able to configure additional premium-type disks when configuring SQL Server settings in the next step and have Azure stripe them for you using Windows Storage Spaces as the location for your SQL Server data files. If you select Standard, you will have to add drives later (premium or standard) and configure striping yourself.

b. **Storage – Storage Account**: Provide the name for the storage account you will be using. You can either accept the default name for a new storage account or click to change this default value. If changing, you can either select an existing storage account if one of the right type exists (it must be a premium account if using premium storage) or create a new storage account with a name of your choice. I am choosing my own storage account name, carlrg1iaasprem.

Notice that a premium storage account type is the configuration choice for the new storage account if you selected premium storage as the disk type. See Figure 2-10.

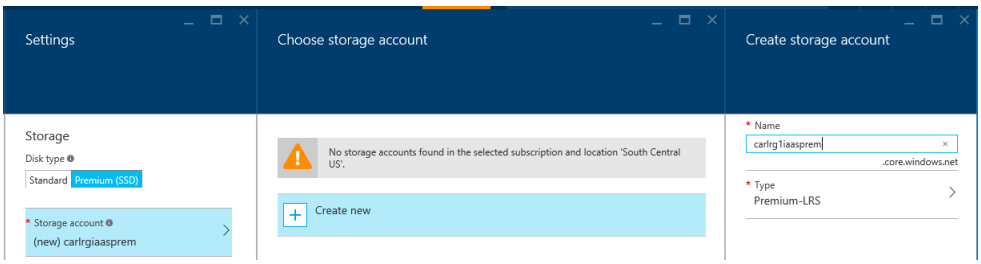


Figure 2-10: New premium storage account

17. Click OK.

18. Define the virtual network, subnet, and public IP address for the network interface. Before continuing, notice the default values for these resources (we will discuss the network security group in just a moment). You can choose to accept the default values for each setting here.

a. **Virtual Network**: You can either accept the default virtual network name, address space, subnet name, and subnet address range or click to change (or view) these default values. If changing, you can either select an existing virtual network if one exists or create a new virtual network with the values of your choice. I will change the virtual network name to carlrg1iaasvnet and then click OK. See Figure 2-11.

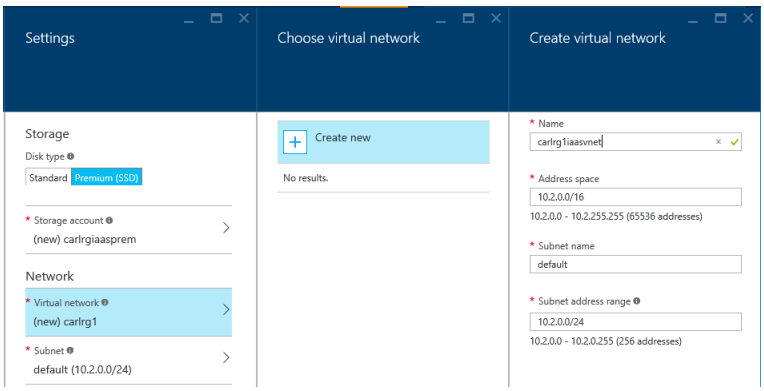


Figure 2-11: New virtual network

b. **Subnet**: This option just displays the subnet name and address range without having to open the virtual network properties and modify the subnet name and range.

c. **Public IP Address**: As discussed earlier, a public IP address is required if you want to communicate with the virtual machine from outside the virtual network. By default, a public IP address is defined with a default name and dynamic TCP/IP address assignment. I changed the name to carliaas. See Figure 2-12.

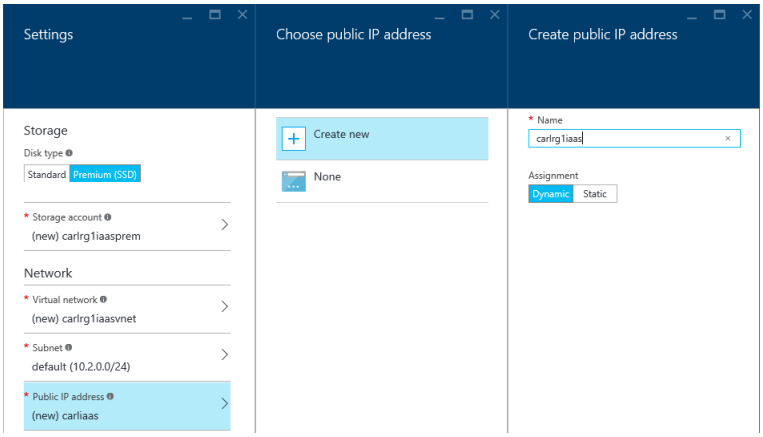


Figure 2-12: Defining a public IP address

19. Define a network security group. Unlike the previous settings that will be used to create resources in Azure for the virtual machine, a network security group is a set of firewall rules you specify here to open firewalls within the Windows operating system inside the virtual machine you are creating. By default, an inbound firewall rule allowing traffic over TCP port 3389 will be configured within the new virtual machine to enable Windows Remote Desktop (RPC) connectivity. See Figure 2-13.

Note On the SQL Server settings blade, we will have the option to create a firewall rule for the SQL Server service.

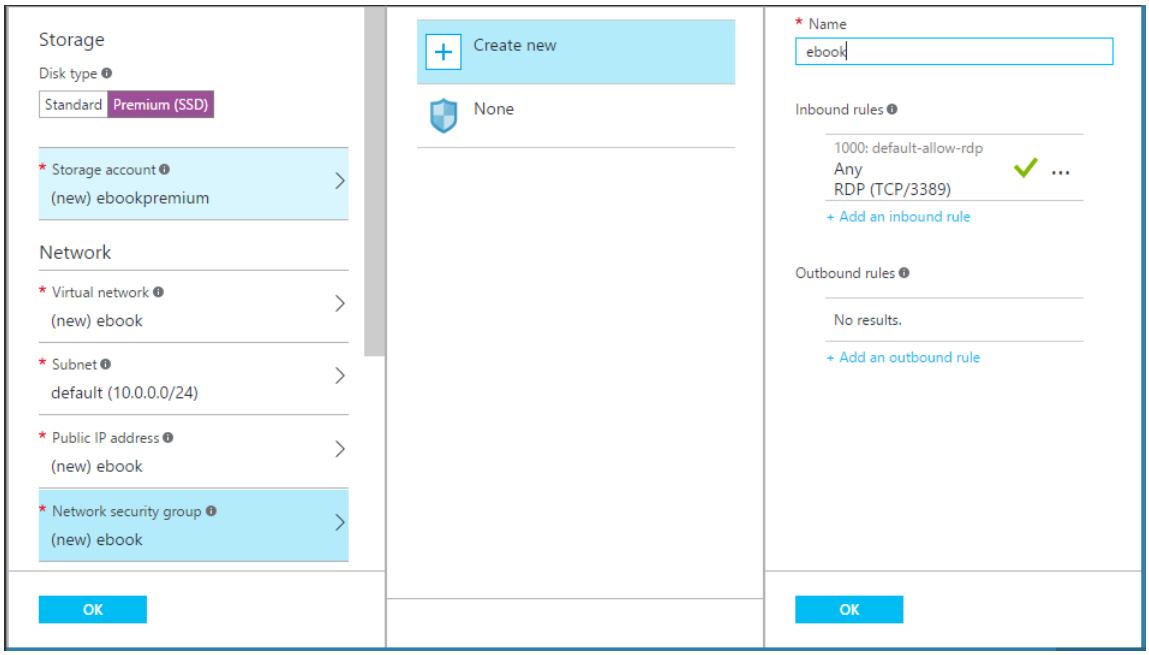


Figure 2-13: Inbound firewall rule for RDP traffic

20. Configure monitoring. Azure provides the option to record metrics for your virtual machine into a new standard storage account (rather than recording it into premium storage at a higher cost). You can configure the metrics you wish to have recorded either in the Azure portal or programmatically after the machine is created. Think of this as a performance monitor trace. Diagnostics are enabled by default. See Figure 2-14.

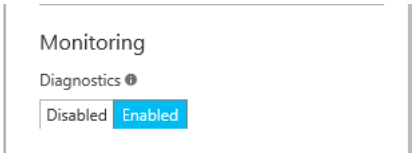


Figure 2-14: Monitoring enabled

21. I will change the name for this storage account to carlrg1metrics and click OK. If you had an existing standard storage account, you might want to use a single storage account for multiple virtual machines. See Figure 2-15.

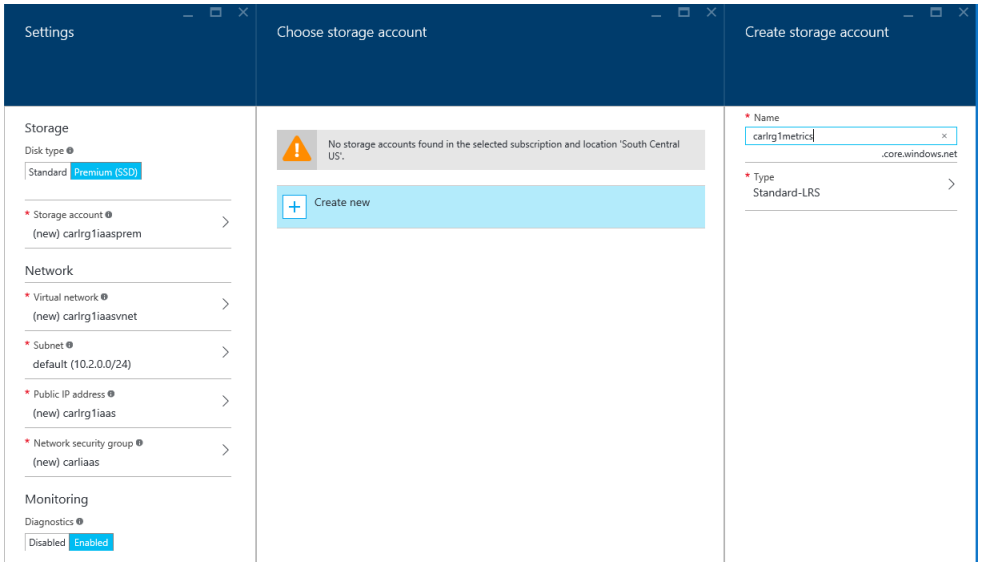


Figure 2-15: Diagnostics storage account

22. Define your availability set. No availability set is defined by default. As we discussed earlier, configuring an Azure virtual machine running SQL Server for high availability using availability sets is beyond the scope of this ebook. If you are curious, click through the options here.

23. When finished, review your input values. See Figure 2-16.

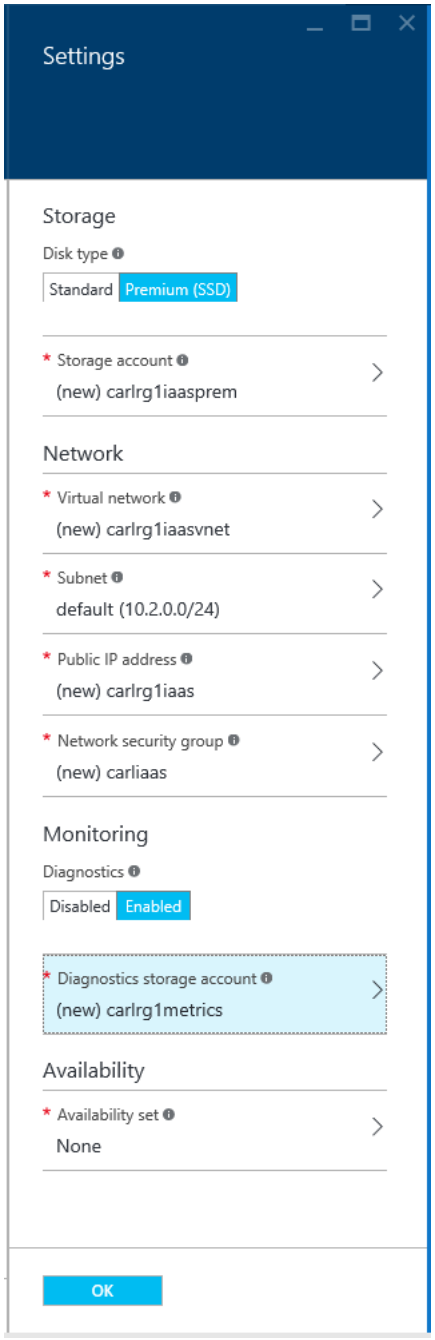


Figure 2-16: Storage Settings blade, completed

24. Click OK to proceed to the Configure SQL Server Settings blade.

25. On the SQL Server Settings blade, you have a number of configuration options I will walk you through. See Figure 2-17.

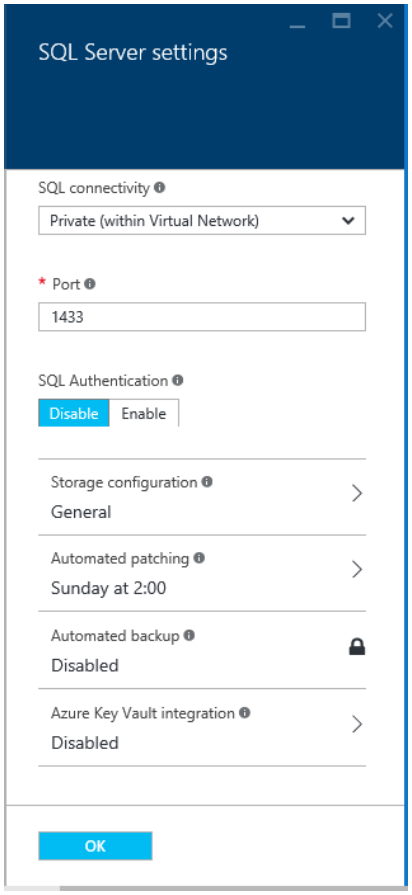


Figure 2-17: SQL Server Settings blade

26. Select the type of connectivity to SQL Server in your virtual machine. By default, SQL Server traffic from outside the virtual machine will not be allowed through the Windows firewall. See Figure 2-18.

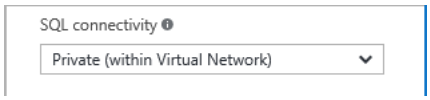


Figure 2-18: Default SQL connectivity configuration

27. Changing this value from Private (Within Virtual Network) to Public (Internet) will define an inbound firewall rule allowing traffic on the port defined for your SQL Server instance (port 1433 by default). See Figure 2-19.

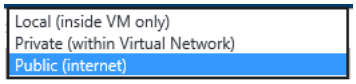


Figure 2-19: Defining inbound firewall rule allowing SQL Server traffic over the SQL Server port

28. Provide the port number on which you wish SQL Server to listen. For our walk-through, I will leave the port on which SQL Server will listen to its default value of 1433. See Figure 2-20.

Figure 2-20: SQL Server port number

29. Configure SQL Server Authentication. Because our virtual machine and many SQL Server virtual machines running in SQL Server are running in virtual machines that are not domain joined, you must select SQL Authentication to enable users to authenticate to SQL Server using SQL logins. Select Enable. Notice that when you do so, the name and password you provided for the local administrator also will be used as the SQL Server login. This account will be given system administrator privileges. See Figure 2-21.

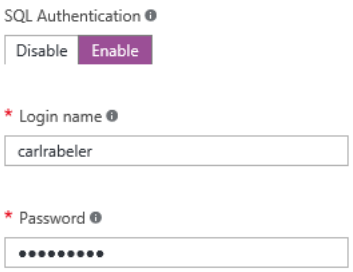


Figure 2-21: Configure SQL Authentication

30. Configure storage. By default, SQL Server will be configured with a single 1-terabyte (TB) disk using premium storage and configured for a general workload. See Figure 2-22.

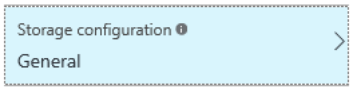


Figure 2-22: Current storage configuration

31. Click the Storage Configuration pane. Here, you will provide the desired performance, storage size, and workload optimization values. Notice you have three slider bars that display the IOPS, Throughput (MBps), and Storage Size (TB) you will get from a single data disk on premium storage. See Figure 2-23.

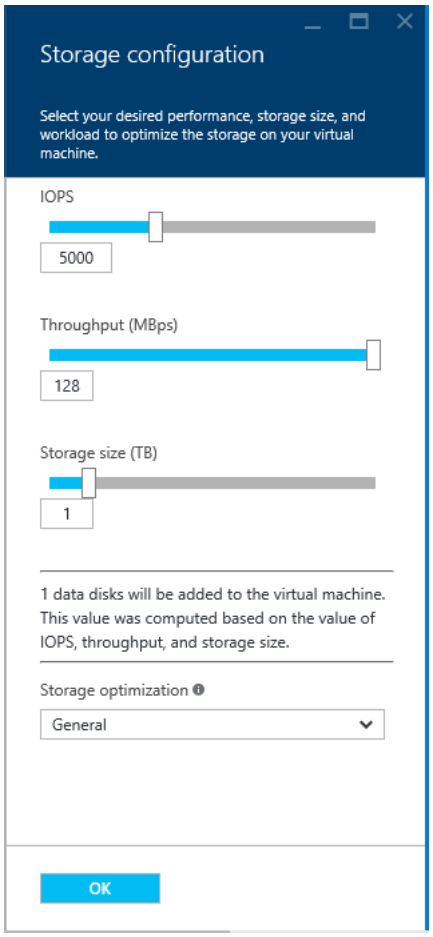


Figure 2-23: Storage Configuration blade

32. If you want more storage or more IOPS, move the sliders to the right. If I select 12,800 IOPS, notice that three data disks will be added. These disks automatically will be configured as a striped set in a Windows Storage Space to provide this level of IOPS. See Figure 2-24.

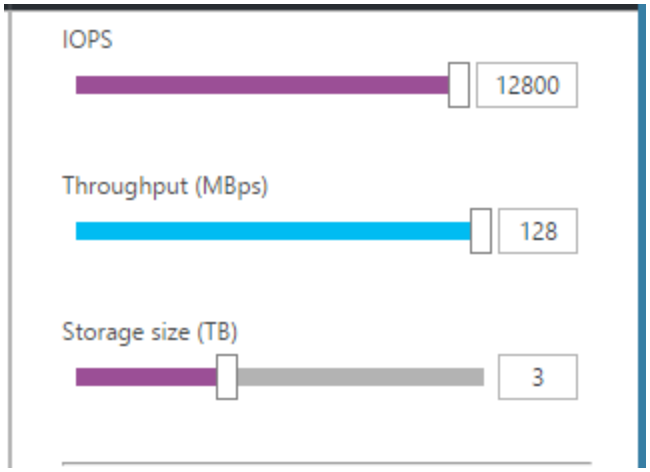


Figure 2-24: 12,800 IOPS

Tip If you want more IOPS than are available in this slider, you will need to select a higher-end virtual machine, such as a DS13. See Figure 2-25.

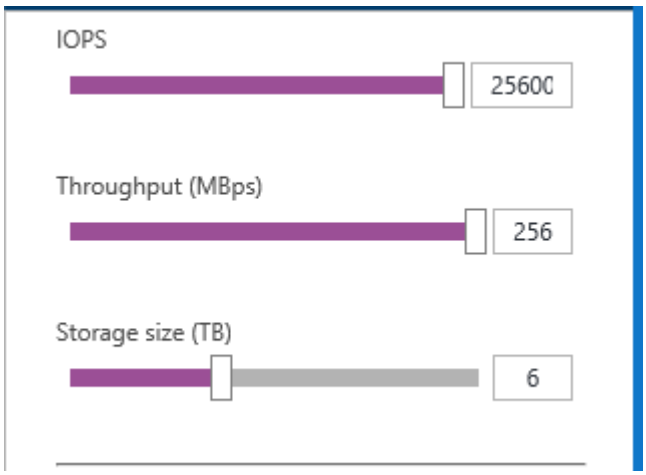


Figure 2-25: 25,600 IOPS

33. On this blade, you also can select to optimize storage for a particular type of workload. For our walk-through, I will select the default setting of General. See Figure 2-26.

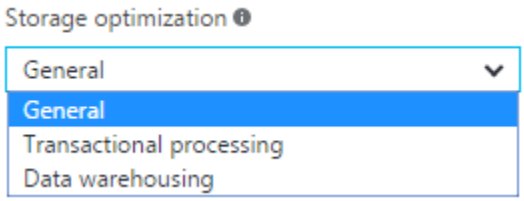


Figure 2-26: Storage optimization

34. When finished, review your settings. See Figure 2-27.

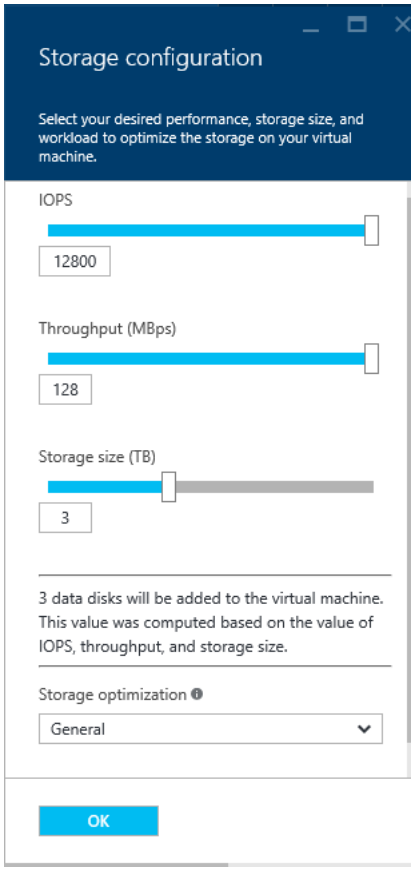


Figure 2-27: Completed Storage Configuration blade

35. When done, click OK to return to the SQL Server Settings blade.

36. Configure automated Windows and SQL Server patching. Click this option to review your choices to have Azure automatically apply Windows and SQL Server patches. Notice that patching is enabled by default and that the patch window is Sundays at 2 A.M. with a patch window of 60 minutes. You can change any of these options. I will keep the default settings. See Figure 2-28.

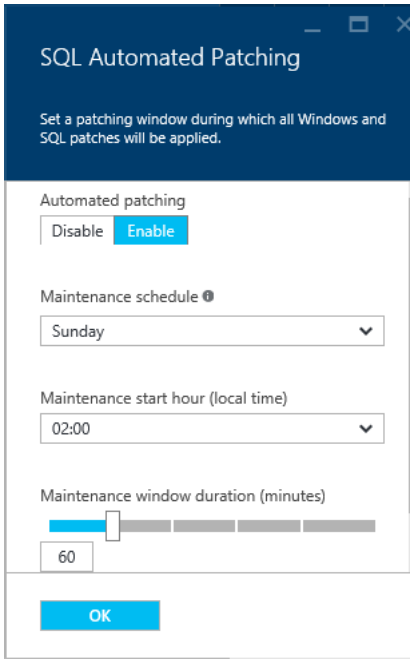


Figure 2-28: SQL Automated Patching

37. Configure automated backups. By default, automated SQL Server backups are disabled. However, you can choose to enable automated backups. Currently, this option is not available for SQL Server 2016. However, it will be when SQL Server 2016 is released (or soon after). See Figure 2-29.

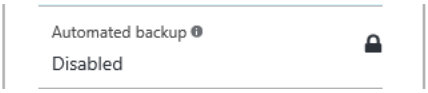


Figure 2-29: Automated backups

38. Configure Azure Key Vault integration. By default, this is disabled. Discussion of Azure Key Vault to encrypt data in the cloud is beyond the scope of this walk-through. For more information, see Key Vault. See Figure 2-30.

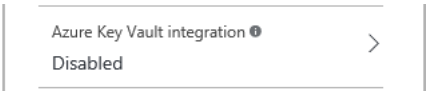


Figure 2-30: Azure Key Vault integration

39. When finished, click OK to proceed to the Summary blade.

40. The summary blade enables you to review all of your configuration values before you ask Azure to create this virtual machine. See Figure 2-31.

Note When you start creating machines with PowerShell cmdlets and JSON templates, these are the values you will provide when creating the necessary resource providers and custom scripts.

Figure 2-31: Summary blade

41. Click OK to have Azure create the virtual machine.

42. On the dashboard, notice that your virtual machine is being created. See Figure 2-32.

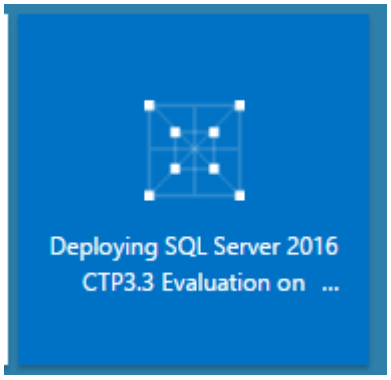


Figure 2-32: Dashboard showing virtual machine being deployed

Viewing and configuring your virtual machine in the Azure portal

1. After the provisioning of your virtual machine is complete, click All Resources in the default blade and, in the All Resources blade, view the resources that were created. See Figure 2-33.

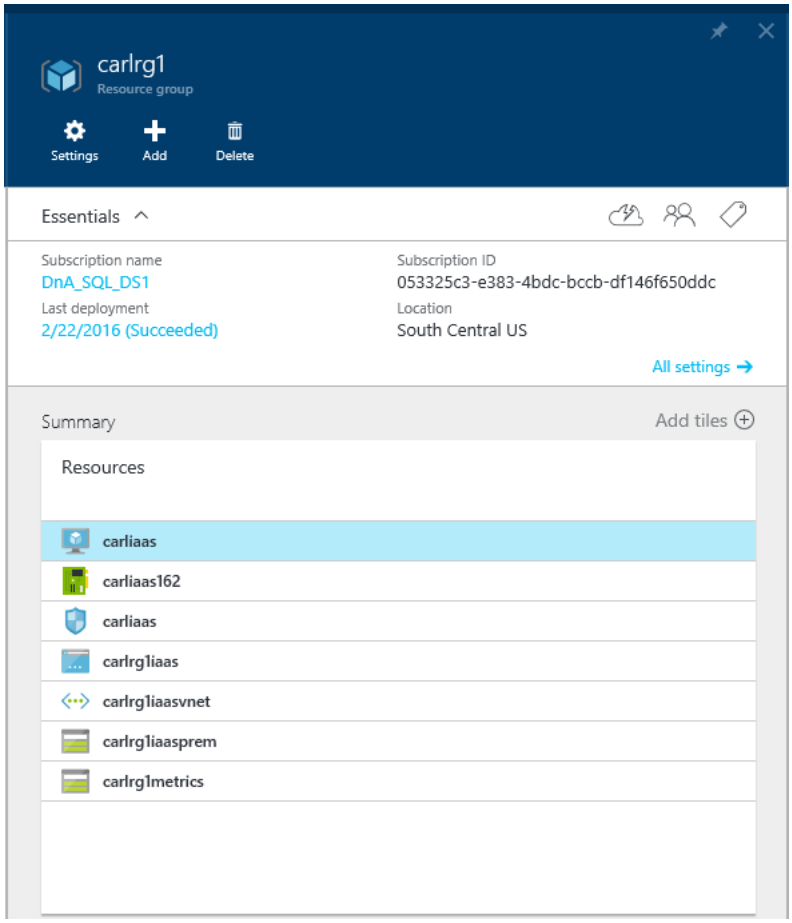


Figure 2-33: All Resources blade

2. Click your virtual machine and review the information displayed about your virtual machine on the blade for your virtual machine and on its Settings blade. See Figure 2-34.

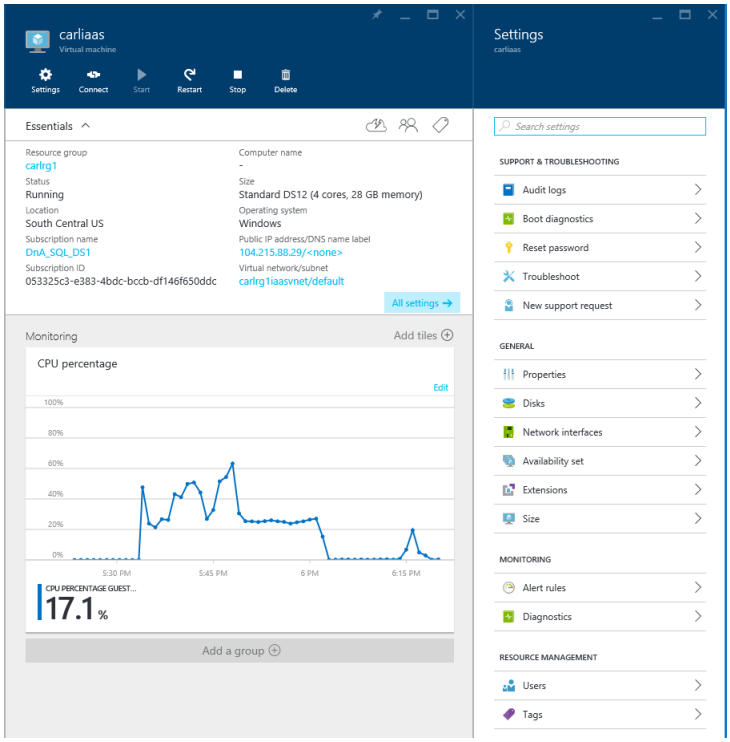


Figure 2-34: Virtual Machine blade

3. Locate the public IP address on this blade. You will use this public IP address to connect to your SQL Server instance in your virtual machine. See Figure 2-35.

   Notice that no DNS name label is defined.

Figure 2-35: Public IP address

4. Click the IP address and, in the Settings blade, click Configuration to open the public IP address Configuration blade. See Figure 2-36.

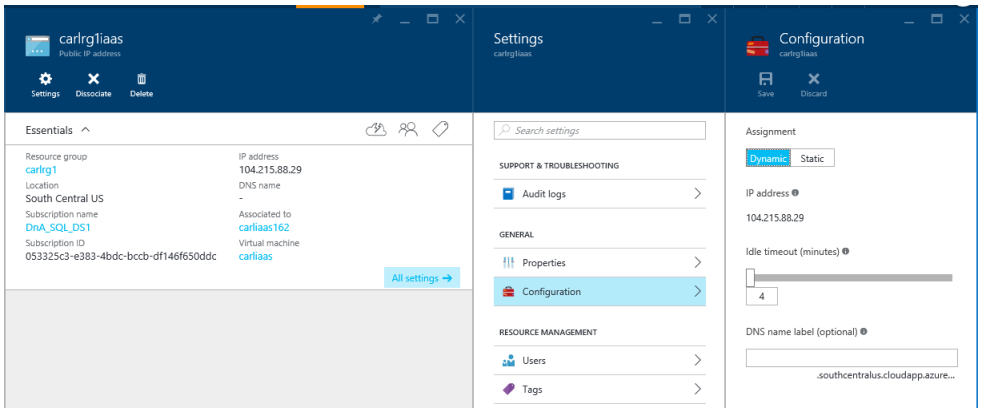


Figure 2-36: Public IP address Configuration blade

5. In the DNS Name Label (Optional) text box, enter a DNS name. This will enable you to connect to your virtual machine and SQL Server resources using its DNS name in addition to connecting using the public IP address. I will use my machine name as my DNS label so the fully qualified

DNS name for my virtual machine will be carliaas.southcentralus.cloudapp.azure.com. See Figure 2-37.

Note It will take a bit of time for this DNS name to be propagated and available to you.

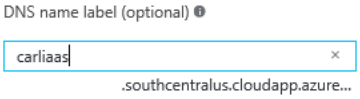


Figure 2-37: DNS name label

6. Click Save on the Configuration blade.
7. Close the Configuration, Settings, and Public IP Address blades.

Tip You can close all three blades at once by closing the Public IP Address blade.

Connecting to your virtual machine

1. On your virtual machine's blade, click Connect. See Figure 2-38.

   An RDP file is downloaded to your Downloads folder.

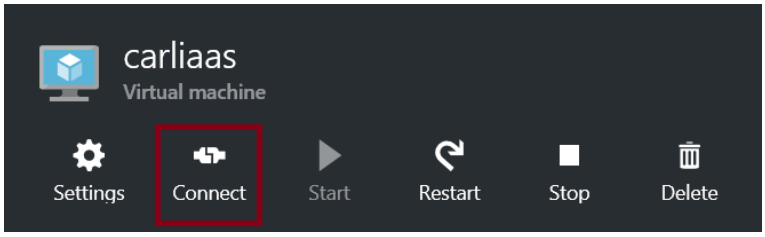


Figure 2-38: Virtual machine actions

2. Open the RDP file and click Connect in the Remote Desktop Connection window. See Figure 2-39.

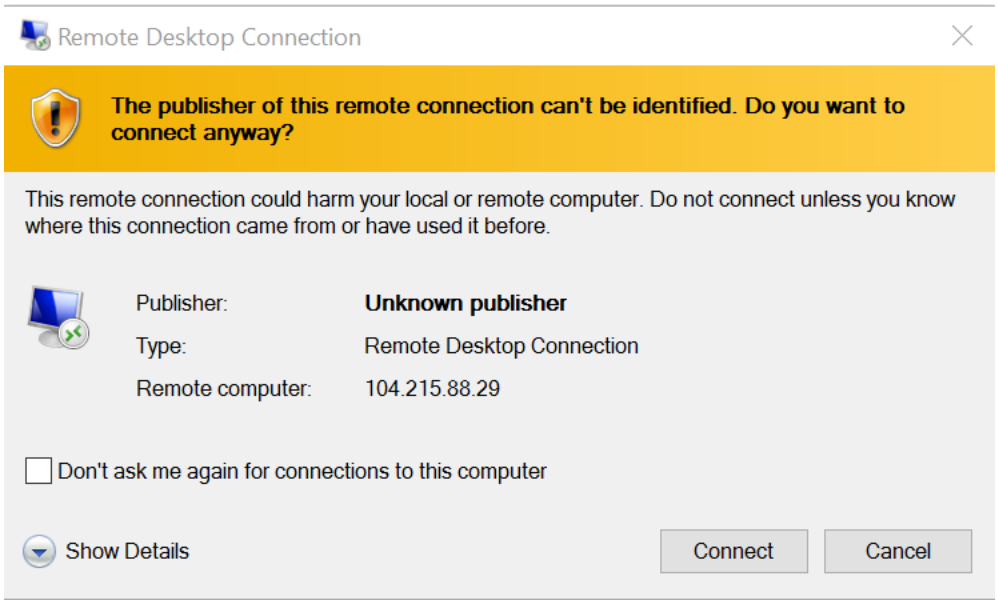


Figure 2-39: Remote Desktop Connection

3. In the Windows Security window, provide the local administrator credentials you specified when defining your virtual machine and then click OK. See Figure 2-40.

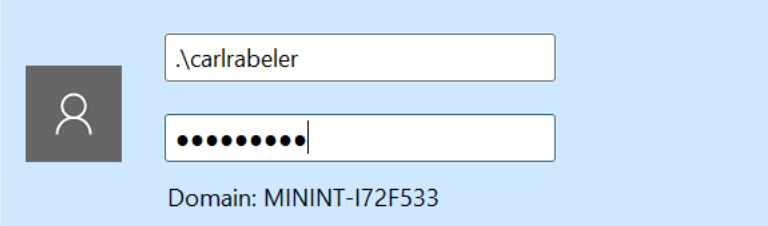


Figure 2-40: Windows Security window

4. In the Remote Desktop Connection window, click Yes to connect to your virtual machine despite the fact that your new virtual machine's certificate is not from a trusted certifying authority.
5. You are connected to your virtual machine, and the sysprep process completes.
6. Click No in the Networks pane.
7. At some point, your will be prompted to install Windows updates. Click Install and then click Install Updates.
8. In the Tools menu of Server Manager, click Computer Management.
9. Click Disk Management to view your disk drives—in particular, your Windows Storage Spaces drive.
10. Open File Explorer and review the files on your F drive. Notice that this drive is configured for storing your SQL Server data and log files for your user databases.

Connecting to SQL Server within the virtual machine

1. Open Microsoft SQL Server Management Studio in your virtual machine.
2. In the Connect to Server dialog box, click Connect to connect to your instance in Object Explorer using Windows Authentication.
3. Right-click your server in Object Explorer, click Properties, and then click Security to verify that SQL Server is configured for SQL Server and Windows Authentication. See Figure 2-41.

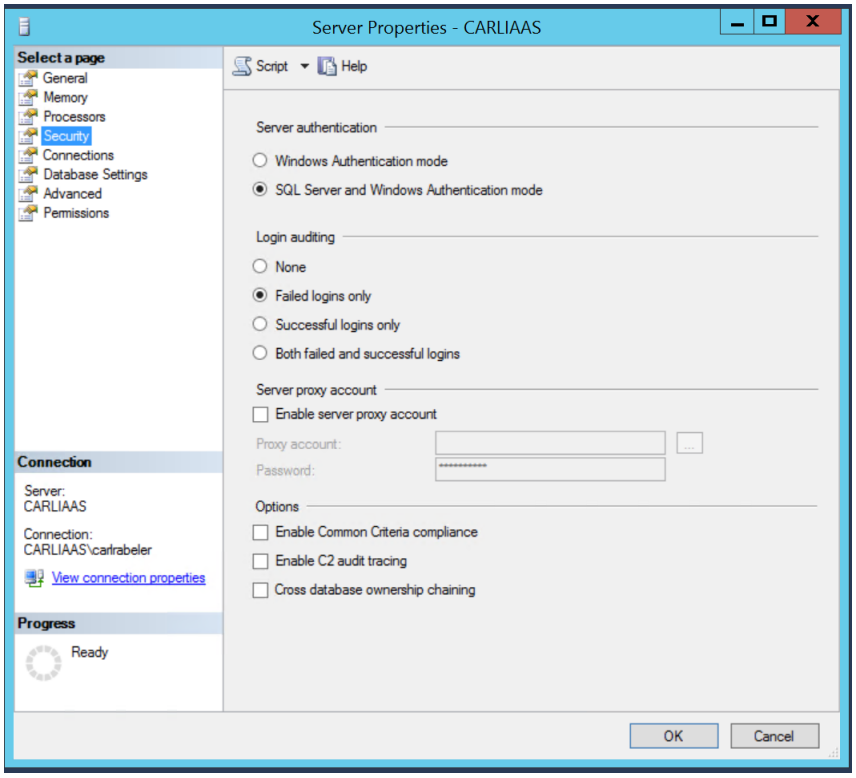


Figure 2-41: Server Authentication

4. Click Database Settings to verify that the F drive is configured as the default database location for new user databases. See Figure 2-42.

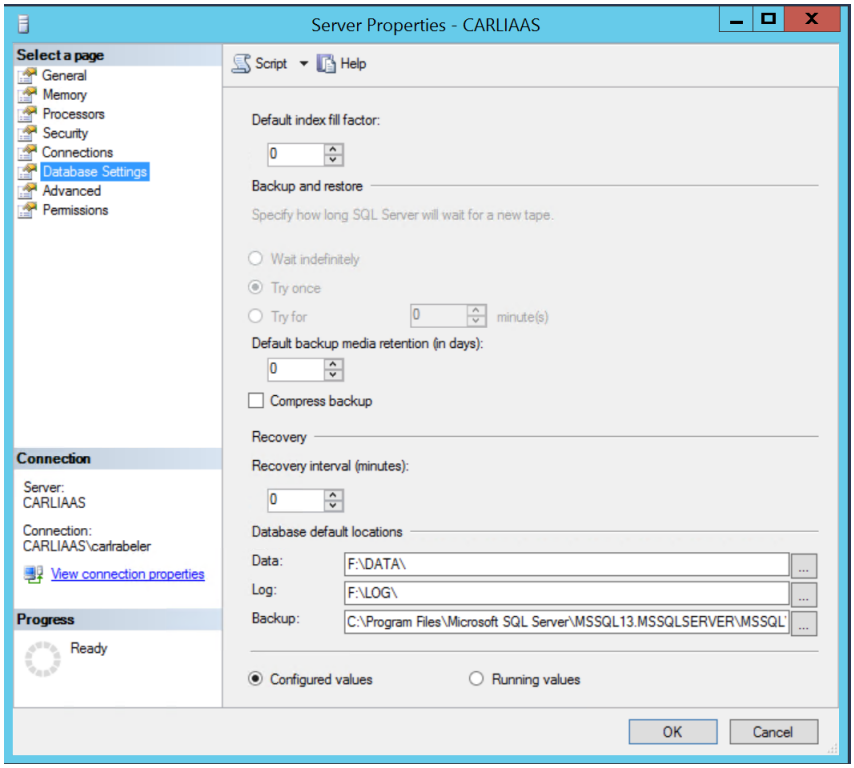


Figure 2-42: Database default locations

Connecting to SQL Server in your virtual machine from another computer

1. On your own computer, get the newest version of SQL Server Management Studio from https://msdn.microsoft.com/en-us/library/mt238290.aspx.

> **Important** The newest release of SQL Server Management Studio supports SQL Server 2005 through SQL Server 2016 versions of SQL Server, and this tool is updated on a monthly basis.

2. Open SQL Server Management Studio.
3. In the Connect to Server dialog box:

   a. Enter the public IP address for your virtual server in the Server Name text box.

> **Note** It will take a few minutes for the DNS name that you configure above to propagate and be available for use, so we first will connect using the IP address.

   b. In the Authentication drop-down menu, select SQL Server Authentication.

   c. Enter your credentials.

   See Figure 2-43.

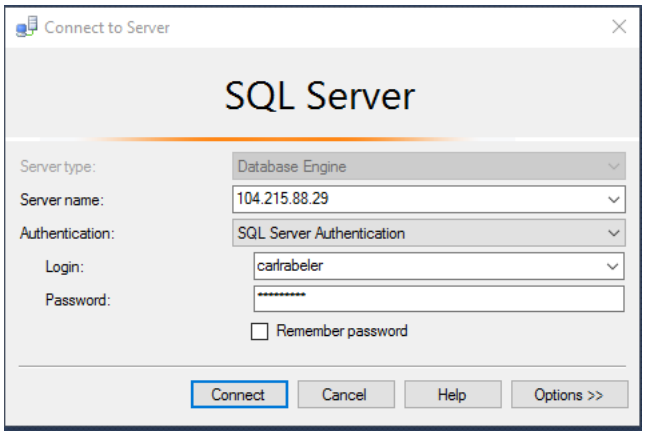


Figure 2-43: The SQL Server Connect to Server dialog box in your virtual machine using SQL Server Authentication

4. Click Connect to connect to your virtual machine's SQL Server instance in Object Explorer. See Figure 2-44.

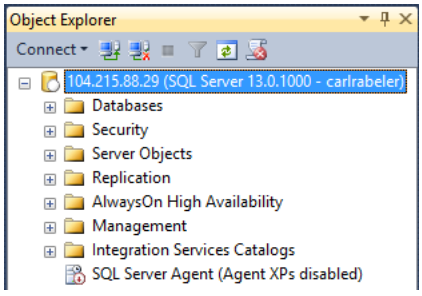


Figure 2-44: Connection in Object Explorer to SQL Server in your virtual machine using IP address

5. In Object Explorer, click Connect and then click Database Engine.
6. In the Connect to Server dialog box:
   a. Change the Server Name to the fully qualified domain name (FQDN) for your virtual machine.
   b. Type your password.

