CHAPTER 4

Deploying Azure SQL

Deploying an Azure SQL Managed Instance or Database is a different but similar experience than deploying SQL Server on Azure Virtual Machine. The experience is the same because you can use the Azure portal and CLI. The difference is that Azure is managing the virtual machine and infrastructure, so several of the options you pick for a virtual machine you do not have to worry about.

In this chapter, you will learn the options and process to deploy and connect to an Azure SQL Managed Instance and Database. You will also learn the options to migrate existing databases into Azure SQL. In addition, you will learn some implementation details of the architecture used to host Azure SQL Managed Instances and Databases.

You have the option to follow along the examples in this chapter. You will need the following to complete these examples:

- An Azure subscription.
- A minimum of Contributor role access to the Azure subscription. You can read more about Azure built-in roles at `https://docs.microsoft.com/en-us/azure/role-based-access-control/built-in-roles`.
- Access to the Azure Portal (web or Windows application).
- Installation of the **az** CLI (see `https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest` for more details). You can also use the Azure Cloud Shell instead since az is already installed. You can read more about the Azure Cloud Shell at `https://azure.microsoft.com/en-us/features/cloud-shell/`.

B. Ward, *Azure SQL Revealed*, https://doi.org/10.1007/978-1-4842-5931-3_4

- You will run some T-SQL in this chapter, so install a tool like SQL Server Management Studio (SSMS) at https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15. You can also use Azure Data Studio at https://docs.microsoft.com/en-us/sql/azure-data-studio/download-azure-data-studio?view=sql-server-ver15.
- For migration scenarios, you will need to download the Data Migration Assistant (DMA) tool from https://docs.microsoft.com/en-us/sql/dma/dma-overview and have access to a SQL Server instance.

Pre-deployment Planning

Before you jump into deploying an Azure SQL Database or Managed Instance, I recommend you spend some time doing some pre-deployment planning. Reviewing your choices and making a few informed decisions will save you time and money.

New Deployment or Migration

One of the first decisions to make which may be easy is whether you plan to migrate an existing database or instance or deploy a new database or instance. The process of deploying will be the same, but migrating implies you need to assess your current SQL Server instance, database, or other database environment before you deploy. Your assessment will give you guidance on what type of deployment choice you need to make based on your current requirements. You must decide what possible changes must be made to your application, schema, scripts, or other aspects to your current deployment with SQL Server or other database platforms. You also must consider how to migrate your actual data into the new deployment. This chapter will include sections specifically geared around what consideration and tools you can use to migrate to Azure SQL Managed Instance and Database. Here are two great resources for you to consider as you think about migration:

- Azure Migration Program – https://azure.microsoft.com/en-us/migration/migration-program/
- The Microsoft Data Migration Guide – https://datamigration.microsoft.com/

Making Deployment Choices

Whether you are migrating or creating a new deployment, you have several choices to make that is worth the time to plan out. Chapter 2 of this book is invaluable to go back and read as it describes choices and differences between Azure SQL Managed Instance and Azure SQL Database.

Having said that, let us quickly review some important choices at a high level that can affect your decision-making:

- If you need SQL Server instance features like SQL Server Agent, Database Mail, and cross-database queries, Managed Instance is the choice you need to make.
- If your database size is > 8TB, your only choice as of the time of writing this book is Azure SQL Database Hyperscale.

Past these two choices, either Azure SQL Managed Instance or Database likely meets your needs. However, as I called out in Chapter 2 of the book, there can be advantages in using Azure SQL Database because Microsoft will manage both the infrastructure and the SQL Server instance to let you focus on the database. Furthermore, Azure SQL Database can offer you more options such as Serverless compute and Automated Tuning for indexes.

The options you will pick as you deploy a Managed Instance or Database will look like the following as seen in Figure 4-1.

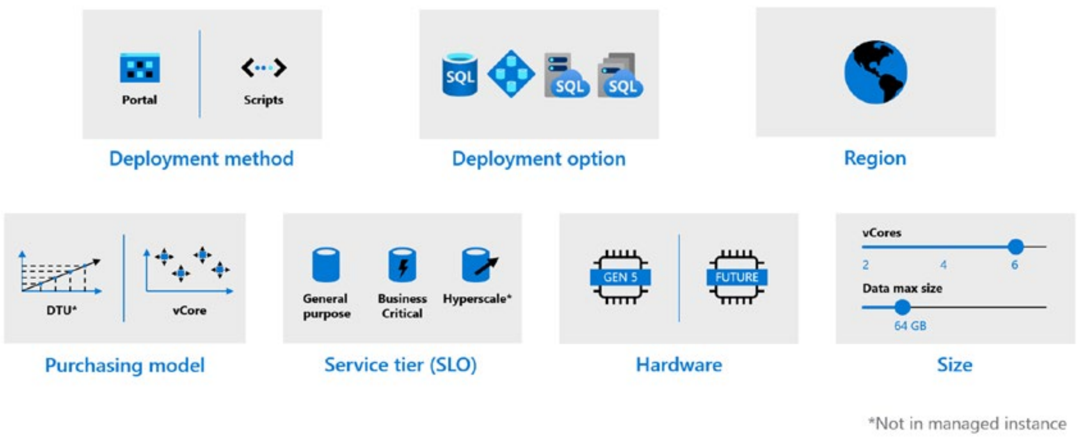


Figure 4-1. *Deployment choices for Azure SQL*

Deployment Method

You can deploy an Azure Managed Instance using the Azure Portal or through a CLI with the az utility, PowerShell, or even REST APIs (az rest can be used if you do not want to write code).

If you are just trying out Azure SQL or doing a proof of concept, you can easily use the Azure Portal. However, for a repeatable process to deploy (imagine if you needed to for some reason redeploy at any time), a script using a CLI is a better option. Remember you also can use Azure templates to help automate deployments. You can read more about using Azure templates for Azure SQL at `https://docs.microsoft.com/en-us/azure/azure-sql/database/arm-templates-content-guide?tabs=single-database`.

Another option for developers to automate deployment is with Azure DevOps and Pipelines. I really like this blog post by my colleague at Microsoft Arvind Shyamsundar on DevOps and Azure SQL at `https://devblogs.microsoft.com/azure-sql/devops-for-azure-sql/`.

Deployment Option

I discussed earlier whether you will consider Azure SQL Managed Instance or Database. Within each of these options are a choice of using a *pool*. Azure SQL Managed Instance offers a *Managed Instance Pool* which could be a better fit for a smaller, cost-effective Managed Instance. Deployment times are also much faster with pools. You can read more about Managed Instance Pools at `https://docs.microsoft.com/en-us/azure/azure-sql/managed-instance/instance-pools-overview`.

Azure SQL Database provides an option called an *elastic pool*. An elastic pool can be a good choice if you plan to use Azure SQL Database to host many databases. ISVs and Software as a Service (SaaS) developers often look at this choice to save costs and manage databases more efficiently. You can read more about elastic pools at `https://docs.microsoft.com/en-us/azure/azure-sql/database/elastic-pool-overview`.

Region

Choosing a specific Azure region can be important just as I described in Chapter 3 on virtual machines. You need to make sure your deployment options are available in your Azure region choice. A full list of Azure products by region can be found at `https://azure.microsoft.com/en-us/global-infrastructure/services/`.

You may have certain compliance and security requirements which also dictate what region you choose.

You may be implementing specific HADR options like Availability Zones, Geo-replication, or Auto-Failover Groups and have specific regions in mind to make those deployments successful.

If you need to move an Azure SQL Managed Instance or database to another region, read a checklist in our documentation at `https://docs.microsoft.com/en-us/azure/azure-sql/database/move-resources-across-regions`.

Azure SQL Database is a "Ring 0" service which means it gets deployed in every region as a default service. Managed Instance is not exactly at that status yet, but it is generally available in all regions.

In addition, you need to consider where your application will be hosted and latency requirements between where the application will be hosted and your Azure SQL deployment. Consider performance and proximity to other services. I was chatting with my colleague Anna Thomas on this topic. She said, "...but I feel that it's not just where the application is – where are the users? Where should the application be? If you have geo-replication or auto-failover groups, how do you build a globally available solution?"

Purchasing Model

For Azure SQL Database only, you will need to choose a *Purchasing model*. The choices are DTU or vCore. I explained these models and the history behind them in Chapters 1 and 2 of the book. While the DTU model may be a valid choice for you, I recommend the vCore model.

If you select the DTU model and want to move to the vCore model at a later date, consult the documentation at `https://docs.microsoft.com/en-us/azure/azure-sql/database/migrate-dtu-to-vcore`.

Service Tier (SLO)

If Azure SQL Managed Instance is your deployment option, then you will need to select a *Service Tier* of General Purpose (GP) or Business Critical (BC). A SLO stands for Service Level Objective and is the combination of choices of Purchasing Model, Service Tier, and Hardware. I described these service tier options in Chapter 2 of the book. While resource limits and performance may differ, one of the primary differences with these tiers is how Availability works which you will learn more about in Chapter 8 of the book. One notable difference for Business Critical is that it supports In-Memory OLTP capabilities. A comparison between GP and BC for Managed Instance can be found at https://docs.microsoft.com/en-us/azure/azure-sql/database/service-tiers-general-purpose-business-critical.