   See Figure 2-45.

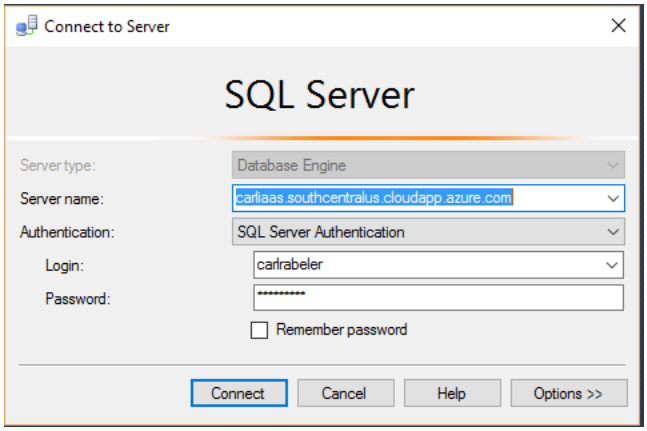


Figure 2-45: Connect using FQDN

7. Click Connect to connect to your virtual machine's SQL Server instance in Object Explorer. See Figure 2-46.

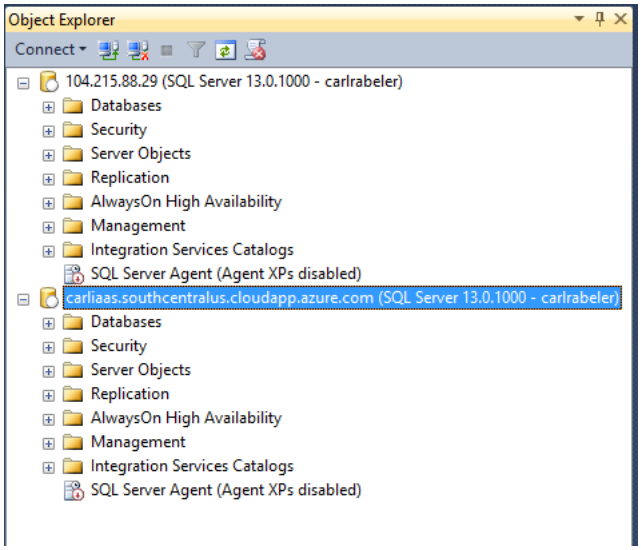


Figure 2-46: Connection in Object Explorer to SQL Server in your virtual machine using IP address and FQDN

Conclusion

In this chapter, we discussed the components of a virtual machine, how to attach a virtual machine image, and how to configure connectivity. We then walked through the entire process of creating a SQL Server virtual machine in Azure and successfully connected to the virtual machine using Windows Remote Desktop and using SQL Server Management Studio from another computer.

In the next chapter, we will contrast the requirements for and the process of creating an Azure SQL Database.

> **Tip** If you are taking a break at this point, shut down your virtual machine to save money.

Get the latest news from Microsoft Press sent to your inbox.

- New and upcoming books
- Special offers
- Free eBooks
- How-to articles

Sign up today at MicrosoftPressStore.com/Newsletters

Microsoft

CHAPTER 3

Getting started with an Azure SQL Database

In this chapter, I will discuss the Azure resources you will configure for your SQL Database logical server and databases. I also will discuss the Azure SQL Database connectivity requirements. I will finish the chapter with a guided walk-through of the creation of a SQL Database logical server and database that will conclude by connecting to the logical server and database using SQL Server Management Studio.

Overview of SQL Database

SQL Database is an Azure V12 SQL database running within the SQL Server platform service and associated with a logical server. An Azure V12 SQL database is similar in most respects to a SQL Server 2016 database, but there are a few differences. The SQL Server platform service provides the functionality to create and manage your database solution, freeing you up to focus on the solution rather than the underlying network, storage, and compute resources in addition to the SQL Server instance.

When creating your SQL database, you specify the amount of resources available to your new SQL database and associate it with either a new or an existing SQL Database logical server. The features and resource limits available to your new SQL database are determined by the service tier and performance level you select for the SQL database. The SQL Database service manages the resources available to each SQL database using a resource governance mechanism and hard resource limits (both of which are discussed below). A SQL database is portable and can be moved between logical servers.

To secure your SQL Database logical server and database instances, a SQL Database firewall limits connectivity to either Azure services or specific IP addresses. The SQL Database firewall operates at both the logical server and the database level.

> **Note** Integration of SQL Database with Azure Active Directory is out of scope of this chapter. It will be discussed in conjunction with security in Chapter 5, "Authentication, authorization, and data resiliency."

Service tiers

There are three service tiers: Basic, Standard, and Premium. The capabilities and maximum performance levels of a database within a tier vary depending upon the tier selected.

Service tiers and performance levels

SQL Database service tiers differ based on performance levels. Within each service tier, performance levels are designed to have a database operate as if it is running in its own machine, isolated from other databases and services. SQL Database accomplishes this by using a resource governor for certain resources and hard resource limits for other resources.

Each database can have its own service tier with its own limits, independent of any other database. This option is best for databases with a consistent load. You also can place multiple databases in an elastic pool where resources can be shared among databases and minimum and maximum resource thresholds can be set for each database in the pool.

The resource governor enforces a maximum amount of resources for either a single database or an elastic pool of databases. If the aggregated resource utilization reaches the maximum available CPU, memory, log I/O, and data I/O resources assigned to the database, the resource governor will queue queries in execution and assign resources to the queued queries as resources become available.

To assist you in choosing the amount of resources you need for your database, Microsoft developed a unit of measure called a Database Transaction Unit (DTU). To develop this benchmark, Microsoft took a set of operations that are typical for an online transaction processing (OLTP) request and then measured how many transactions could be completed per second under fully loaded conditions within each performance level.

If you are migrating an existing SQL Server database, you can use a third-party tool, the Azure SQL Database DTU Calculator, to get an estimate of the performance level and service tier your database might require in SQL Database.

Figure 3-1 shows the service tiers and their capabilities and performance limits for a single database.

	Basic	Standard				Premium				
		S0	S1	S2	S3	P1	P2	P4	P6/P3	P11
DTUs	5	10	20	50	100	125	250	500	1,000	1,750
Max storage (GB)	2	250				500				1,000
Max In-memory OLTP storage (GB)	N/A	N/A	N/A	N/A	N/A	1	2	4	8	14
Max concurrent workers	30	60	90	120	200	200	400	800	1,600	2,400
Max concurrent logins	30	60	90	120	200	200	400	800	1,600	2,400
Max concurrent sessions	300	600	900	1,200	2,400	2,400	4,800	9,600	19,200	32,000
Point-in-time restore	Any point last 7 days	Any point last 14 days				Any point last 35 days				
Disaster recovery	Active Geo-Replication, up to 4 offline or online (readable) secondary backups									

Figure 3-1: Single database limits

Figure 3-2 shows the service tiers and their capabilities and performance limits for an elastic pool of databases.

Elastic Pool	Basic					Standard					Premium				
eDTUs per pool	100	200	400	800	1,200	100	200	400	800	1200	125	250	500	1,000	1,500
Max storage per pool (GB)*	10	20	39	73	117	100	200	400	800	1,200	250	500	750	750	750
Max number of databases per pool	200	400				200	400				50				
Max concurrent workers per pool	200	400	800	1,600	2,400	200	750	1,300	1,850	2,400	200	750	1,300	1,850	2,400
Max concurrent logins per pool	200	400	800	1,600	2,400	200	750	1,300	1,850	2,400	200	750	1,300	1,850	2,400
Max concurrent sessions per pool	4,800	9,600	19,200	28,800	28,800	4,800	9,600	19,200	28,800	28,800	2,400	4,800	9,600	19,200	28,800

Elastic Database			
Max eDTUs per database	5	10, 20, 50, 100	125, 250, 500, 1000
Min eDTUs per database	0, 5	0, 10, 20, 50, 100	0, 125, 250, 500, 1000
Max storage per database (GB)*	2	250	500
Point in-time restore	Any point last 7 days	Any point last 14 days	Any point last 35 days
Disaster recovery	Active Geo-Replication, up to 4 readable secondary backups		

** Elastic databases share pool storage, so database storage is limited to the smaller of remaining pool storage and max storage per database*

Figure 3-2: Elastic pool limits

DTU resource limits

Single database: A single database can have between 5 and 1,750 DTUs. A Basic database has 5 DTUs, which means it can complete 5 "typical" transactions per second, while a Premium P11 database supports 1,750 DTUs. See Figure 3-3.

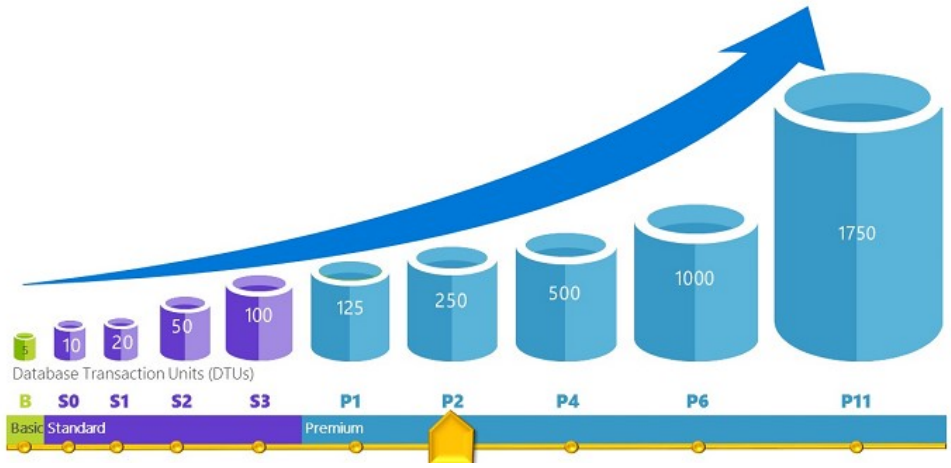


Figure 3-3: DTUs for a single database from the Basic tier through the Premium tier

Elastic pool database: A database can be a member of an elastic pool with between 100 and 1,500 DTUs. All databases within a pool share a common set of resources. See Figure 3-4.

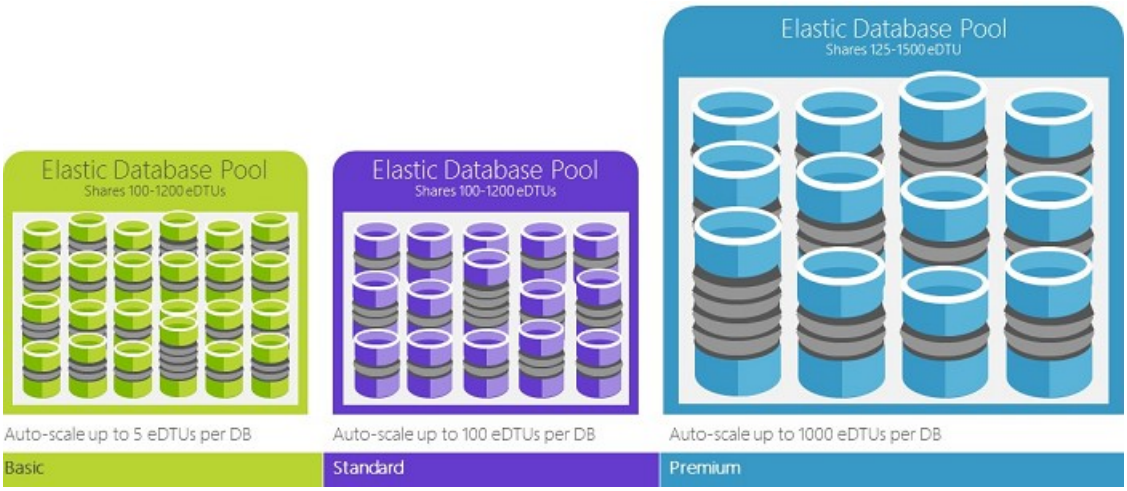


Figure 3-4: Elastic database pool

Other resource limits

SQL Database enforces limits on other resources by denying all new requests when limits are reached. These limits include the following:

- Maximum database size: Between 5 GB and 1 terabyte (TB)
- Max in-memory OLTP storage: Between not available and 10 GB
- Max concurrent requests: Between 30 and 2,400
- Max concurrent logins: Between 30 and 2,400
- Max sessions: Between 300 and 32,000
- Max databases using automated export: 10

Service tiers and capabilities

SQL Database service tiers also differ based on capabilities related to business continuity and in-memory features.

Business continuity services

All service tiers provide mechanisms to enable a business to continue operating in the face of disruption to its computing infrastructure. The types of potential disruption scenarios from which these mechanisms protect you are:

- Local hardware and software failures, such as disk failure
- Data corruption and deletion, such as application bugs and user errors
- Data center outage, such as a natural disaster
- Upgrade or maintenance errors, such as unanticipated issues during an application upgrade requiring a rollback to a previous state

The next two sections discuss these business continuity mechanisms and how they differ across service tiers.

Automatic database backup and self-service geo-restore services

To protect your data and enable point-in-time restore services, SQL Database automatically takes full database backups every week, multiple differential backups every day, and log backups every five minutes. Backup files are stored in a geo-redundant storage account with read access (RA-GRS) to ensure backups' availability for disaster recovery purposes in case of catastrophic disaster at any single data center. The first full backup is scheduled immediately after a database is created. After the first full backup, all further backups are scheduled automatically and managed silently in the background. The exact timing of full and differential backups is determined by the system to balance overall load. Figure 3-5 illustrates both local redundancy within a data center and storage geo-replication of these backup files to protect you from both user error and catastrophic failures.

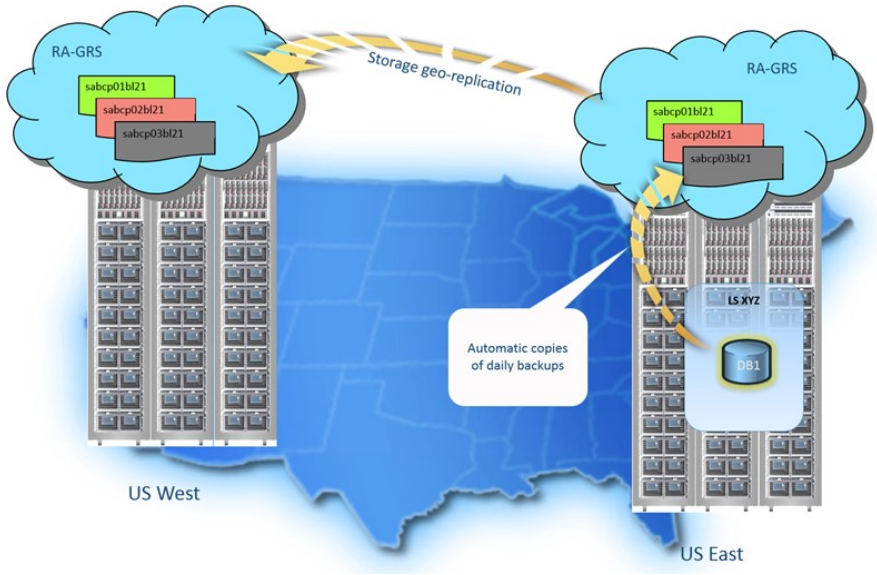


Figure 3-5: Storage geo-replication

SQL Database provides this service for all databases regardless of server tier, but they have different retention periods for the database backups as follows:

- Basic: Any restore point within the past 7 days
- Standard: Any restore point within the past 14 days
- Premium: Any restore point within the past 35 days

SQL Database provides both database copy and point-in-time restore services on top of this automated backup system, which enable you to restore an existing or deleted database to a new database as of a specified point in time, including the current point in time, within the retention period supported by the service tier. This capability is offered within the current data center.

> **Note** You cannot choose to overwrite an existing database, only to create a new database from the automated backups.

Storage geo-replication provides the ability to restore a database to another data center from the last geo-redundant backup to create a new database in any geographic region. This capability is called geo-restore. The geo-replication of the backup blobs from the automated backups guarantees that daily backups are available even after an outage or massive failure in the primary region.

Active geo-replication

In addition to automated backups and storage geo-replication, SQL Database offers active geo-replication for all databases regardless of service tier. Active geo-replication provides a mechanism to create and maintain asynchronous secondary replicas in up to four regions. These secondary replicas can be either offline or online and readable, and they can be in the same or different data centers.

A readable secondary is priced at 1X price of the selected performance level. A non-readable secondary is charged at a discounted rate of 3/4X price of the selected performance level. To ensure that transactions being applied to the secondary do not bottleneck the primary, the secondary must be at the same or higher performance level than the primary.

Figure 3-6 illustrates a single non-readable offline secondary, and Figure 3-7 illustrates multiple readable secondaries.

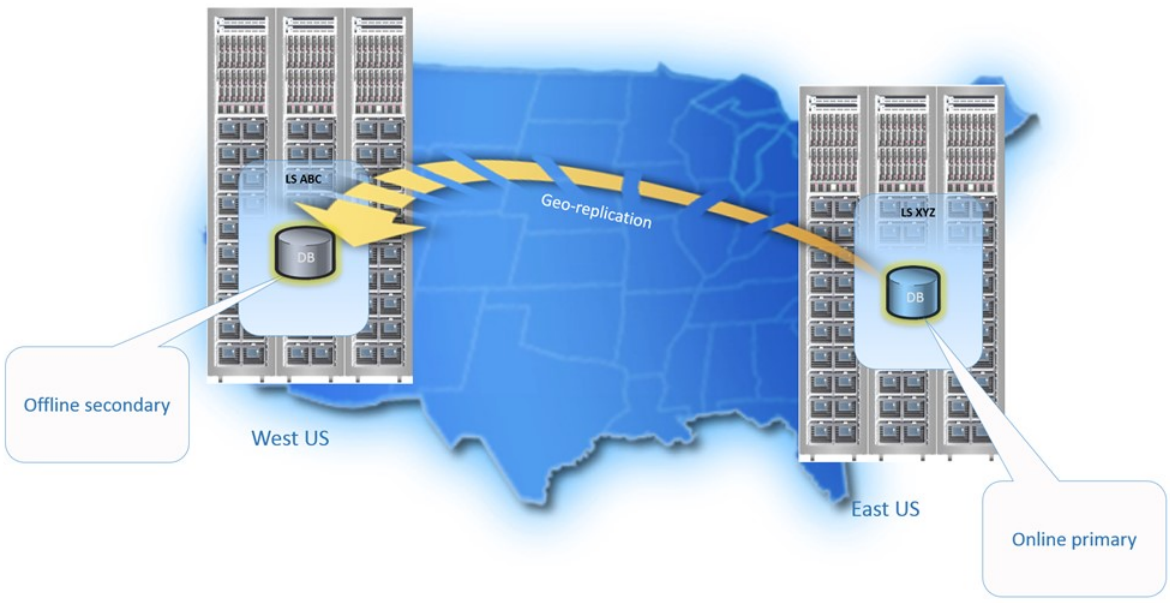


Figure 3-6: Geo-replication with single non-readable offline secondary

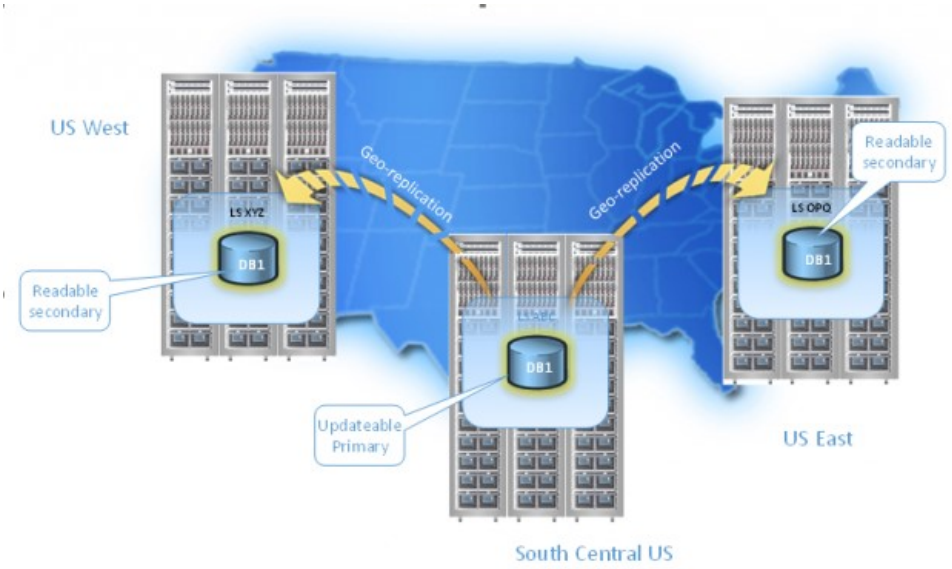


Figure 3-7: Geo-replication with multiple readable secondaries

Non-readable secondaries are available for failover if needed due to any of the disaster scenarios discussed above. Readable online secondary replicas can be used for load balancing and/or low-latency read access to different geographic regions. When performing an application upgrade or maintenance operation, the continuous replication can be frozen immediately prior to the upgrade or maintenance operation so that you can easily fall back. In case of a data center outage, you can manually fail over to one of the secondary replicas. You also can design and deploy a small worker role application that monitors your primary database and triggers a failover if necessary. For a discussion of active replication topologies using a combination of readable and non-readable secondaries for both disaster recovery scenarios and application load-balancing scenarios, see Design an application for cloud disaster recovery using geo-replication in SQL Database.

In-memory services

In-memory services are available only in the Premium service tier, with different size limits for memory storage based on the performance level within the Premium tier. In-memory services encompass two features: in-memory OLTP and Operational Analytics.

- In-memory OLTP is a feature that can greatly improve OLTP performance by storing selected OLTP tables in memory (5X–20X improvement) and by using compiled stored procedures (100X improvement). In-memory OLTP has been available in SQL Server beginning with SQL Server 2014 and has been enhanced significantly in SQL Server 2016. These enhancements also are available in SQL Database.
- Operational Analytics provides the ability to run both analytics and OLTP workloads in the same database at the same time without a significant degradation in performance. Besides running analytics in real time, you can eliminate the need for loading data into a data warehouse. Operational Analytics is a feature introduced in SQL Server 2016, with the same feature available in SQL Database.

A two-minute video about the benefits of using the in-memory OLTP feature is available at Azure SQL Database - In-Memory Technologies.

SQL Database logical server

A SQL Database server is a logical construct into which you can organize one or more SQL databases. Although much of the focus when working with SQL Database is at the database level, you do create some objects at the server level (for example, see the SQL Database firewall discussion below and see the discussion of logins in Chapter 5).

In addition, each server has a DTU quota per logical server of 45,000 DTUs and 5,000 databases per logical server. This quota represents the DTUs a logical server can host, based on the sum of the DTUs of the performance level of each database on the server. For example, a logical server with five Basic databases (5 X 5 DTUs maximum), two Standard S1 databases (2 X 20 DTUs maximum), and three Premium P1 databases (3 X 100 DTUs maximum) has consumed 365 DTUs of its 45,000 DTU quota. When you exceed the DTUs for a server, you will need to move a SQL database to another server.

Firewall rules

To protect your data in a SQL database hosted by the SQL Database service, the SQL Database firewall prevents all access to your SQL Database logical server until you open the firewall. You can create a firewall rule that opens the firewall to specific IP addresses and/or opens the firewall to all Azure services. SQL Database firewall rules can be established at the server level and at the database level.

- A firewall rule grants access based on the originating IP address of each request. You can permit individual addresses or a range of IP addresses. Use this option for connections from the Internet and for Azure connections using static IP addresses.
- Allowing access to Azure services opens the firewall to all Azure connections, which still must authenticate. Use this option for Azure connections using dynamic IP addresses.

Connection attempts from the Internet and Azure must pass through the firewall before they can reach your logical server or database. See Figure 3-8.

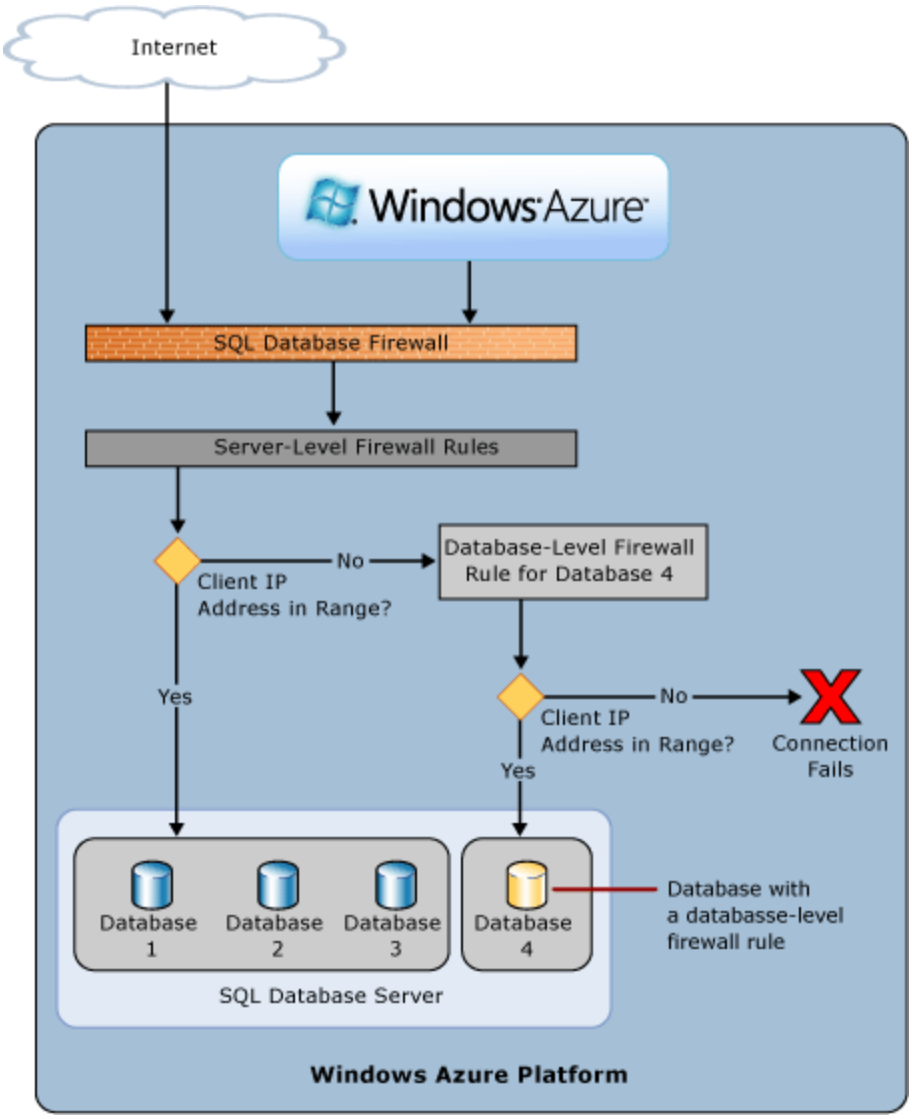


Figure 3-8: SQL Database server and database-level firewall rules workflow

Server-level firewall rules

Server-level firewall rules enable users to access the entire SQL Database logical server through the SQL Database firewall, including all databases on the logical server based either on permitted IP address ranges or for Azure services. Actually, access is based on permissions in the individual databases. Server-level firewall rules are stored in the master database. Use server-level firewall rules for administrative access to enable management of all databases on a server by server-level principals.

Note As of this writing, only server-level firewall rules can be created in the portal. Server-level firewall rules also can be created programmatically using PowerShell and the REST API. After the first server-level firewall rule has been created in the Azure portal, additional server-level firewall rules can be created using Transact-SQL.

Database-level firewall rules

Database-level firewall rules enable permitted IP address ranges or Azure services to access a specific database. Database-level firewall rules are stored in the individual databases. Use database-level firewall rules for database portability and to restrict database access within a logical server. After you have configured the first server-level firewall rule, you may want to restrict access to certain databases. If you specify an IP address range in the database-level firewall rule that is outside the range specified in the server-level firewall rule, only clients that have IP addresses in the database-level range can access the database. You can have a maximum of 128 database-level firewall rules for a database.

> **Note** As of this writing, database-level firewall rules for master and user databases can be created and managed using Transact-SQL, the REST API, PowerShell, or through the classic portal.

Walk-through: Getting started with an Azure SQL Database

In this walk-through, we will provision an Azure SQL Database logical server and a sample database using the Azure portal and then connect to this logical server and database using SQL Server Management Studio.

Provisioning a SQL Database logical server and database using the Azure portal

Let's start by provisioning a SQL Database containing a sample database. For simplicity, we will create a logical server at the same time that we create a SQL Database. If you have an existing logical server, feel free to use that logical server instead.

1. If you are not still connected to the Azure portal, connect to the Azure portal now.
2. To create a SQL Database in the Azure portal, click SQL Databases on the default blade to open the SQL Databases blade.

   Initially, there are no databases displayed. As you create SQL databases, you will be able to see and manage your SQL databases on this blade. To manage your logical SQL Database servers, click SQL Servers on the default blade (which will appear only after you have at least one SQL Database logical server defined). See Figure 3-9.

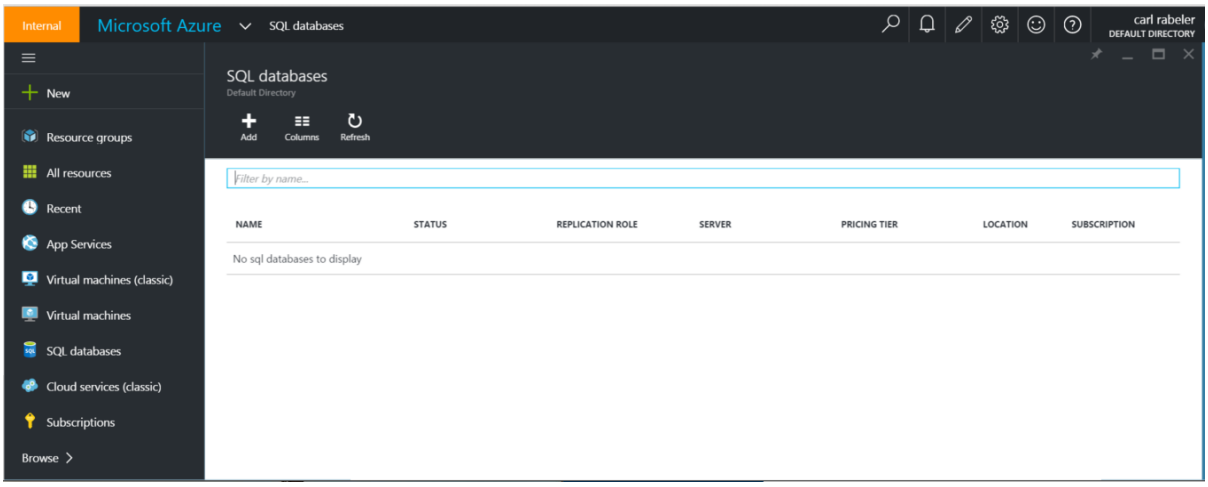


Figure 3-9: SQL Databases blade with no SQL databases

3. Click Add. See Figure 3-10.

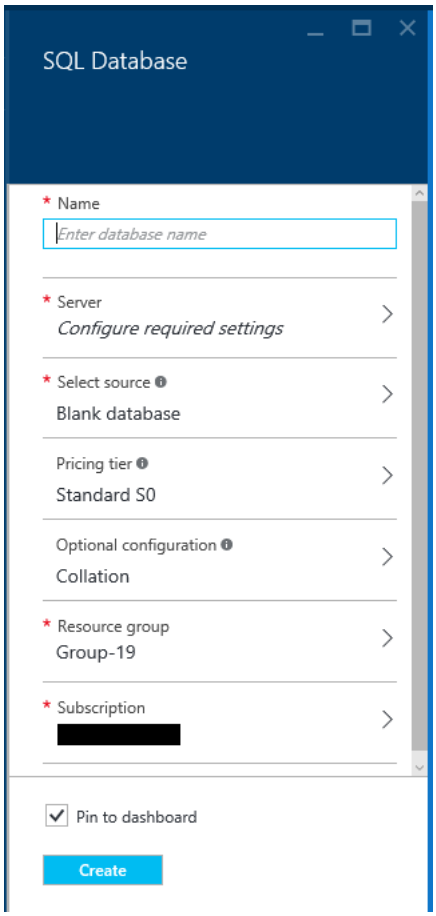


Figure 3-10: Creating a new database

4. Provide a database name of your choice. I will call mine carlpaasdb.

5. To configure your logical server, click the Server node to open the Server blade. See Figure 3-11.

Notice that you are prompted to Configure Required Settings. This value will change to display your selected server name and location after configuration. This is a required value.

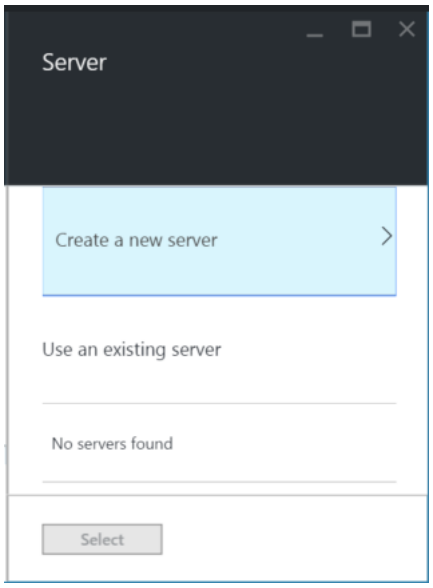


Figure 3-11: Selecting a new logical server

6. Because we have no existing servers from which to choose, click Create a New Server to open the New Server blade. See Figure 3-12.

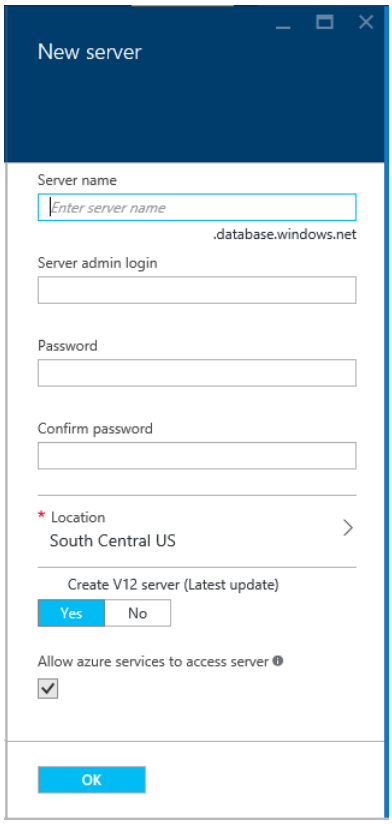


Figure 3-12: Configure a new server

7. Provide a name for your logical server. I will call mine carlpaassrv. See Figure 3-13.

   Notice that the logical server has a fully qualified server name. In my case, it will be carlpaassrv.database.windows.net.

Figure 3-13: New logical server name

8. Provide a server admin login and password.
9. Choose an Azure region as the location. I will choose South Central US.
10. Verify that Yes is the default for Create a V12 Server (Latest Update).
11. Verify that the check box to Allow Azure Services to Access Server is selected. See Figure 3-14.

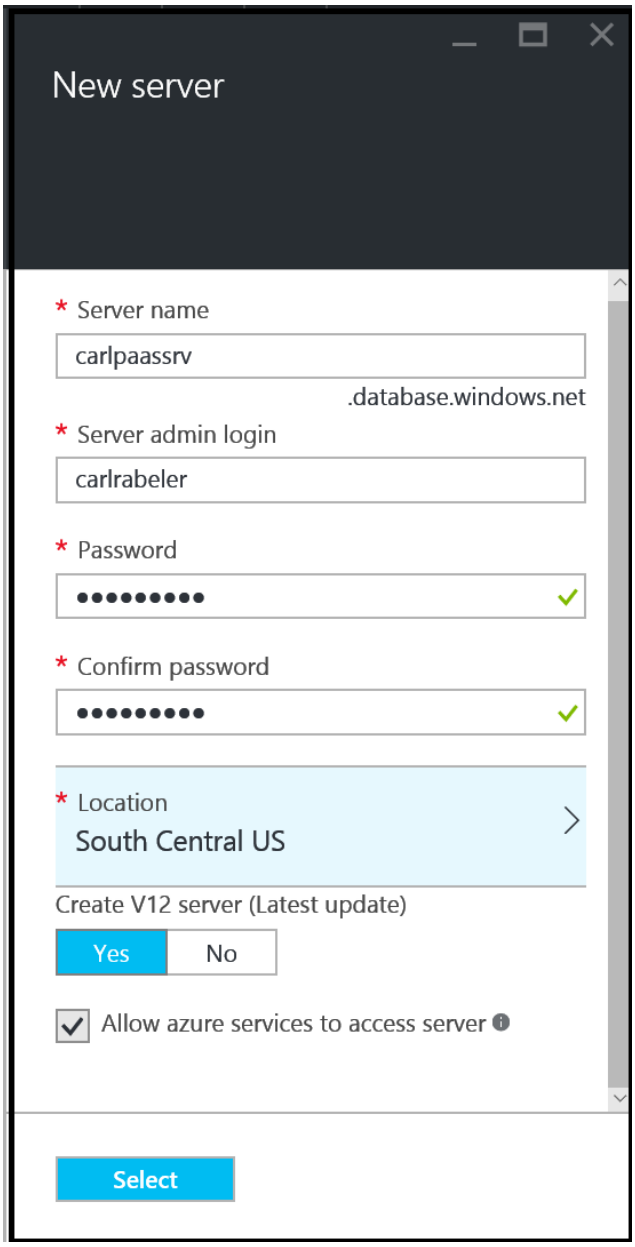


Figure 3-14: Completed New Server blade

12. Click Select to return to the SQL Database blade to continue configuring the SQL Database.
13. To select a database for your new SQL Database, click the Select Source node to open the Select Source blade.

    Notice that Blank Database is the default, which you are not required to change.
14. On the Select Source blade, review your options for a database source. See Figure 3-15.

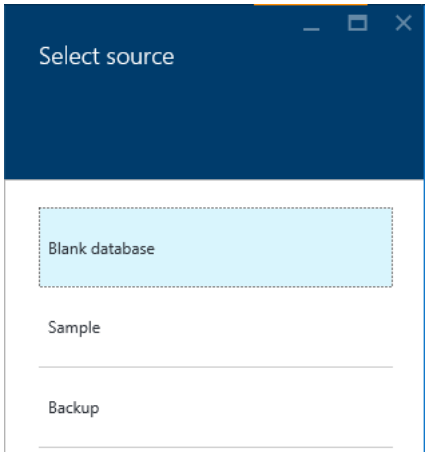


Figure 3-15: Select database source

15. If you select Sample, your database source will be the AdventureWorksLT (V12 database). See Figure 3-16.

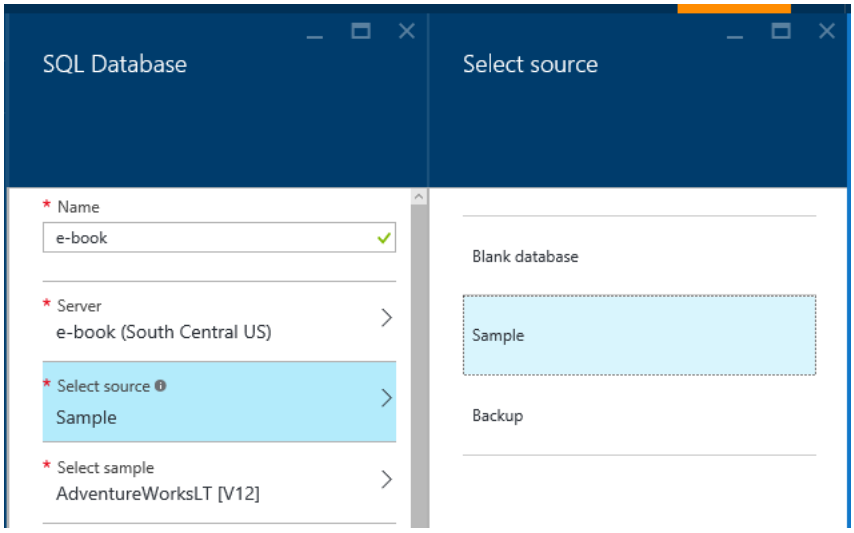


Figure 3-16: AdventureWorksLT (V12) sample

16. If you select Backup, you will have the option to restore an existing SQL database or a deleted SQL database to a point in time from its server blade. See Figure 3-17.

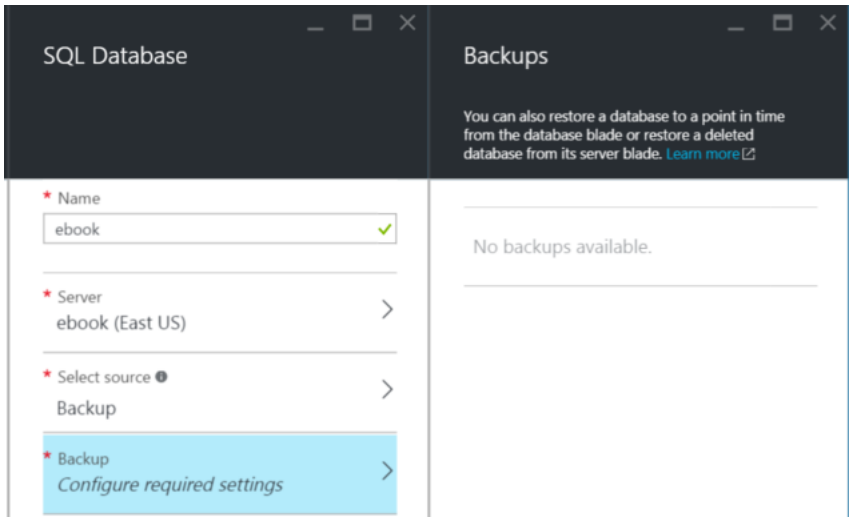


Figure 3-17: Backup as source

17. Because I have no backups and I want to have a sample database to start with, I will select Sample.

18. On the Pricing Tier node, notice that the default service tier and performance level is Standard S0. Changing this is optional.

19. Click the Pricing Tier node to open the Choose Your Pricing Tier blade.

    This blade displays all of the pricing tiers that are available to you, along with their DTU ratings, service features, and estimated cost per month. See Figure 3-18.

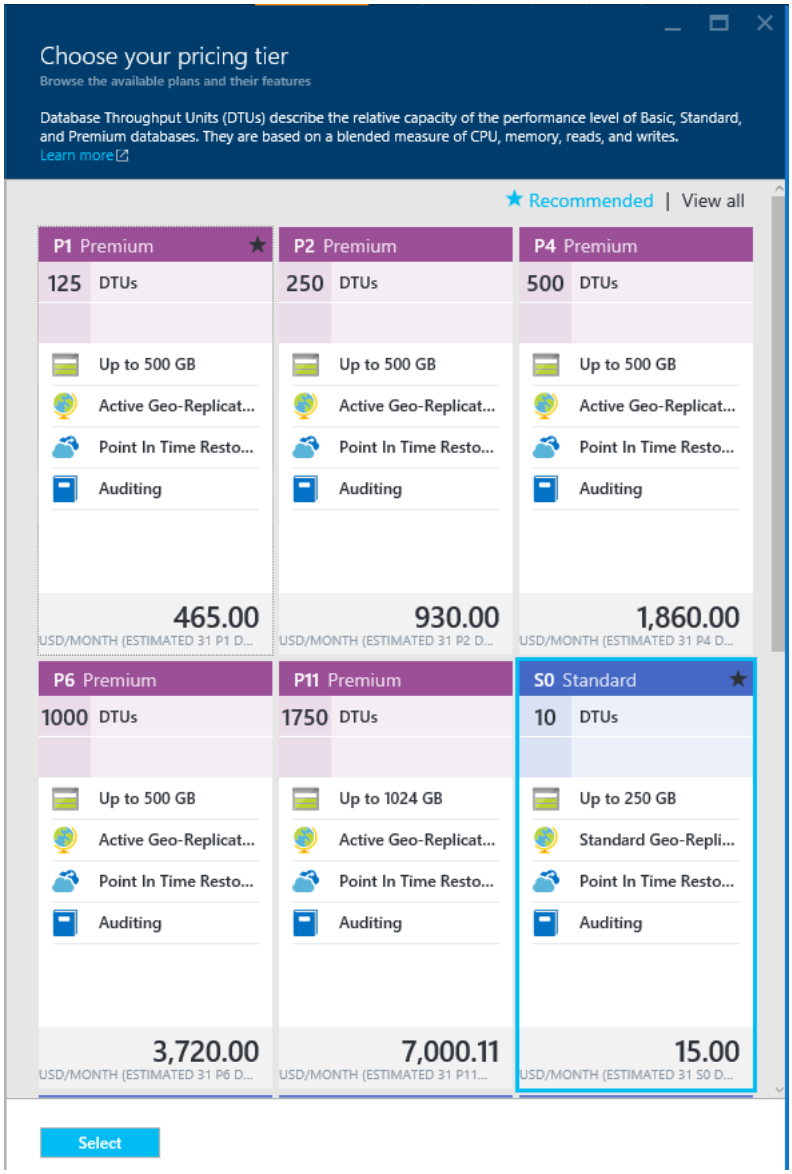


Figure 3-18: Pricing tier, all options

20. Click Recommended and review the recommended pricing tiers that are available to you to get started. See Figure 3-19.

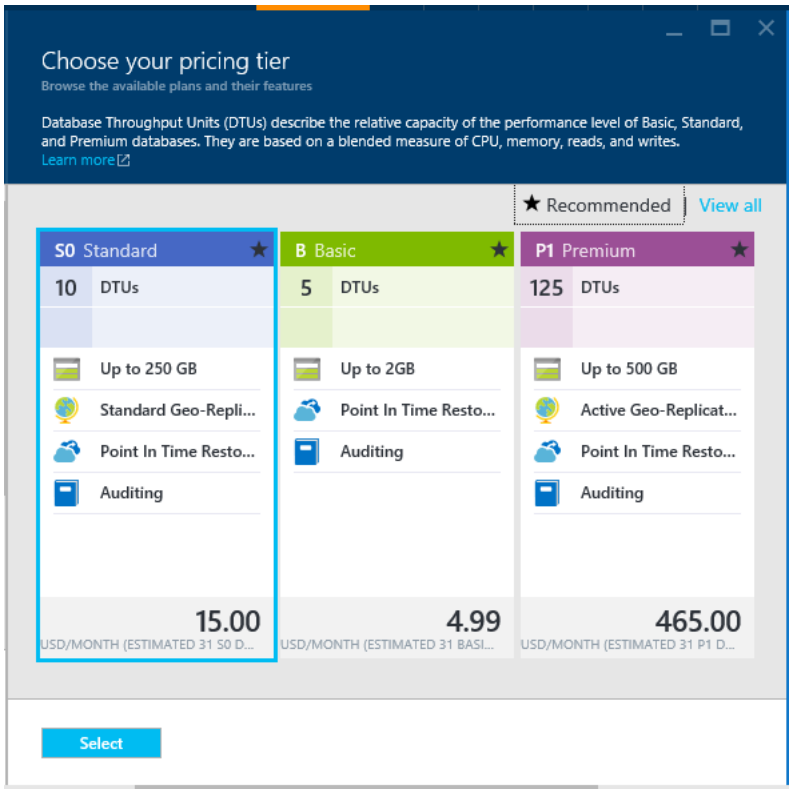


Figure 3-19: Pricing tier, recommended options

21. Because we can change service tier and performance level dynamically, I will start with S0.

22. Although we can change the collation settings, a discussion of collations is well beyond the scope of this walk-through.

23. Next, we need to define a resource group into which to place this SQL Database logical server and database. See Figure 3-20.

    Notice the default resource group name. I am going to change this to carlrgpaas.

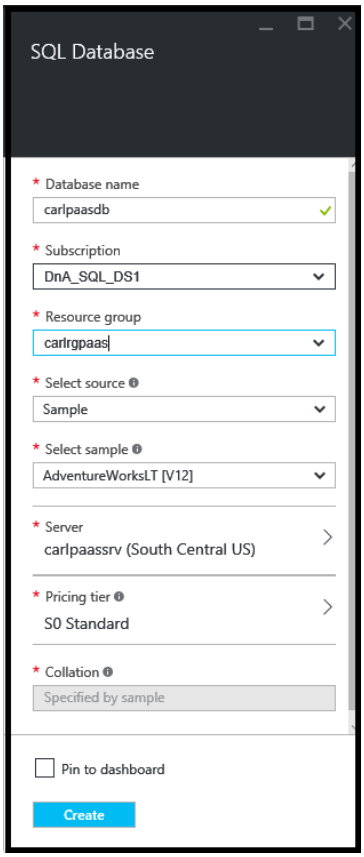


Figure 3-20: Completed SQL Database blade

24. We are almost ready to create a SQL Database. Notice that by default, the new SQL Database will be pinned to the dashboard. With lots of databases and other resources, you probably will not want all of them on the dashboard. But in our walk-through, let's pin it to the dashboard.

25. Click Create.

26. On the dashboard, notice that your SQL Database is being created. See Figure 3-21.

Figure 3-21: Creating a SQL Database on the dashboard

Viewing and configuring your SQL Database in the portal

1. After the provisioning of your SQL Database is complete, a number of panes appear on the carlpaas SQL Database pane. See Figure 3-22.

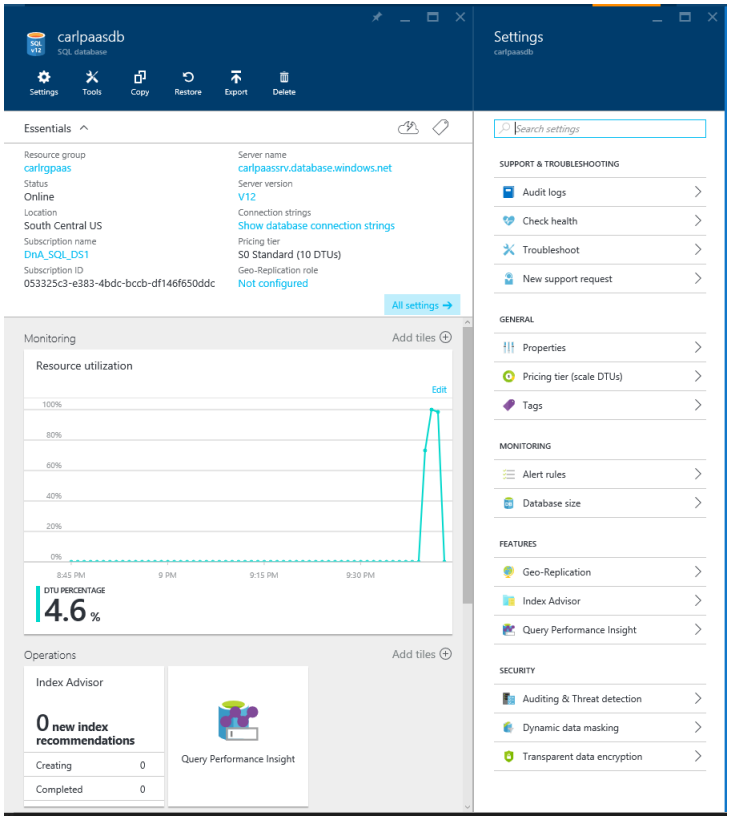


Figure 3-22: The newly created database in the Azure portal

2. Review the Essentials and Monitoring panes and the information on the Settings blade. Notice there are additional panes for Operations and Geo-Replication. Geo-Replication is out of scope for this ebook.

3. In the Essentials pane, click the link for your Server Name to open the carlpaassrv SQL Server blade. See Figure 3-23.

Figure 3-23: Server name link in the Essentials pane

4. On the carlpaassrv SQL Server blade, review the Essentials and Databases panes and the Settings blade. Notice also the Elastic Database Pools pane at the bottom of this pane. See Figure 3-24.

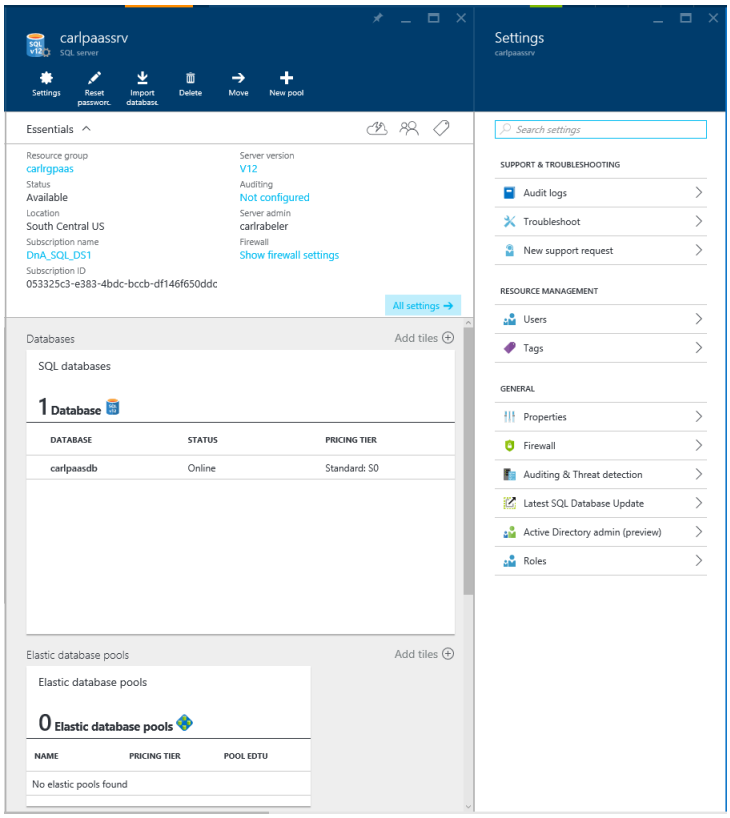


Figure 3-24: carlpaassrv SQL Server blade

5. In the Essentials pane, click Show Firewall Settings. See Figure 3-25.

Figure 3-25: Link to show firewall settings

6. On the Firewall Settings blade, notice that you have two firewall options: to allow access to Azure services and to allow access to specific IPs. Notice also that your client's IP address is listed. See Figure 3-26.

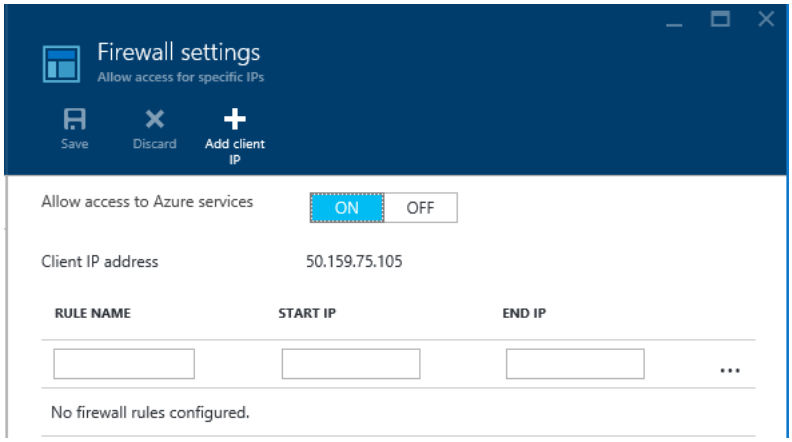


Figure 3-26: Firewall settings

7. Enter a rule name, a starting IP address, and an ending IP address. See Figure 3-27.

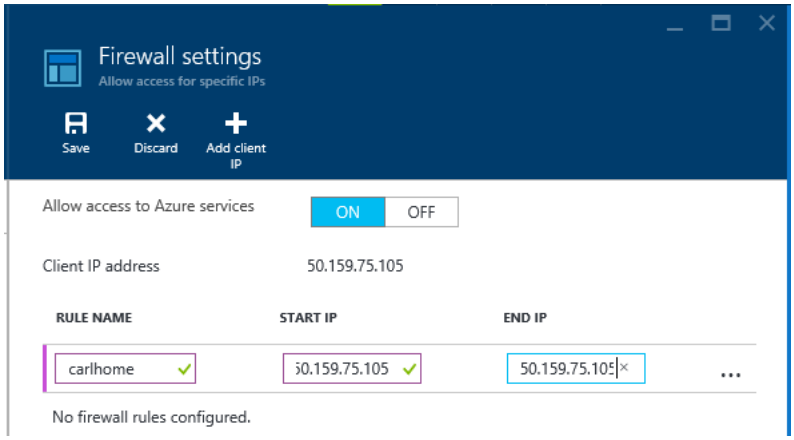


Figure 3-27: Completed firewall rule

8. Click Save to update the firewall rules and then click OK.

9. Close the Firewall Settings blade and the carlpaasserv SQL Server blade.

10. In the Essentials pane, click Show Database Connection Strings. See Figure 3-28.

Figure 3-28: Show database connection strings

11. Review these connection strings on the Database Connection Strings pane. Notice that complete connection strings are provided for ADO.NET, ODBC, PHP, and JDBC. See Figure 3-29.

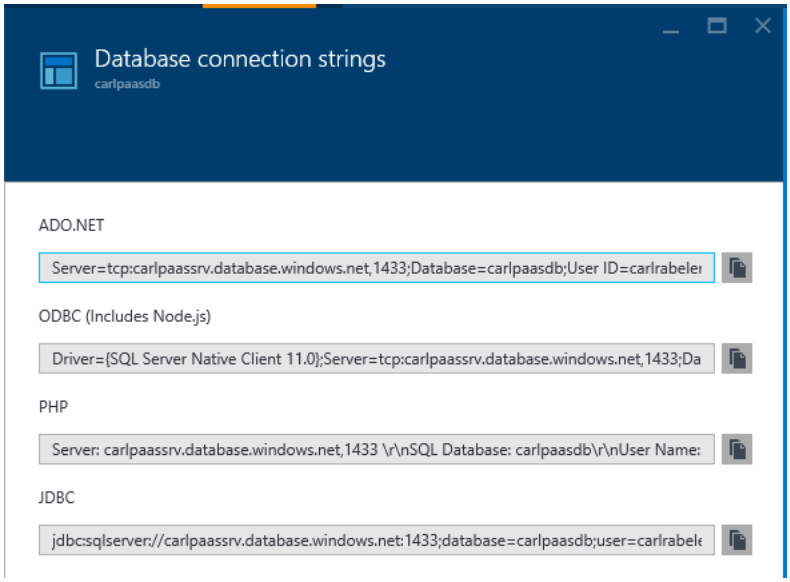


Figure 3-29: Database connection strings

Connecting to your SQL Database using SQL Server Management Studio

1. On your own computer, if you have not already done so, get the newest version of SQL Server Management Studio (SSMS) from https://msdn.microsoft.com/en-us/library/mt238290.aspx.

Important The newest release of SQL Server Management Studio supports SQL Server 2005 through SQL Server 2016 versions of SQL Server, and this tool now is updated on a monthly basis.

2. Open SQL Server Management Studio.
3. In the Connect to Server dialog box:
   a. Enter the fully qualified server name for your SQL Database logical server in the Server Name text box.
   b. In the Authentication drop-down box, click SQL Server Authentication.
   c. Enter your credentials.

   See Figure 3-30.

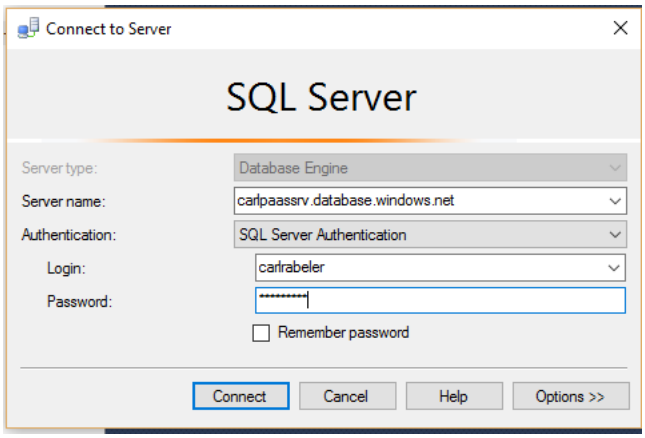


Figure 3-30: Connect to Azure SQL Database using SQL Server Authentication

4. Click Connect to connect to your Azure SQL Database in Object Explorer. See Figure 3-31.

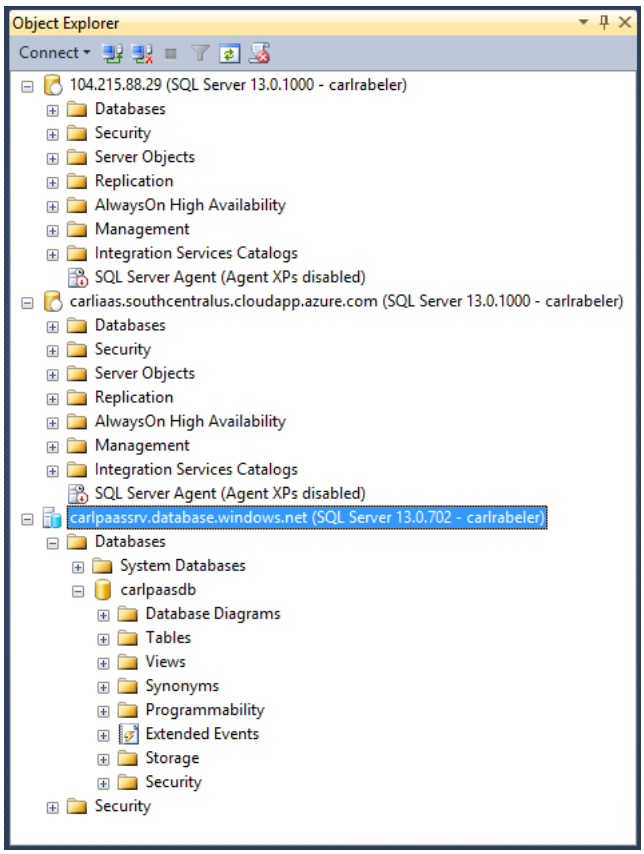


Figure 3-31: Connection in Object Explorer to Azure SQL Database

Conclusion

In this chapter, we discussed the components of an Azure SQL Database logical server and database and how to configure connectivity. We then walked through the relatively simple process of creating an Azure SQL database, creating a firewall rule, and connecting to the Azure SQL Database logical server and database using SQL Server Management Studio from another computer.

In the next chapters, we will discuss importing data into an Azure virtual machine running SQL Server and into an Azure SQL Database.

CHAPTER 4

Migrating a database to Azure

In this chapter, I will discuss migrating a user database to Azure. Your source database can be a Microsoft SQL Server database, or it can be any number of non-Microsoft SQL Server databases. Your destination database can be either SQL Server in an Azure virtual machine or Azure SQL Database. The migration method and complexity depend on your source database, your destination database, and your tolerance for downtime. I will discuss the migration options for each type of source database into both destination options. I will also briefly discuss managing metadata that is stored outside the user databases. Finally, I will briefly discuss setting the appropriate database compatibility level after the migration is complete. We will finish the chapter with a guided walk-through of migrating a SQL Server 2008 R2 database into SQL Server in an Azure virtual machine and into Azure SQL Database.

Overview of migrating a user database to Azure

SQL Server provides a variety of mechanisms to help you migrate some or all of the user objects and data from both Microsoft SQL Server and non-Microsoft SQL Server databases to an Azure environment:

- **Migrating from a SQL Server source database**: You can migrate from a SQL Server 2005–2016 database to an Azure environment, but you may have compatibility issues to resolve either before the migration can be completed or after the migration but before the database is ready to use.

    - Migrating from a recent version of SQL Server to a supported version of SQL Server in an Azure virtual machine will require few if any changes at the user database level.
    - Migrating from an older version of SQL Server database to a newer version of SQL Server may have breaking changes that you will have to resolve before, during, or after the migration.
    - When you are migrating from a SQL Server database to Azure SQL Database, you may have objects that are not compatible that you will need to fix prior to the migration.
    - Microsoft provides a number of mechanisms to minimize downtime during the migration, depending upon both your source environment and your destination.
- **Migrating from a non–SQL Server source database**: You can migrate from the following non–SQL Server source databases into an Azure environment:
    - Access
    - DB2
    - MySQL
    - Oracle
    - Sybase

When you are migrating from a non–SQL Server database, SQL Server provides mechanisms to convert a substantial percentage of the database objects automatically and will assist you in identifying the objects that cannot be converted, providing you guidance in how to convert those remaining objects. These changes will have to be made either before or during the migration. In most cases, you cannot complete the migration until you have established compatibility with the destination environment.

Regardless of the migration method, you will have to manage metadata that is stored outside the user databases, plan to minimize user downtime, and determine how to redirect users from the old database to the new database.

After the migration is complete, you will want to set the appropriate database compatibility level for the best performance and to take advantage of the newest database capabilities and provide partial backward compatibility with earlier versions of SQL Server until you have converted applications that use deprecated and discontinued functionality. By default, a migrated database retains its existing compatibility if it is at least equal to the minimum supported by the instance to which the database is being migrated.

Finally, how you perform the migration will be affected by your tolerance for downtime in your existing production environment. You will need to plan your migration to handle the amount of downtime required. For example, you may perform an initial migration of a copy of your database, perform the required testing and development work on the copy, and later migrate the production system by using one of the mechanisms discussed later in this chapter to minimize downtime. Or, you may discover that your database requires minimal or no changes and can be migrated during a scheduled maintenance window by using one of the simple migration methods discussed in this chapter.

> **Note** The focus in this chapter is on migrating a user database rather than migrating an entire instance or migrating system databases because the typical migration does not include system databases. I briefly will discuss the migration of system database objects later in this chapter.

Managing breaking changes between SQL Server versions

When you migrate a database to a newer version of SQL Server or to SQL Database, you may encounter breaking changes from the version of SQL Server on which the database was running. These changes might break applications, scripts, or functionalities that are based on earlier versions of SQL Server. There are two types of breaking changes: changes that will be detected by the migration tool and prevent the migration and changes that are not detected but affect functionality. With respect to changes that affect functionality, running a migrated database at an older compatibility level is a short-term fix, but you will need to run your database at the highest compatibility level possible to take advantage of the newest capabilities of SQL Server or SQL Database.

Before you begin a migration, you should review the Microsoft documentation with respect to any breaking changes if you are migrating from an older version of SQL Server to a newer version of SQL Server. See the following links for details regarding breaking changes:

- SQL Server 2016 Breaking Changes
- SQL Server 2014 Breaking Changes
- SQL Server 2012 Breaking Changes
- SQL Server 2008 R2 Breaking Changes
- SQL Server 2008 Breaking Changes
- SQL Server 2005 Breaking Changes

Microsoft provides an upgrade advisor to help you detect and prepare for a version upgrade. These advisors will examine scripts, stored procedures, triggers, and trace files, but they will not analyze desktop applications or encrypted stored procedures. See the following links for information about how to install and use the upgrade advisor for the version of SQL Server to which you are upgrading and for information about where to find or install the upgrade advisor.

- SQL Server 2016
- SQL Server 2014
- SQL Server 2012
- SQL Server 2008 R2
- SQL Server 2008
- SQL Server 2005

Important The SQL Server 2016 Upgrade Advisor is currently in preview. It is a new and substantially enhanced version, with more features to come soon. In late April or early May 2016, a new version of the SQL Server 2016 Upgrade Advisor will be released and will be named the Data Migration Assistant (DMA), also as a preview. Among the enhancements is the ability not only to detect upgrade issues, but also to fix some issues and to perform the upgrade and migration for you using database backup functionality.

With respect to issues you need to fix due to breaking changes, when you fix them depends on the source and the destination.

- If you are migrating from SQL Server on-premises to SQL Server in a virtual machine, you can fix them in the source before the migration—in the source database (perhaps affecting the production system), in an on-premises copy as part of the migration, or after the database is restored. A SQL Server database backup can be restored to a newer version of SQL Server, even with issues that must be changed before the database will function properly (discontinued syntax, for example).
- If you are migrating from a non–SQL Server source database or are migrating to SQL Database, you must fix the issues detected before the migration because the migration mechanisms do not permit you to fix them after the migration.

To fix issues detected, you can use either SQL Server Management Studio or SQL Server Data Tools for Visual Studio (SSDT). With SSDT, you create a database project from the database schema and then modify the schema within the project without affecting the source database. You then can merge the changes back into the source database or to a database copy and then synchronize the data.

Migrating and managing metadata stored outside a user database

When you migrate a user database to an Azure environment, metadata that is stored outside the user database generally is not migrated as part of the database migration. In a typical SQL Server environment, an application has dependencies on the master and msdb databases (and sometimes multiple user databases) and on metadata in the user database itself. Anything stored outside a user database that is required for the correct functioning of that database must be made available on the destination server instance, or the application must be re-architected to find that metadata in the user database. For example, the logins for an application traditionally are stored as metadata in the master database, and they must be either re-created on the destination server or re-created as contained users (see Chapter 5, "Authentication, authorization, and data resiliency"). For issues with orphaned logins, see Troubleshoot Orphaned Users.

- **Migrating to SQL Server in an Azure VM**: If you are migrating a SQL Server database to SQL Server in an Azure virtual machine (Azure VM), you can either migrate all of the user databases using backup and restore or re-create the needed metadata. For more information, see Manage Metadata When Making a Database Available on Another Server Instance (SQL Server). Although not typically done, you could also migrate all of the system databases (although this is beyond the scope of this chapter).
- **Migrating to SQL Database**: If you are migrating a SQL Server database to SQL Database, you will need to re-architect how your database and application are designed to work within the constraints of SQL Database. For example, SQL Server Agent jobs are not supported in SQL Database. If an application or database maintenance plan depends on SQL Server Agent jobs, you must use other Azure functionality to achieve these tasks, such as elastic database jobs, Azure scheduler, or Azure automation. Although elastic database jobs are still in preview, they generally should be your first option to replace SQL Server Agent jobs, such as using contained users.

Migrating a SQL Server user database to a SQL Server instance in an Azure virtual machine

When migrating a SQL Server user database to a SQL Server instance in an Azure virtual machine, the underlying mechanism generally uses the SQL Server backup functionality (using transactional

replication also is an option, and this option is discussed in its own section later in this chapter). The options that use a variation of the SQL Server backup technology are discussed in this section.

Using database backups, you can:

- Use a full database backup. For simplicity, you can perform the user database migration using a SQL Server full database backup. This method is most appropriate for small databases where the downtime is short; for scenarios in which you can afford the amount of downtime required to back up, copy, and restore the database to the SQL Server instance in the Azure virtual machine; and for scenarios in which you just want to test SQL Server in an Azure virtual machine prior to an actual migration.
- Use a combination of full database and transaction log backups to minimize downtime on your production system.
- Use AlwaysOn Availability Groups and create a replica of your existing on-premises user database on your Azure virtual machine (or replicas of a group of user databases that must be migrated together).

Note To migrate a SQL Server 2000 database to a supported configuration in an Azure virtual machine, you must first upgrade the database to SQL Server 2005 SP4 or greater.

Caution When migrating using these methods, encrypt your backups and use encrypted connections to ensure your data is secure during the migration process.

Migrating using full database backup for simplicity

To use the full database backup method with SQL Server, select one of the following mechanisms and do the following:

- Manually back up your source database and then restore it to your SQL Server instance in your virtual machine using SQL Server Management Studio. You can back it up and restore it using one of these methods:
  - Back up to a local file, copy backup to your virtual machine (or to blob storage), and then restore your backup file into your SQL Server instance in your virtual machine. We will walk through this method at the end of this chapter.
  - Backup to URL and then restore from URL. This method is supported only for SQL Server 2012 SP1 CU2 or newer.
- Use the Deploy Database to a Microsoft Azure VM Wizard in SQL Server 2016 SQL Server Management Studio to back up your source database and then restore it to your SQL Server instance in your virtual machine. This method currently is not recommended because this wizard has not yet been updated to support Azure Resource Manager (ARM)–deployed virtual machines.
- Use the Microsoft SQL Server 2016 Upgrade Advisor to back up your source database and then restore it to your SQL Server instance in your virtual machine. As mentioned previously, the SQL Server 2016 Upgrade Advisor currently is in preview, and a new version called Data Migration Assistant (DMA) will be released in late April or early May 2016. As this tool matures, it will become the preferred method. Until Data Migration Assistant is available, this method is not recommended.

Note You also could detach your data and log files, copy those files into your virtual machine, and then attach those files. However, this method has no advantage over the database backup and restore method.

Migrating using full database and transaction log backups to minimize downtime

To minimize downtime involved in a migration using a full database backup, you can use a combination of a full database backup, a differential backup (manual method only), and multiple transaction log backups. By using this method, you will keep downtime to a minimum. This method commonly is used with medium to large databases where minimizing downtime is of paramount importance. To test your database in Azure before migrating a production system, either use the full database backup method or use this method to test and verify the migration steps before the actual production migration.

- **Manual method**: Back up, copy, and then restore (without recovery) a full database backup, a differential backup (optional), and multiple transaction logs while keeping your production system operational. Continue taking and applying transaction log backups without recovery until you are ready to switch over. Apply the final transaction log backup with recovery when you are ready to switch your clients/applications over to connect to the SQL Server instance in the Azure virtual machine rather than the on-premises database.
- **AlwaysOn Availability Group method**: If you have an on-premises AlwaysOn Availability Group, you can extend the availability group to SQL Server in your virtual machine. Configure your SQL Server instance in the Azure virtual machine as an AlwaysOn replica, seed the replica with a full database backup, keep the replica current with transaction log backups that are applied automatically, and then fail over to the Azure replica when you are ready to switch over your clients to the SQL Server instance in the Azure virtual machine (and turn off AlwaysOn in the source database).

Migrating a SQL Server user database to an Azure SQL Database

When migrating a SQL Server user database to an Azure SQL Database, the underlying mechanism generally is a SQL Server BACPAC file, although you can use transactional replication (discussed later in this chapter). A BACPAC file is a Windows file with a .bacpac extension that encapsulates a database's schema and data within the file for transport between systems. A BACPAC file format essentially is a .zip file with a different extension. You can open and see the XML files inside by renaming it as a .zip file.

To migrate to SQL Database using a BACPAC file, you export a transactionally consistent copy of a database that is compatible with Azure SQL Database V12 using one of several tools and then import that BACPAC file into SQL Database. Unlike a SQL Server backup file, an export to a BACPAC file is not inherently transactionally consistent. To achieve a transactionally consistent export, you must either cease all write activity to the database or export from a transactionally consistent copy of the database (generally achieved by using backup and restore).

> **Note** To migrate a SQL Server 2000 database to an Azure SQL Database using a BACPAC file, you must first upgrade the database to SQL Server 2005 or greater.

Determining and resolving Azure SQL Database V12 compatibility issues

To export to a BACPAC file, the database must be compatible with Azure SQL Database V12. The export will fail if the database is not compatible. With SQL Database V12, there are very few remaining

compatibility issues other than server-level and cross-database operations. Databases and applications that rely on partially supported or unsupported functions will need some reengineering to fix these incompatibilities before the SQL Server database can be migrated. Review these partially supported or unsupported functions before migrating the SQL Database.

The tooling to detect and fix these incompatibilities is improving, but it still requires some manual intervention in many cases. The list below describes the tooling as it currently exists, including new tooling coming out in preview mode in May 2016.

- The Microsoft SQL Server 2016 Upgrade Advisor preview (UA): This stand-alone tool that currently is in preview will detect and generate a report of SQL Database V12 incompatibilities. This tool does not yet have all of the most recent compatibility rules (but will soon). If no errors are detected, you can continue and complete the migration to SQL Database. If errors are detected, you must use another tool to fix any detected compatibility issues. SSDT is the recommended tool.

  As mentioned previously, UA will be updated in late April or early May 2016 and become Data Migration Assistant (DMA). DMA will contain the most recent compatibility rules and will, over time, enable you to fix incompatibilities directly in the tool. This functionality is still coming online, so stay tuned.

- SQL Server Data Tools for Visual Studio (SSDT): For Azure SQL Database V12, the best tool today is SSDT because it uses the most recent compatibility rules to detect SQL Database V12 incompatibilities. To use this option, download the newest version of SSDT. If incompatibilities are detected, you can fix detected issues directly in SSDT.

- SqlPackage: SqlPackage is a command-prompt utility that tests for and, if found, generates a report containing detected compatibility issues. If you use this tool, make sure you use the most recent version (follow the link provided) to use the most recent compatibility rules. If errors are detected, you must use another tool to fix any detected compatibility issues. SSDT is the recommended tool.

- The Export Data-tier Application Wizard in SQL Server Management Studio: This wizard will detect and report errors to the screen. If no errors are detected, you can continue and complete the migration to SQL Database. If errors are detected, you must use another tool to fix any detected compatibility issues. SSDT is the recommended tool.

- SQL Azure Migration Wizard (SAMW): For Azure SQL Database V11, the best tool was SAMW. This is a community-supported CodePlex tool that has not been updated fully for Azure SQL Database V12 and generally is not the best method to use to migrate to Azure SQL Database V12.

Figure 4-1 shows the general workflow for determining and resolving compatibility issues and then migrating a compatible database to Azure SQL Database.

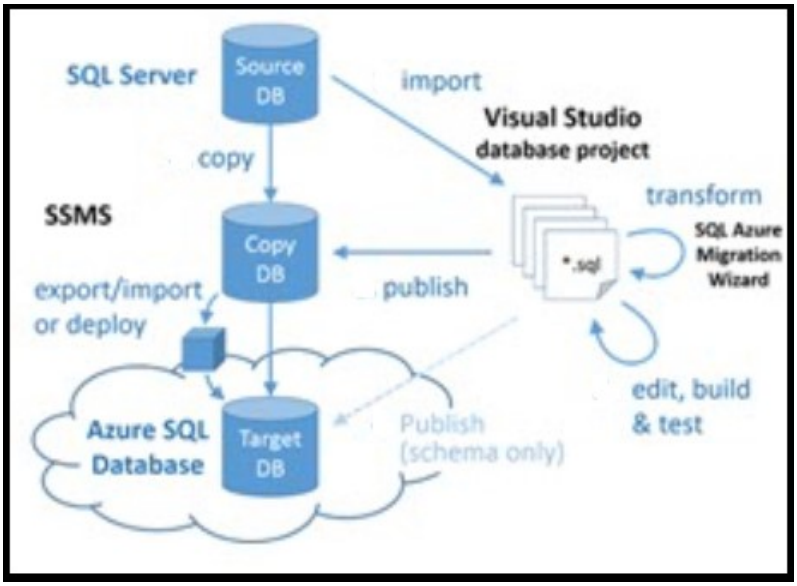


Figure 4-1: SQL Database migration workflow

The older the version of SQL Server for your source database and the more complex your database solution, the more potential incompatibilities you will encounter during the migration process. Use the following resources in addition to a targeted Internet search using your search engine of choice:

- SQL Server database features not supported in Azure SQL Database
- Discontinued Database Engine Functionality in SQL Server 2016
- Discontinued Database Engine Functionality in SQL Server 2014
- Discontinued Database Engine Functionality in SQL Server 2012
- Discontinued Database Engine Functionality in SQL Server 2008 R2
- Discontinued Database Engine Functionality in SQL Server 2005

In addition to searching the Internet and using these resources, another way to identify how best to fix an incompatibility issue is to use the MSDN SQL Server community forums or StackOverflow.

Migrating a compatible SQL Server database to SQL Database

To migrate a compatible SQL Server database to Azure SQL Database, Microsoft provides several migration methods for various scenarios. The method you choose depends upon your tolerance for downtime, the size of your SQL Server database, and the speed and quality of your connection to the Microsoft Azure cloud.

If you can afford some downtime or you are performing a test migration of a production database for later migration, consider one of the following three methods:

- SSMS Migration Wizard: For small to medium databases, migrating a compatible SQL Server 2005 or later database is as simple as running the Deploy Database to Microsoft Azure SQL Database Wizard in SQL Server Management Studio. We will walk through this method at the end of this chapter.
- Export to BACPAC file and then Import from BACPAC file: If you have connectivity challenges (no connectivity, low bandwidth, or timeout issues) and for medium to large databases, use this manual method. With this method, you export the SQL Server schema and data to a BACPAC file

and then import the BACPAC file into SQL Database using either the Deploy Database to Microsoft Azure SQL Database Wizard in SQL Server Management Studio or the SqlPackage command-prompt utility.

- Use BACPAC and BCP together: Use a BACPAC file and BCP for very large databases to achieve greater parallelization for increased performance, albeit with greater complexity. With this method, migrate the schema and the data separately.
  - Export the schema only to a BACPAC file.
  - Import the schema only from the BACPAC file into SQL Database.
  - Use BCP to extract the data into flat files and then parallel load these files into Azure SQL Database.

See Figure 4-2 for an illustration of these options.

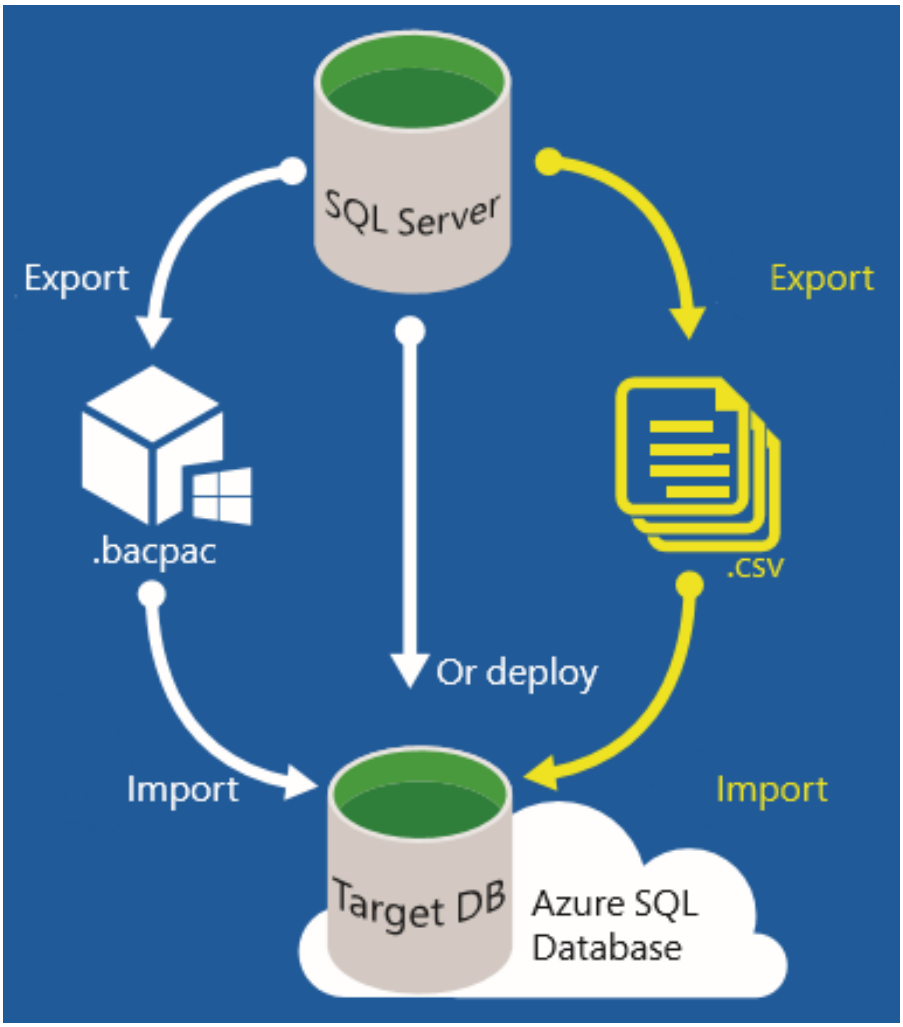


Figure 4-2: SQL Database import processes for compatible databases

Using transactional replication to migrate a SQL Server user database to SQL Server in an Azure virtual machine or to Azure SQL Database

To minimize downtime, you can use transactional replication to migrate an entire database or a subset of the source database to either SQL Server in a virtual machine or an Azure SQL Database. This method will keep downtime to a minimum and commonly is used in environments that have already implemented transactional replication. See Figure 4-3.