Tip You will see later in this chapter that the time it takes to deploy for Managed Instance can be lengthy. Changing between GP and BC is possible but could result in significant downtime.

If Azure SQL Database is your deployment option, then you also have the choice of General Purpose (GP) vs. Business Critical (BC) service tiers. In addition, you have the choice of *Hyperscale*. If you choose General Purpose, you also have the choice of *Provisioned* vs. *Serverless*. This is also called a *computer model or tier*. I covered all these options in Chapter 2 of the book.

GP vs. BC is a similar choice as with Azure SQL Managed Instance. Read the same documentation page at https://docs.microsoft.com/en-us/azure/azure-sql/database/service-tiers-general-purpose-business-critical for a comparison. Hyperscale is your best choice for very large databases and has some attractive capabilities for scaling, replicas, and restore performance. You can read more about Hyperscale at https://docs.microsoft.com/en-us/azure/azure-sql/database/service-tier-hyperscale. Serverless is a unique option we have created for autoscale and scenarios where your database may always not be utilized. It provides a new cost-effective way to deploy and use an Azure SQL Database. Read more about Serverless at https://docs.microsoft.com/en-us/azure/azure-sql/database/serverless-tier-overview.

Switching between GP and BC for Azure SQL Database is typically significantly faster than with Managed Instance. You can also switch between Serverless and Provisioned easily. However, Hyperscale is the one option you cannot switch back once you choose it without completing migrating your database to the new deployment option.

Hardware

Even though for Azure SQL we abstract you from the infrastructure and virtualization used for the deployment, we provide options for a *hardware generation*.

We are constantly looking to take advantage of new hardware supplied within the Azure infrastructure so these choices may be new by the time you are reading this book.

As of summer of 2020, the only hardware generation choices aside from the default Gen5 generation are with Azure SQL Database. The Fsv2-series is available in certain regions for General Purpose. This hardware option provides more CPU performance per vCore than Gen5. The M-series option for Business Critical offers more memory and vCores. Keep track of the latest on hardware generations at `https://docs.microsoft.com/en-us/azure/azure-sql/database/service-tiers-vcore?tabs=azure-portal#hardware-generations`.

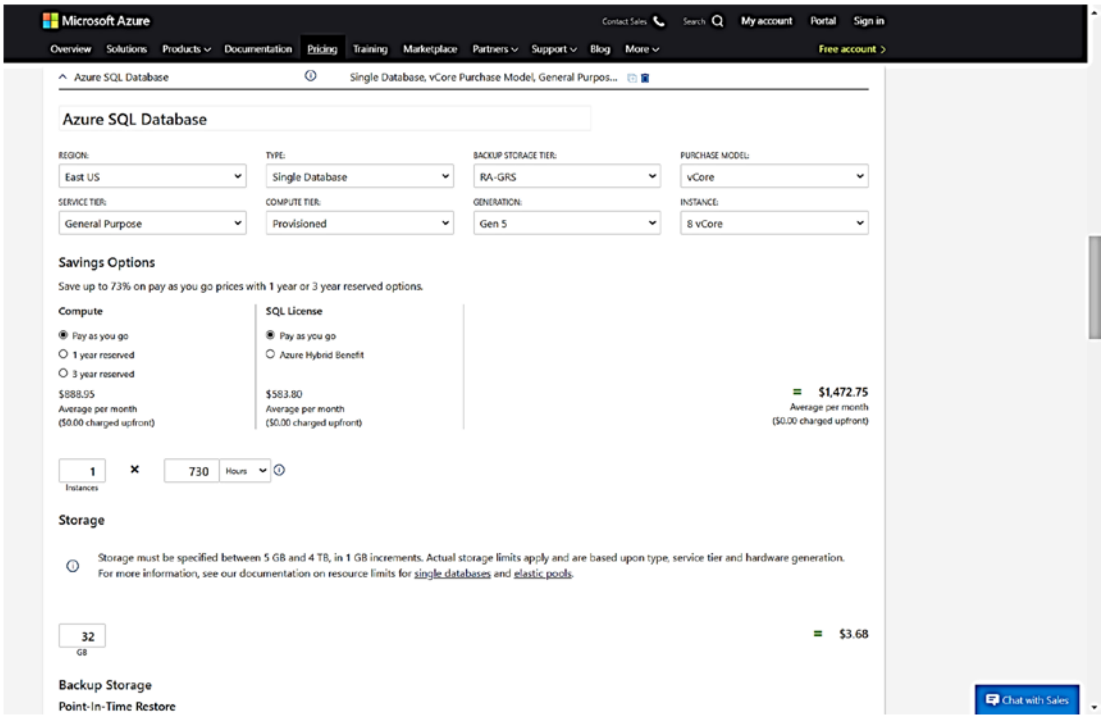
Note You might see some evidence of Gen4 hardware as you use Azure SQL. This hardware generation is being phased out, so focus on Gen5 or newer hardware generations.

Sizes

Once you have figured these options, you have choices on size. The DTU model for Azure SQL Database has a DTU number you can choose (and a data size). For the vCore purchasing you model, you have both number of vCores and database size to select. There are a few differences on how these options work depending on your other choices. I will describe these differences as I walk you through the deployment process in the rest of this chapter.

Price

Just like with Azure Virtual Machine, take advantage of the Azure Pricing Calculator to plug in some of these choices to get an idea of your costs. This includes using Azure Hybrid Benefit. You can find the pricing calculator for Azure SQL Managed Instance at `https://azure.microsoft.com/en-ca/pricing/details/azure-sql/sql-managed-instance/single/` and Azure SQL Database at `https://azure.microsoft.com/en-us/pricing/calculator/?service=sql-database`. Figure 4-2 shows an example of the pricing calculator for Azure SQL Database.



Azure SQL Database — Single Database, vCore Purchase Model, General Purpos...

Azure SQL Database

REGION: East US	TYPE: Single Database	BACKUP STORAGE TIER: RA-GRS	PURCHASE MODEL: vCore
SERVICE TIER: General Purpose	COMPUTE TIER: Provisioned	GENERATION: Gen 5	INSTANCE: 8 vCore

Savings Options

Save up to 73% on pay as you go prices with 1 year or 3 year reserved options.

Compute: [✓ Pay as you go] 1 year reserved 3 year reserved
$888.95 Average per month ($0.00 charged upfront)

SQL License: [✓ Pay as you go] Azure Hybrid Benefit
$583.80 Average per month ($0.00 charged upfront)

= $1,472.75 Average per month ($0.00 charged upfront)

1 Instances × 730 Hours

Storage

Storage must be specified between 5 GB and 4 TB, in 1 GB increments. Actual storage limits apply and are based upon type, service tier and hardware generation. For more information, see our documentation on resource limits for single databases and elastic pools.

32 GB = $3.68

Backup Storage

Point-In-Time Restore

Figure 4-2. *The pricing calculator for Azure SQL Database*

Consider Resource Limits

Your choices from deployment options, service tier, and sizes can affect your resource limits. These are the following resource limits to consider as you make these choices. I call these out because these may not be obvious as you deploy through the Azure Portal:

- Max Memory
- Max Log Size
- Log Rate Governance
- IOPS and I/O latency
- Max size of Tempdb
- Max concurrent workers
- Backup Retention

I will discuss more about Log Rate Governance and IOPS and I/O latency in Chapter 7 of this book. For now, keep these concepts in mind as they can affect performance of applications such as those that are heavy transaction log users.

To see the specific limits for Azure SQL Managed Instance, please see these very well-documented tables at https://docs.microsoft.com/en-us/azure/sql-database/sql-database-managed-instance-resource-limits.

To see the specific limits for Azure SQL Database, please see the table at https://docs.microsoft.com/en-us/azure/sql-database/sql-database-vcore-resource-limits-single-databases.

You should also know that you have overall Azure SQL limits per subscription per region. You can view what these limits are at https://docs.microsoft.com/en-us/azure/sql-database/sql-database-managed-instance-resource-limits#regional-resource-limitations and https://docs.microsoft.com/en-us/azure/sql-database/sql-database-resource-limits-database-server#maximum-resource-limits.

It is possible to make a request to Microsoft increase your subscription limits. This is called a *quota increase request*. Read more at https://docs.microsoft.com/en-us/azure/sql-database/quota-increase-request.

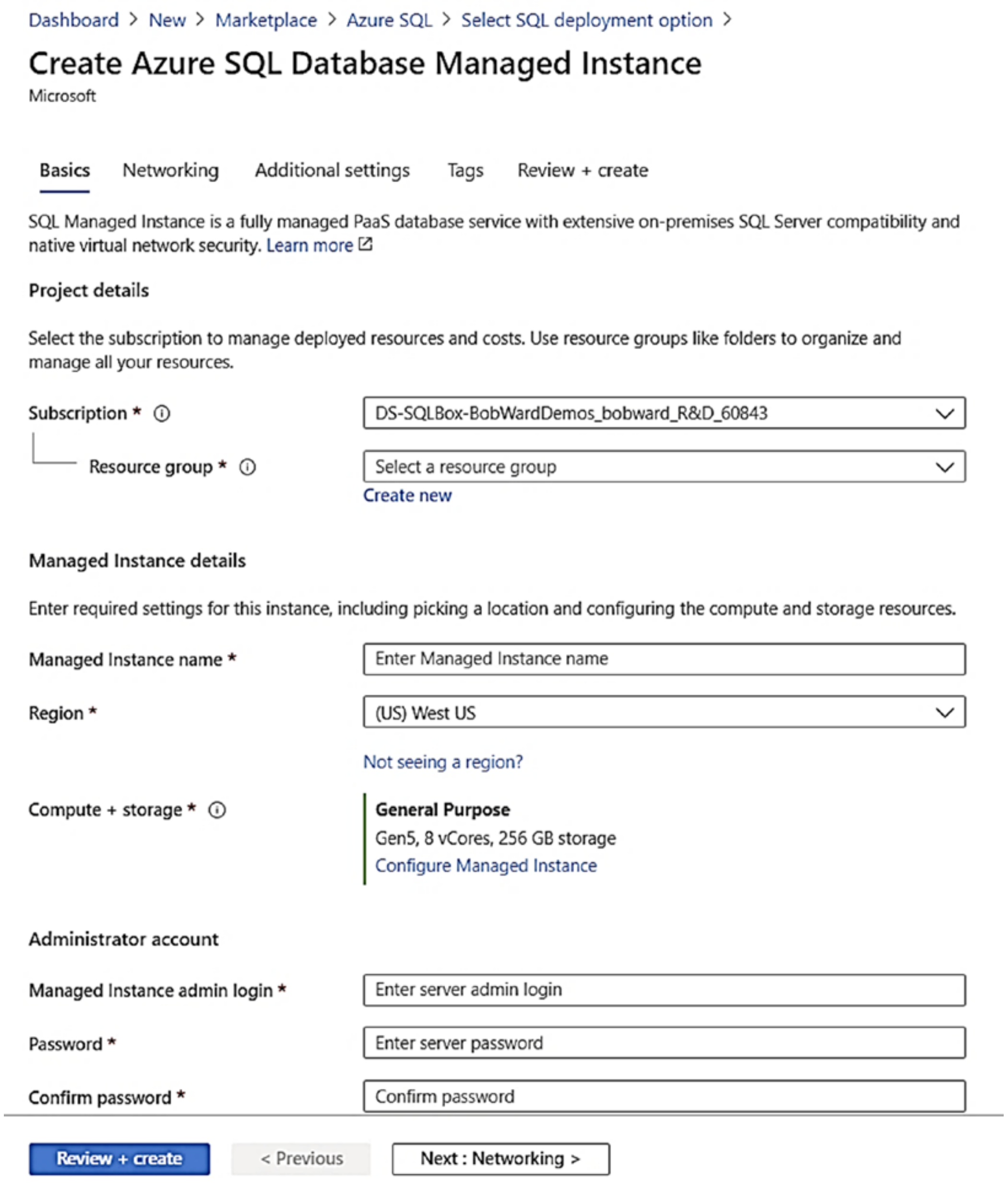
Deploying Azure SQL Managed Instance

Similar to the process I documented in Chapter 3 for a virtual machine, deploying an Azure SQL Managed Instance through the Azure Portal starts by using the Azure SQL option from the Azure Marketplace (I showed you this view in Figure 3-1).

Using the three Azure SQL choices, you would select SQL Managed Instance and single instance and then click **Create**.

Note At the time of writing this book, an instance pool can only be created through PowerShell. I will talk more about Instance Pools in the section titled "Implementation Details." You can read more about Instance Pools at https://docs.microsoft.com/en-us/azure/azure-sql/managed-instance/instance-pools-overview.

Like the experience of deploying a virtual machine, you will have options but not the same options. You can see in Figure 4-3 you will have options for *Basics*, *Networking*, *Additional settings*, and *Tags*. Basics is the only required set of fields, while the others are optional.



Dashboard > New > Marketplace > Azure SQL > Select SQL deployment option >

Create Azure SQL Database Managed Instance

Microsoft

Basics | Networking | Additional settings | Tags | Review + create

SQL Managed Instance is a fully managed PaaS database service with extensive on-premises SQL Server compatibility and native virtual network security. Learn more

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	DS-SQLBox-BobWardDemos_bobward_R&D_60843
Resource group *	Select a resource group Create new

Managed Instance details

Enter required settings for this instance, including picking a location and configuring the compute and storage resources.

Managed Instance name *	Enter Managed Instance name
Region *	(US) West US Not seeing a region?
Compute + storage *	General Purpose Gen5, 8 vCores, 256 GB storage Configure Managed Instance

Administrator account

Managed Instance admin login *	Enter server admin login
Password *	Enter server password
Confirm password *	Confirm password

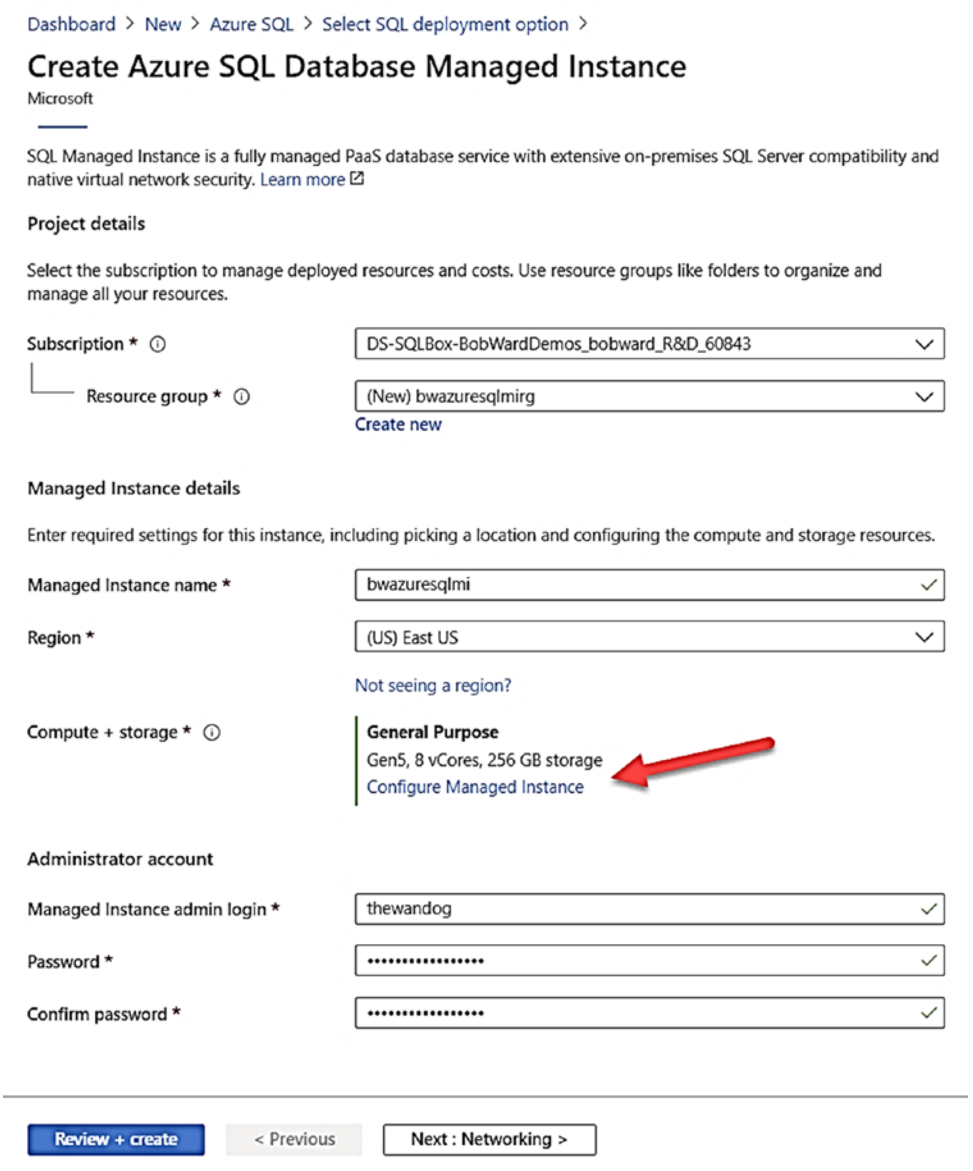
Figure 4-3. *The initial Azure SQL Managed Instance screen in the portal*

Deployment and Options

Let us walk through each screen with deployment options of a Managed Instance and then deploy it through the Azure Portal.

Basics

Figure 4-4 shows the fields you complete on the Basics screen, including choosing Subscription, Resource group, Region, Managed Instance name (this becomes part of the server name and @@SERVERNAME of the instance), and Administrator account and password.