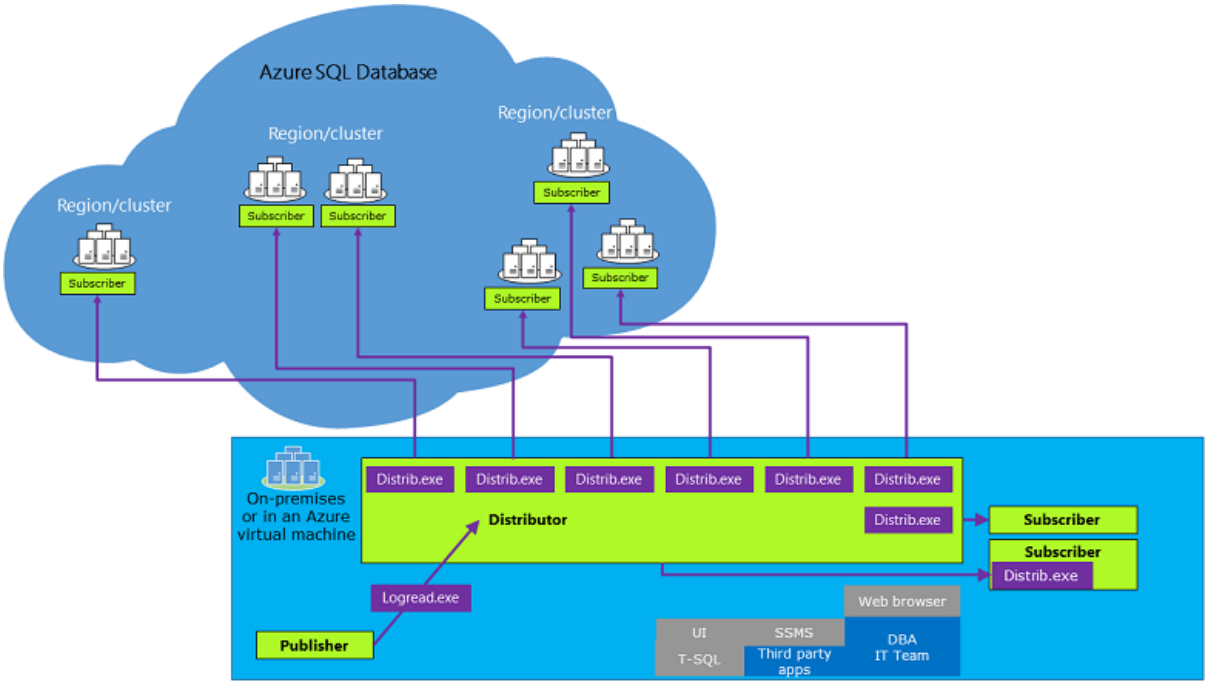


Figure 4-3: Transactional replication

To use transactional replication, do the following:

- Configure your SQL Server instance in the Azure virtual machine as a transactional replication subscriber.
- Allow it to catch up with the publisher.
- When you are ready, switch over your clients to the SQL Server instance in the Azure virtual machine (and turn off replication on the source database).

Unlike the methods discussed previously, there are two primary limitations to this approach.

- First, this scenario is supported only when the publisher and the distributor are one of the following SQL Server versions:
  - SQL Server 2016 CTP3 (preview) and above
  - SQL Server 2014 SP1 CU3 and above
  - SQL Server 2014 RTM CU10 and above
  - SQL Server 2012 SP2 CU8 and above

    - SQL Server 2012 SP3
- Second, each replicated table must have a primary key. For additional limitations, see Transactional Replication Limitations.

Migrating a non–SQL Server user database to SQL Server in an Azure virtual machine or an Azure SQL Database

SQL Server supports the migration of the following non–SQL Server user databases to either SQL Server in an Azure virtual machine or an Azure SQL Database using the SQL Server Migration Assistant (SSMA):

- SQL Server Migration Assistant for Access
- SQL Server Migration Assistant for DB2
- SQL Server Migration Assistant for MySQL
- SQL Server Migration Assistant for Oracle
- SQL Server Migration Assistant for Sybase

SSMA is a free web download. The links above introduce you to the tool itself, point you to where to download the tool, and provide detailed guidance in using SSMA with each type of source.

When migrating a non–SQL Server user database to either SQL Server in an Azure virtual machine or to Azure SQL Server, take the following general steps:

- Connect to source and destination.
- Map the source schemas to the SQL Server schemas.
- Assess the source schemas for conversion:
    - Some objects can be converted automatically.
    - Some objects must be converted manually.
- Convert the schemas.
- Load the converted database objects.
- Bulk-import the data.
- Test and validate the migrated objects and data.

The detailed steps will vary depending upon the source database technology, the source database complexity, and the Azure destination. For help and support when using SSMA, Microsoft provides both assistant support and premier support.

Walk-through: Migrating a SQL Server database to SQL Server in an Azure virtual machine using SQL Server backup and restore

In this walk-through, please follow along as we back up a SQL Server 2008 R2 user database, copy the database backup to our Azure virtual machine, and then restore the SQL Server 2008 R2 user database backup to SQL Server 2016 in an Azure virtual machine.

Backing up a SQL Server 2008 R2 user database to local storage

Let's start by connecting to a SQL Server 2008 R2 instance of SQL Server using SQL Server Management Studio and backing up the AdventureWorks2008R2 database.

1. Open SQL Server Management Studio and connect to your SQL Server 2008 R2 instance (or the instance of your choice) in Object Explorer.
2. Right-click AdventureWorks2008R2 (or the database of your choice), point to Tasks, and click Back Up. See Figure 4-4.

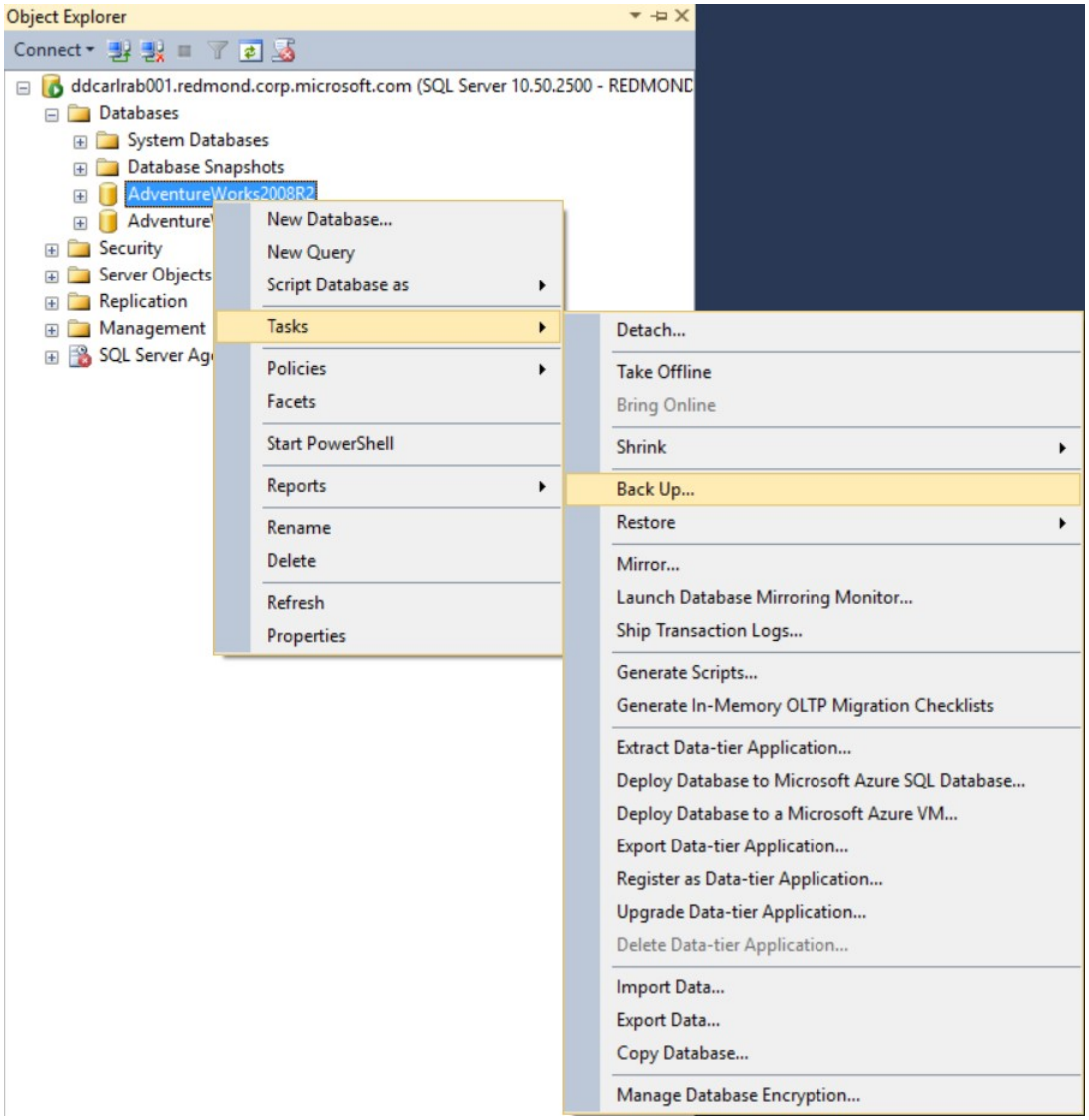


Figure 4-4: Initiating a database backup

3. In the Back Up Database window, verify that the backup type is Full and click the Copy-Only Backup check box. A copy-only backup is a SQL Server backup that is independent of the sequence of conventional SQL Server backups and preserves the existing log archive point. Therefore, it does not affect the sequencing of regular log backups. See Figure 4-5.

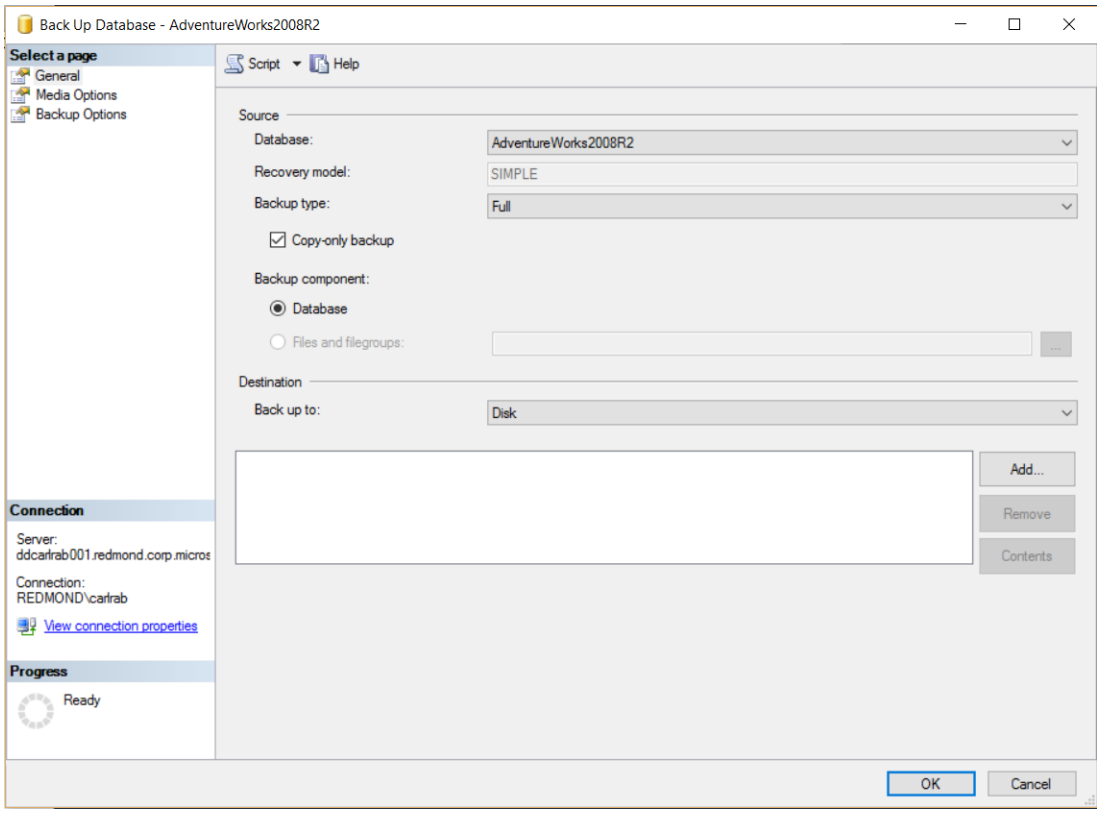


Figure 4-5: Specifying backup type

4. Click Add to define the file name and destination on disk for the backup file and then click OK. See Figure 4-6.

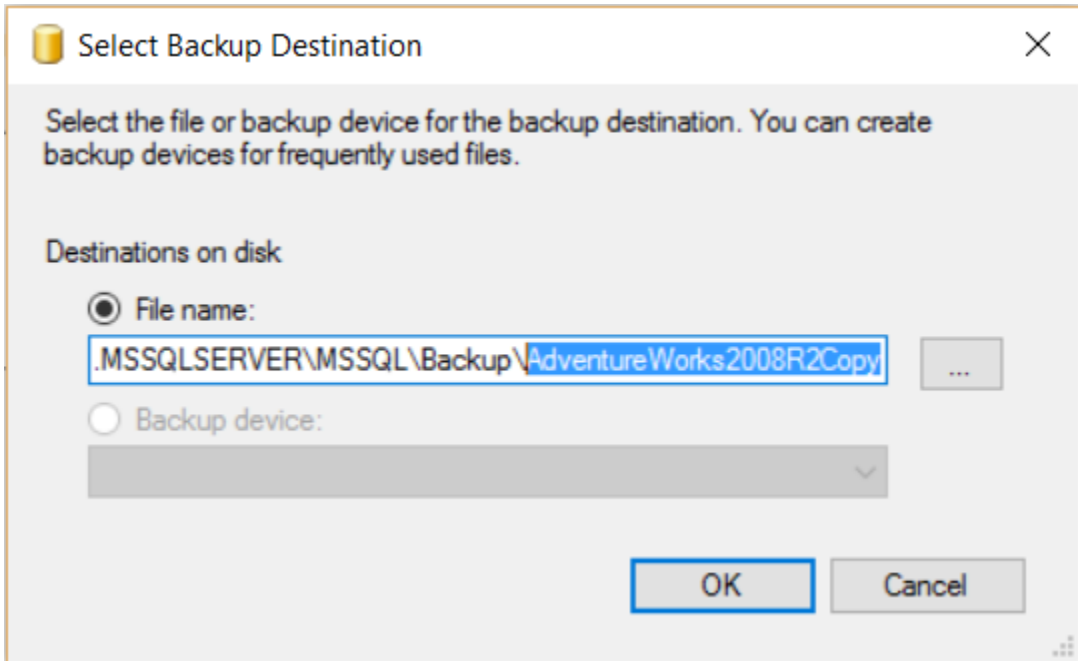


Figure 4-6: Specifying backup destination

5. Click OK to create the database backup.

6. Verify that the database was created successfully and then click OK to close the message box. See Figure 4-7.

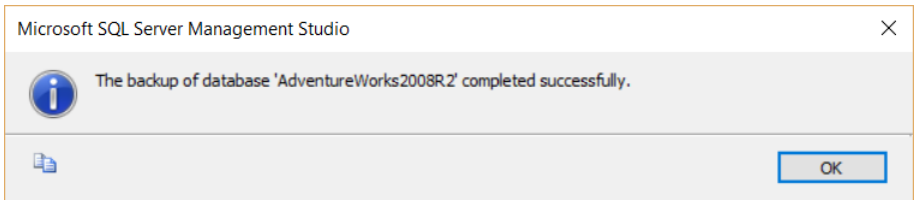


Figure 4-7: Successful backup message

Copying the backup file from local storage to Azure blob storage

Now that we have a full database backup, we will copy it to Azure blob storage (and ultimately to the Azure virtual machine). There are a number of methods we could choose to perform this task. I will use the AzCopy program. Although you could use the copy and paste method to copy it directly to the Azure virtual machine, this method is subject to transient network errors, does not have built-in retry mechanisms, and is much slower than AzCopy.

> **Note** Although SQL Server Management Studio has a wizard that will perform this task for you, it only works with virtual machines deployed using classic mode and then only for virtual machines created using the most recent images in the Azure portal. (Microsoft will be releasing a new wizard for SSMS later this year.)

1. Download the latest version of AzCopy either on the computer containing the database backup file or on a computer with connectivity to that storage location.
2. Install the Microsoft Azure Storage Tools following the prompts.
3. After the installation finishes, open a command prompt and navigate to the install location for AzCopy. In my case, it is C:\Program Files (x86)\Microsoft SDKs\Azure\AzCopy. See Figure 4-8.

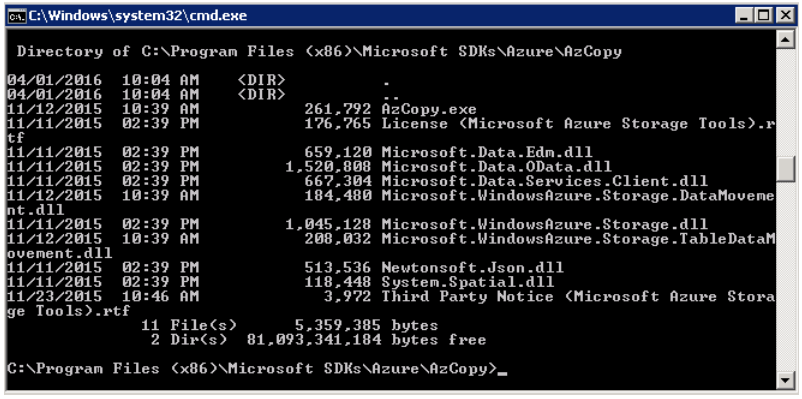


Figure 4-8: AzCopy file location

4. Switch to the Azure portal and open your standard storage account for your Azure virtual machine. We will create a container into which we will copy this backup file. See Figure 4-9.

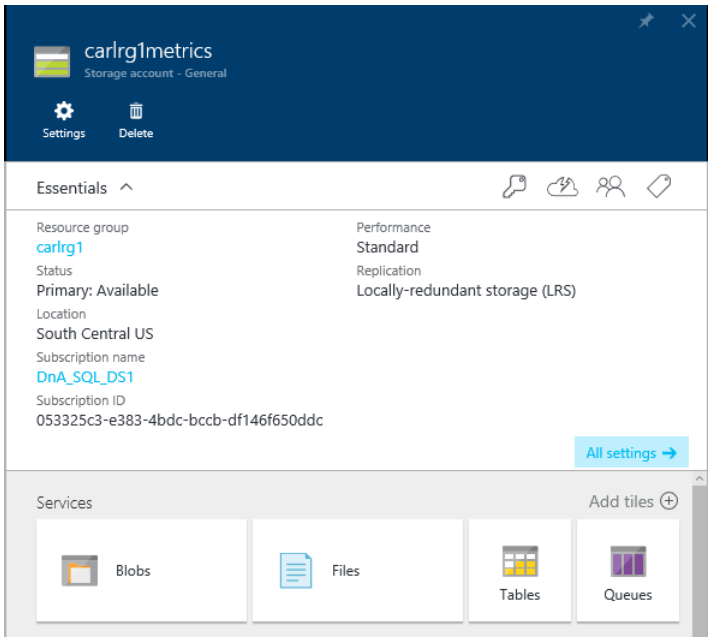


Figure 4-9: Azure standard storage account

5. Click Blobs to open the Blob Service blade. Notice that there already is a blob in this container that contains the metrics for your Azure virtual machine. See Figure 4-10.

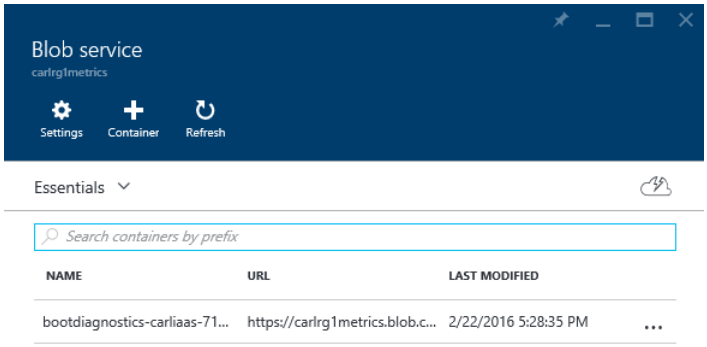


Figure 4-10: Blob Service blade

6. Click the plus sign to create a new container. See Figure 4-11.

Figure 4-11: Click + to configure a new container

7. In the New Container blade, specify a name for this container and leave the access type as Private. See Figure 4-12.

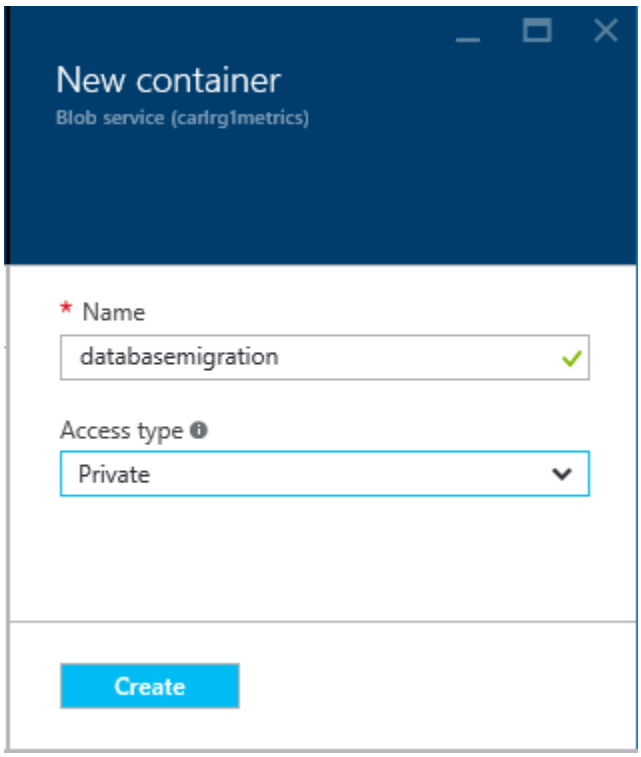


Figure 4-12: New container creation

8. Click Create.
9. After the container is created, click the new container to open the container. See Figure 4-13.

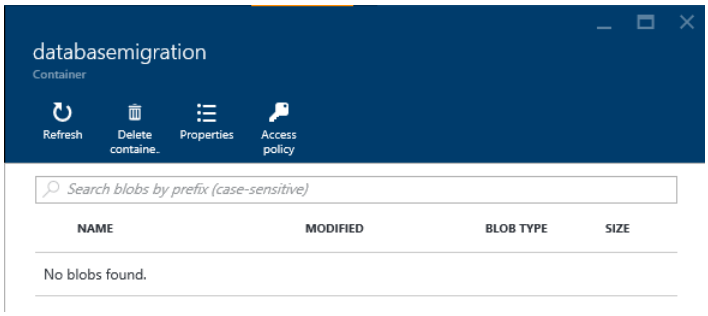


Figure 4-13: New container

10. Click Properties to open the Container Properties blade. See Figure 4-14.

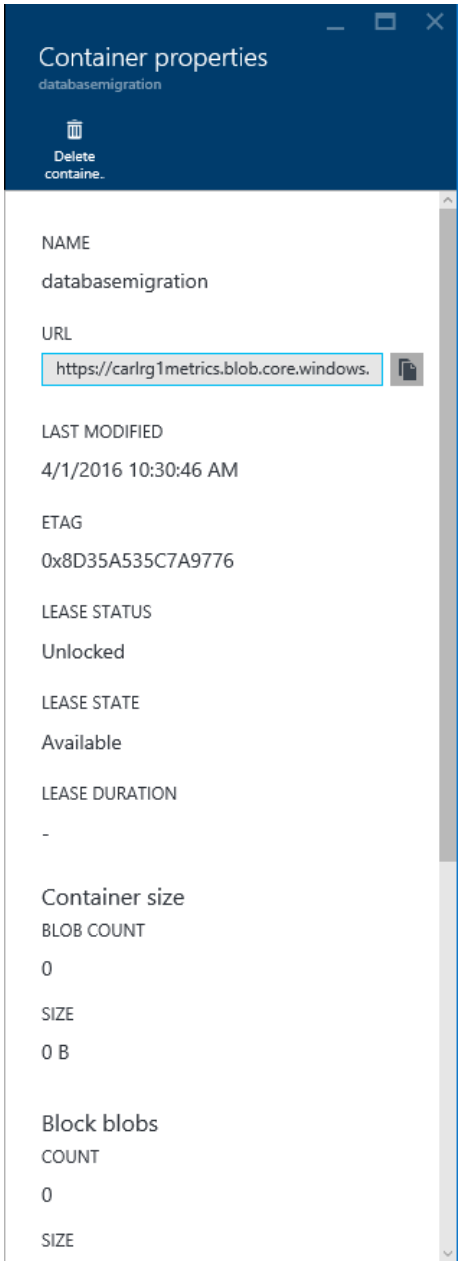


Figure 4-14: Container properties

11. Copy the URL for the container to your clipboard. See Figure 4-15.

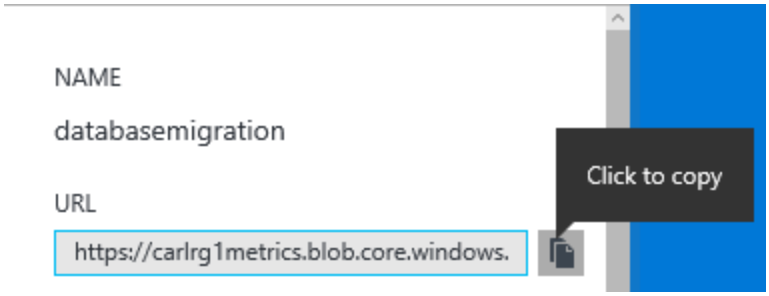


Figure 4-15: Copy URL

12. Paste this URL to a scratch pad location.
13. On the Settings blade for your storage account, click Access Keys. See Figure 4-16.

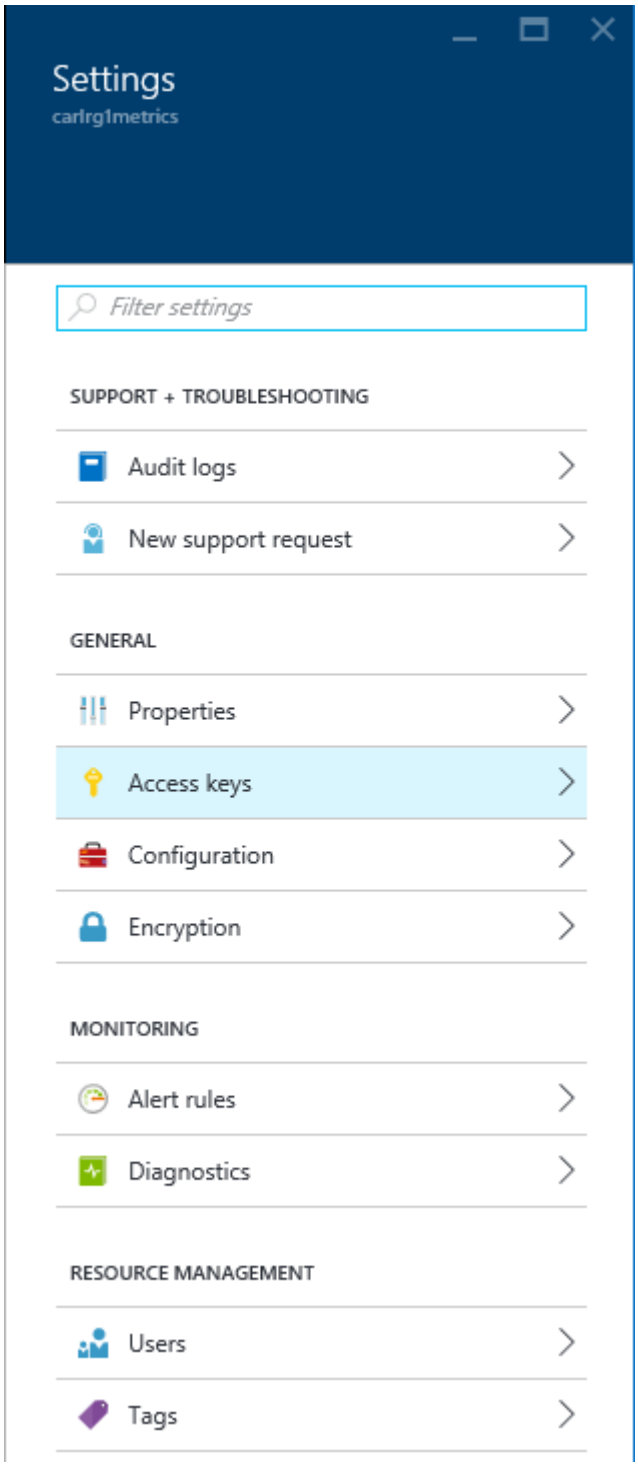


Figure 4-16: Settings blade

14. Copy one of your access keys to a scratch pad location.

15. Switch back to your command prompt window and then modify and paste the following command into the command prompt window, modifying the bracketed items to match your environment:

```
AzCopy /Source:<"C:\Program Files\Microsoft SQL
Server\MSSQL10_50.MSSQLSERVER\MSSQL\Backup\mybackupupfile.bak">
/Dest:<https://myaccount.blob.core.windows.net/mycontainer> /DestKey:<accesskey>
/Pattern:"<mybackupfile.bak>"
```

16. Execute the command and verify success. See Figure 4-17.

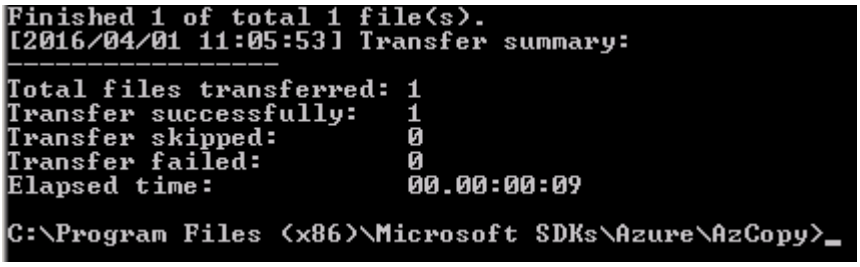


Figure 4-17: Successful copy to Azure blob storage

Notice that the copy is very fast.

Copying the backup file from Azure blob storage to the Azure virtual machine

Now that we have a full database backup in Azure blob storage, we will copy it to the virtual machine by using the AzCopy program.

1. Establish a remote desktop connection to your Azure virtual machine.
2. Download the latest version of AzCopy either on the computer containing the database backup file or on a computer with connectivity to that storage location.
3. Install the Microsoft Azure Storage Tools following the prompts.
4. After the installation finishes, open a command prompt and navigate to the install location for AzCopy.
5. In the command prompt window, modify and paste the following command, modifying the bracketed items to match your environment:

```
AzCopy /Source:<https://myaccount.blob.core.windows.net/mycontainer> /Dest:<"C:\Program Files\Microsoft
SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Backup\mybackupupfile.bak"> /SourceKey:<accesskey>
/Pattern:"<mybackupfile.bak>"
```

6. Execute the command and verify success. See Figure 4-18.

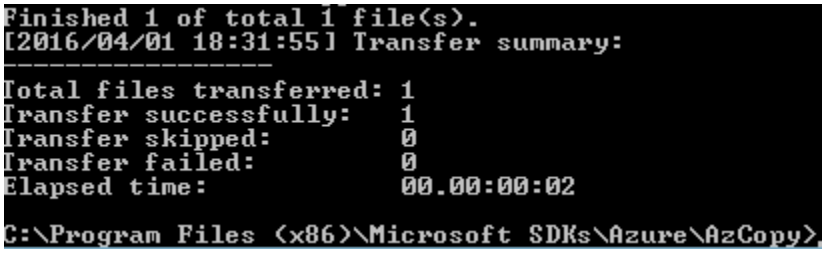


Figure 4-18: Successful copy from Azure blob storage to Azure virtual machine

7. Using File Explorer, verify that the copied backup file appears in the backup folder. See Figure 4-19.

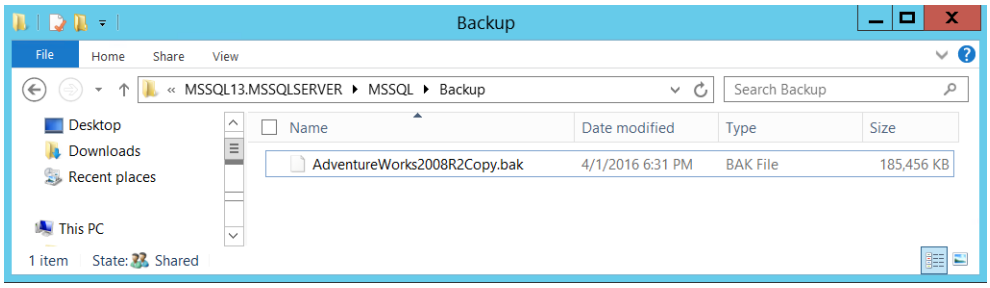


Figure 4-19: Backup file in Azure virtual machine file system

Restoring the SQL Server 2008 R2 user database backup to SQL Server 2016 in an Azure virtual machine

Now that you have copied the backup into the file system for the Azure virtual machine, you are ready to restore the backup file into SQL Server 2016 in your virtual machine, which automatically will upgrade the database as well.

1. Open SQL Server Management Studio in your Azure virtual machine and connect to your SQL Server 2016 instance in Object Explorer.

2. Right-click Databases and then click Restore Database. See Figure 4-20.

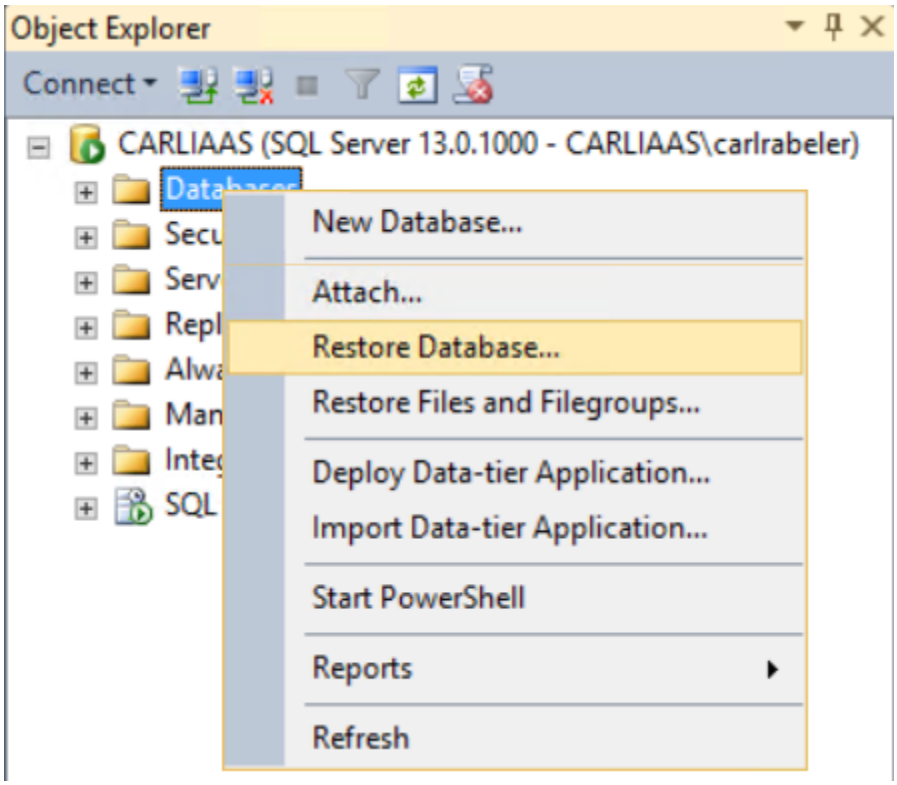


Figure 4-20: Restore database

3. In the Restore Database window, click Device and then click the ellipsis to select the backup device. See Figure 4-21.

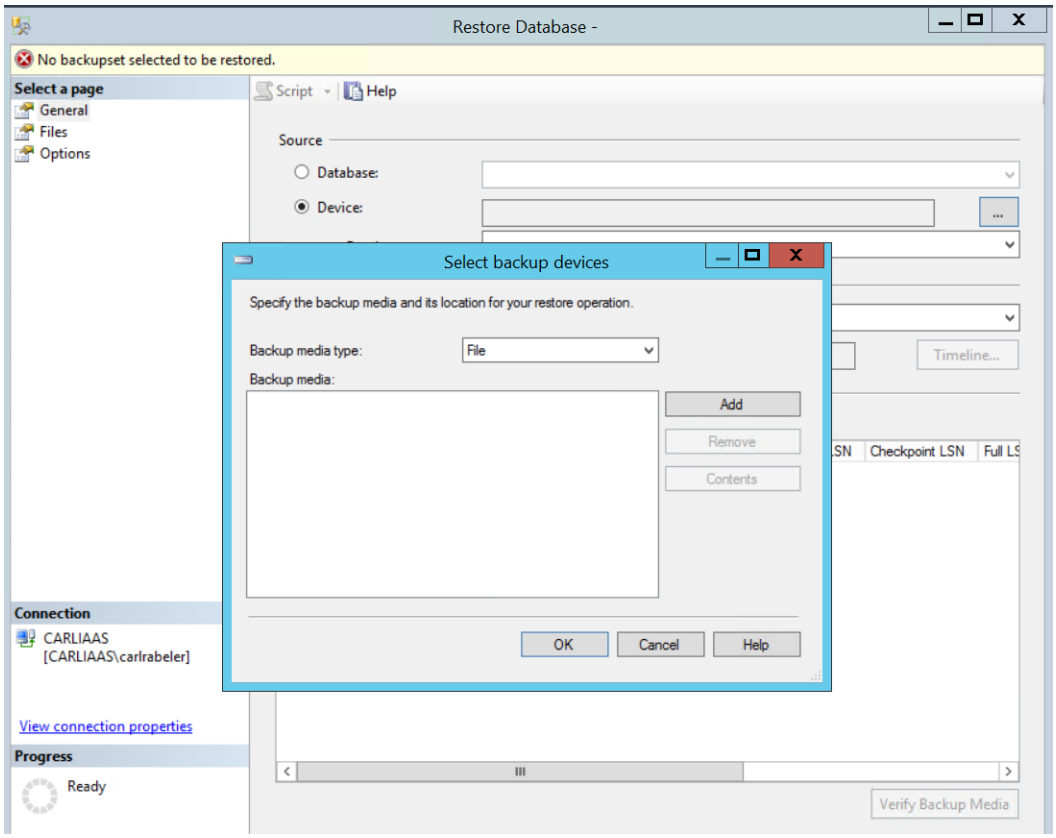


Figure 4-21: Select backup device

4. In the Select Backup Devices window, click Add and then, in the Locate Backup File window, select your backup file and click OK. See Figure 4-22.

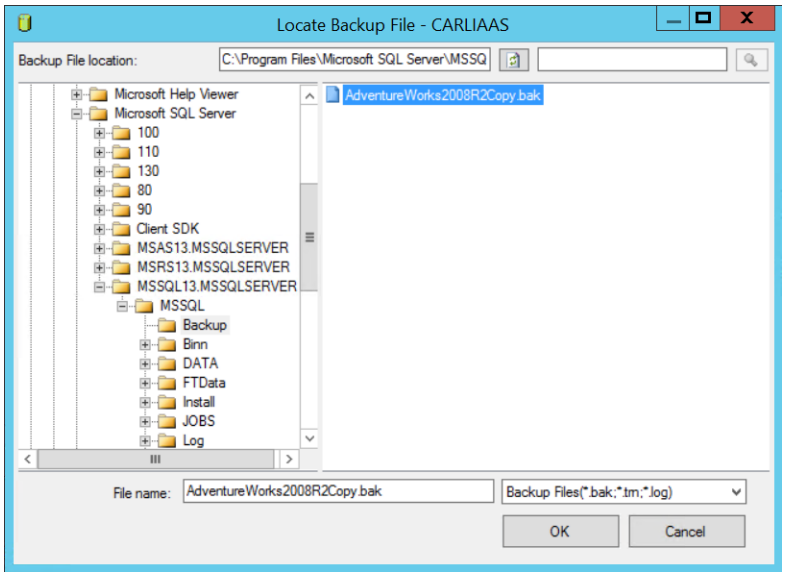


Figure 4-22: Specify the backup file

5. Click OK in the Select Backup Devices window and then review the information in the Restore Database window. See Figure 4-23.