Dashboard > New > Azure SQL > Select SQL deployment option >
Create Azure SQL Database Managed Instance
Microsoft

SQL Managed Instance is a fully managed PaaS database service with extensive on-premises SQL Server compatibility and native virtual network security. Learn more

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	DS-SQLBox-BobWardDemos_bobward_R&D_60843
Resource group *	(New) bwazuresqlmirg

Managed Instance details

Enter required settings for this instance, including picking a location and configuring the compute and storage resources.

Managed Instance name *	bwazuresqlmi
Region *	(US) East US
Compute + storage *	General Purpose Gen5, 8 vCores, 256 GB storage

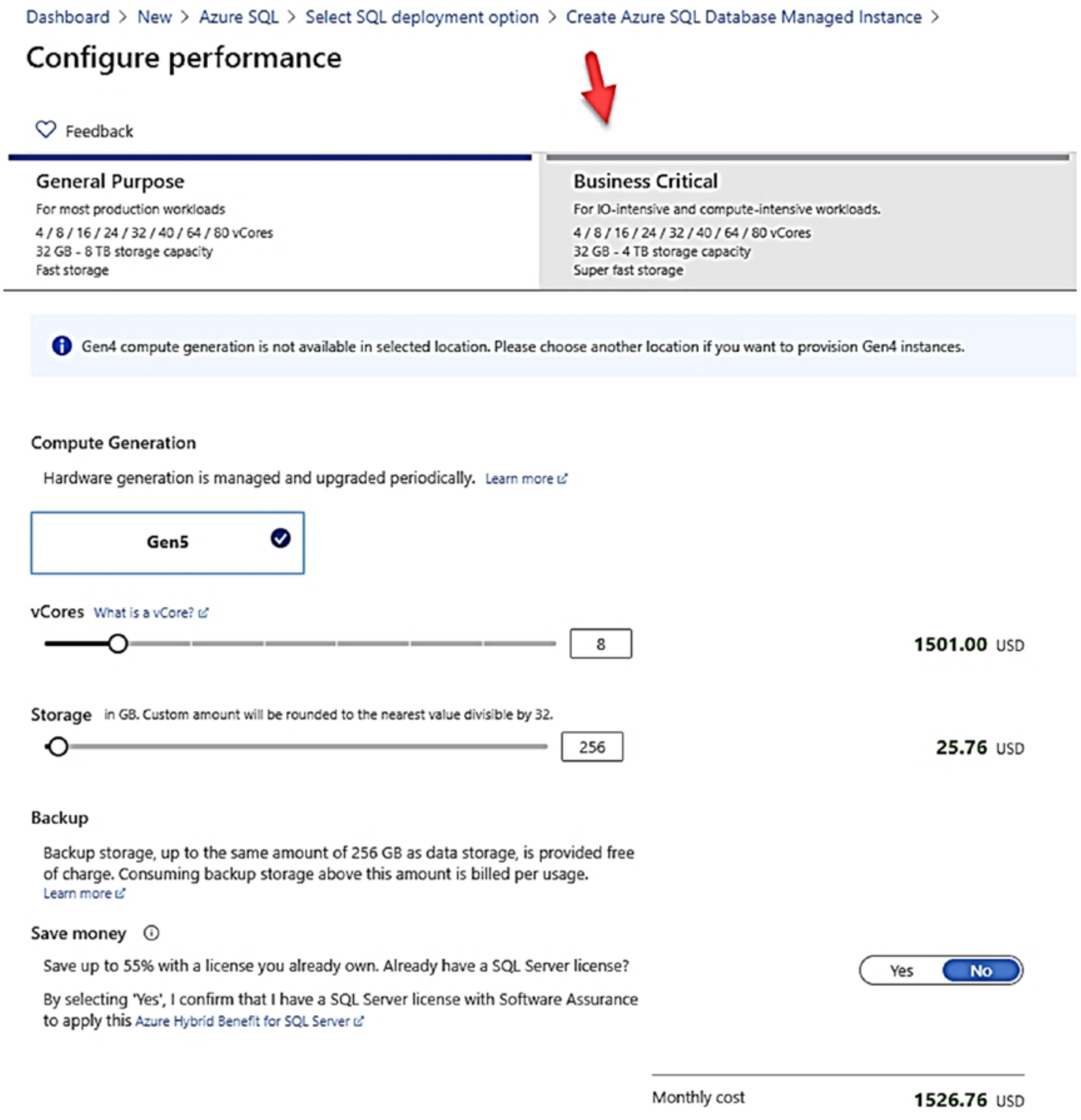
Administrator account

Managed Instance admin login *	thewandog
Password *	•••••••••••••••••
Confirm password *	•••••••••••••••••

Figure 4-4. *Filling out the basics for an Azure SQL Managed Instance deployment*

The Administrator account becomes a SQL Server login assigned to the sysadmin role.

Notice in the middle is an option called **Compute + storage**. This is where you will choose Service Tier and Size (vCores + Storage). Notice for me the default is General Purpose with 8 vCores and 256Gb max storage. Click **Configure Managed Instance** to see what your options are which should look like Figure 4-5.



Dashboard > New > Azure SQL > Select SQL deployment option > Create Azure SQL Database Managed Instance >

Configure performance

Feedback

General Purpose
For most production workloads
4 / 8 / 16 / 24 / 32 / 40 / 64 / 80 vCores
32 GB - 8 TB storage capacity
Fast storage

Business Critical
For IO-intensive and compute-intensive workloads.
4 / 8 / 16 / 24 / 32 / 40 / 64 / 80 vCores
32 GB - 4 TB storage capacity
Super fast storage

Gen4 compute generation is not available in selected location. Please choose another location if you want to provision Gen4 instances.

Compute Generation
Hardware generation is managed and upgraded periodically. Learn more

Gen5 ✓

vCores What is a vCore? 8 — 1501.00 USD

Storage in GB. Custom amount will be rounded to the nearest value divisible by 32. 256 — 25.76 USD

Backup
Backup storage, up to the same amount of 256 GB as data storage, is provided free of charge. Consuming backup storage above this amount is billed per usage.
Learn more

Save money
Save up to 55% with a license you already own. Already have a SQL Server license?
By selecting 'Yes', I confirm that I have a SQL Server license with Software Assurance to apply this Azure Hybrid Benefit for SQL Server

Yes [✓ No]

Monthly cost 1526.76 USD

Figure 4-5. Azure SQL Managed Instance Service Tier options

At the top of this screen, you can see that you can choose General Purpose (GP) or Business Critical (BC), and some of the resource limits and performance expectations are listed with each. From pre-deployment planning, remember there are other significant reasons to choose BC including

- Access to the In-memory OLTP feature
- Higher availability because BC uses a local storage and replica architecture

Below this are *slider bars* to choose the desired number of vCores and Maximum Storage. As you use each slider, the expected costs are updated to the right. vCores are only allowed in the increments shown on the screen. Managed Instance today supports up to 80 vCores for both GP and BC. 4 vCores is the minimum choice (Instance Pools support 2 vCore deployments).

The maximum storage value is called the *maximum instance size* and is the maximum size allowed for **all** database and transaction log files associated with databases for the managed instance. You should think of this like the maximum of a storage drive for your databases. The maximum storage size is different depending on your vCore choice and choice of GP and BC:

- 4 vCores have a max storage limit of 2TB for GP and 1TB for BC.
- Any other vCore choice past this supports up to 8TB for GP and 4TB for BC.

Note I discussed the architecture of Business Critical deployments in Chapter 2 of the book and will elaborate more on this architecture in Chapter 8. The reason for the lower limit of storage for BC is the fact that databases are stored on local SSD drives which have lower capacity than using Azure Storage.

Tempdb max sizes are dependent on vCore selections but are counted toward the overall maximum storage limit. In fact, all databases including system databases count toward the overall max instance storage size.