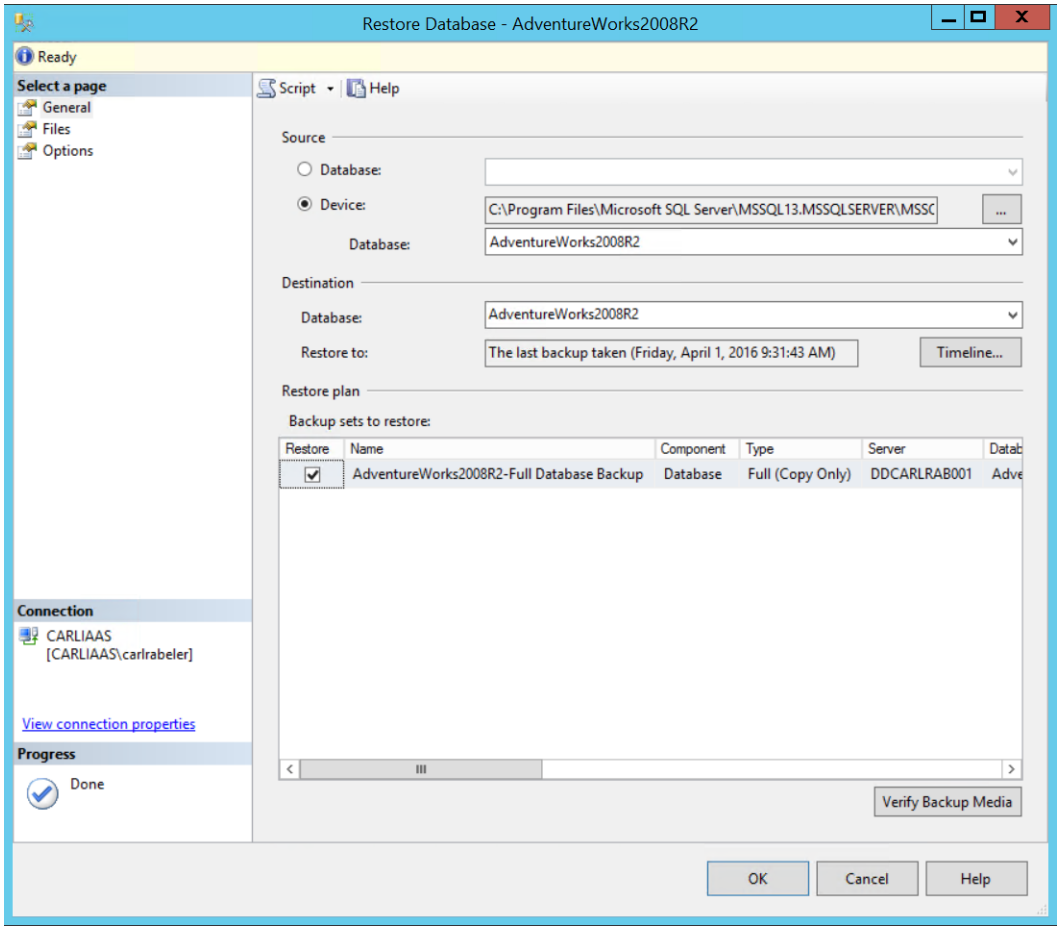


Figure 4-23: Ready to restore

6. Click OK to restore and upgrade from your backup file.

7. Verify that your database was restored successfully. See Figure 4-24.

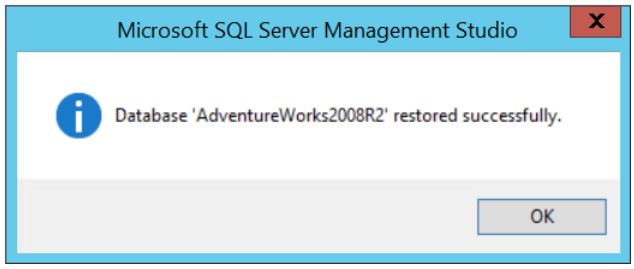


Figure 4-24: Restore success

8. Click OK to close the message box.

9. In Object Explorer, expand Databases to see your restored, upgraded, and migrated database. See Figure 4-25.

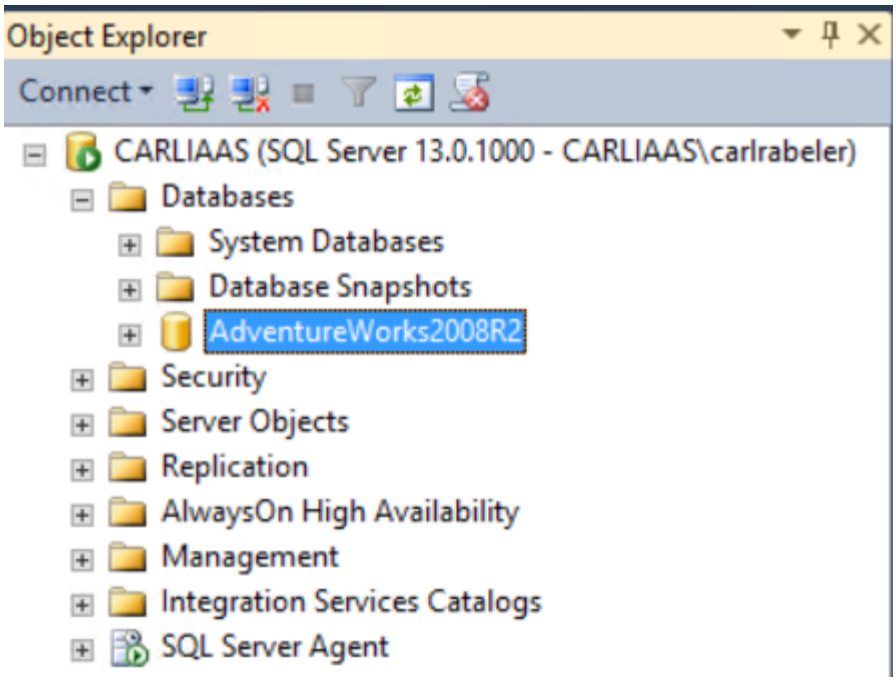


Figure 4-25: Restored, upgraded, and migrated database

10. Right-click the migrated database and click Properties.

11. In the Database Properties window, click Options. Notice the compatibility level is SQL Server 2008 (100). See Figure 4-26.

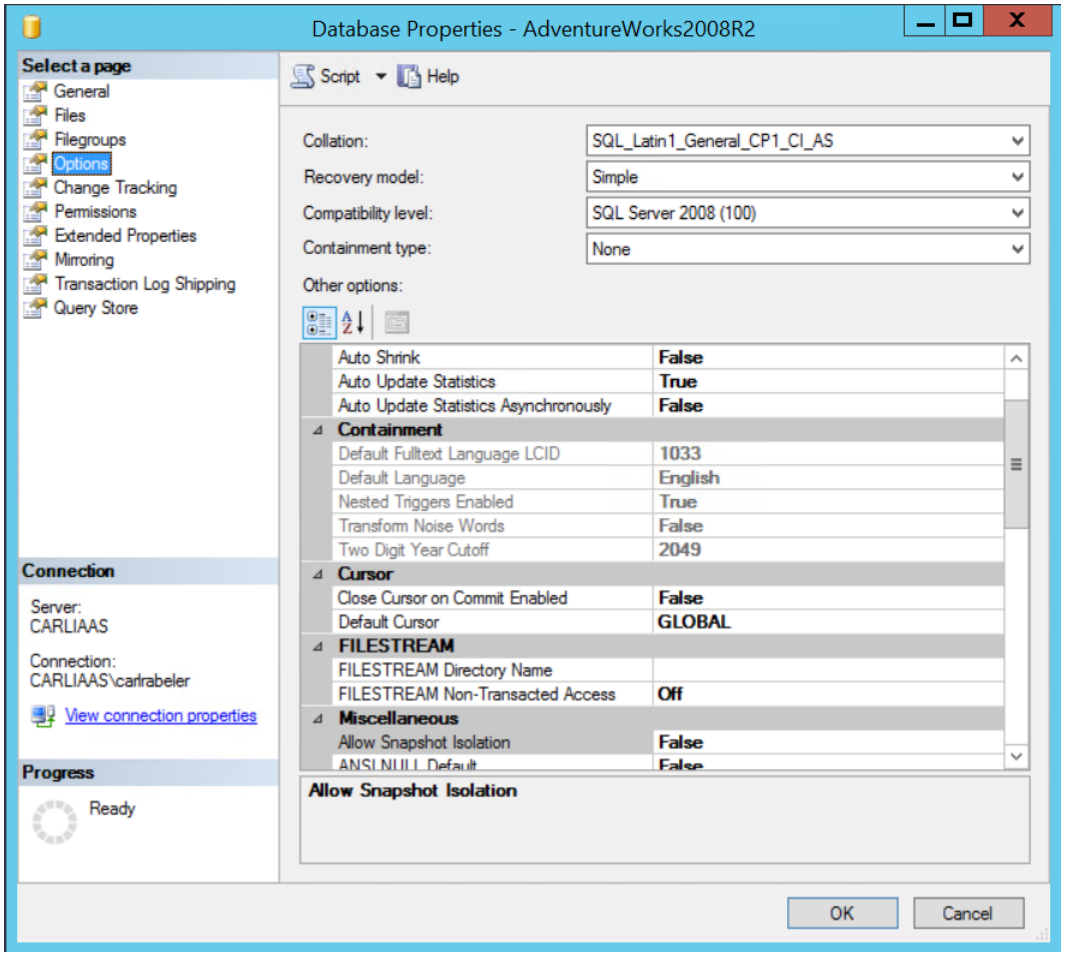


Figure 4-26: Database properties options

12. The final step in this migration and upgrade process is to set the compatibility level at which you want to run your migrated database. In SQL Server 2016, some changes are enabled only after the DATABASE_COMPATIBILITY level for a database has been changed to 130. For more information on changing the compatibility level to 130, see Change the Database Compatibility Mode and Use the Query Store.

Walk-through: Migrating a SQL Server database to SQL Database using a BACPAC file

In this walk-through, please follow along as we export a SQL Server 2008 R2 user database to a BACPAC file and then import that BACPAC file into SQL Database. Rather than perform these steps individually, we will use a Microsoft wizard to accomplish this task. There are currently two wizards we could use: the Deploy Database to Azure SQL Database Wizard in the newest version of SQL Server Management Studio and the preview version of Upgrade Advisor (as discussed earlier in this chapter). For this walk-through, I will use the former because its interface has been finalized. Upgrade Advisor will be replaced by Data Migration Assistant in late April or early May 2016 with a new UI.

Using the Deploy Database to Azure SQL Database Wizard to migrate a user database to SQL Database

1. Using the newest version of SQL Server Management Studio, connect to your SQL Server 2008 R2 instance (or your instance of choice).

2. Right-click the database that you wish to migrate, point to Tasks, and then click Deploy Database to Microsoft Azure SQL Database. See Figure 4-27.

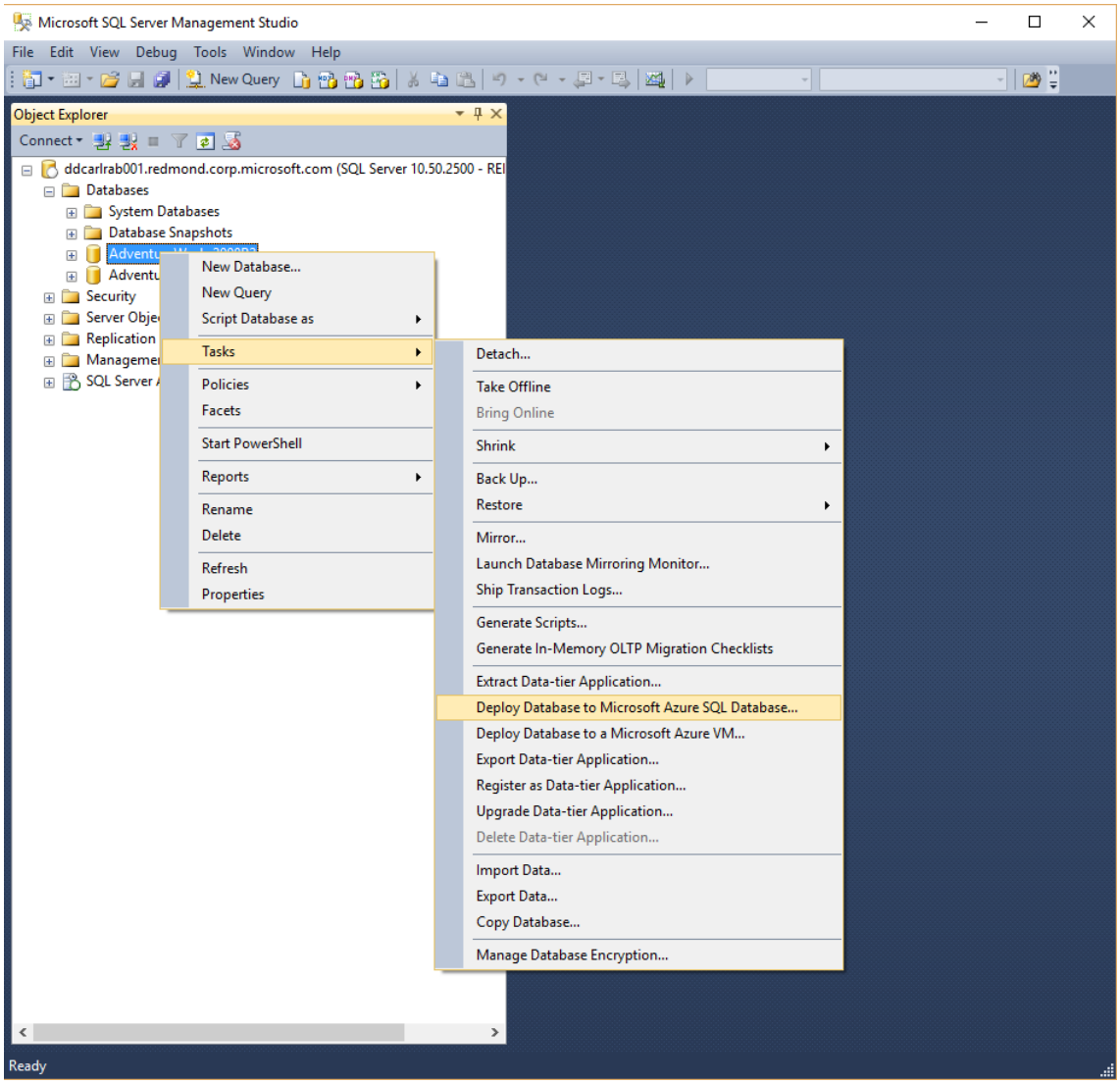


Figure 4-27: Deploy Database to Microsoft Azure SQL Database

3. On the Introduction page, click Next. See Figure 4-28.

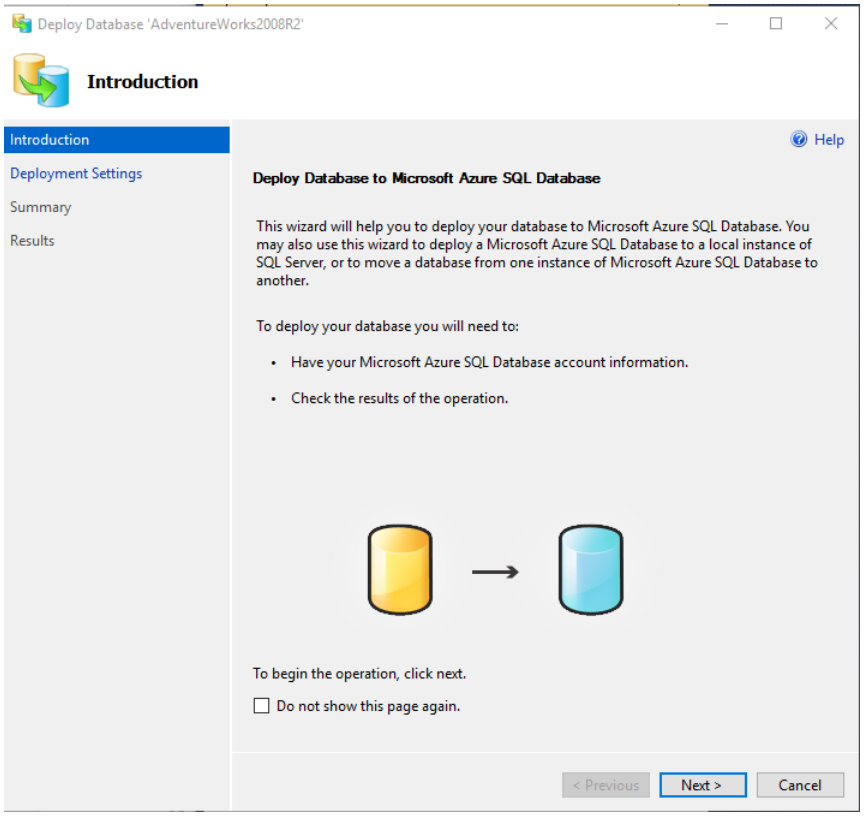


Figure 4-28: Introduction to the Deploy Database to Microsoft Azure SQL Database Wizard

4. On the Deployment Settings page, review the information completed for you for the destination database name and the temporary file name for the BACPAC file that will be exported and then imported. See Figure 4-29.

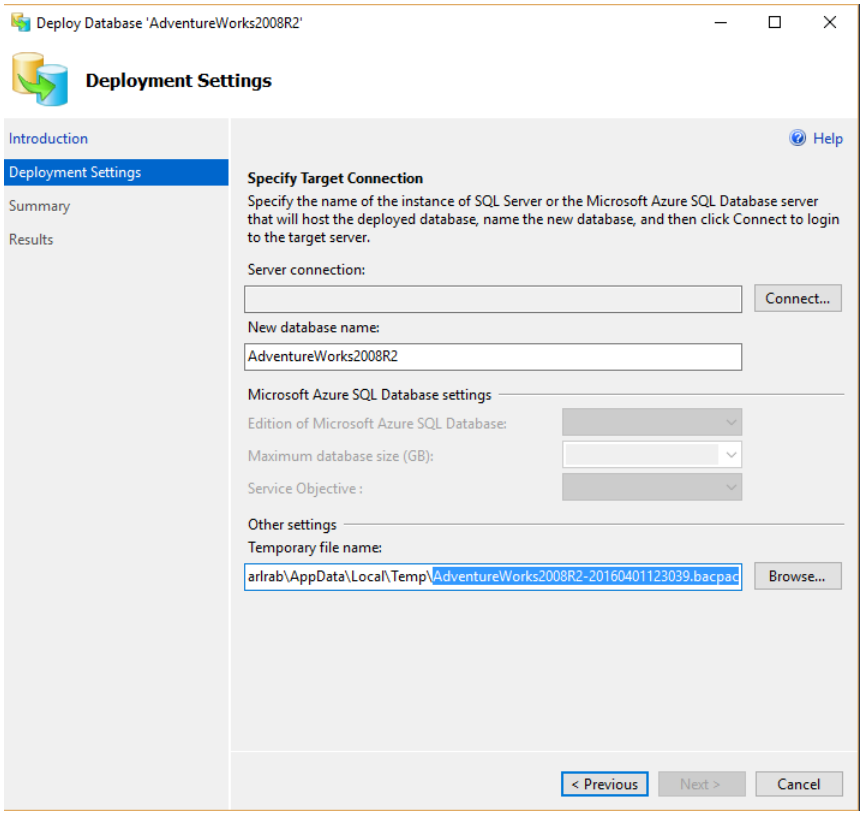


Figure 4-29: Deployment Settings default values

5. Click Connect to establish a connection to your existing SQL Database logical server. See Figure 4-30.

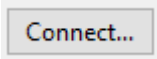


Figure 4-30: Connect button

6. In the Connect to Server dialog box, specify the Azure SQL Database logical server name and your authentication information and then click Connect. See Figure 4-31.

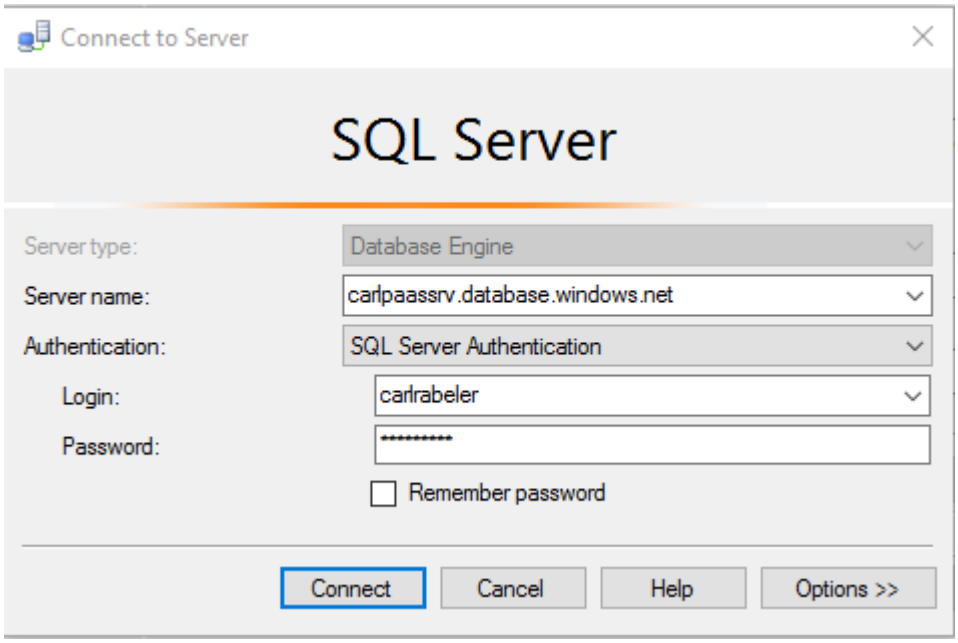


Figure 4-31: Connect to Azure SQL Database logical server

7. On the Deployment Settings page, review the Microsoft Azure SQL Database settings for this migrated database. Notice that the default is Basic edition, 2 GB in size, and Basic Service Objective. See Figure 4-32.

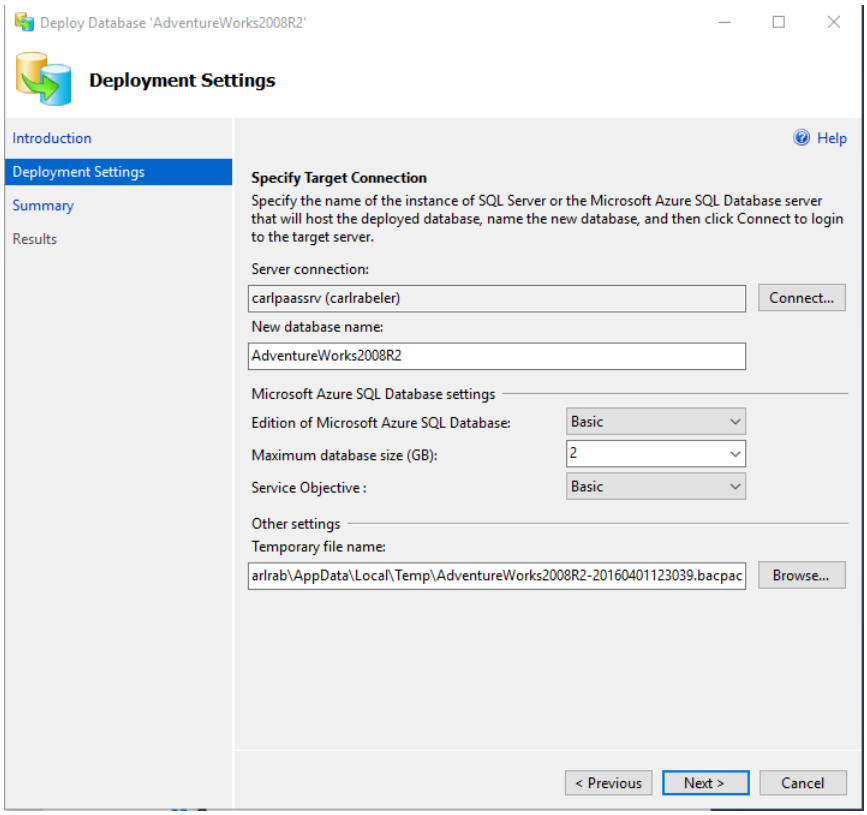


Figure 4-32: Microsoft Azure SQL Database settings

8. Change the SQL Database settings as desired. For this walk-through, I will change the edition to Standard, change the maximum database size to 10 GB, and change the service objective to S3. See Figure 4-33.

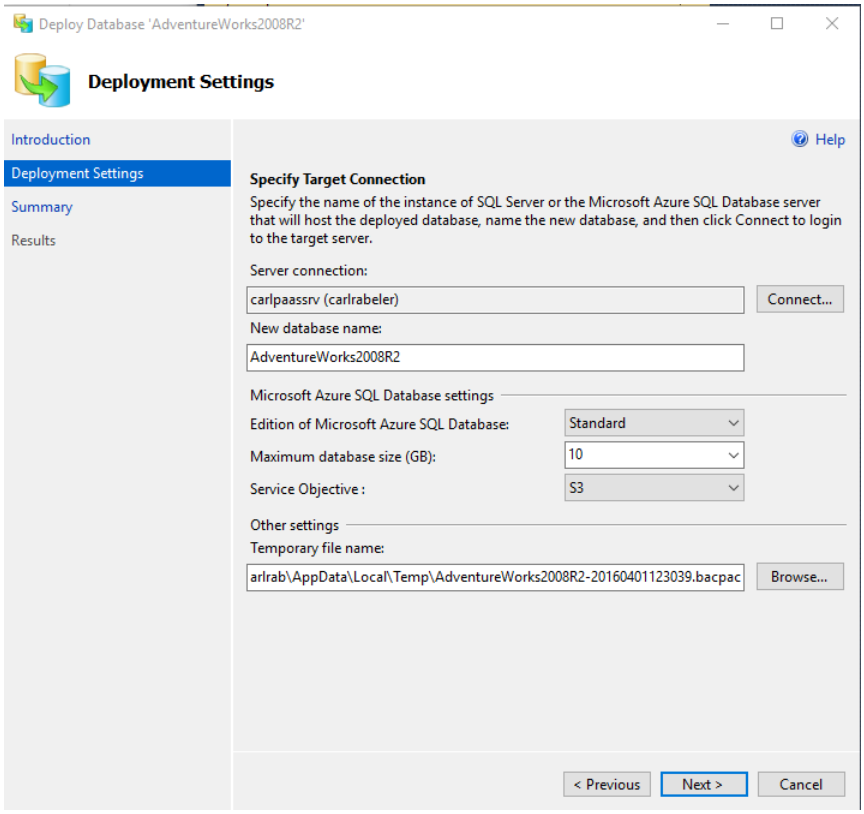


Figure 4-33: Modified SQL Database settings

9. Click Next to advance to the Summary page. See Figure 4-34.

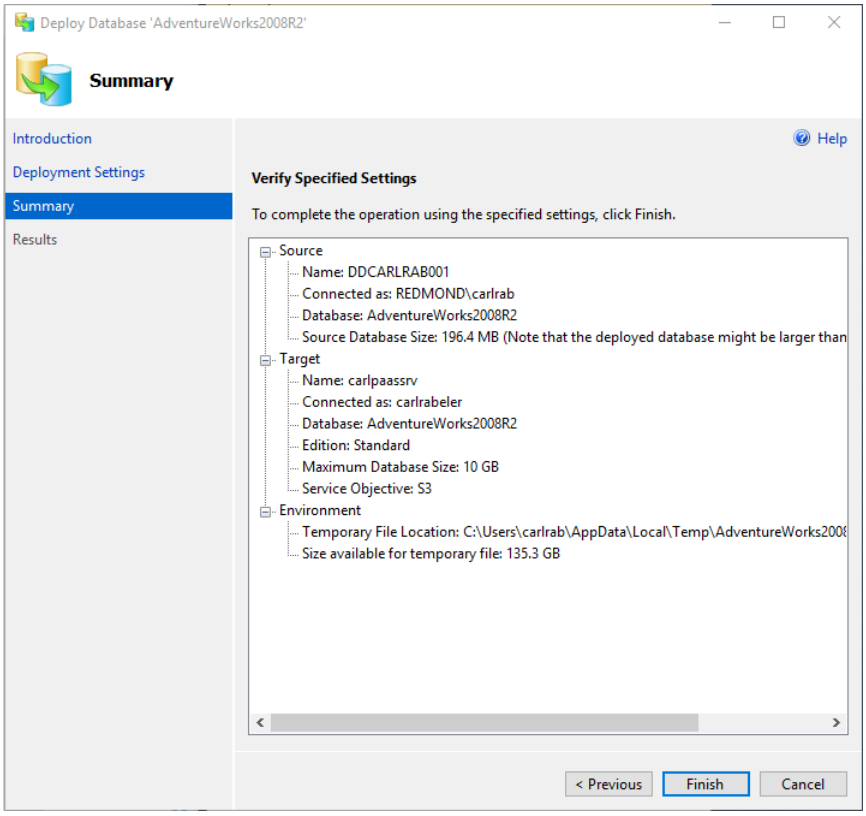


Figure 4-34: Summary page

10. Review the specified settings and click Finish. See Figures 4-35, 4-36, and 4-37. Notice in Figure 4-35 that the first step is to create the BACPAC file. Notice in Figure 4-36 that the next steps are to validate the schema and, if successful, export the data. If your source database is not compatible, this wizard will fail and you will need to fix compatibility errors using SQL Server Data Tools for Visual Studio. Figure 4-37 shows the importing of the schema and the data to your Azure SQL Database.

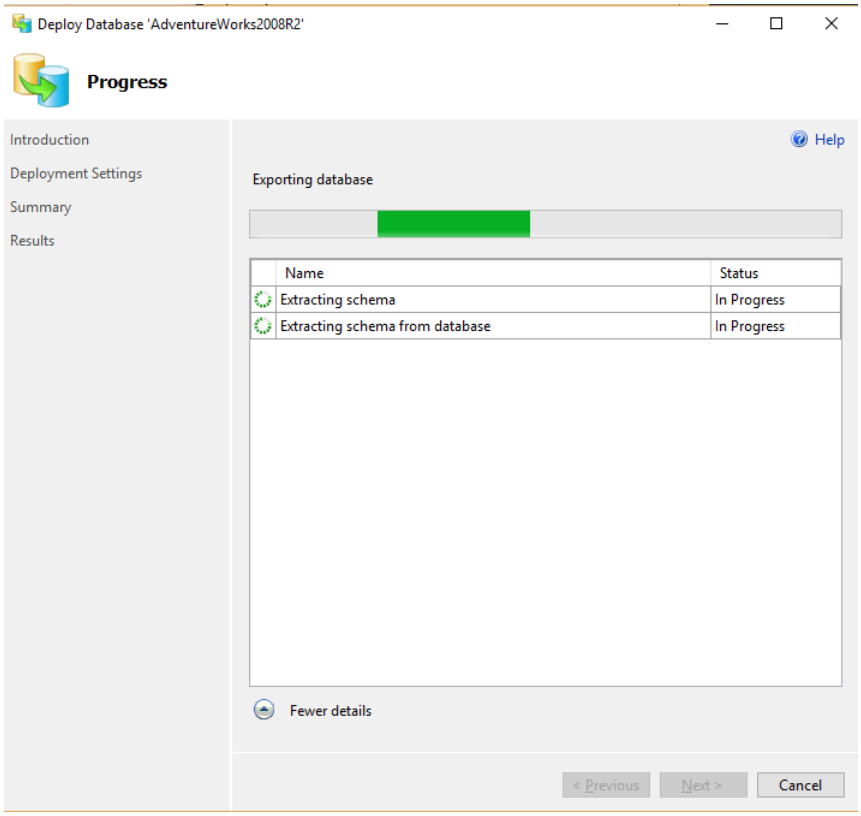


Figure 4-35: Creating BACPAC file

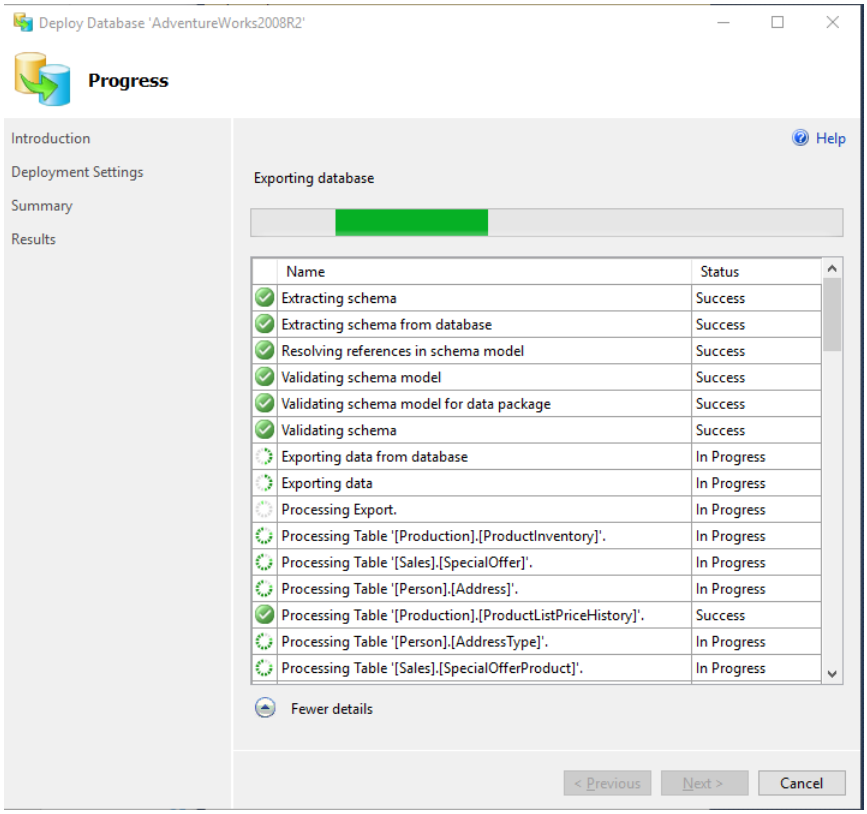


Figure 4-36: Validating BACPAC file and exporting data

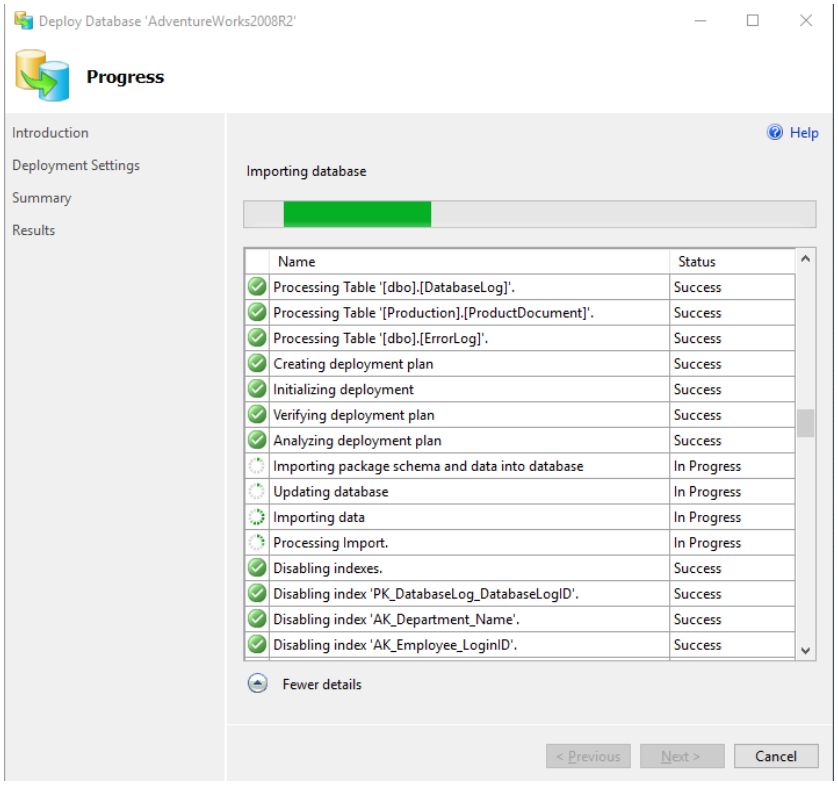


Figure 4-37: Importing schema and data

11. When the wizard is complete, verify that no errors appear and then click Close. See Figure 4-38.

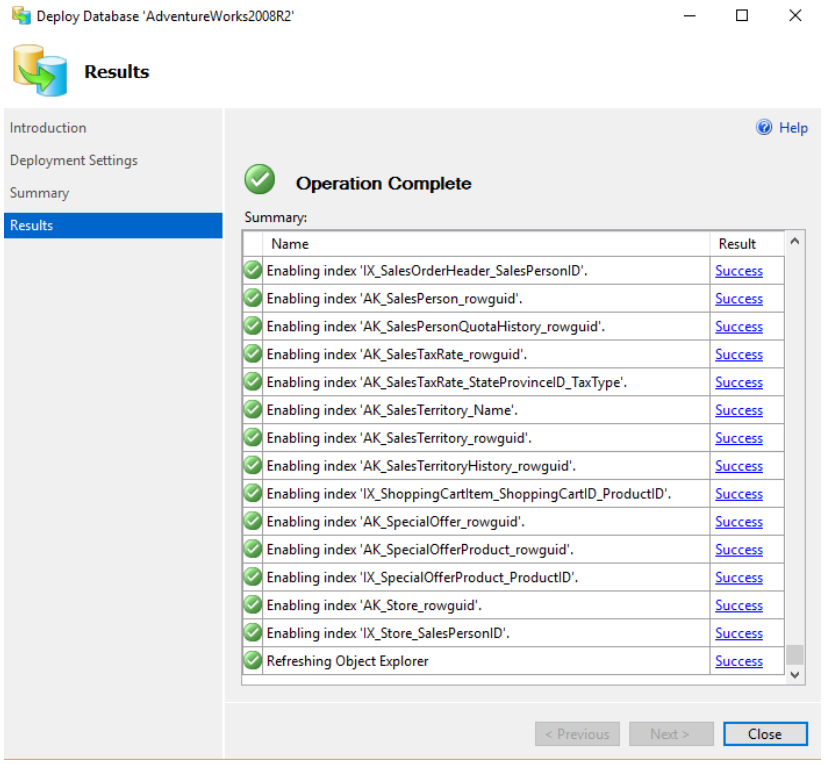


Figure 4-38: Migration complete

12. Click Close.

13. Next, using SQL Server Management Studio, connect to your Azure SQL Database logical server and verify that your database was migrated successfully. See Figure 4-39.

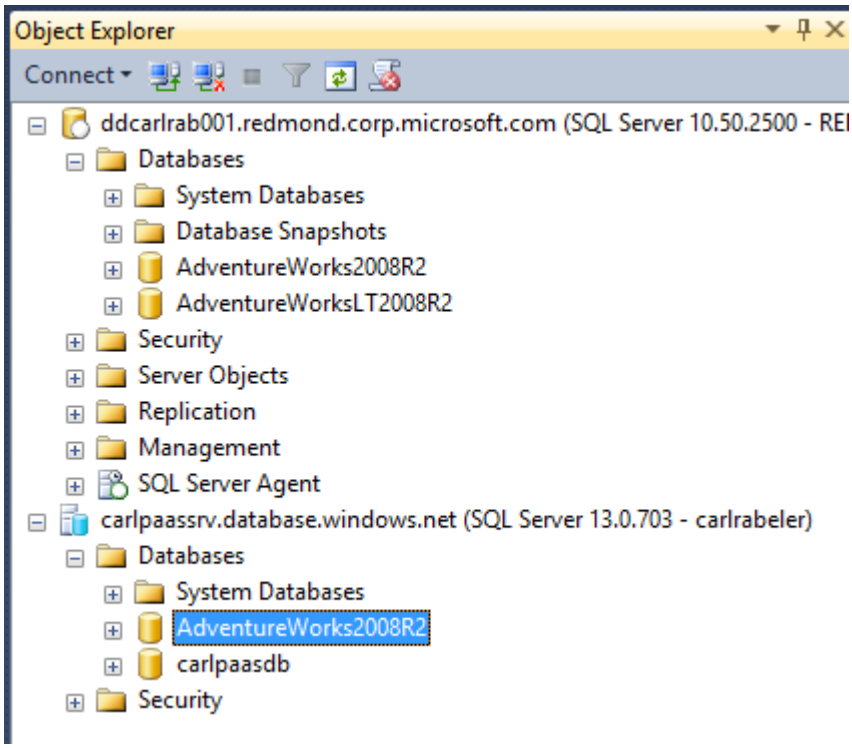


Figure 4-39: Migrated database

14. In Object Explorer, right-click your migrated database and click Properties.

15. In the Database Properties dialog box, click Options. Notice the greyed-out version of the compatibility level for this migrated database. This value is just showing what the value was before migration. See Figure 4-40.

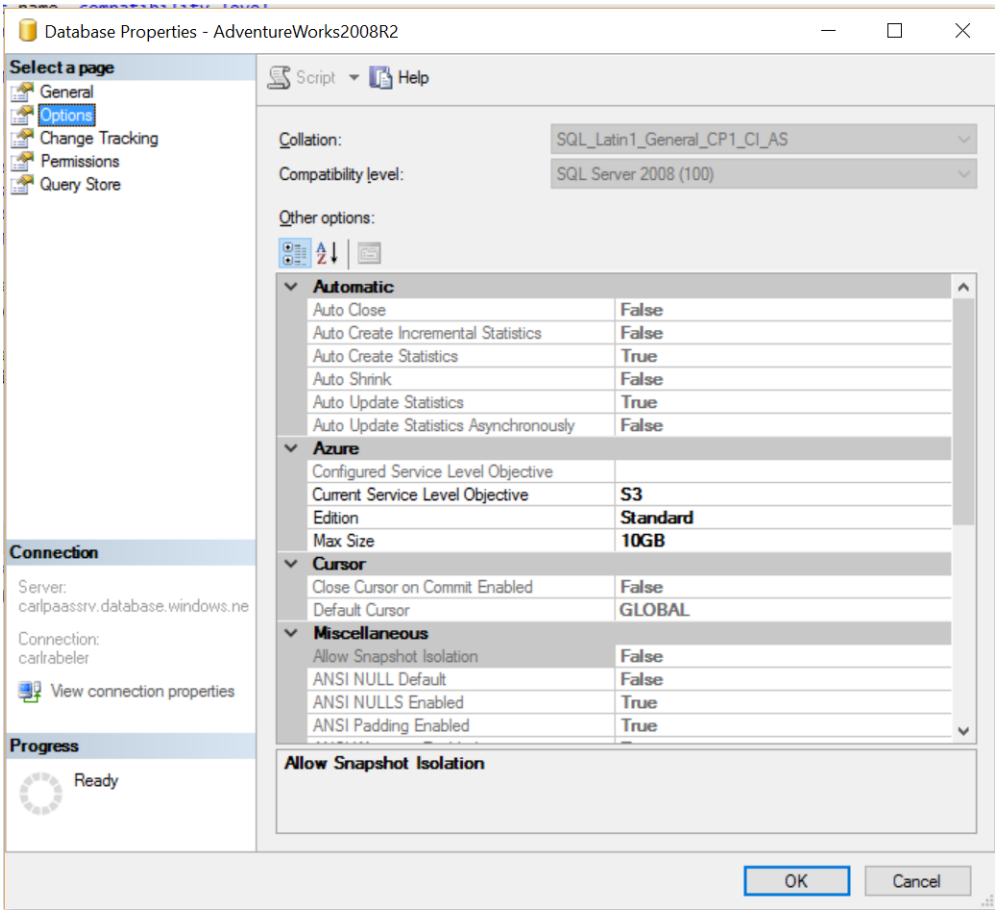


Figure 4-40: Azure SQL Database compatibility level

16. Click Cancel.

17. Right-click your migrated database and then click New Query.

    The final step in this migration and upgrade process is to set the compatibility level at which you want to run your migrated database.