You can run the following query after you deploy your managed instance to see how much space your databases are taking up in relation to the max storage for the instance:

```
select top 1 used_storage_gb = storage_space_used_mb/1024,
    max_storage_size_gb = reserved_storage_mb/1024
from sys.server_resource_stats order by start_time desc
```

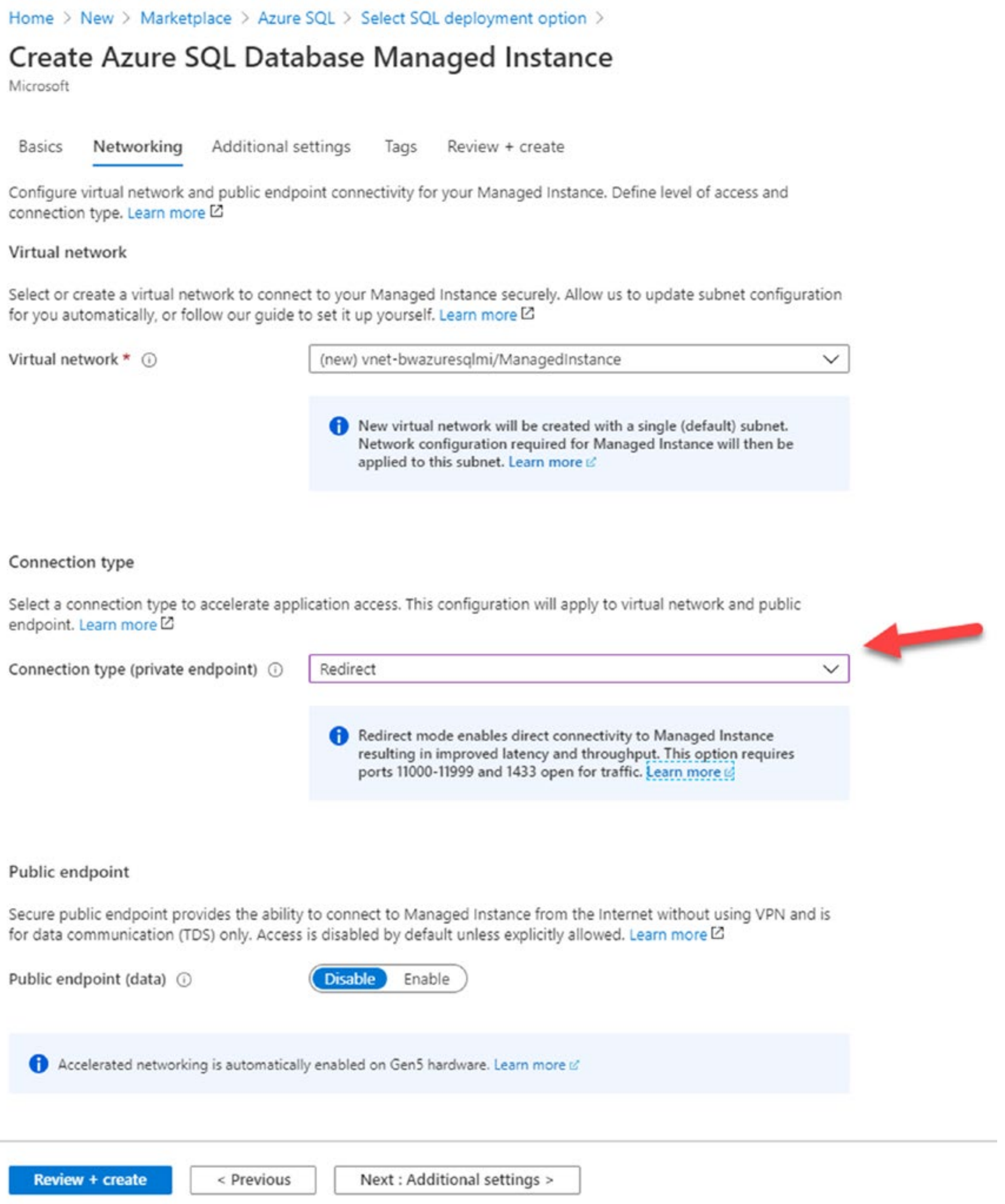
You also have a choice to save money using your existing SQL Server licenses with Azure Hybrid Benefit (AHB).

There is also a statement about backup storage and costs on this page. I will discuss more about Backups and Managed Instance in Chapter 8 of the book. Click **Apply** after you make any changes (Apply is only enabled if you change the defaults. You can click the X to get back to the Basics screen). For my example, I will leave the choice at 8 vCores and 256Gb of max storage.

Tip Hitting Apply will not deploy the Managed Instance yet but take your time to get your choices here as close to correct as possible. Why? You can change them later, but Managed Instance changes to tiers and sizes can be a long operation. Instance pools will not require as much time.

Networking

Click **Next: Networking** > to review your networking choices. Your screen should look like Figure 4-6.


Home > New > Marketplace > Azure SQL > Select SQL deployment option >
Basics
Networking
Additional settings
Tags
Review + create
Review + create
< Previous
Next : Additional settings >

Create Azure SQL Database Managed Instance

Microsoft

Configure virtual network and public endpoint connectivity for your Managed Instance. Define level of access and connection type. Learn more

Virtual network

Select or create a virtual network to connect to your Managed Instance securely. Allow us to update subnet configuration for you automatically, or follow our guide to set it up yourself. Learn more

Virtual network * : (new) vnet-bwazuresqlmi/ManagedInstance

New virtual network will be created with a single (default) subnet. Network configuration required for Managed Instance will then be applied to this subnet. Learn more

Connection type

Select a connection type to accelerate application access. This configuration will apply to virtual network and public endpoint. Learn more

Connection type (private endpoint): Redirect

Redirect mode enables direct connectivity to Managed Instance resulting in improved latency and throughput. This option requires ports 11000-11999 and 1433 open for traffic. Learn more

Public endpoint

Secure public endpoint provides the ability to connect to Managed Instance from the Internet without using VPN and is for data communication (TDS) only. Access is disabled by default unless explicitly allowed. Learn more

Public endpoint (data): [✓ Disable] Enable

Accelerated networking is automatically enabled on Gen5 hardware. Learn more

Figure 4-6. *Azure SQL Managed Instance Networking options*

Virtual Network

You can see at the top of the screen that a new virtual network will be created to host the Azure SQL Managed Instance. One of the advantages of Managed Instance is that it is deployed in a private virtual network. You could deploy your own Azure Virtual Network first (you can use the Azure portal or CLI) and select that virtual network on this deployment screen. If you choose to use your own virtual network, you must configure it a specific way which you can read at https://docs.microsoft.com/en-us/azure/azure-sql/managed-instance/vnet-existing-add-subnet.

Connection Type

Notice on my screen I have chosen a connection type of *Redirect*. The default is *proxy*. A proxy connection requires that any connection to the Managed Instance by a tool or application (a connection to the TDS port 1433) must always go through a *gateway*. A redirect connection type uses the gateway to find the direct virtual private IP address of the node containing the Managed Instance. All subsequent traffic flows directly to the node. You can read more about these connection types at https://docs.microsoft.com/en-us/azure/azure-sql/managed-instance/connection-types-overview. Proxy can be more secure, but redirect can be faster. The virtual network and included subnet will have all the appropriate Network Security Group (NSG) rules applied for these types if you choose to create the virtual network as part of this deployment step.

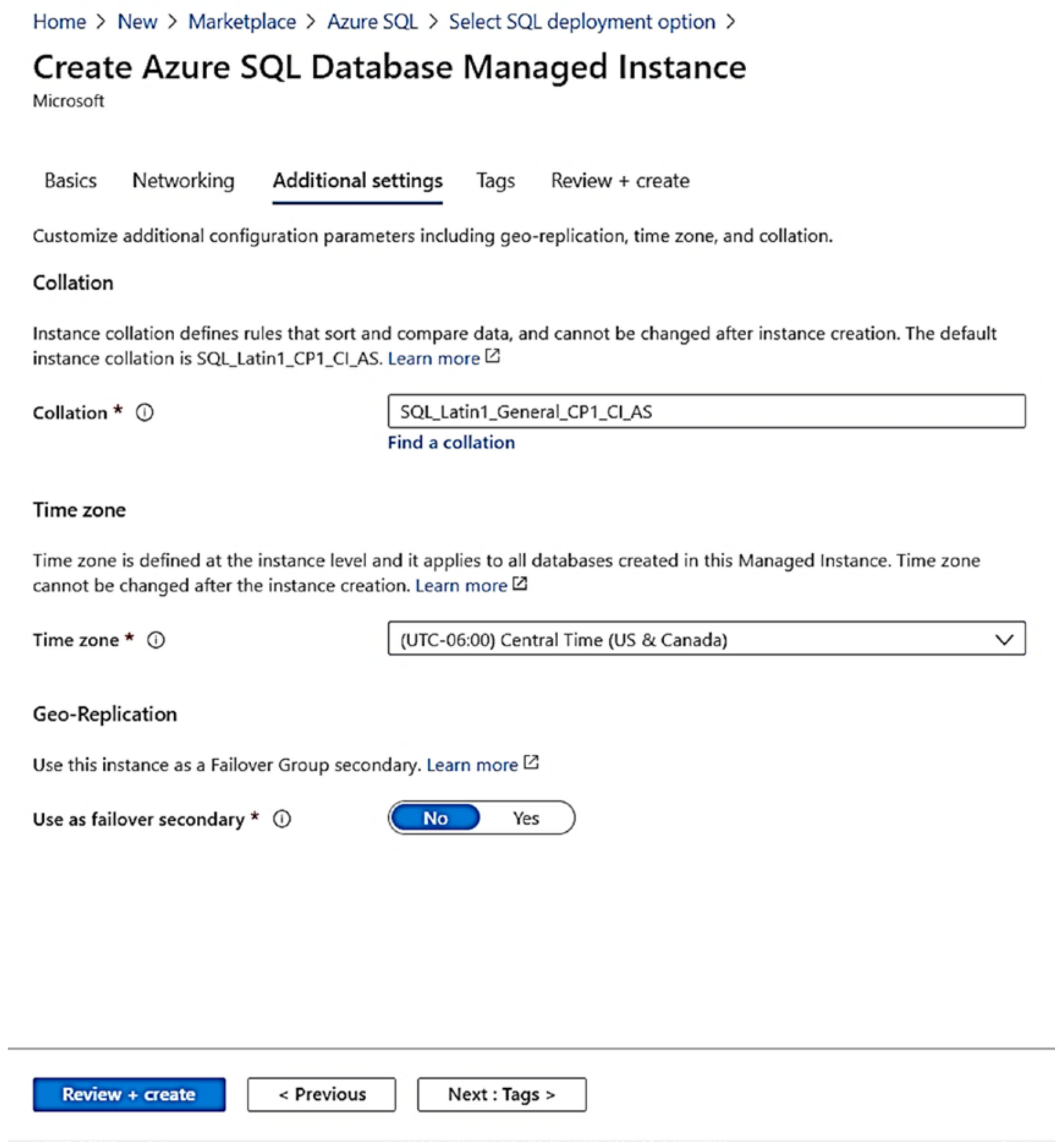
Public Endpoint

You have the option to enable TDS traffic on a public endpoint. The public endpoint will be enabled on port 3342 (and get redirected in the virtual network to the node instance port 1433). While I do not recommend using this option, it is one of the quickest ways to get connected to a Managed Instance. There could be other scenarios where you want to enable this. You can read more about the public endpoint for Managed Instance at https://docs.microsoft.com/en-us/azure/azure-sql/managed-instance/public-endpoint-overview.

Notice that Accelerated networking is automatically enabled for the Managed Instance.

Additional Settings

Click **Next: Additional settings** > to enable a few additional options for the deployment as seen in Figure 4-7.



Home > New > Marketplace > Azure SQL > Select SQL deployment option >

Create Azure SQL Database Managed Instance

Microsoft

Basics | Networking | Additional settings | Tags | Review + create

Customize additional configuration parameters including geo-replication, time zone, and collation.

Collation

Instance collation defines rules that sort and compare data, and cannot be changed after instance creation. The default instance collation is SQL_Latin1_CP1_CI_AS. Learn more

Collation * : SQL_Latin1_General_CP1_CI_AS

Find a collation

Time zone

Time zone is defined at the instance level and it applies to all databases created in this Managed Instance. Time zone cannot be changed after the instance creation. Learn more

Time zone * : (UTC-06:00) Central Time (US & Canada)

Geo-Replication

Use this instance as a Failover Group secondary. Learn more

Use as failover secondary * : [✓ No] Yes

Figure 4-7. Additional settings for Managed Instance

The **Collation** here is like setting a collation for a SQL Server. It is important to know that you cannot change the instance collation after it is supplied here.

Note You can set and change database collations on Managed Instance after deployment.

The **Time zone** is the recognized time zone by the SQL Server engine on the Managed Instance node. I have changed this to my local time zone, but it can be UTC or whatever time zone you want to choose. You cannot change this after deployment.

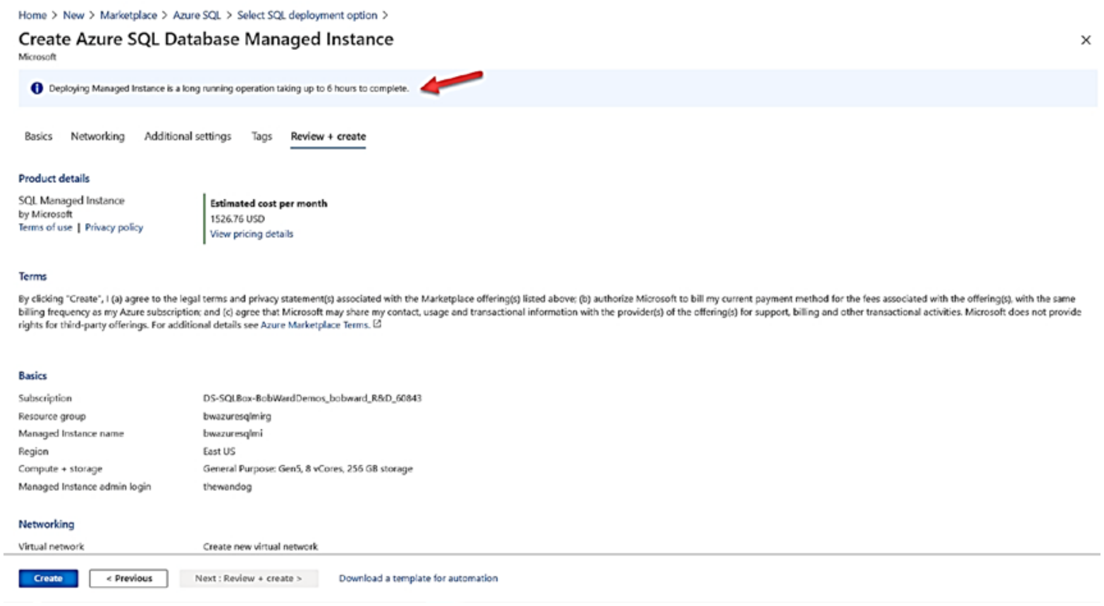
The Managed Instance can be part of a **failover group** which we will talk more about in Chapter 8 of the book. You will use that option when you review those topics later. For now, leave it to the default of No.

Tags

Click **Next: Tags** > to define a tag like how I described in Chapter 3 with Azure Virtual Machines. I will use a Name = Environment and Value = Development with a SQL managed instance resource.

Deploy!

Click **Next: Review + create** > to view the final screen before deploying as seen in Figure 4-8.


Home > New > Marketplace > Azure SQL > Select SQL deployment option >
Create Azure SQL Database Managed Instance
Microsoft
Basics
Networking
Additional settings
Tags
Review + create
Create
< Previous
Next : Review + create >
Download a template for automation

Deploying Managed Instance is a long running operation taking up to 6 hours to complete.

Product details

SQL Managed Instance
by Microsoft
Terms of use | Privacy policy

Estimated cost per month
1526.76 USD
View pricing details

Terms

By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. For additional details see Azure Marketplace Terms.

Basics

Subscription	DS-SQLBox-BobWardDemos_bobward_R&D_60843
Resource group	bwazuresqlmirg
Managed Instance name	bwazuresqlmi
Region	East US
Compute + storage	General Purpose: Gen5, 8 vCores, 256 GB storage
Managed Instance admin login	thewandog

Networking

Virtual network	Create new virtual network

Figure 4-8. *The validation screen for Managed Instance before deploy*

Like Figure 3-16 for Azure Virtual Machine, this screen shows estimated costs, Terms of use, Privacy Policy, a review of all the options you have chosen, and the option to download an Azure template that describes these deployment options. Notice the important warning at the top of the screen on the time it takes to deploy. So, click **Create** to deploy, leave this screen open to see the progress, and read on to the next section about deploying with a CLI and then some architecture and implementation details you might find interesting why the deployment runs. This section will explain a bit why the deployment can take so long.

Deploying with a CLI

An Azure Managed Instance can be deployed with command-line interfaces (CLI) through the **az sql mi** (`https://docs.microsoft.com/en-us/cli/azure/sql/mi?view=azure-cli-latest`) command interfaces or through **New-AzSQLInstance** PowerShell cmdlet (`https://docs.microsoft.com/en-us/powershell/module/az.sql/New-AzSqlInstance`).

I went down the path to build an example with az sql mi and found that I needed to run several az CLI commands to create the virtual network, subnet, and all associated settings. Therefore, I only recommend you use the az sql mi CLI with *Azure templates*. An example template can be found at `https://docs.microsoft.com/en-us/azure/azure-sql/managed-instance/create-template-quickstart?tabs=azure-cli`.

PowerShell requires you to set up the virtual network and other context before executing New-AzSQLInstance. There is a good tutorial on using PowerShell at `https://docs.microsoft.com/en-us/azure/azure-sql/managed-instance/scripts/create-configure-managed-instance-powershell`.

Implementation Details

Note These implementation details may change over time as we change and improve the service. I offer up some of these details so you can understand how we build, manage, and run the service.

Azure SQL Managed Instance is deployed on nodes (virtual machines) powered by Azure Service Fabric in a concept called a *ring* or *virtual cluster*. A virtual cluster is a dedicated set of isolated virtual machines that run in a virtual network subnet. Using a dedicated ring or cluster gives Managed Instance the isolation and private connection that was lacking for years from Azure SQL Database (which you will see has also been resolved).

When you deploy your first managed instance in a new virtual network (subnet) as I did in my example, you are in fact deploying an entire virtual cluster. This explains why the initial deployment can take so long to complete. You can deploy other managed instances in the *same virtual network subnet*, and the deployment is much faster.