18. Execute the Transact-SQL script below to change the compatibility level of your migrated database to 130. For Azure SQL Database, some changes are enabled only after the DATABASE_COMPATIBILITY level for a database has been changed to 130. For more information on changing the compatibility level to 130, see Change the Database Compatibility Mode and Use the Query Store.

```
ALTER DATABASE Adventureworks2008r2 SET COMPATIBILITY_LEVEL = 130
```

19. Execute the Transact-SQL script below to verify the change. See Figure 4-41.

```
select name, compatibility_level from sys.databases
```

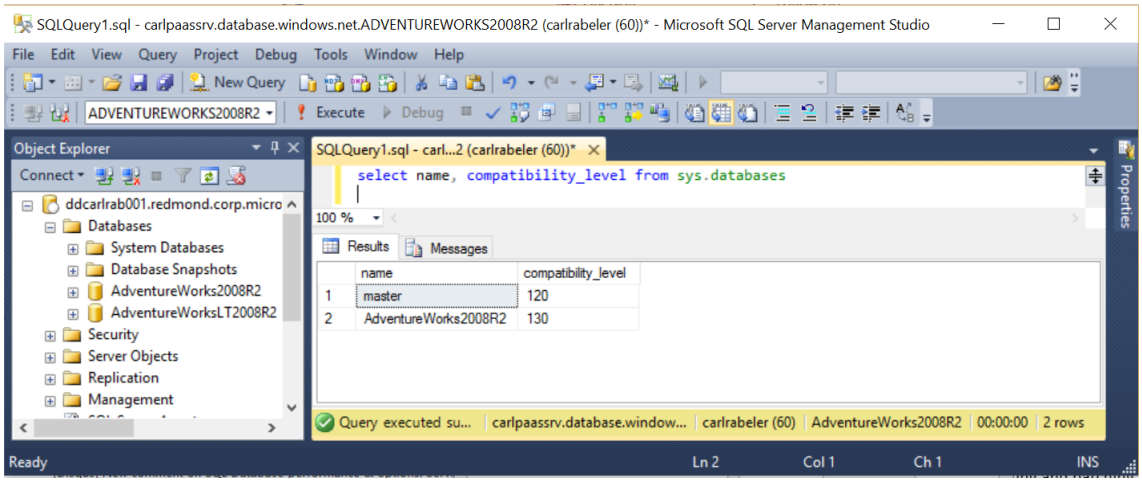
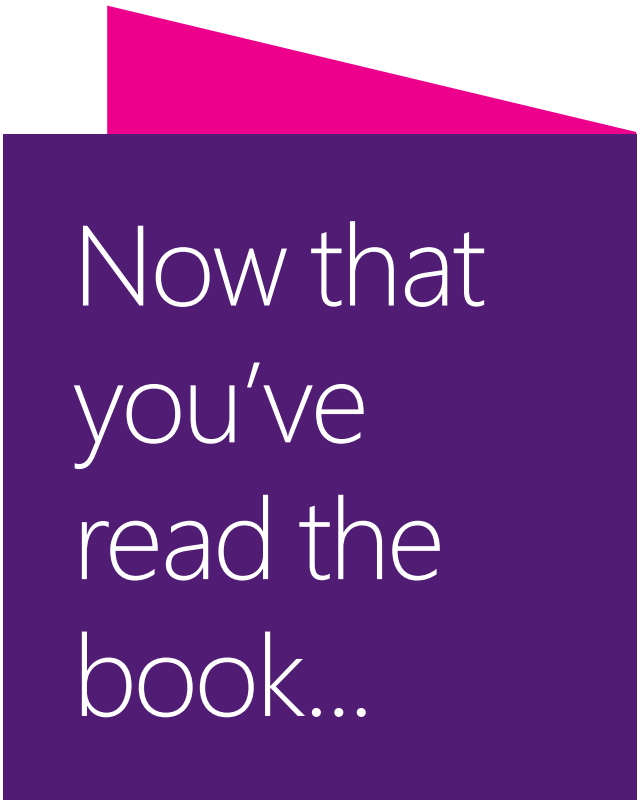


Figure 4-41: Changed database compatibility level

Conclusion

In this chapter, you learned how to migrate your on-premises database to SQL Server in a virtual machine or the Azure SQL Database service. In the walk-throughs, you learned how simple it is to perform this migration with a compatible database.



Tell us what you think!

Was it useful?
Did it teach you what you wanted to learn?
Was there room for improvement?

Let us know at http://aka.ms/tellpress

Your feedback goes directly to the staff at Microsoft Press,
and we read every one of your responses. Thanks in advance!

Microsoft

CHAPTER 5

Authentication, authorization, and data resiliency

In this chapter, I will discuss protecting access to your data through authentication over a secure network and granting permissions to authenticated users to perform tasks and access data. I will also discuss protecting and archiving your data using database backups and database exports. I will compare and contrast the mechanisms available to you with SQL Server in an Azure virtual machine and with Azure SQL Database. We will finish the chapter with a guided walk-through of configuring authentication and granting permissions in Azure SQL Database and a walk-through demonstrating how to use Azure SQL Database automated backups to recover from user error and to archive data.

Securing connections

Protecting your data begins with securing connections to SQL Server in the virtual machine and to Azure SQL Database. In the next two sections, I will talk about the security options you have to restrict and secure connections, independent of user authentication and authorization.

Configuring and securing connections to SQL Server in an Azure virtual machine

With an Azure virtual machine, you have several options to restrict and secure connections to your SQL Server instance. The virtual network for your Azure virtual machine is a logical isolation of the Azure cloud dedicated to your subscription. You can fully control the IP address blocks, DNS settings, security policies, and route tables within this network, similarly to how you use these mechanisms to control your on-premises network. You can also segment your virtual network into subnets to further control access to the virtual machines on your virtual network that host your SQL Server instances.

In addition, you can connect the virtual network to your on-premises network using one of the connectivity options available for Azure virtual machines. In essence, you can expand your on-premises network to your Azure virtual network, delivering the benefit of enterprise scale that Azure provides. Finally, you can set up and configure an Azure virtual machine as a domain controller and join your SQL Server virtual machine to this Azure domain controller. This Azure domain controller can be federated with Azure Active Directory, be federated with your on-premises Active Directory, or be a controller within your existing on-premises Active Directory. A full discussion of your options and best practices for configuring a secure connection to your on-premises network is beyond the scope of this ebook. If you do so, you can join your virtual machine to your on-premises Active Directory environment and use Windows user accounts for authentication.

In addition to configuring and securing the virtual network to which your Azure virtual network is connected, you should take these security steps:

- Use a unique local administrator account for your virtual machine that does not have a name of Administrator.
- Use complex strong passwords for all of your accounts, Windows and SQL Server. For more information about how to create a strong password, see the Create Strong Passwords article in the Safety and Security Center.
- Enable encrypted connections for your SQL Server instance and configure your SQL Server instance with a signed certificate.
- Use Windows firewall rules to control database engine access.
- If your virtual machines should be accessed only from a specific network, use network security groups (NSGs) to control traffic and restrict access to certain IP addresses or network subnets. An NSG contains access control rules that allow or deny traffic based on traffic direction, protocol, source address and port, and destination address and port.

Configuring and securing connections to Azure SQL Database

With Azure SQL Database, Azure configures the network for you and by default restricts all access using Azure SQL Database firewall rules, as discussed in Chapter 3, "Getting started with an Azure SQL Database." If a firewall rule does not exist, Azure will reject all connection attempts from IP addresses that have not been whitelisted explicitly.

- Use database-level firewall rules in conjunction with contained users (discussed below) whenever possible to make your database more portable.
- Use server-level firewall rules when you have many databases that have the same access requirements and you don't want to spend time configuring each database individually.

Additionally, Azure SQL Database requires encrypted connections at all times while data is "in transit" to and from the database. In your application's connection string, you must specify parameters to

encrypt the connection and *not* to trust the server certificate (this is done for you if you copy your connection string out of the Azure portal). If you do not, the connection will not verify the identity of the server and will be susceptible to "man-in-the-middle" attacks. For the ADO.NET driver, for instance, these connection string parameters are Encrypt=True and TrustServerCertificate=False. For more information, see Azure SQL Database Connection Encryption and Certificate Validation. The code block below shows a sample ADO.NET connection string.

```
Server=tcp:[your_sql_database_server_name_here].database.windows.net,1433;Database=[your_sql_database_na
me_here]>;User
ID=[your_username_here]@[your_sql_database_server_name_here];Password={your_password_here};Encrypt=True;
TrustServerCertificate=False;Connection Timeout=30;
```

Unlike SQL Server in an Azure virtual machine, with Azure SQL Database you do not have to secure the operating system itself. However, all Azure subscription administrators have access to the SQL Database instance, and you should limit subscription administrators.

User authentication

When configuring user authentication with SQL Server in an Azure virtual machine and with Azure SQL Database, you can use SQL Server Authentication. If you have configured Active Directory and/or Azure Active Directory, you can also use Active Directory accounts. In addition, with Azure SQL Database, Azure Role-Based Access Control (RBAC) can be used to enable an Azure user to manage an Azure SQL Database instance. In the next two sections, I will discuss these user authentication options in SQL Server in an Azure virtual machine and in Azure SQL Database using both SQL Server and Windows user accounts.

User authentication with SQL Server in an Azure virtual machine

The default authentication mode for SQL Server in an Azure virtual machine (and in a SQL Server on-premises installation) is Windows Authentication mode. With this mode, users must connect and be authenticated using an Active Directory user account. In this mode, SQL Server Authentication is disabled and the SA login account is also disabled. You can change the SQL Server Authentication mode to mixed mode authentication. Mixed mode enables both Windows Authentication and SQL Server Authentication. Windows Authentication always is available and cannot be disabled.

User authentication using mixed mode

If you do not join your Azure virtual machine to an Active Directory domain, you generally will use SQL Server Authentication for user authentication. SQL Server Authentication is less secure than Active Directory authentication for the following reasons:

- The encrypted SQL Server Authentication login password must be passed over the network at the time of the connection. Some applications that connect automatically will store the password at the client. These are additional attack points.
- If a user is an Active Directory domain user who has a login and password, the user must provide SQL Server login and password to connect. Keeping track of multiple names and passwords is difficult for many users. Having to provide SQL Server credentials every time one connects to the database can be annoying.
- SQL Server Authentication cannot use Kerberos security protocol.

However, SQL Server Authentication may be the best option for SQL Server in an Azure virtual machine in the following scenarios:

- If the SQL Server database users do not have Active Directory user accounts and cannot be authenticated by an Active Directory domain

- If the web-based applications accessing SQL Server in the virtual machine allow users to create their own identities
- If the software applications accessing the SQL Server instance use a complex permission hierarchy based on known SQL Server logins

To use SQL Server Authentication, you must enable mixed mode authentication and restart the SQL Server instance. When you do so, take the following steps to increase the security for your SQL Server instance:

- Use an account other than the SA login for SQL Server administration.
- Enforce password expiration.
- Enforce Windows password policies, including password length and complexity.
- Require users to change their password at their next login.

When configuring SQL Server in an Azure virtual machine using the Resource Manager deployment model, you have the option to have Azure configure SQL Server for mixed authentication mode when SQL Server is configured on the Azure virtual machine. When you do so, the default SQL Server login account and password are the same as the local administrator account for the virtual machine, although you can define a different account. Password policies are enforced, and the SA login account is not enabled.

If you do not change the authentication mode during the initial configuration of the SQL Server instance from the SQL Server default of Windows Authentication mode to mixed mode, you can use Windows remote desktop to connect to the virtual machine and then use SQL Server Management Studio to connect to SQL Server locally and change the authentication mode if desired.

> **Note** On the blade for your virtual machine in the Azure portal, you (and any subscription administrator) have the option to change the password of the local administrator account. This will also enable you (and any subscription administrator) to remote into the virtual machine, log onto SQL Server using the local administrator account, and manage SQL Server in the Azure virtual machine (including reading any data).

User authentication using Windows Authentication mode

If you join your Azure virtual machine to an Active Directory domain, you can leave SQL Server in Windows Authentication mode and require your users to connect and be authenticated using Active Directory authentication. When a user connects through an Active Directory user account, SQL Server validates the account name and password using the Active Directory principal token in the operating system. This means the user identity is confirmed by Active Directory. SQL Server does not ask for the password and does not perform the identity validation. Active Directory authentication uses Kerberos security protocol, provides password policy enforcement with regard to complexity validation for strong passwords, provides support for account lockout, and supports password expiration.

In addition, using Windows Authentication mode enables you to manage authentication using groups rather than individual user accounts and avoid the proliferation of user identities across database servers.

> **Note** If you do not join your machine to Active Directory, you still can use Windows Authentication mode by using local user accounts. By default, the local administrator account of your Azure virtual machine is added automatically as a member of the sysadmin server role.

User authentication with Azure SQL Database

Unlike SQL Server in an Azure virtual machine, the default authentication mode for Azure SQL Database is mixed mode authentication. Furthermore, with Azure SQL Database, you cannot switch from mixed mode to Windows Authentication mode.

User authentication using SQL Server Authentication

When you create the logical server for your database, you specify a server admin login with a username and password. Using these credentials, you can authenticate to (and manage) any database on that server as the database owner, or dbo. In Figure 5-1, I use SQL Server Management Studio (SSMS) to connect to the carlpaassrv logical server. I am authenticated by the master database. Notice that my user account has a login in the security folder and a user account in the logical master database. Notice also that my account does not have a user account in the carlpaasdb database, but I can manage it through my membership in the dbo or server admin role.

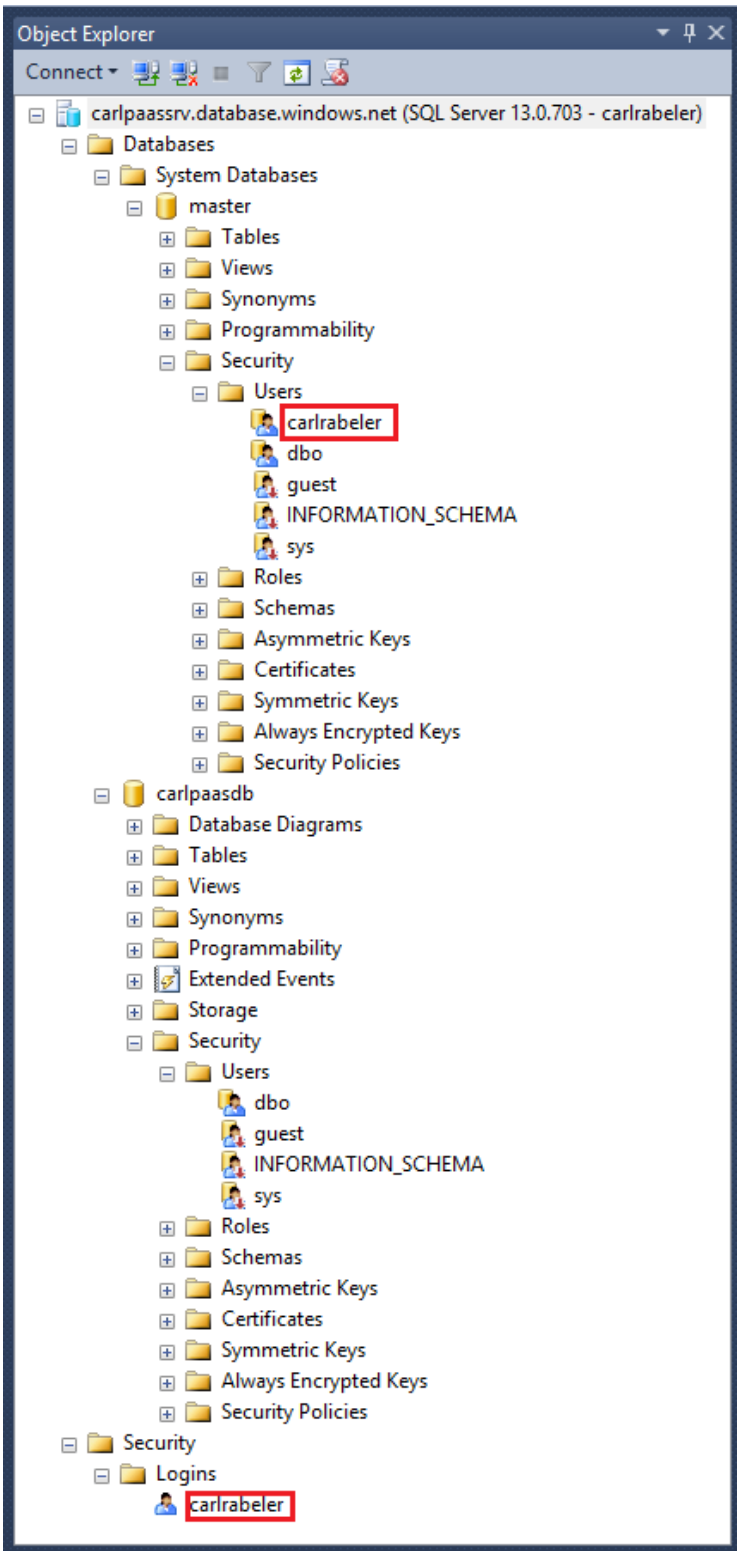


Figure 5-1: SQL Server login

User authentication using Azure Active Directory authentication

If you join your Azure SQL Database logical server to Azure Active Directory (Azure AD), you can use Active Directory authentication and require your users to connect using Active Directory authentication. Azure AD can be a stand-alone implementation of Active Directory, can be federated with your on-premises Active Directory, or can be an infrastructure as a service (IaaS) Active Directory implementation. The security and administrative benefits of using Active Directory user accounts are the same as with SQL Server on-premises or SQL Server in a virtual machine. See Figure 5-2.

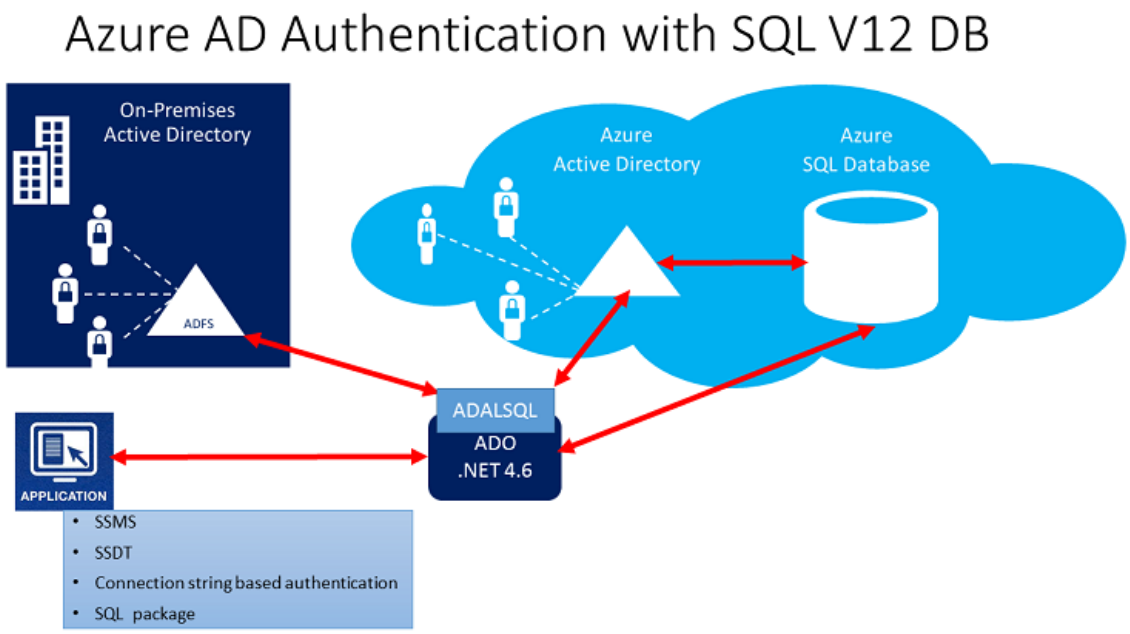


Figure 5-2: Azure AD authentication

Configuring Azure Active Directory (Azure AD) is beyond the scope of this ebook. Use the following links to learn how to set up Azure Active Directory:

- Getting started with Azure AD
- Add your custom domain name to simplify sign-in using Azure Active Directory
- Connecting to SQL Database By Using Azure Active Directory Authentication

After you have configured Azure AD, you create another server admin called the Azure AD admin (these are the only two server principals you can create). See Figure 5-3.

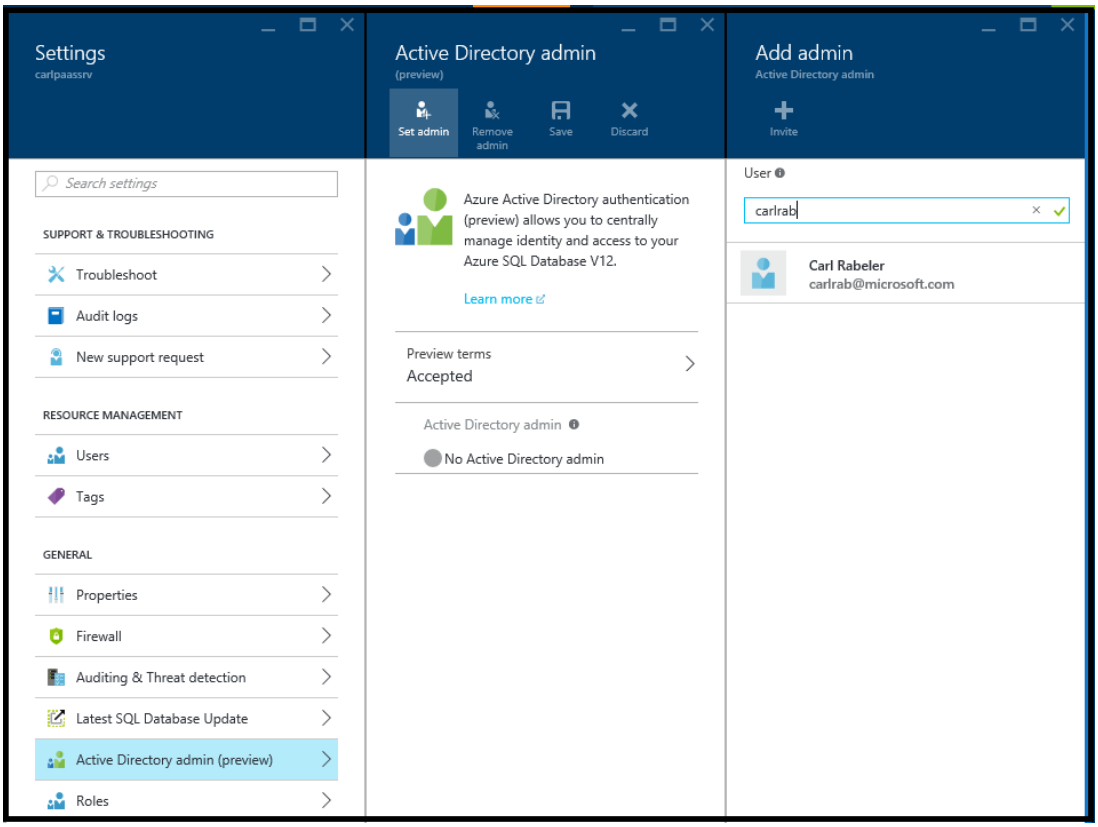


Figure 5-3: Setting Active Directory admin

See Connecting to SQL Database by Using Azure Active Directory Authentication for a walk-through of how to create an Azure AD admin to enable Azure Active Directory authentication.

The Azure AD administrator login can be an Azure AD user or an Azure AD group. When the administrator is a group account, any group member can use it, enabling multiple Azure AD administrators for the SQL Server instance. Using a group account as an administrator enhances manageability by allowing you to add and remove group members centrally in Azure AD without changing the users or permissions in SQL Database. Only one Azure AD administrator (a user or group) can be configured at any time.

The Active Directory admin account will be added as a user account in the master database as a contained user (more on contained users in the next section). No login for this account is created. See Figure 5-4.

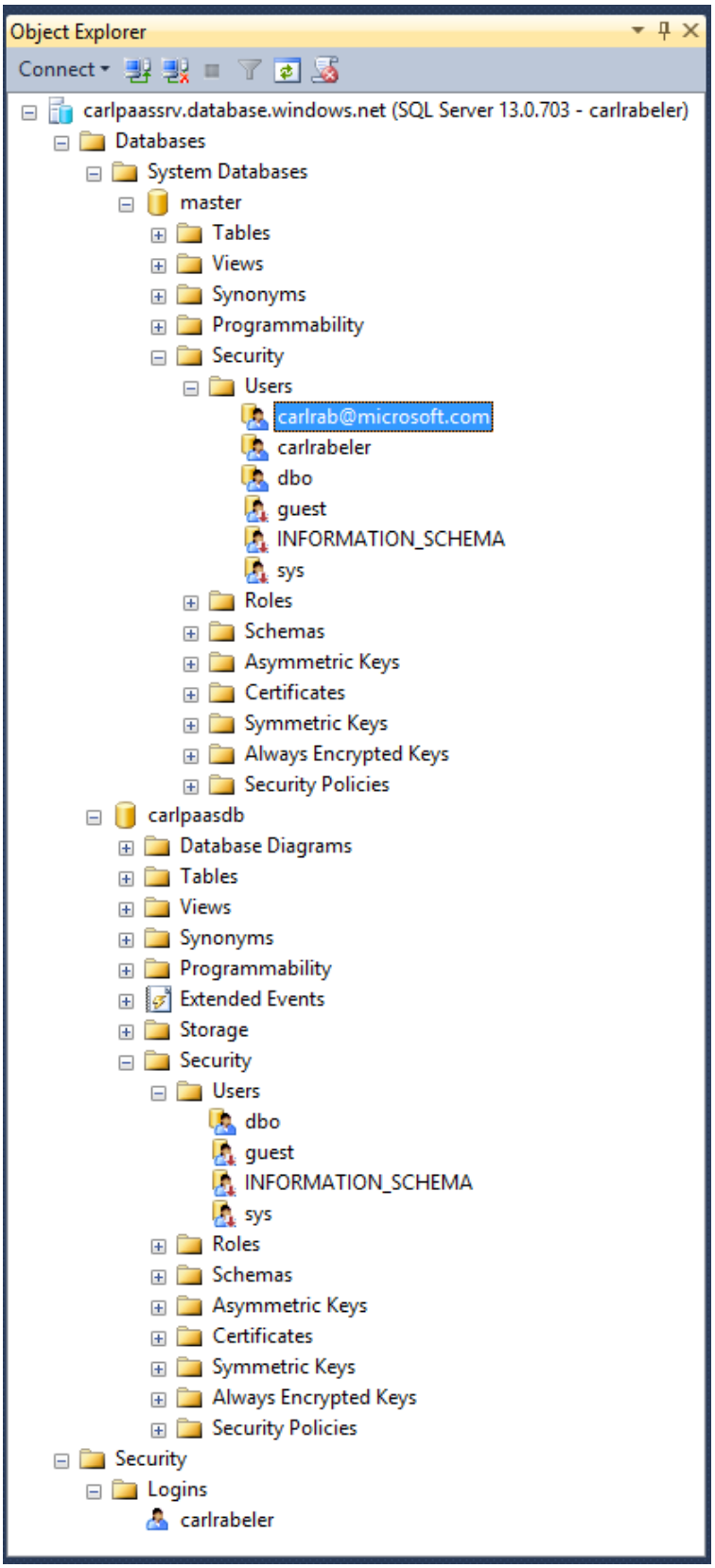


Figure 5-4: Two administrator accounts

When using Azure AD authentication, there will be two administrator accounts for the SQL Database server: the original SQL Server administrator and the Azure AD administrator. Only the Azure AD administrator is allowed to administer Azure AD users and groups. This admin can also perform all operations a regular server admin can. As administrators, the server administrator accounts are members of the db owner role in every user database and enter each user database as the dbo user. See Figure 5-5.

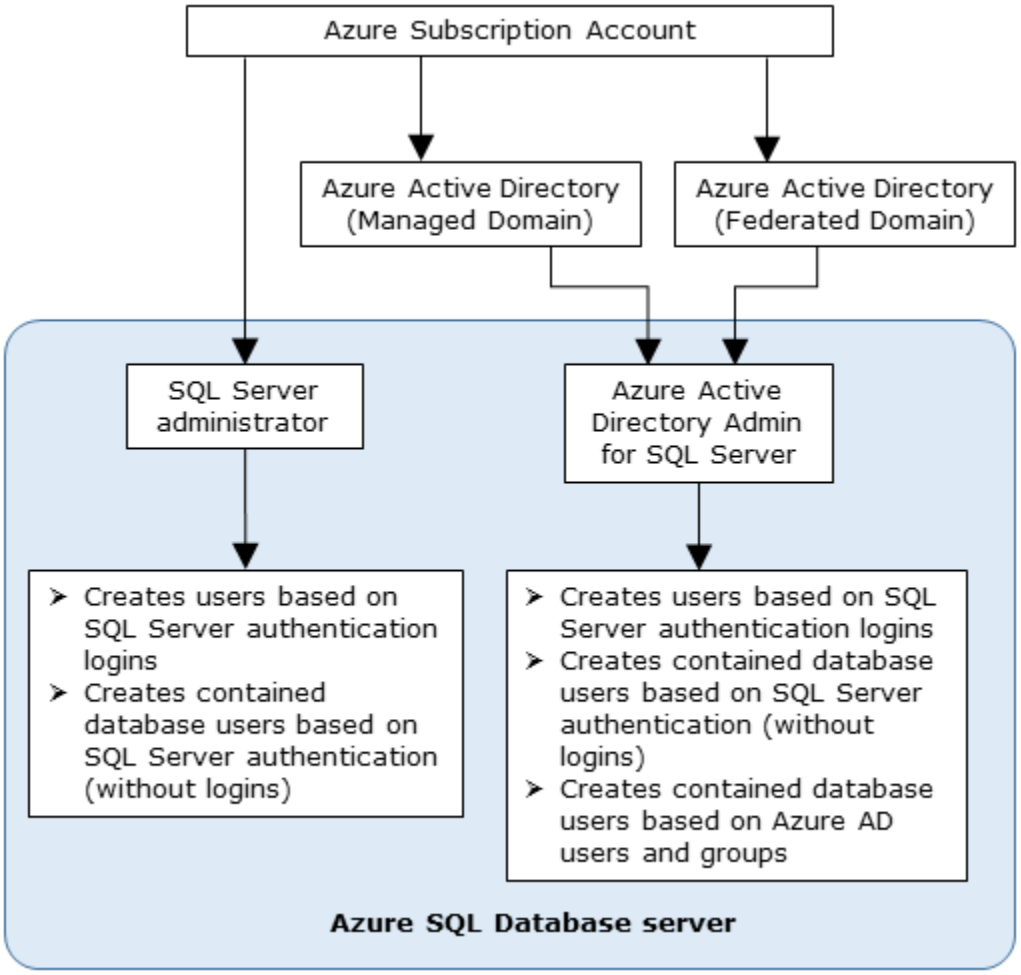


Figure 5-5: Authentication flow diagram

Contained users

SQL Server in an Azure virtual machine (and in an on-premises installation) and Azure SQL Database support contained users that are authenticated at the database level rather than the server level. A contained database is a database that is isolated from other databases and from the server instance (and the master database) that hosts the database. Contained users are supported for both Active Directory and SQL Server logins.

In the traditional connection model, Active Directory users or members of Active Directory groups connect to SQL Server by providing user or group credentials authenticated by Active Directory or by providing a SQL Server name and password when connecting using SQL Server. With this model, the master database must have a login that matches the connecting credentials. After SQL Server confirms the Active Directory authentication credentials or authenticates the SQL Server Authentication credentials, the connection typically attempts to connect to a user database. To connect to a user

database, the login must be able to be mapped to (that is, associated with) a database user in the user database. The connection string may also specify connecting to a particular database. This is optional in SQL Server but required in SQL Database. The connection to the user database has a dependency upon the login in the master database that limits the ability of the database to be moved to a different hosting SQL Server or Azure SQL Database server.

In the contained database user model, the login in the master database is not present. Instead, authentication is at the user database level, and a database user does not have an associated login in the master database. The contained database user model supports both Windows Authentication (in SQL Server) and SQL Server Authentication (in both SQL Server and SQL Database). To connect as a contained database user, the connection string must always contain a parameter for the user database so that SQL Server knows which database is responsible for managing the authentication process. The activity of the contained database user is limited to the authenticating database. With SQL Database, contained users must create a new connection to change the database context. With SQL Server, contained users can change databases if an identical user is present in another database.

Important With Azure SQL Database, there is no guarantee that the logical database and the logical server are located on the same physical hardware. To decrease authentication times, logins at the server level are cached in each database containing a mapped user. For this reason, it is recommended that you use database-level firewall rules.

Although using contained users is optional, you may in the future be required to switch to the contained database user model and database-scoped firewall rules to attain the higher availability service-level agreement (SLA) and higher max login rates for a given database. Microsoft encourages you to consider such changes today.

Note When using the traditional model, the server-level roles and server-level permissions can limit access to all databases. When using the contained database model, database owners and database users with the ALTER ANY USER permission can grant access to the database. This reduces the access control of highly privileged server logins and expands the access control to include highly privileged database users.

User authorization

Authorization refers to what an authenticated user can do, which is based on the user account's role membership and permissions. In the next two sections, I will discuss authorization in an Azure virtual machine and in Azure SQL Database.

User authorization with SQL Server in an Azure virtual machine

User authorization with SQL Server in an Azure virtual machine is identical to authorization with SQL Server on-premises. As such, I will sketch only the outlines to contrast this model with the SQL Database model.

Server-level permissions

SQL Server has nine fixed server roles and user-defined server roles to which users can be added. These roles are used to grant users necessary permissions at the server level.

Database-level permissions

SQL Server has nine fixed database roles and flexible database roles to which users can be added. These are used to grant users necessary permissions at the database level. Any database user with the ALTER ANY USER permission can create additional users in a database. Typically, you will add admin

users to the db_owner role. Furthermore, users can be granted more granular permissions directly on various objects (securables) in a database.

User authorization with Azure SQL Database

User authorization with Azure SQL Database has a number of distinct differences from SQL Server in an Azure virtual machine. Azure SQL Database permissions are managed through the Azure portal, through the master database, and through each user database.

Azure Role-Based Access Control

Azure Role-Based Access Control (RBAC) comes with a number of built-in roles for managing Azure services, including Azure SQL Database. The SQL Database roles include the following:

- **SQL DB Contributor**: Can create and manage SQL databases but not their security-related policies
- **SQL Security Manager**: Can create and manage the security-related policies of SQL servers and databases
- **SQL Server Contributor**: Can manage SQL servers and databases but not their security-related policies

Members of these Azure roles can perform these database tasks using the REST API and PowerShell. For example, an Azure web app can create user accounts and grant permissions to these user accounts.

Master database–level permissions

An Azure SQL Database logical server is an abstraction that defines a grouping of databases. You (using one of the two server-level principal accounts) perform server-level administration for all of them in a single database called master. Within the master database, there are two server-level security roles to which users in the master database can be added.

- **loginmanager role**: Membership in the loginmanager role authorizes a user to create logins. This role is similar to the securityadmin fixed server role with SQL Server.
- **dbmanager role**: Membership in the dbmanager role authorizes a user to create databases. This role is similar to the dbcreator fixed server role with SQL Server.

Use the ALTER ROLE Transact-SQL statement to add a user to either or both of these roles. See the following example:

```
ALTER ROLE dbmanager ADD MEMBER user1
ALTER ROLE loginmanager ADD MEMBER login1user;
```

To view membership in these roles, use the following Transact-SQL statement:

```
SELECT r.name role_principal_name, m.name AS member_principal_name
FROM sys.database_role_members rm
JOIN sys.database_principals r
    ON rm.role_principal_id = r.principal_id
JOIN sys.database_principals m
    ON rm.member_principal_id = m.principal_id
WHERE r.name IN ('loginmanager', 'dbmanager');
```

Database-level permissions

The database-level permission model in Azure SQL Database is same as an on-premises instance of SQL Server. Users can be added to database roles or be granted more granular permissions on securables in a database.

Any database user with the ALTER ANY USER permission can create additional users in a database. However, the first contained user must be created by a server-level principal.

Database backups and restores

Next, I will discuss protecting your data from disaster, hardware failures, user errors, and database corruption and fulfilling archive requirements. In this space, there are substantial differences between SQL Server in an Azure virtual machine and Azure SQL Database. In the next two sections, I will discuss these differences as they relate to disaster recovery, backup automation, backup performance, restore effort, and restore performance.

Database backups with SQL Server in an Azure virtual machine

For the most part, database backups with SQL Server in an Azure virtual machine are the same as SQL Server backups in an on-premises environment. You develop a backup strategy and manage the scheduling and storage of backups yourself as the DBA. You can store the backup in an attached drive or to Azure blob storage using SQL Server backup to URL. You can also use managed backup to have Microsoft manage your backup strategy. If you store your database files in Azure, you can use file-snapshot backups for nearly instantaneous backups and very fast restores. Although all of these options are available in an on-premises environment, they are more useful in the Azure virtual machine context. The next three sections discuss each of these options in the context of an Azure virtual machine.

SQL Server backup to URL

With SQL Server backup to URL, you specify an Azure blob storage container as the destination for your full, differential, and transaction log backups, and you manage the backup process and the retention strategy. A primary reason to use SQL Server backup to URL is to avoid the cost of backing up to attached disks that use premium storage (although you always can back up to attached premium storage for performance and then archive to blob storage). See Figure 5-6.

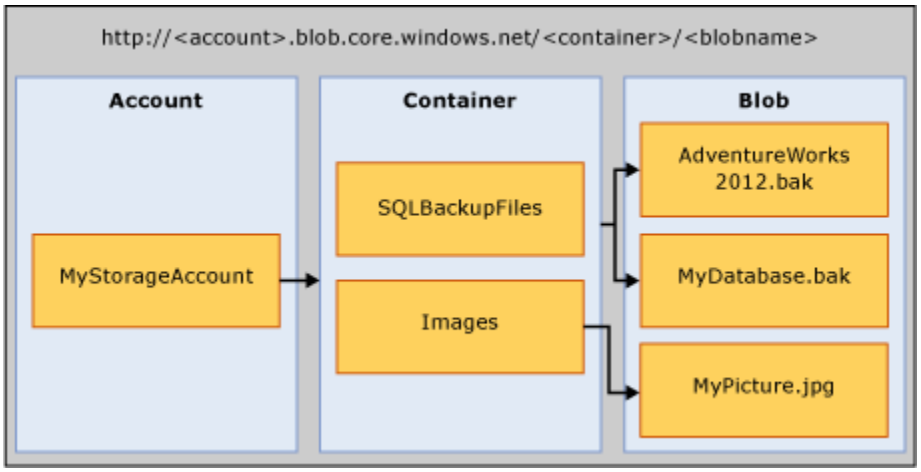


Figure 5-6: Backup to URL architecture

To use this backup mechanism, take the following steps:

- Create an Azure storage account.
- Create a container within the storage account to contain your backup files.
- Generate a Shared Access policy and apply the Shared Access Signature (SAS) token to the Azure blob container.

> **Note** You can create the SAS token on the blob directly or create a SAS policy and apply it to multiple containers. Creating a SAS policy is a best practice because it can be revoked without regenerating the storage account keys.

- Create a SQL Server credential based on the SAS token.

```
CREATE CREDENTIAL <mycredentialname> WITH IDENTITY = 'SHARED ACCESS SIGNATURE', SECRET = '<SAS_TOKEN>' ;
```

- Specify TO URL as an argument with your BACKUP DATABASE and BACKUP LOG commands.

```
BACKUP DATABASE AdventureWorks2016 TO URL =
'https://<mystorageaccountname>.blob.core.windows.net/<mycontainername>/AdventureWorks2016.bak;
```

Backup to URL is a strategy you will want to use to save money when your Azure virtual machine is using premium storage because SQL Server backup to URL uses standard storage. The maximum size for a single file using a SAS token is 200 GB, and the maximum backup size using a striped backup set is 12.8 terabytes (TB).

> **Note** You can back up to a single backup with a maximum size of 1 TB using the legacy SQL Server 2014 syntax of WITH CREDENTIAL that utilizes the storage account access key rather than a SAS key. This syntax is being deprecated. For an example of this syntax, see To URL using storage account identity and access key.

SQL Server Managed Backup to Windows Azure

You can use SQL Server Managed Backup to Windows Azure and have SQL Server manage and automate SQL Server backups to Azure blob storage. With this option, you specify the blob storage location and the retention period and optionally customize backup at the database level. The maximum backup size is 12.8 TB. Managed backup supports point-in-time restore during the retention period. By default, managed backups are enabled at the instance level, with the same retention period for all databases (including newly added databases). You can change the retention period on an individual database, create a custom backup schedule, or disable managed backup for an individual database using Transact-SQL. You also have the option to encrypt the database backups for additional security.

Managed backup was introduced with SQL Server 2014, and the functionality increased substantially with SQL Server 2016. For the purposes of this ebook, I will focus on SQL Server 2016. To understand and use managed backup with SQL Server 2014, see SQL Server 2014 Managed Backup to Windows Azure.

> **Important** With SQL Server Managed Backup to Windows Azure, the maximum retention period is 30 days. To retain a backup for longer than 30 days, you must either perform a separate full database backup or copy an existing database backup from the Azure blob storage container to another storage location.

With SQL Server 2016, if you do not specify a custom schedule, the type of backups scheduled and the backup frequency is determined based on the workload of the database.

- A full database for a database is taken in the following instances:

    - When managed backup is enabled for the first time
    - Log growth since the last full backup is 1 GB or more
    - One week has passed since the last full backup
    - The log chain is broken, such as through a manual backup
- A transaction log backup is taken in the following instances:
    - No log backup history can be found. This usually is true when SQL Server Managed Backup to Windows Azure is enabled for the first time.
    - The transaction log space used is 5 MB or larger.
    - The maximum time interval of two hours since the last log backup is reached.
    - Anytime the transaction log backup is lagging behind a full database backup. The goal is to keep the log chain ahead of full backup.

To use SQL Server managed backup, you create the following prerequisites:

- Create an Azure storage account.
- Create a container within the storage account.
- Generate a Shared Access policy and apply the Shared Access Signature (SAS) token to the Azure blob container.

> **Note** You can create the SAS token on the blob directly or create a SAS policy and apply it to multiple containers. Creating a SAS policy is a best practice because it can be revoked without regenerating the storage account keys.

- Create a SQL Server credential based on the SAS token.

```
CREATE CREDENTIAL <mycredentialname> WITH IDENTITY = 'SHARED ACCESS SIGNATURE', SECRET = '<SAS_TOKEN>' ;
```

After satisfying these prerequisites, you use Transact-SQL to interact with SQL Server Managed Backup to Windows Azure. There are 13 system stored procedures and functions for enabling, configuring, and monitoring SQL Server Managed Backup to Windows Azure. System functions are used to retrieve existing configuration settings, parameter values, and backup file information. Each of these system stored procedures and functions provides code examples for you.

Extended events are used to surface errors and warnings. Alert mechanisms are enabled through SQL Agent jobs and SQL Server Policy-Based Management.

PowerShell cmdlets are also available to configure SQL Server Managed Backup to Windows Azure. SQL Server Management Studio supports restoring backups created by SQL Server Managed Backup to Windows Azure by using the Restore Database task.

File-snapshot backups

File-snapshot backup in SQL Server 2016 uses Azure snapshots to provide nearly instantaneous backups and very fast restores for database files stored using the Azure blob storage service. This new backup method is not available for database files stored within the Azure virtual machine itself.

Understanding the capabilities of file-snapshot backup will inform your decision with respect to the type of storage you will use for SQL Server in an Azure virtual machine. For all database sizes, as discussed earlier in this ebook, you can choose between premium and standard storage for virtual machines, and you can choose Azure blob storage. For best performance, you will use premium

storage with your SQL Server virtual machine rather than storing your files in Azure blob storage and therefore will not have the ability to take advantage of the benefits provided by file-snapshot backups. However, for very large databases, storing some or all of the data files in Azure blob storage is the only choice due to size limitations on using attached storage with Azure virtual machines. For many users, the performance benefits of using premium storage will outweigh the benefits of file-snapshot backups. To make your own decision, read on to understand the benefits of file-snapshot backups.

Using the file-snapshot capability enables you to simplify your backup and restore policies. A file-snapshot backup consists of a set of Azure snapshots of the blobs containing the database files (of all data and log files) plus a backup file containing pointers to these file-snapshots. The backup is almost instantaneous because it is just a metadata operation to establish the snapshot point. The restore is much faster because you need to read less data to perform the restore, as the following paragraphs explain in more detail.

Each file-snapshot is stored in the container with the base blob. See Figure 5-7.

Caution Deleting the base blob will also delete all file-snapshot backups. The Azure portal protects you from deleting base blobs containing snapshots, but you can override these protections by using PowerShell or the REST API.

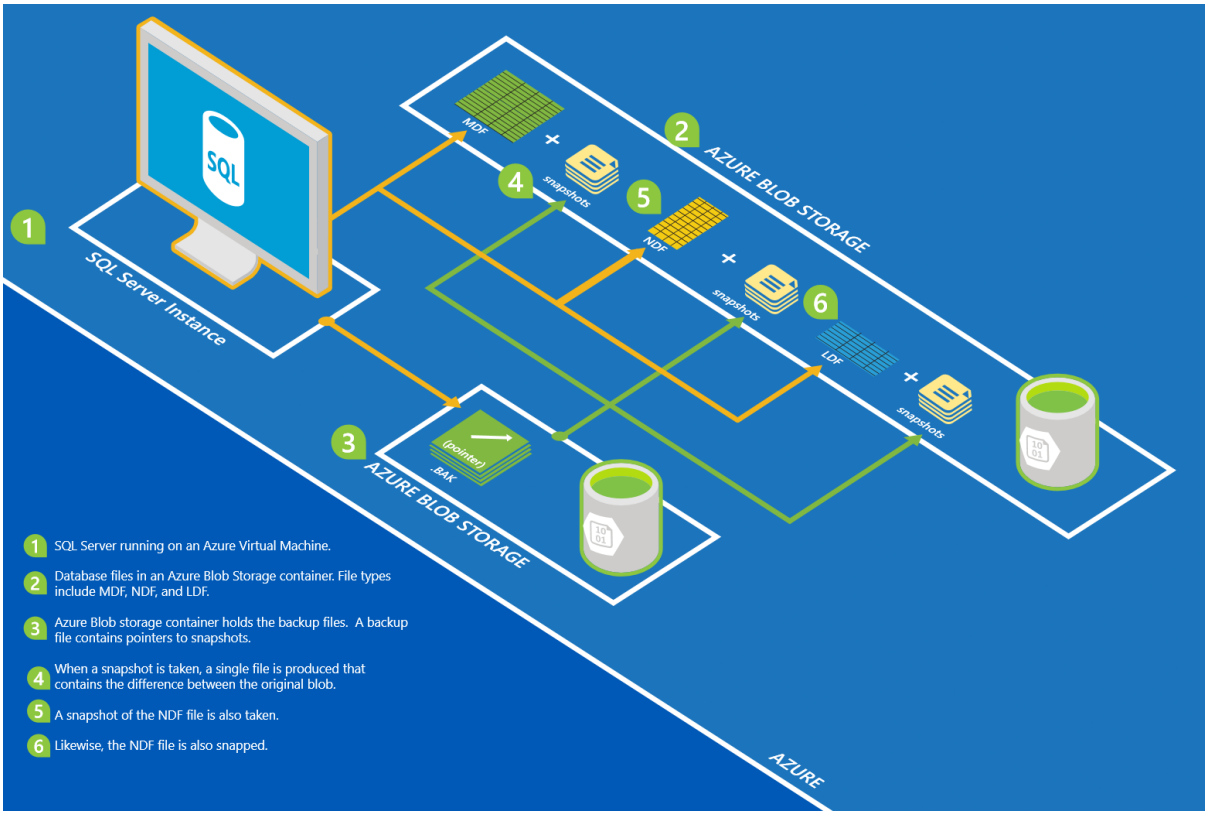


Figure 5-7: File-snapshot backup architecture

You can specify that the backup file be written to URL, disk, or tape. Backup to URL is recommended. The backup file must be written to standard storage.

- **Full database backup**: Performing a full database backup using file-snapshot backup creates an Azure snapshot of each data and log file comprising the database, establishes the transaction log backup chain, and writes the location of the file-snapshots into the backup file.
- **Transaction log backup**: Performing a transaction log backup using file-snapshot backup creates a file-snapshot of each database file (not just the transaction log), records the file-snapshot location information into the backup file, and truncates the transaction log file.

After the initial full backup that is required to establish the transaction log backup chain (which can be a file-snapshot backup), you only need to perform transaction log backups because each transaction log file-snapshot backup set contains file-snapshots of all database files and can be used to perform a database restore or a log restore. After the initial full database backup, you do not need additional full or differential backups because the Azure blob storage service handles the differences between each file-snapshot and the current state of the base blob for each database file. See Figure 5-8.

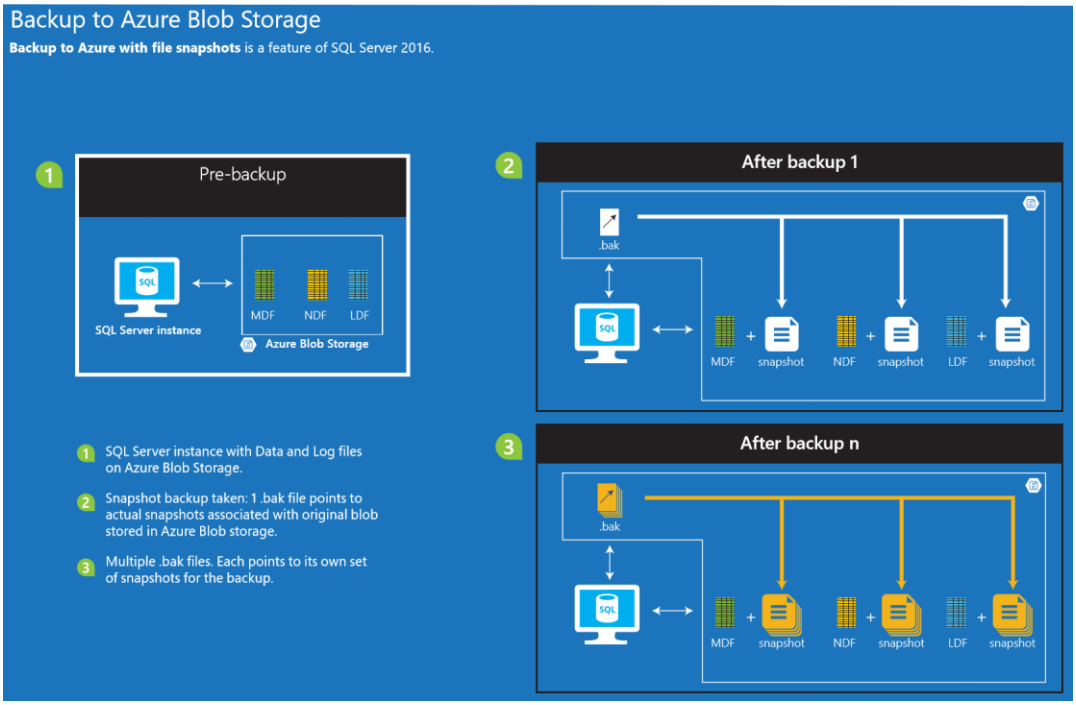


Figure 5-8: File-snapshot backup artifacts

Because each file-snapshot backup set contains a file-snapshot of each database file, a restore process requires at most two adjacent file-snapshot backup sets. This is true regardless of whether the backup set is from a full database backup or a log backup. This is different from the restore process when using traditional (streaming) backup files. With traditional streaming backup, the restore process requires the use of an entire chain of backup sets: the full backup, a differential backup, and one or more transaction log backups. The recovery portion of the restore process remains the same regardless of whether the restore is using a file-snapshot backup or a streaming backup set.

The syntax for performing a file-snapshot backup is almost identical to a traditional backup. See the following sample syntax:

```
BACKUP DATABASE AdventureWorks2016
TO URL = 'https://<mystorageaccountname>.blob.core.windows.net/<mycontainername>/AdventureWorks2016.bak'
WITH FILE_SNAPSHOT;
```

The syntax for performing database restores is identical to restoring from a traditional set of backup files. The only difference is that the file used for a full database restore is the set of snapshots from the transaction log backup file (or files if performing a point-in-time restore) containing your restore point.

You manage file-snapshot backups by using system stored procedures and a system function. For a tutorial on using SQL Server 2016 with the Microsoft Azure blob storage service, see Tutorial: Using the Microsoft Azure Blob storage service with SQL Server 2016 databases.

Database backups with Azure SQL Database

Database backups with Azure SQL Database are performed for you automatically, and you have no option to perform manual database backups. To protect your data and enable point-in-time restore services, SQL Database takes full backups every week, multiple differential backups every day, and log backups every five minutes. Backup files are stored in geo-redundant storage with read access (RA-GRS) to ensure backups' availability for disaster recovery purposes. The first full backup is scheduled immediately after a database is created. After the first full backup, all backups are scheduled automatically and managed silently in the background. The exact timing of full and differential backups is determined by the system to balance overall load.

SQL Database provides a point-in-time restore self-service feature for all databases regardless of service tier, but with different restore points as follows:

- **Basic**: Any restore point within 7 days
- **Standard**: Any restore point within 14 days
- **Premium**: Any restore point within 35 days

Point-in-time restore (geo-restore) is built on top of the SQL Database automated backup system and enables you to restore an existing or deleted database to a new database as of a specified point in time. You can restore using the Azure portal, PowerShell, or the REST API. The walk-through below will demonstrate restoring a database to a point in time using the Azure portal. See Figures 5-34 through 5-40.

When restoring an existing database, you can restore to the same logical server or to a different server. When restoring to the same server, you create a new database. Unlike SQL Server, you cannot restore over an existing database. As a result, you will have additional storage costs while both databases exist—unless you choose to delete the existing database before restoring from the backups.

Caution Deleting a logical server also deletes all of its databases and their database backups, after which they cannot be restored regardless of the retention period discussed previously for the automated backups.

Note To store data for longer than the period of time provided by your service tier, you will need to export data to a BACPAC file. This option is discussed in the next section of this chapter.

Data archiving

The final topic I will discuss in this chapter is data archiving. Data archiving is a scenario that primarily requires discussion in the context of Azure SQL Database. However, there are a couple of points to discuss in the context of SQL Server in an Azure virtual machine.

Data archiving with SQL Server in an Azure virtual machine

The data archive mechanism with SQL Server in an Azure virtual machine generally will be a full SQL Server backup that you store for the length of time required. If you are managing your own backups, you will want to store your full database backups in Azure blob storage standard storage (or archive outside Azure). If you are not already using backup to URL to store your backups in Azure blob storage, you will want to copy from your attached disks in premium storage to Azure blob storage using standard storage.

If you are using managed backup and want to archive a full database backup for longer than 30 days, you will need to copy a full backup to a new storage location or periodically perform a full database backup (to URL) on a scheduled basis and store it in standard storage for the length of time required.

If you are using file-snapshot backup, you have another option to archive your database: you periodically can copy the backup snapshots for a given backup set to long-term storage. At any point in time, you can choose to restore to a new database when needed and create a traditional (streaming) SQL Server backup for long-term storage.

Data archiving with Azure SQL Database

The data archive mechanism with Azure SQL Database generally will be an export from Azure SQL Database. Export creates a non–transactionally consistent copy of some or all database tables using the BACPAC format (schema and data). This BACPAC file can be stored in Azure blog storage and/or downloaded to an on-premises environment and imported when desired to either Azure SQL Database or SQL Server.

To obtain a transactionally consistent export (without using third-party tools), you must either ensure that no writes occur on the source database during the export or create a database copy (which is guaranteed to be transactionally consistent).

Microsoft provides a number of options for you to use to export your data to a BACPAC file. These are as follows:

- Using the Azure portal
- Using the Export Data Tier Application Wizard in Microsoft SQL Server Management Studio, version 13.0.11000 or higher (Download SQL Server Management Studio)
- Using the SQLPackage.exe command-line utility
- Using the Start-AzureSqlDatabaseExport PowerShell cmdlet, which can be combined with the Start-AzureSqlDatabaseCopy PowerShell cmdlet

Generally, for databases smaller than 200 GB, using the Azure portal is the simplest method. If the operation goes over 20 hours, it may be cancelled. If the export does not have a clustered index, it may fail if it takes longer than 6 to 12 hours.

Walk-through: Configuring authentication and authorization with Azure SQL Database

In this walk-through, we will create logins and users and contained users. We will also grant permissions to a new user to perform a variety of management tasks and then verify that the user can perform those tasks. For this walk-through, we will use SQL logins.

Caution In this walk-through, you will receive errors intentionally, with text explaining why those errors occurred.

Creating a login and a user

Let's start by creating a login in the master database and a user in a user database and validate authentication and our ability to connect to our databases using the login and the user account.

1. Open SQL Server Management Studio and connect to your SQL Database instance in Object Explorer using your server admin login.

2. On the toolbar, click New Query. A new query window opens, and you automatically are connected to the master database using your server admin login. See Figure 5-9.

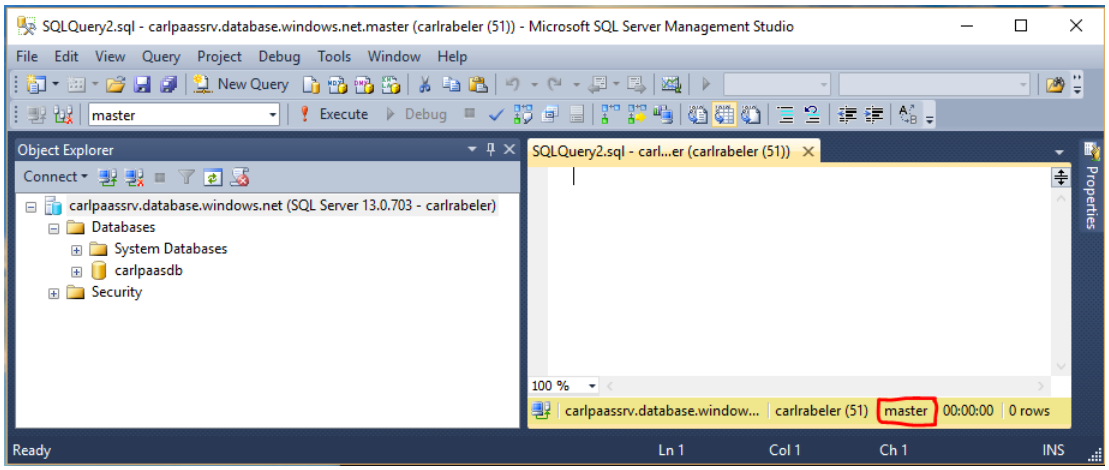


Figure 5-9: Query window with master database connection

3. Use the following Transact-SQL statement to create a login in the master database.

```
CREATE LOGIN login1 WITH PASSWORD ='p@ssw0rd1';
```

4. Review the results of this command. Notice that the new login appears in the Logins folder within the Security folder for the instance. See Figure 5-10.

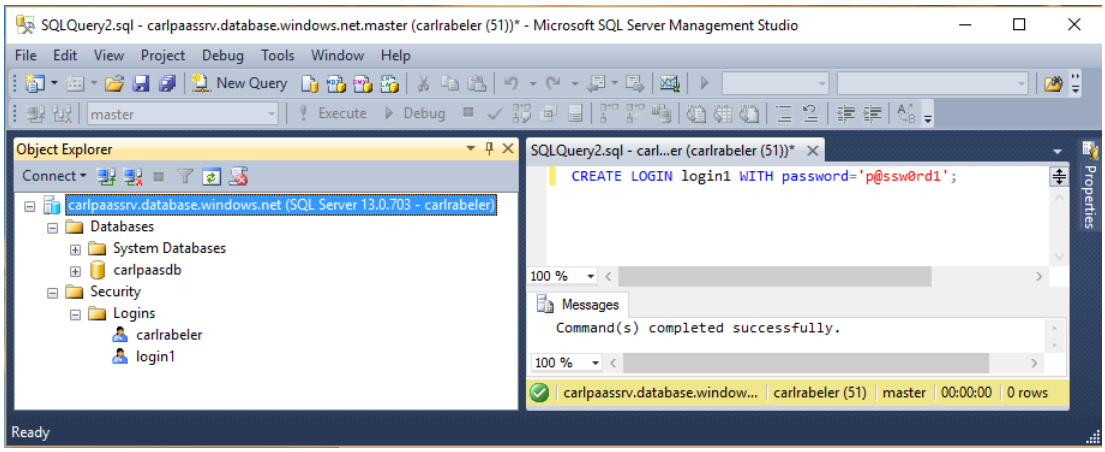


Figure 5-10: New login

5. In the query window, right-click, point to Connection, and click Change Connection. See Figure 5-11.

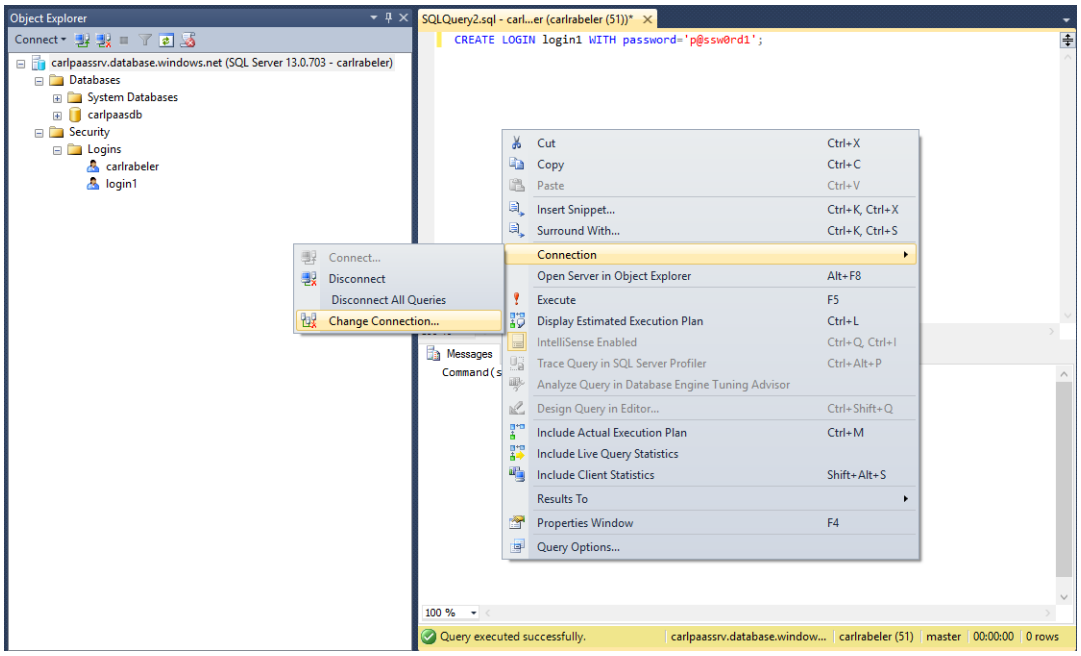


Figure 5-11: Change connection

6. In the Connect to Database Engine window, change the login to login1, enter the password, and click Connect. Notice that you cannot connect to the master database. See Figure 5-12.

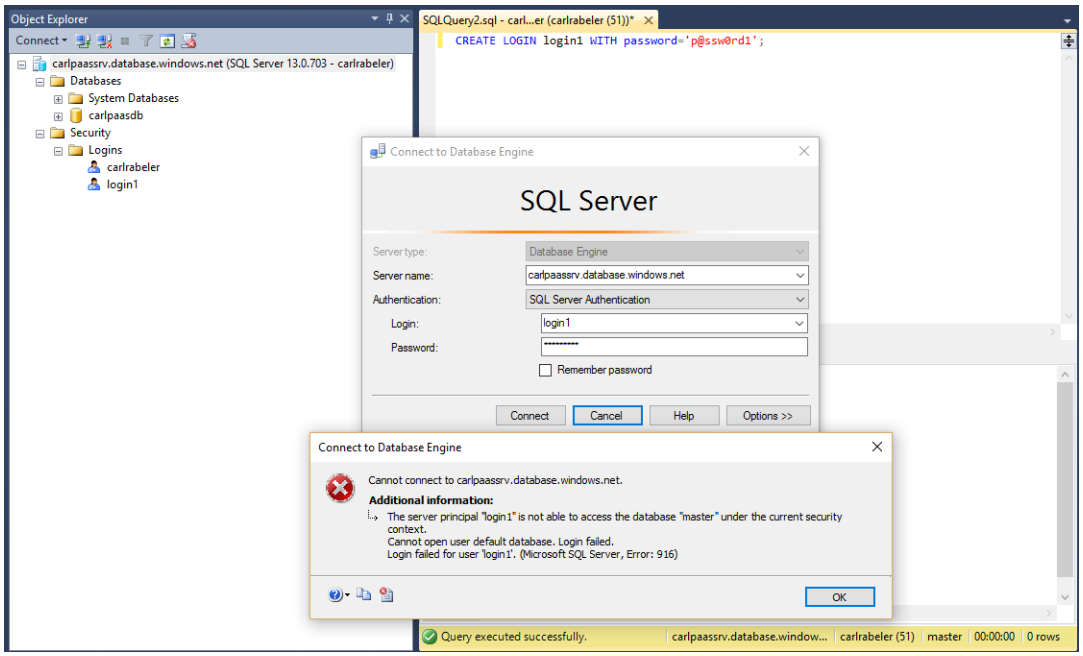


Figure 5-12: Failure to connect message

7. Close the error message and then close the Connect to Database Engine window.

8. Use the following Transact-SQL statement to create a user in the master database to enable login1 to connect to the master database.

```
CREATE USER login1User FROM LOGIN login1;
```

9. Review the results of this command. Notice that the new user appears in the Users folder within the Security folder for the master database. See Figure 5-13.

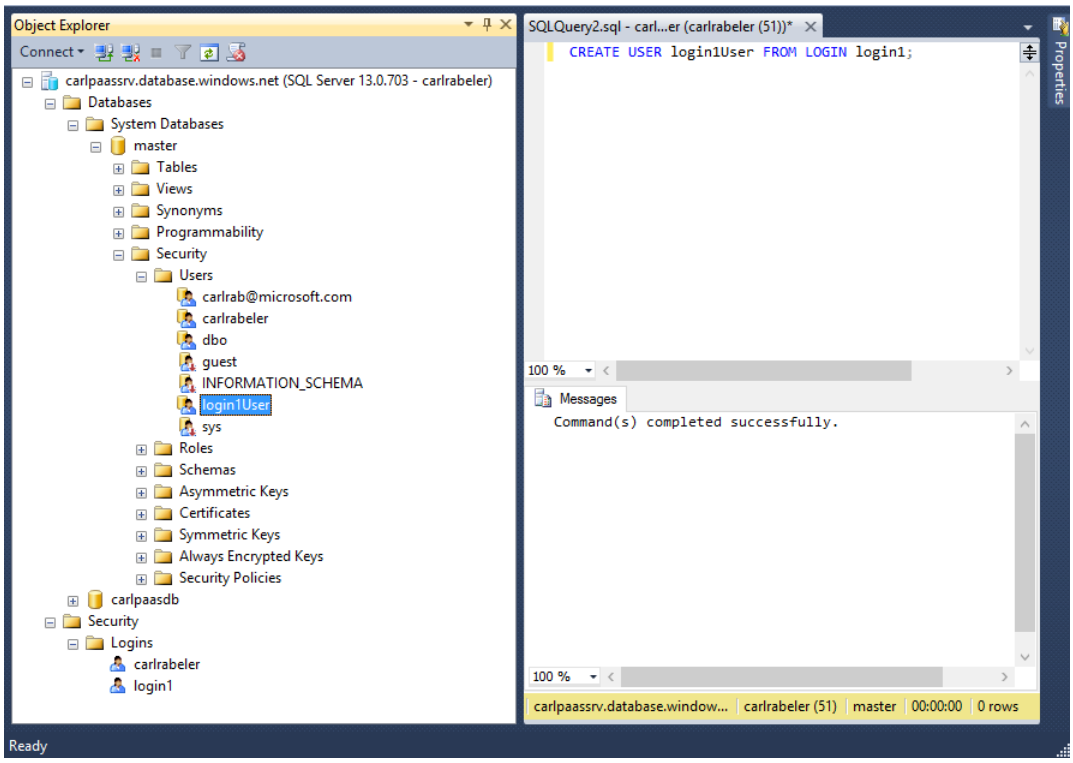


Figure 5-13: New user linked to an existing login created

10. In the query window, right-click, point to Connection, and click Change Connection.

11. In the Connect to Database Engine window, change the login to login1, enter the password, and click Connect. Notice that you can connect to the master database. See Figure 5-14.

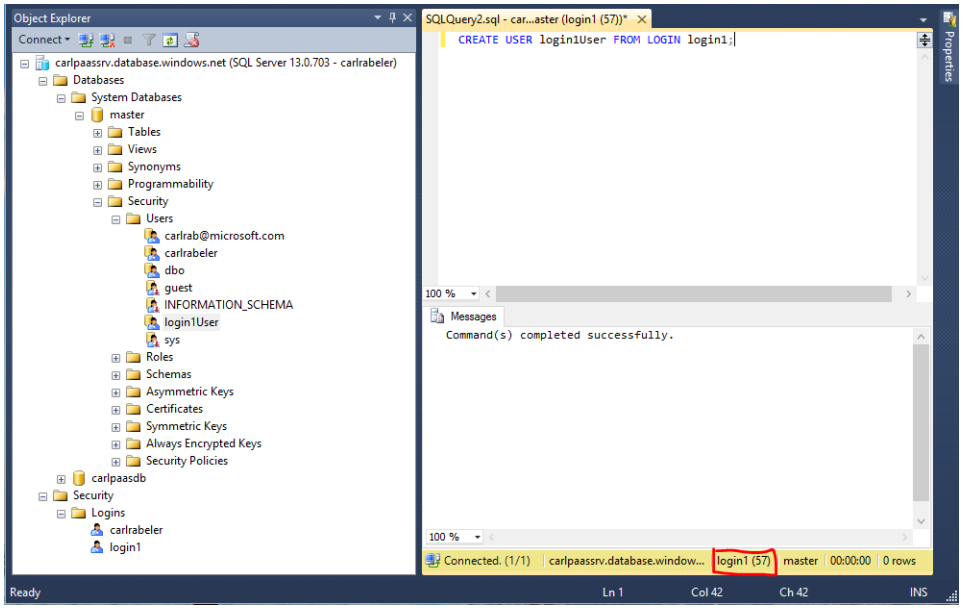


Figure 5-14: Connection using login1

12. In the query window, right-click, point to Connection, and click Change Connection.

13. In the Connect to Database Engine window, change the login to login1User, enter the password, and click Connect. Notice that you cannot connect to the master database using the user account, but you can connect using the login because login1User is not a contained user. See Figure 5-15.

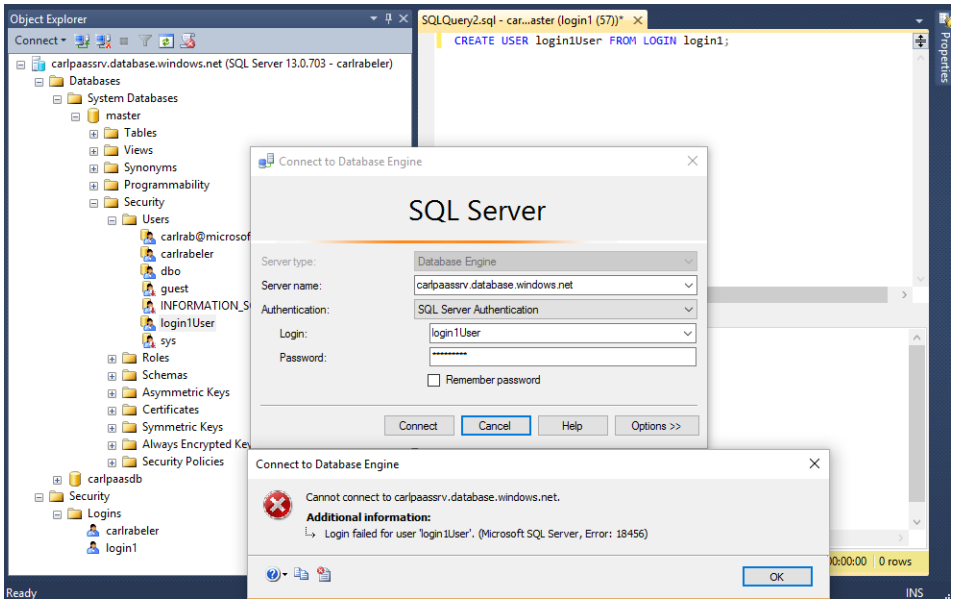


Figure 5-15: Cannot connect with user account if it is mapped to a login

14. Close the error message.

15. In the Connect to Database Engine window, change the login to login1, enter the password, and click Options. See Figure 5-16.

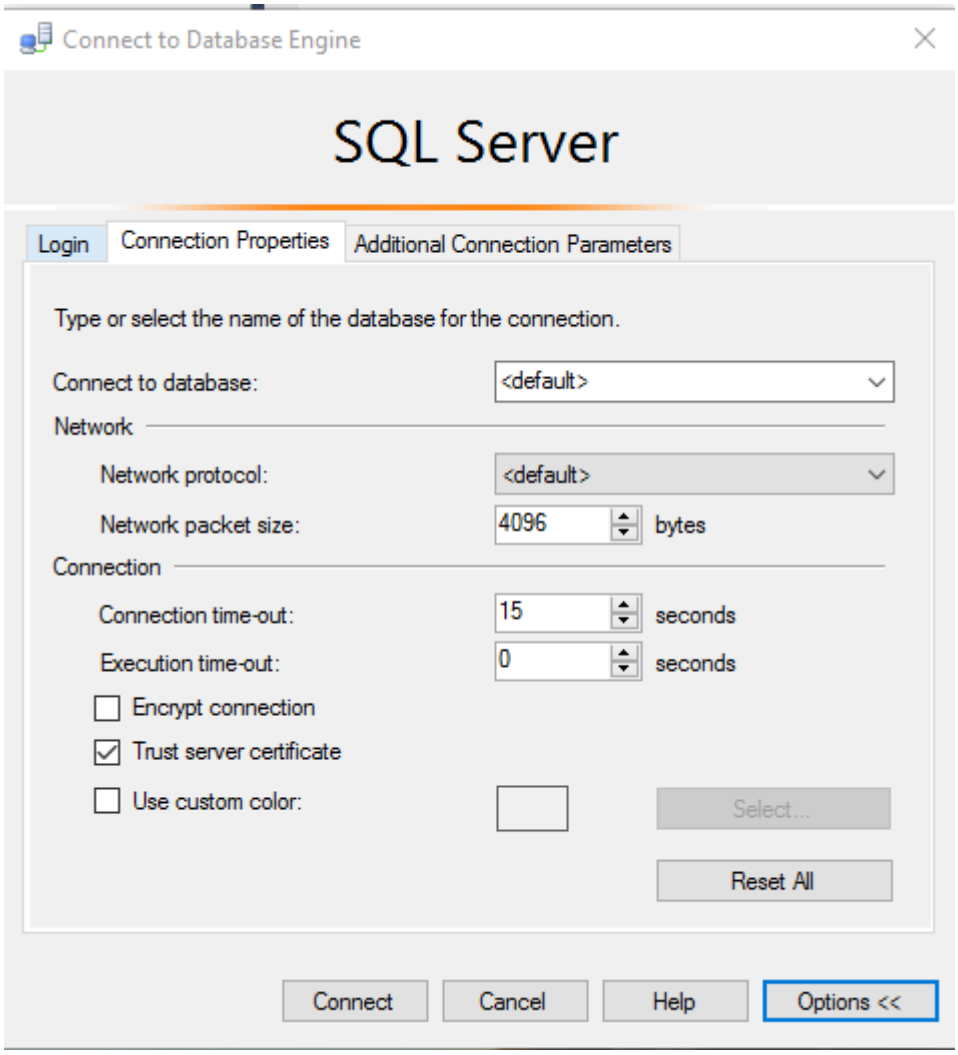


Figure 5-16: Connection properties

16. In the Connect to Database text box, change the database context to your user database (in my case, carlpaasdb) and click Connect. Notice that login1 is not able to access the user database. See Figure 5-17.

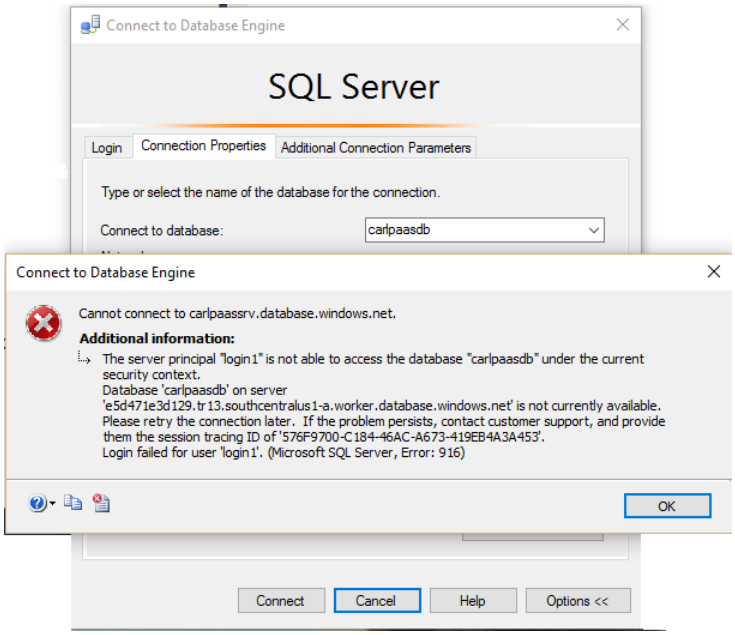


Figure 5-17: Change database error

17. Close the error window.

18. In the Connect to Database Engine window, click the Login tab (do not change the Connect to Database value).

19. In the Connect to Database Engine window, change the login to your server admin login, enter the password, and click Connect. Notice that you are now connected to your user database using your server admin login. See Figure 5-18.

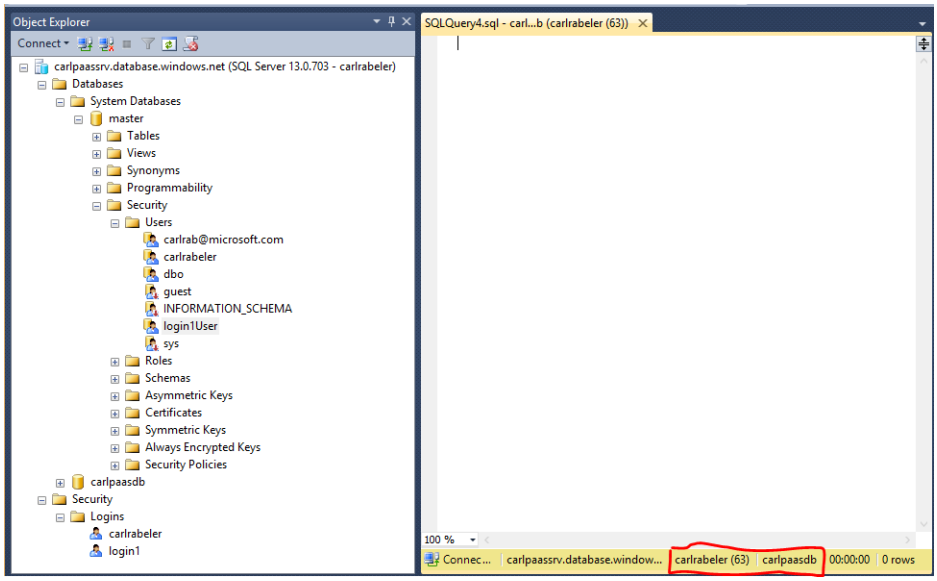


Figure 5-18: Server admin connection to user database

20. Use the following Transact-SQL statement to create a user in your user database to enable login1 to connect to this database.

```
CREATE USER login1User FROM LOGIN login1;
```

21. Review the results of this command. Notice that the new user appears in the Users folder within the Security folder for your user database. See Figure 5-19.

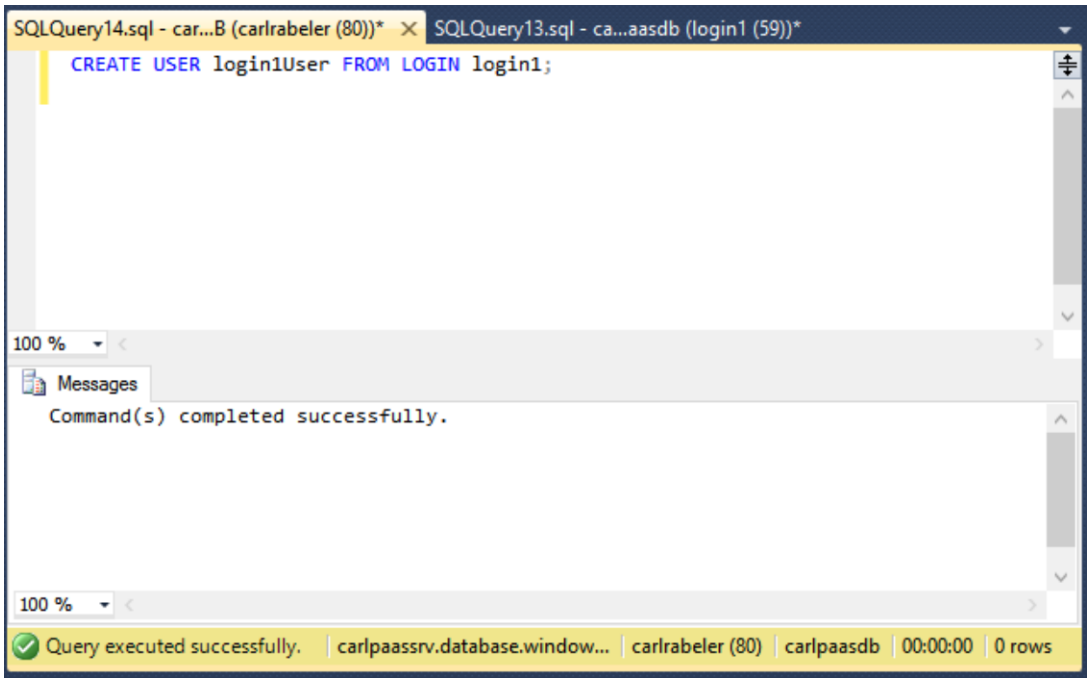


Figure 5-19: Creating user in user database

22. In the query window, right-click, point to Connection, and click Change Connection.

23. In the Connect to Database Engine window, change the login to login1, enter the password, and click Options.

24. In the Connect to Database text box, change the database context to your user database (in my case, carlpaasdb) and click Connect. Notice that you can now connect to your user database using the user account. See Figure 5-20.

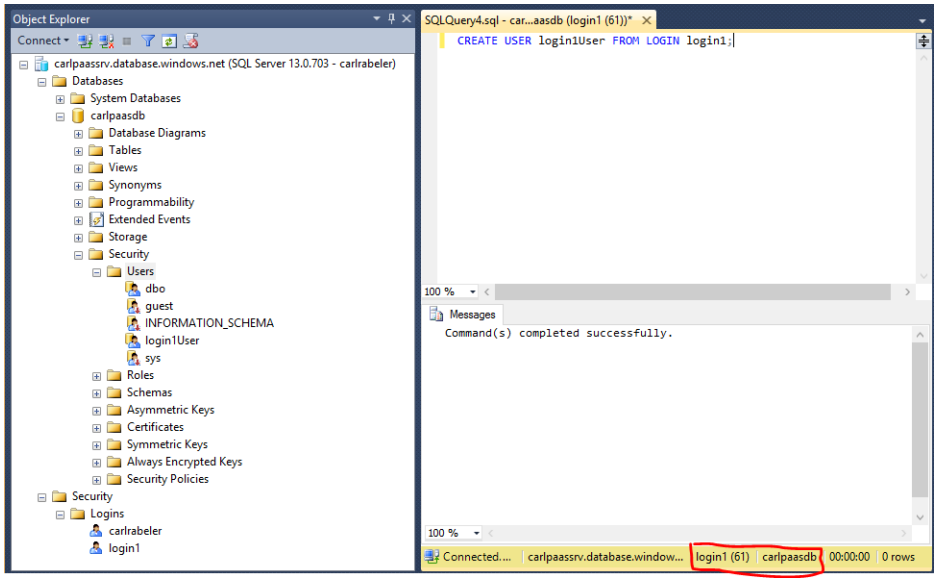


Figure 5-20: User login to user database

Creating a contained user

Let's create a contained user in a user database and validate authentication and our ability to connect to the user database using the user account.

1. In SQL Server Management Studio, connect to your user database using your server admin account.

2. Use the following Transact-SQL statement to create a contained user in your user database.

```
CREATE USER user1 WITH PASSWORD ='p@ssw0rd1';
```

3. Review the results of this command. Notice that the new user appears in the Users folder within the Security folder for your user database. See Figure 5-21.

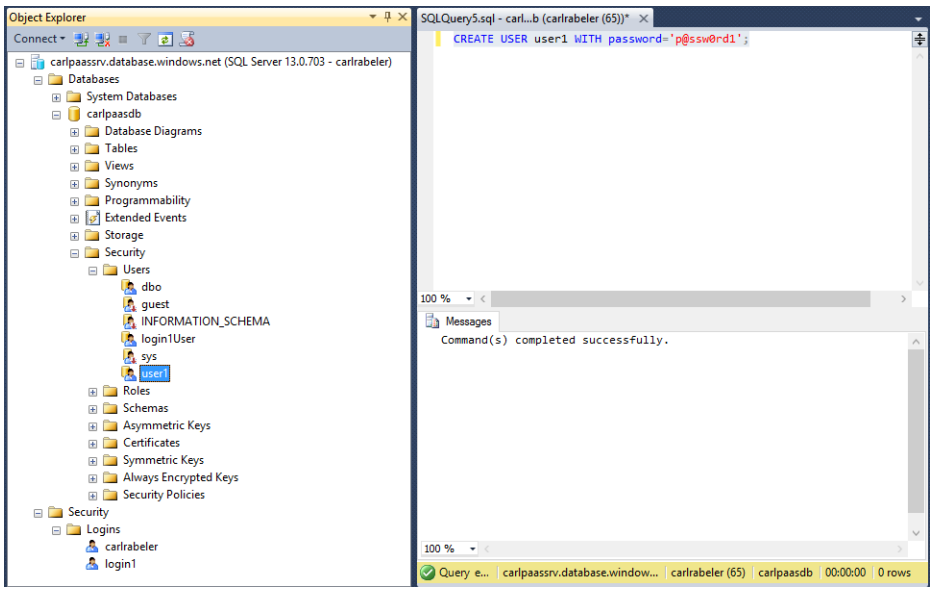


Figure 5-21: Contained user account

4. In the query window, right-click, point to Connection, and click Change Connection.

5. In the Connect to Database Engine window, change the login to user1, enter the password, and click Connect. Notice that you can connect to your user database. See Figure 5-22.