A managed instance is a full SQL Server engine database instance deployed in a dedicated virtual machine in the virtual cluster. Microsoft will decide how to deploy these virtual machines on various nodes of the cluster. It is possible that a node may have one virtual machine with an instance or multiple virtual machines. As per the promise of Platform as a Service (PaaS), Microsoft abstracts you from those details. Your interface with the Managed Instance is through either standard SQL Server interfaces such as T-SQL or Azure interfaces such as the portal, CLI, or REST API. You will never directly access the underlying virtual machines.

This architecture explains also why certain management operations such as scaling vCores can also take a long period of time as some of these operations can require a deployment of a new virtual cluster with either attaching files from Azure Storage or reseeding a replica.

I described some of the architecture of General Purpose (GP) vs. Business Critical (BC) tiers in Chapter 2 of the book. I will describe them further in Chapter 8 of the book. Either of these service tiers uses the same virtual cluster architecture just with different storage and HA implementations.

Resource limits for a Managed Instance such as memory limits, max storage size, and others are enforced through several mechanisms. For example, memory limits are enforced with Windows Job Objects (and you cannot configure "max server memory"). You can read more about *Windows Job Objects* at `https://docs.microsoft.com/en-us/windows/win32/procthread/job-objects`. Storage capacity (or max size) is enforced by The File Server Resource Manager (FSRM) which you can read more about at `https://docs.microsoft.com/en-us/windows-server/storage/fsrm/fsrm-overview`.

A *Managed Instance pool deployment* can be far faster because an instance pool can be a set of SQL Server instances running in the same virtual machine. Isolation and resource limits are applied using Windows Job Objects. Figure 4-9 represents a visual comparing managed instance and pools. This figure comes from the documentation at `https://docs.microsoft.com/en-us/azure/azure-sql/managed-instance/instance-pools-overview#architecture`.

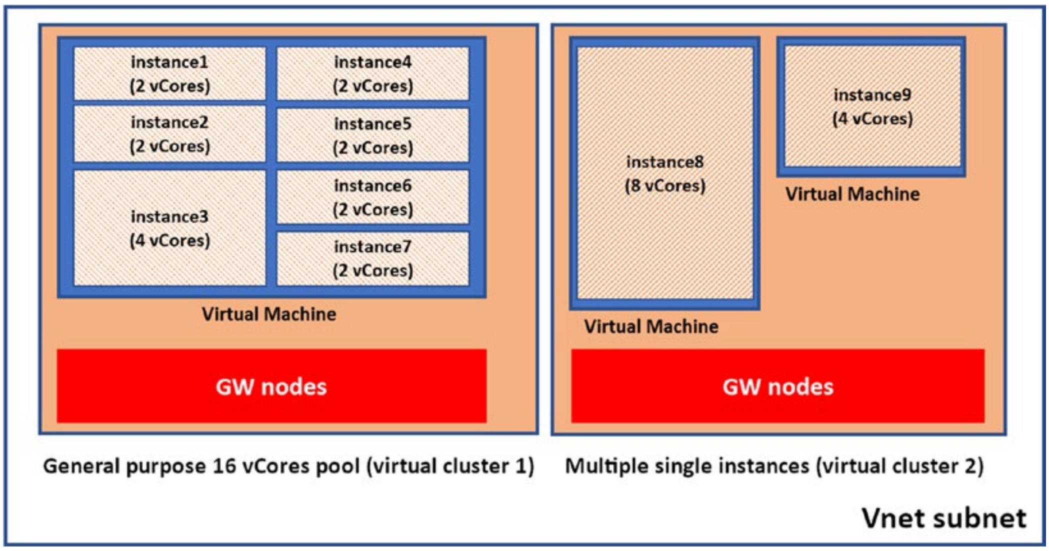
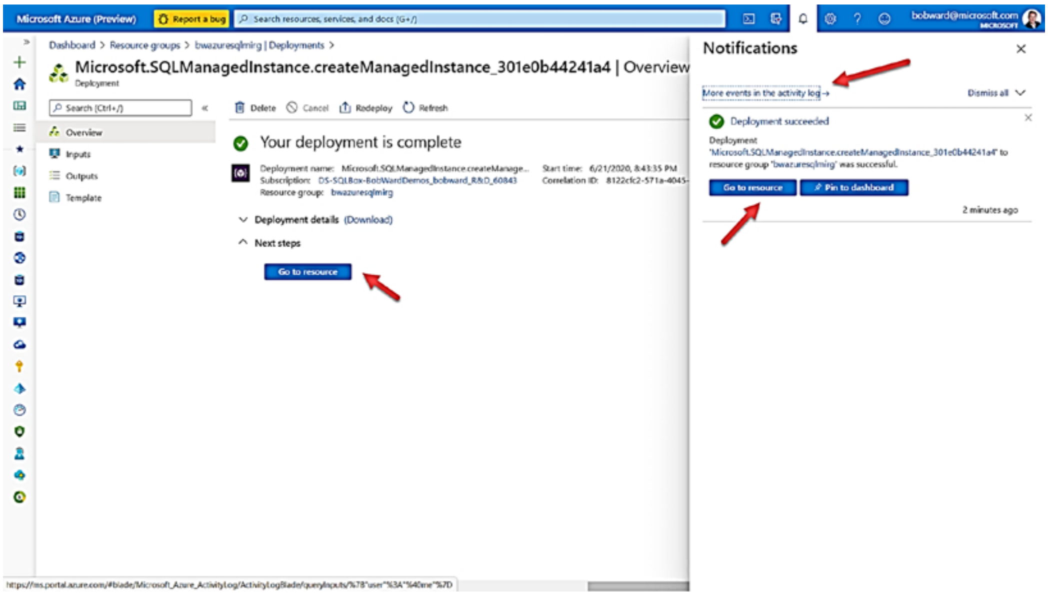


Figure 4-9. *The Managed Instance and pool architecture*

Connecting and Verifying Deployment

Once your deployment is complete, you can view the details of the deployment using the Activity Log like an Azure Virtual Machine as seen in Figure 4-10.



Microsoft.SQLManagedInstance.createManagedInstance_301e0b44241a4 | Overview
Deployment

Your deployment is complete

Deployment name: Microsoft.SQLManagedInstance.createManage...
Subscription: DS-SQLBox-BobWardDemos_bobward_R&D_60843
Resource group: bwazuresqlmirg
Start time: 6/21/2020, 8:43:35 PM
Correlation ID: 8122cfc2-571a-4045-

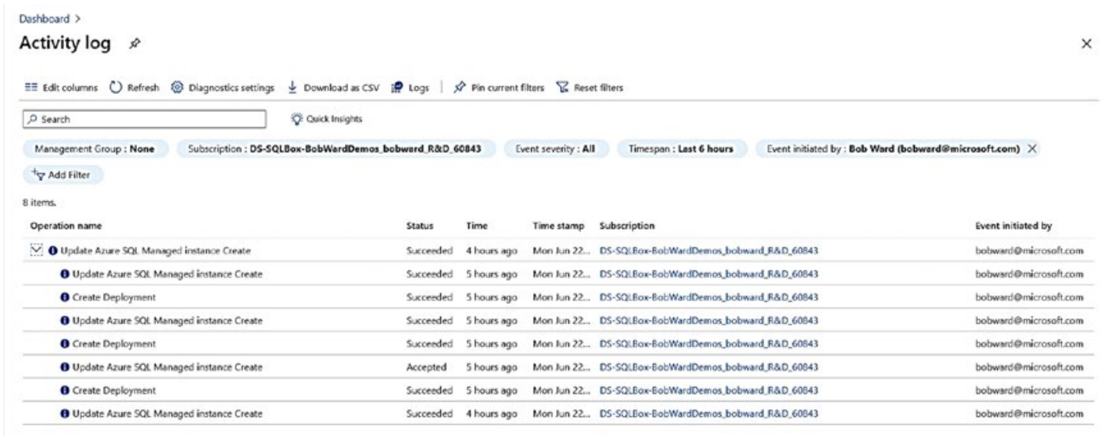
Deployment succeeded

Deployment 'Microsoft.SQLManagedInstance.createManagedInstance_301e0b44241a4' to resource group 'bwazuresqlmirg' was successful.

2 minutes ago

Figure 4-10. *Deployment complete for Managed Instance*

If you select More events in the activity log, you will see a screen like in Figure 4-11.