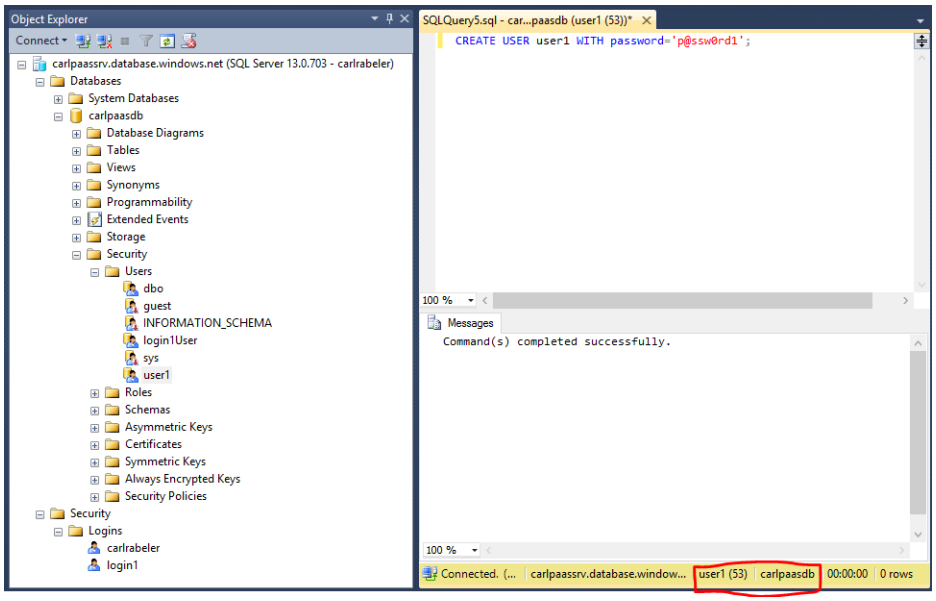


Figure 5-22: Contained user connecting to user database

Granting server-level permissions to users

Next, let's grant server-level permissions to a user. The first requirement is that the user must have a user account in the master database, either a login with a user account or a contained user account.

1. In SQL Server Management Studio, connect to the master database using the login1 account.

   Use the following Transact-SQL statement to attempt to create a login.

   ```
   CREATE LOGIN login2 WITH PASSWORD = 'p@ssw0rd1';
   ```

   Notice that the login1 user account does not have permission to perform this action. See Figure 5-23.

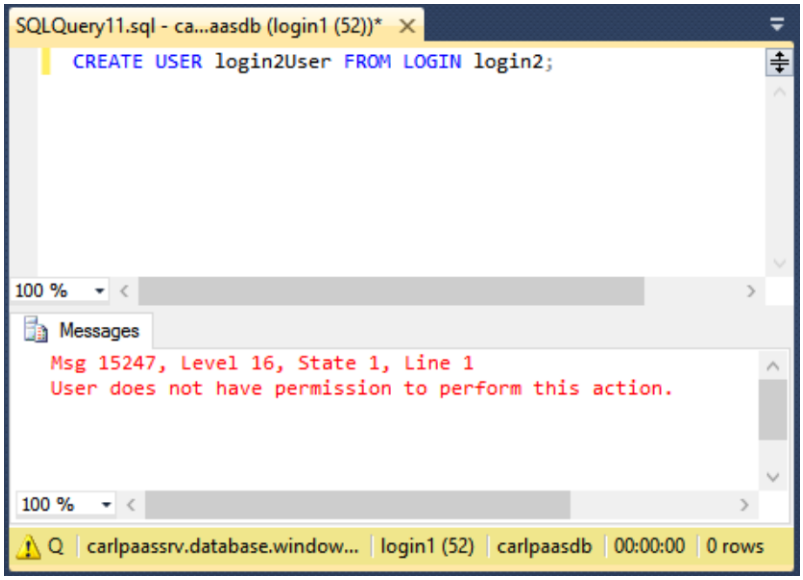


Figure 5-23: No permissions to create a new login

2. In Object Explorer, right-click the master database and click New Query to open a new query window with a connection to master by your server admin account.

3. In the query window, execute the following Transact-SQL statements to grant the login1User account permissions to create logins and create databases. See Figure 5-24.

```
ALTER ROLE dbmanager ADD MEMBER login1user;

ALTER ROLE loginmanager ADD MEMBER login1user;
```

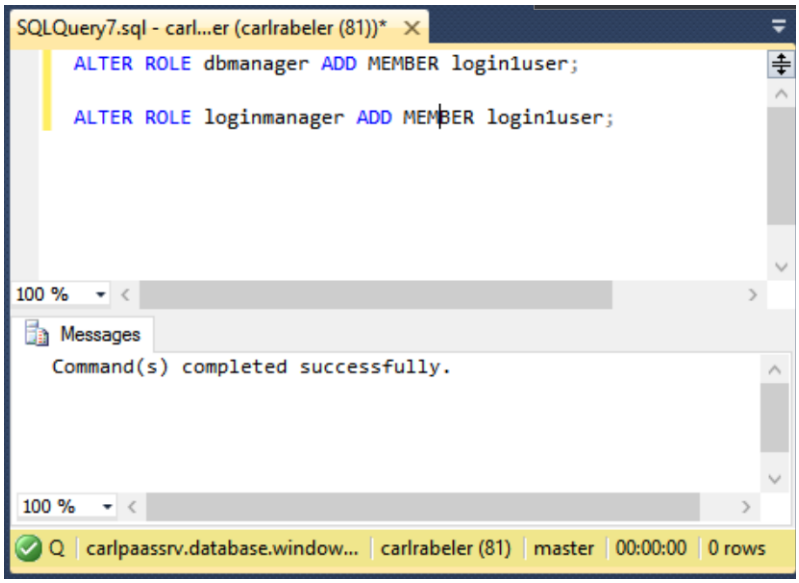


Figure 5-24: Granting server-level permissions to a login

4. Switch to the previous query window and re-execute the CREATE LOGIN Transact-SQL statement. Notice that this user can now create a login. See Figure 5-25.

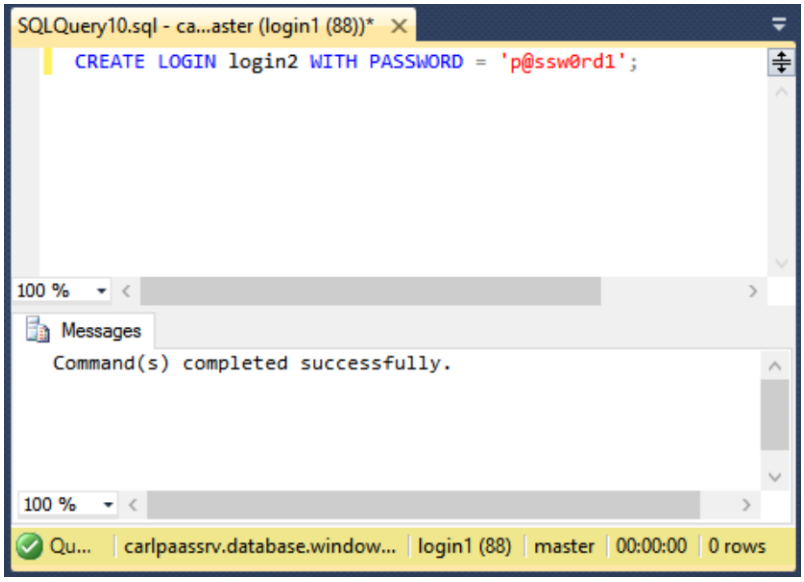


Figure 5-25: Member of loginmanager role creating a login

5. Next, change your connection to your user database and attempt to execute the following Transact-SQL statement to create a user account in your user database for this login using the login1 account.

```
CREATE USER login2User FROM LOGIN login2;
```

Notice that while membership in the loginmanager role grants permission to create logins, it does not grant any permissions to create users (in any database, including master). See Figure 5-26.

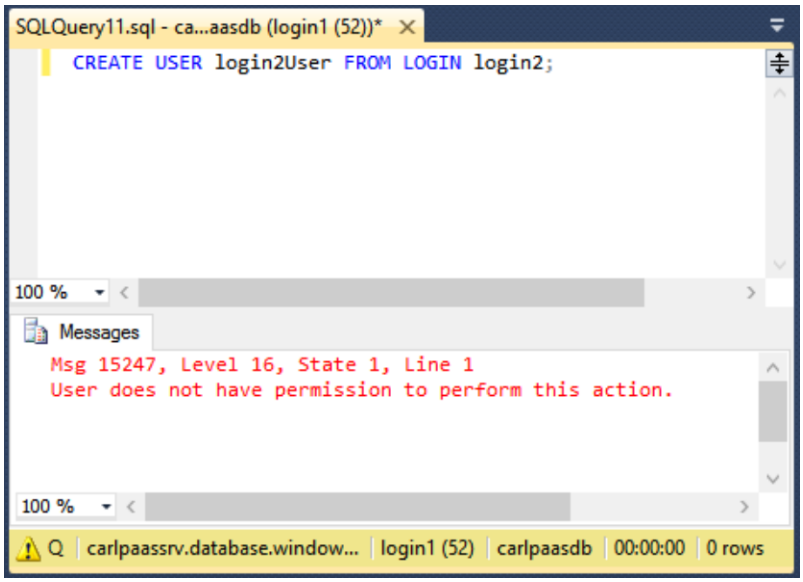


Figure 5-26: Member of loginmanager role cannot create a user

6. Switch back to the query window in which you are connected using your server admin account.
7. On the toolbar, change your connection to your user database.
8. Execute the following Transact-SQL statement to add the login1User account to db_owner role in the user database. See Figure 5-27.

```
ALTER ROLE db_owner ADD MEMBER login1user;
```

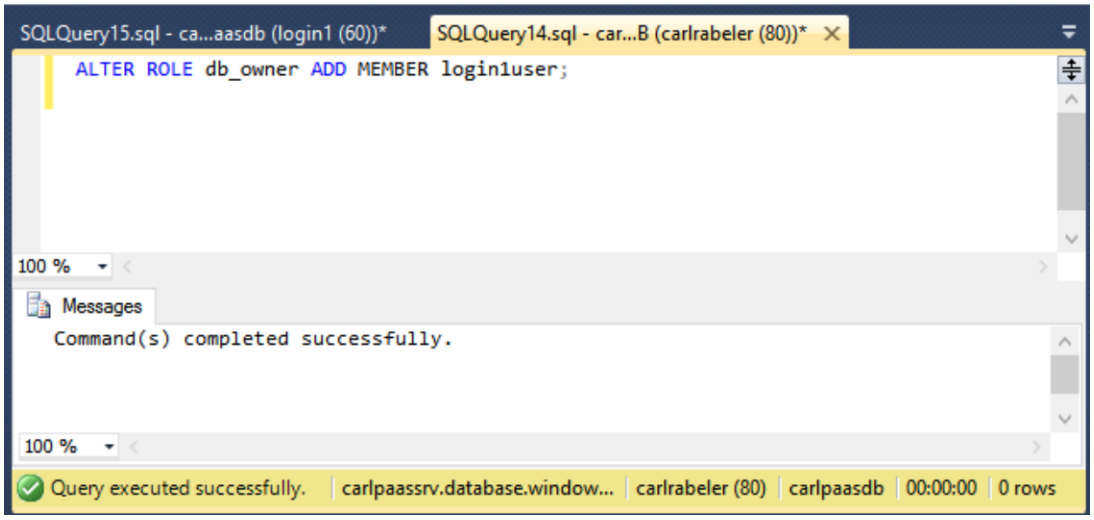


Figure 5-27: Add user to db_owner role

9. Switch to the previous query window and re-execute the CREATE USER Transact-SQL statement. Notice that this user can now create a new user. See Figure 5-28.

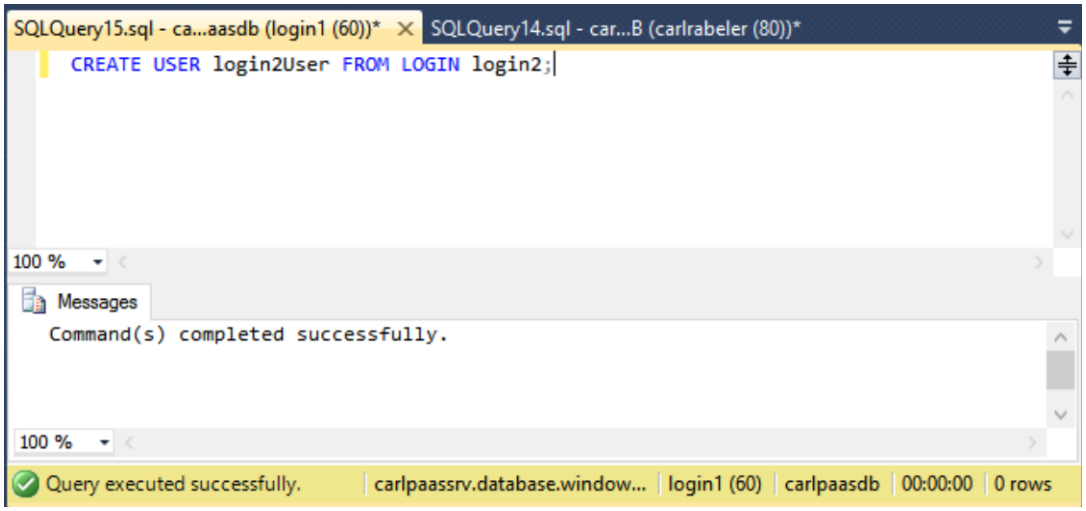


Figure 5-28: User with db_owner permissions and loginmanager permissions creating a user mapped to a login

10. Use the following Transact-SQL statement to create a contained user in your user database. See Figure 5-29.

```
CREATE USER user2 WITH PASSWORD ='p@ssw0rd1';
```

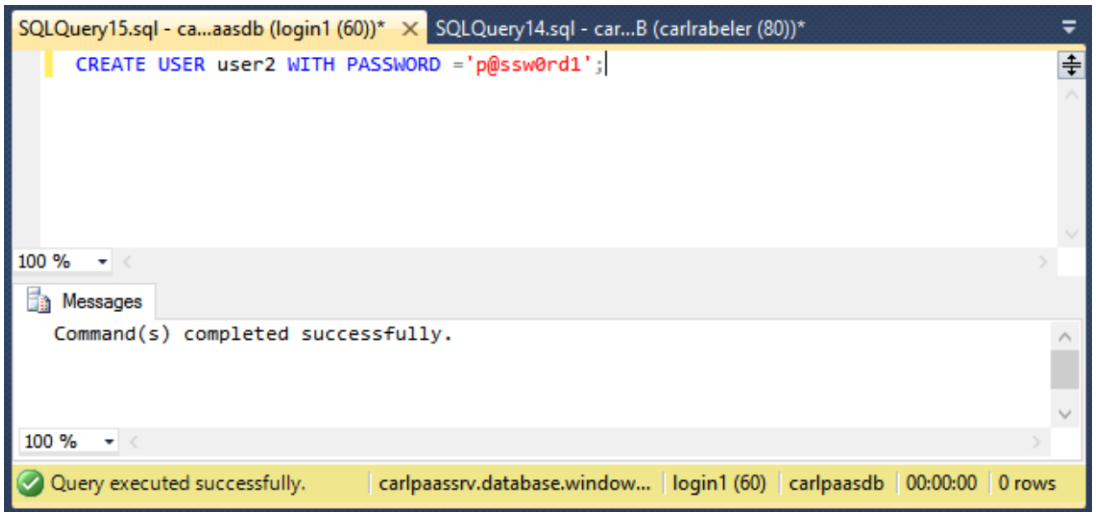


Figure 5-29: User with db_owner permissions creating a contained user

Walk-through: Creating a database copy using automated backups

In this walk-through, we will explore the automated Azure SQL Database backups and create a database copy on a logical server.

Using automated backups to create a database copy

1. Connect to the Azure portal.
2. On the default blade, click SQL Databases and then click your user database.

3. On the blade for your database, click Copy. See Figure 5-30.

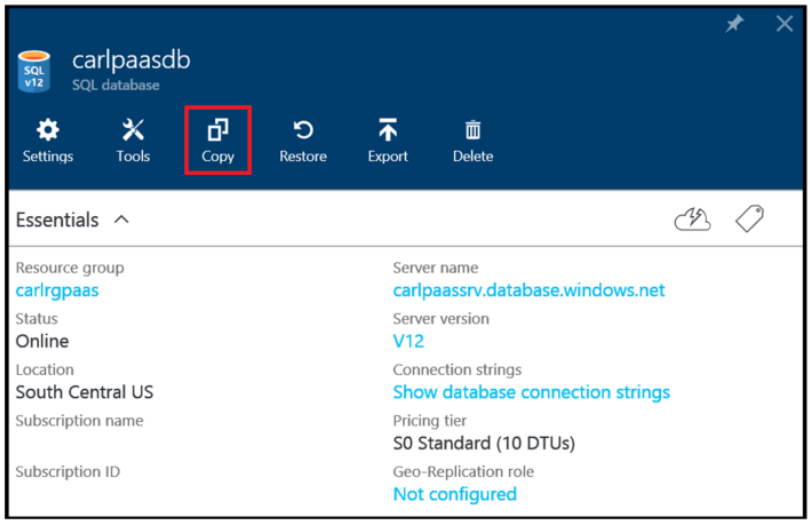


Figure 5-30: SQL Database copy option

4. On the Copy blade, choose a name for the database copy. I will keep the default database name. See Figure 5-31.

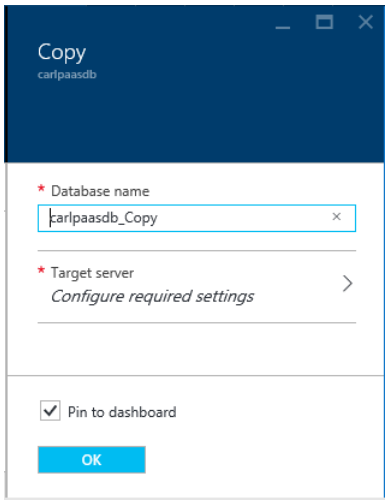


Figure 5-31: Copy blade

5. Click Configure Required Settings to specify the target server. Notice that you can either create a new server or restore a copy of the database on an existing server. See Figure 5-32.

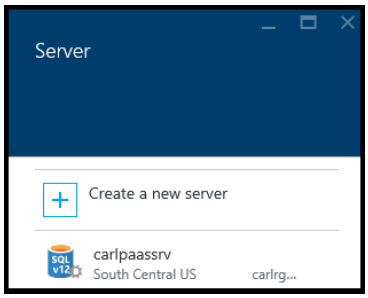


Figure 5-32: Target Server blade

6. Select your existing logical server and then click OK.
7. When the copy of the SQL Database is complete, switch to SQL Server Management Studio.
8. Refresh the Databases node to display the database copy. See Figure 5-33.

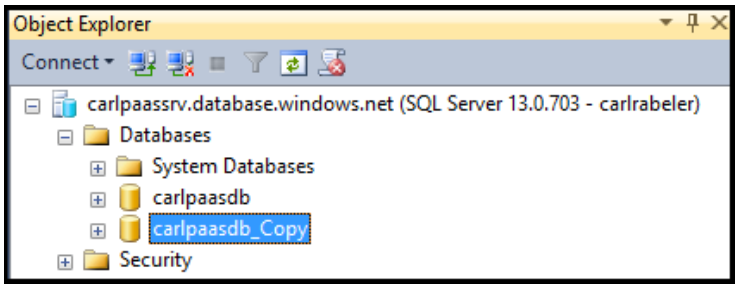


Figure 5-33: Database copy complete

Walk-through: Restoring a database copy using automated backups

In this walk-through, we will explore the automated Azure SQL Database backups and restore a database as of a point in time.

Using automated backups to restore a database to a point in time

1. Switch to the Azure portal.
2. On the blade for your database, click Restore. See Figure 5-34.

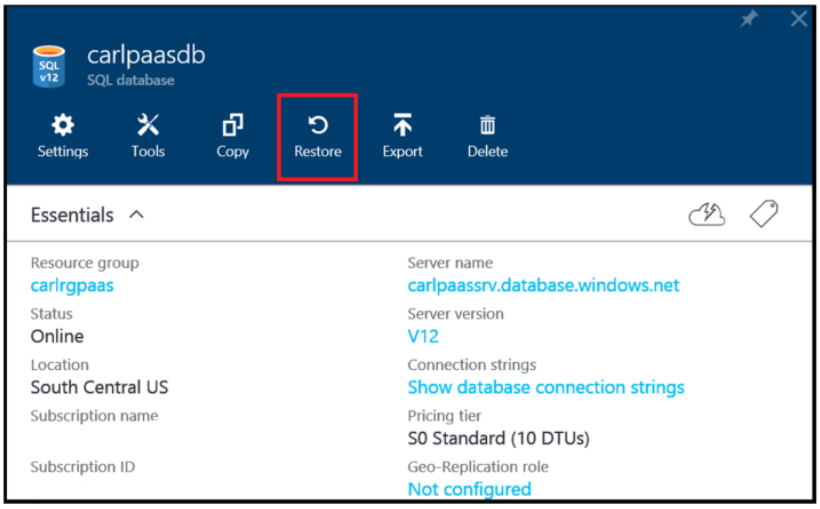


Figure 5-34: SQL Database restore option

3. On the Restore blade, review the options you have to restore to a point in time. The oldest restore point will depend on when your database was created and the service tier. Notice also that the default is to restore as a new database with a time stamp in the name to your existing database. See Figure 5-35.

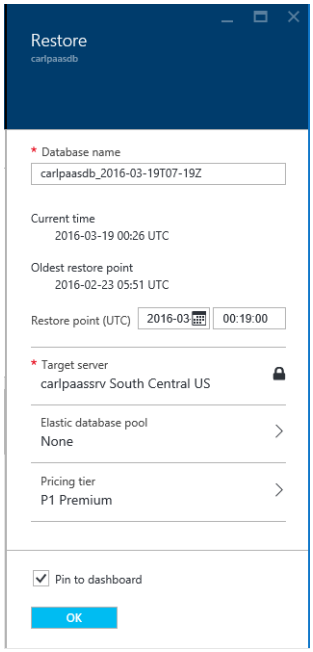


Figure 5-35: Restore blade

4. I am going to select a restore point of March 1, 2016, with a time of 00:00:00 to have a database copy containing a transactionally consistent copy of the AdventureWorks2008R2 data through the end of February. See Figure 5-36.

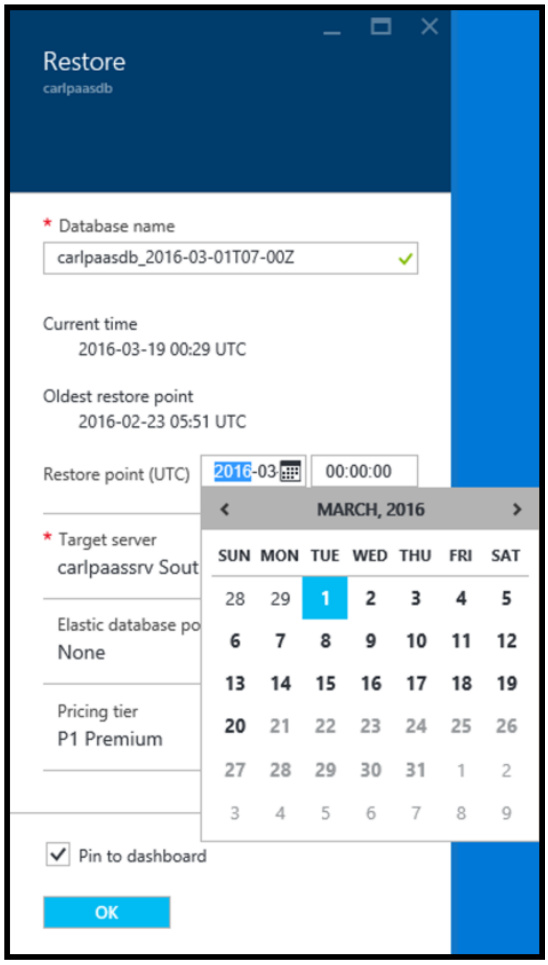


Figure 5-36: Selecting a restore point

5. Click P1 Premium to change the pricing tier. The Choose Your Pricing Tier blade opens. See Figure 5-37.

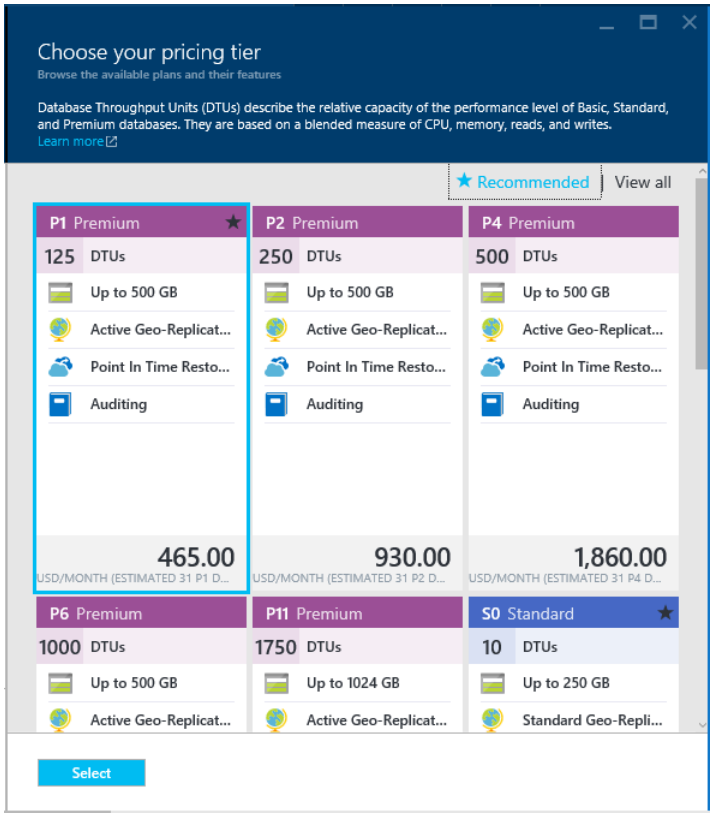


Figure 5-37: Pricing tiers

6. Because for this scenario, I am restoring to the end of February to perform some analysis on the February data at a later point in time and to export to an archive, I do not need the performance of a P1. I could select an S1 service tier. I can combine databases with different service tiers on the same logical server. See Figure 5-38.

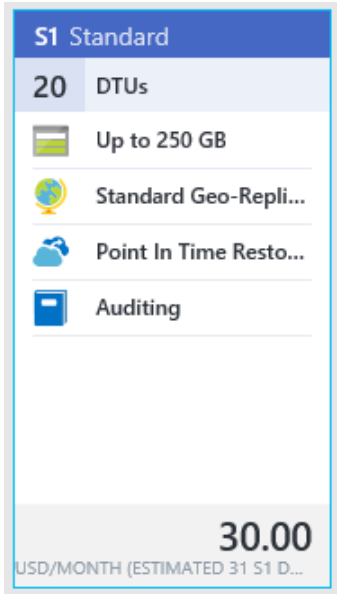


Figure 5-38: S1 Standard tier

7. Change the database name to reflect the restore point and then click OK. See Figure 5-39.

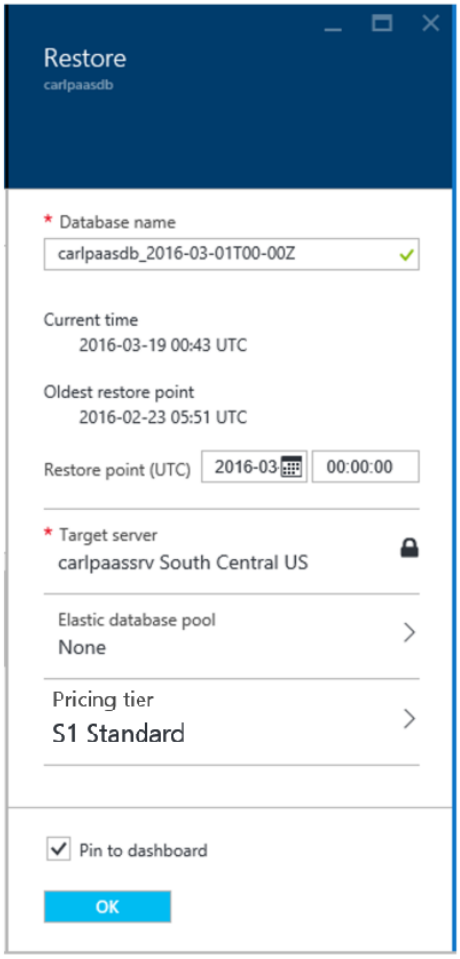


Figure 5-39: Restoring for February

8. When the copy of the SQL Database is complete, switch to SQL Server Management Studio.
9. Refresh the Databases node to display the database restore. See Figure 5-40.

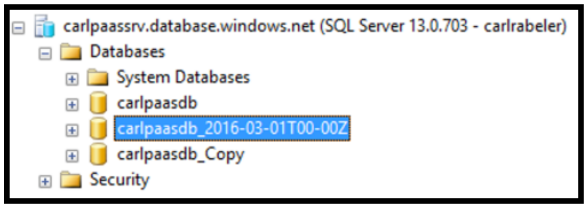


Figure 5-40: Restored database containing data through February 2016

Creating an archive

In this walk-through, we will export data and schema from the database we just created to have an archive of the database containing only February 2016 data. We will store it in Azure blob storage. Because we do not yet have a storage account and container configured for our archive, we will start there.

1. In the Azure portal, click Storage Accounts on the default blade.
2. On the Storage Accounts blade, click Add. See Figure 5-41.

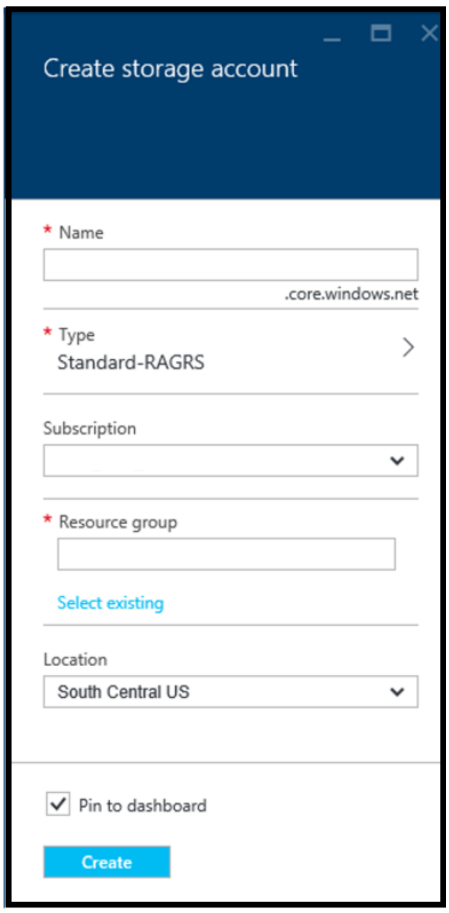


Figure 5-41: Create Storage Account blade

3. Define your storage account, storage type, subscription, resource group, and location. Click Create. See Figure 5-42.

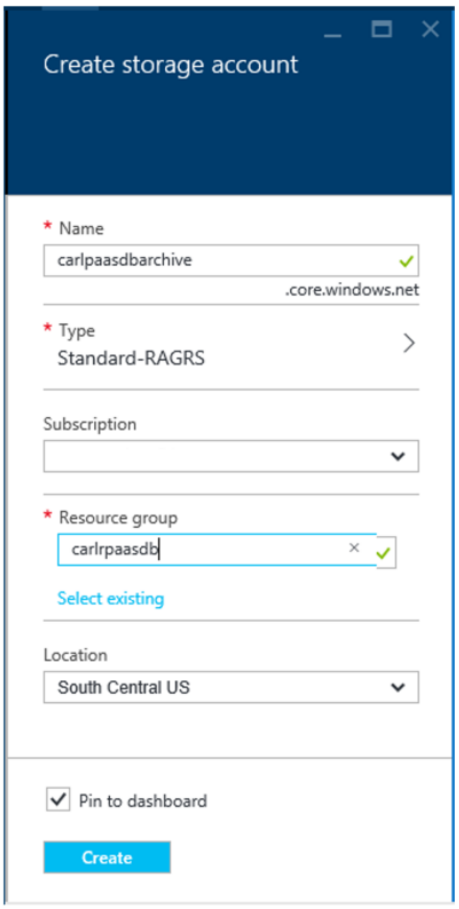


Figure 5-42: Creating new storage account

4. Now that you have your storage account, you are ready to export. On the default blade, click SQL Databases and then click the database containing the February only data.

5. On the SQL Database blade for this database, click Export. See Figure 5-43.

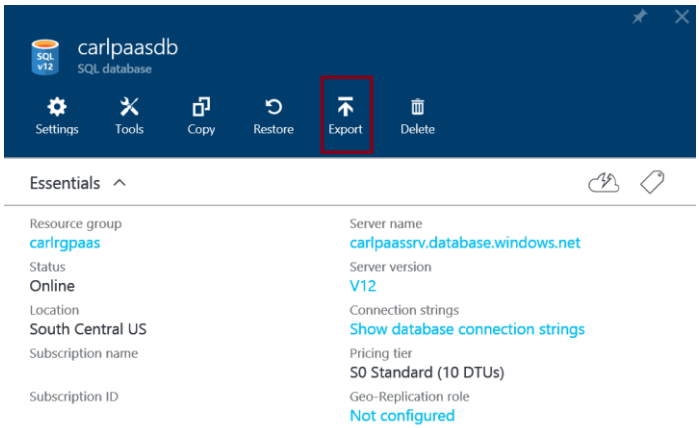


Figure 5-43: Export

6. On the Export Database blade, review the requested information. See Figure 5-44.

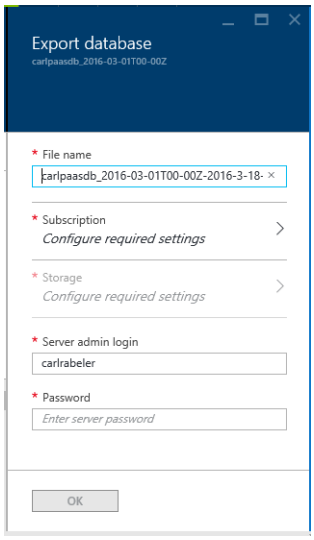


Figure 5-44: Export database requested information

7. Click Configure Required Settings for your subscription information and select your subscription.

8. Click Configure Required Settings for your storage information and select the storage account you created previously.

9. On the Containers blade, create a new container by clicking the Container link. See Figure 5-45.

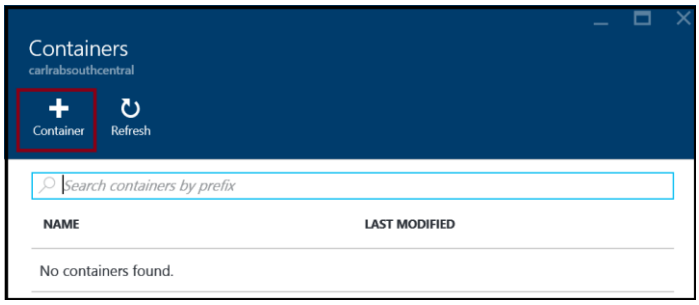


Figure 5-45: Creating a new container

10. On the New Container blade, specify a name for the archive container. See Figure 5-46.

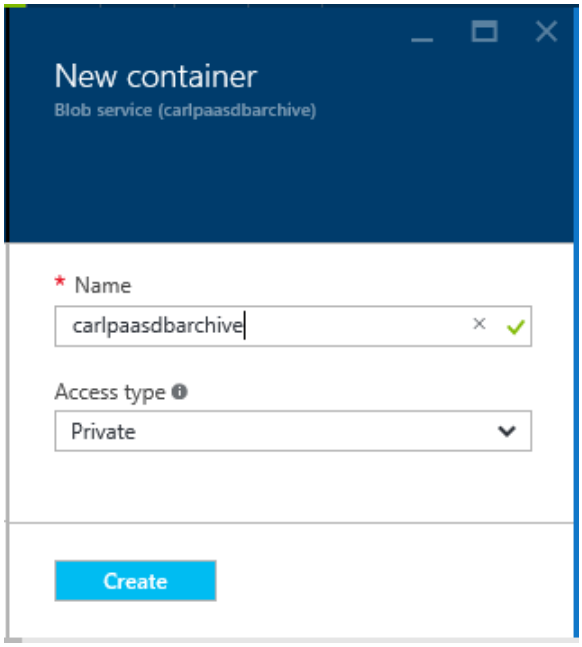


Figure 5-46: New container

11. Choose the type of container: Private, Blob, or Container. Your choice will depend on how publicly you will be sharing the container contents. Because this container will contain sensitive information, and because I can grant selected users read access later using a Shared Access Key, I will select Private and then click Select.

12. On the Containers blade, click this newly created container and click Select. See Figure 5-47.

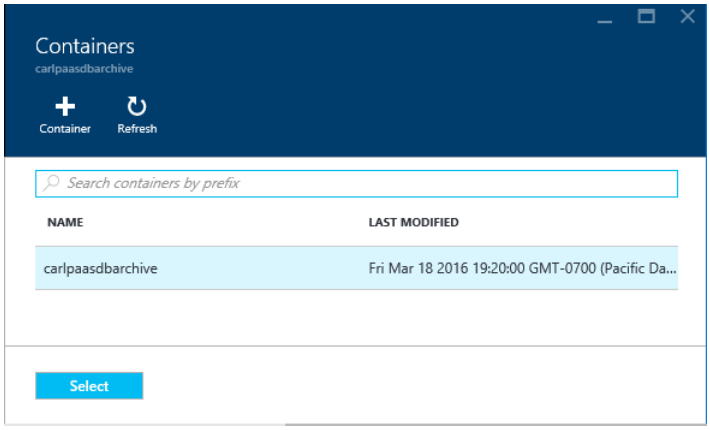


Figure 5-47: Selecting the container for the export

13. On the Export Database blade, provide the password for the server admin login and click OK. See Figure 5-48.

Figure 5-48: Export operation submitted

14. While the export is happening, let's configure SQL Server Management Studio to view the contents of our archive container. Switch to SQL Server Management Studio.

15. Click the Connect drop-down list in Object Explorer and select Azure Storage. See Figure 5-49.

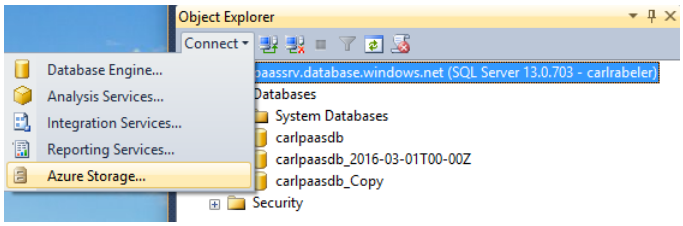


Figure 5-49: Connecting to Azure Storage

16. In the Connect to Microsoft Azure Storage window, enter the storage account and account key for the storage account you created previously and click Connect. See Figure 5-50.

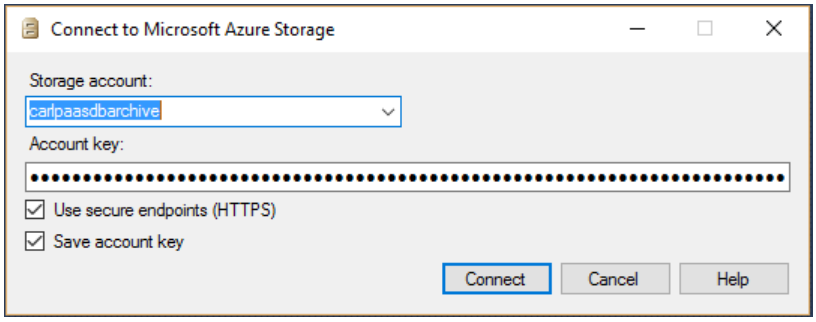


Figure 5-50: Storing Azure storage account information in SQL Server Management Studio

Note To retrieve the account key, open the blade for this storage account, navigate to the Account Keys blade, and copy the key.

17. In Object Explorer, expand Containers, expand your archive container, and notice that your BACPAC file already has been archived to this container. See Figure 5-51.

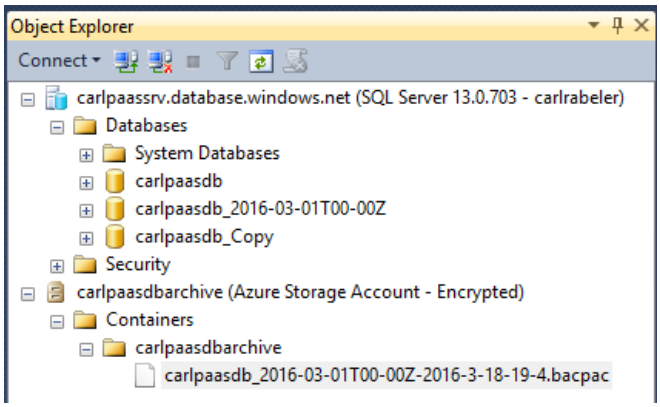


Figure 5-51: Archived BACPAC file

18. To import this BACPAC into a new database to an on-premises server, connect to the on-premises server and use the Import Tier Application Wizard.

Conclusion

Congratulations. You now know the differences between these two options to use SQL Server in the Azure cloud and how to get started with each of these offerings. You also know to import, backup, restore, and archive data, and you have been introduced to how to connect an application to a database in the Azure cloud. Finally, you know how to secure your data by using firewalls, using SQL Server and Active Directory authentication, granting permissions within the database itself, and using encrypted connections. You are now ready to dive into using SQL Database in an Azure virtual machine and SQL Database. Have fun!

About the author

Carl Rabeler is currently the content lead for SQL Database solution content on Azure.com. Carl is the author of multiple books and SQL CAT white papers, as well as various current MSDN and Azure.com articles on SQL Database and SQL Server 2016. He has been working with SQL Server since SQL Server 6.5 and with Microsoft Hyper-V and virtualization of SQL Server since Windows Server 2008. He has also been working with the Microsoft Azure cloud and SQL Database since its inception. In addition, Carl has been a speaker at multiple SQL PASS conferences and SQL Saturday events.

From technical overviews to drilldowns on special topics, get *free* ebooks from Microsoft Press at:

www.microsoftvirtualacademy.com/ebooks

Download your free ebooks in PDF, EPUB, and/or Mobi for Kindle formats.

Look for other great resources at Microsoft Virtual Academy, where you can learn new skills and help advance your career with free Microsoft training delivered by experts.

Microsoft Press