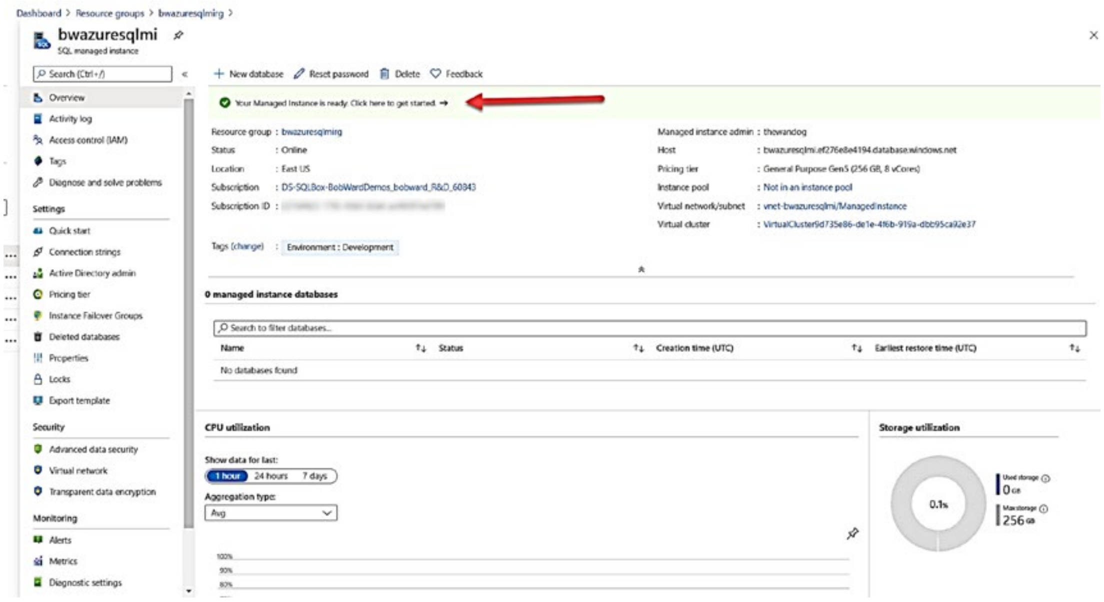
Management Group : None | Subscription : DS-SQLBox-BobWardDemos_bobward_R&D_60843 | Event severity : All | Timespan : Last 6 hours | Event initiated by : Bob Ward (bobward@microsoft.com)

8 items.

Operation name	Status	Time	Time stamp	Subscription	Event initiated by
Update Azure SQL Managed instance Create	Succeeded	4 hours ago	Mon Jun 22...	DS-SQLBox-BobWardDemos_bobward_R&D_60843	bobward@microsoft.com
Update Azure SQL Managed instance Create	Succeeded	5 hours ago	Mon Jun 22...	DS-SQLBox-BobWardDemos_bobward_R&D_60843	bobward@microsoft.com
Create Deployment	Succeeded	5 hours ago	Mon Jun 22...	DS-SQLBox-BobWardDemos_bobward_R&D_60843	bobward@microsoft.com
Update Azure SQL Managed instance Create	Succeeded	5 hours ago	Mon Jun 22...	DS-SQLBox-BobWardDemos_bobward_R&D_60843	bobward@microsoft.com
Create Deployment	Succeeded	5 hours ago	Mon Jun 22...	DS-SQLBox-BobWardDemos_bobward_R&D_60843	bobward@microsoft.com
Update Azure SQL Managed instance Create	Accepted	5 hours ago	Mon Jun 22...	DS-SQLBox-BobWardDemos_bobward_R&D_60843	bobward@microsoft.com
Create Deployment	Succeeded	5 hours ago	Mon Jun 22...	DS-SQLBox-BobWardDemos_bobward_R&D_60843	bobward@microsoft.com
Update Azure SQL Managed instance Create	Succeeded	4 hours ago	Mon Jun 22...	DS-SQLBox-BobWardDemos_bobward_R&D_60843	bobward@microsoft.com

Figure 4-11. *The activity log for the deployment for Managed Instance*

If you click **Go to Resource**, you will see a screen like in Figure 4-12.



Your Managed Instance is ready. Click here to get started. →

Resource group : bwazuresqlmirg
Status : Online
Location : East US
Subscription : DS-SQLBox-BobWardDemos_bobward_R&D_60843
Subscription ID : [redacted]

Managed instance admin : thewandog
Host : bwazuresqlmi.ef276e8e4194.database.windows.net
Pricing tier : General Purpose Gen5 (256 GB, 8 vCores)
Instance pool : Not in an instance pool
Virtual network/subnet : vnet-bwazuresqlmi/ManagedInstance
Virtual cluster : VirtualCluster9d735e86-de1e-4f6b-919a-dbb95ca92e37

Tags (change) : Environment : Development

0 managed instance databases

Search to filter databases...

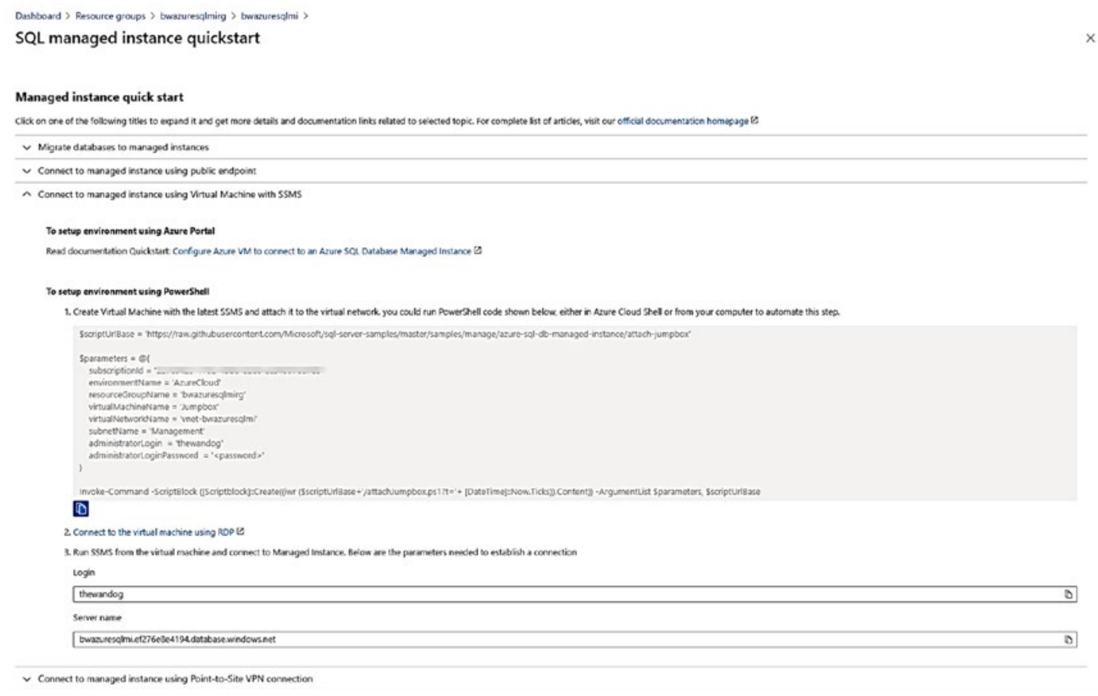
Name	Status	Creation time (UTC)	Earliest restore time (UTC)
No databases found			

Figure 4-12. *The Overview screen for Azure SQL Managed Instance*

One of the first steps you will want to do at this point is try and connect to the new instance and verify the deployment. I like to verify my SQL Server installations with a set of simple T-SQL queries, but I will show you other navigation details as well.

Connect to a Managed Instance

You can see at the top of Figure 4-12 guidance to get started for a Managed Instance. If you click this message, you will see choices like the following in Figure 4-13.



Dashboard > Resource groups > bwazuresqlmirg > bwazuresqlmi >

SQL managed instance quickstart

Managed instance quick start

Click on one of the following titles to expand it and get more details and documentation links related to selected topic. For complete list of articles, visit our official documentation homepage

- Migrate databases to managed instances
- Connect to managed instance using public endpoint
- Connect to managed instance using Virtual Machine with SSMS

To setup environment using Azure Portal

Read documentation Quickstart: Configure Azure VM to connect to an Azure SQL Database Managed Instance

To setup environment using PowerShell

1. Create Virtual Machine with the latest SSMS and attach it to the virtual network. you could run PowerShell code shown below, either in Azure Cloud Shell or from your computer to automate this step.

```
$scriptUrlBase = 'https://raw.githubusercontent.com/Microsoft/sql-server-samples/master/samples/manage/azure-sql-db-managed-instance/attach-jumpbox'

$parameters = @{
   subscriptionId = '[redacted]
   environmentName = 'AzureCloud'
   resourceGroupName = 'bwazuresqlmirg'
   virtualMachineName = 'Jumpbox'
   virtualNetworkName = 'vnet-bwazuresqlmi'
   subnetName = 'Management'
   administratorLogin  = 'thewandog'
   administratorLoginPassword  = '<password>'
}

Invoke-Command -ScriptBlock ([Scriptblock]::Create((iwr ($scriptUrlBase+'/attachJumpbox.ps1?t='+ [DateTime]::Now.Ticks)).Content)) -ArgumentList $parameters, $scriptUrlBase
```

2. Connect to the virtual machine using RDP
3. Run SSMS from the virtual machine and connect to Managed Instance. Below are the parameters needed to establish a connection

Login

thewandog

Server name

bwazuresqlmi.ef276e8e4194.database.windows.net

- Connect to managed instance using Point-to-Site VPN connection

Figure 4-13. *Connecting to Azure SQL Managed Instance with a VM*

The list on this screen points to guidance on how to perform various tasks for a newly deployed Azure SQL Managed Instance. Since I did not enable the Public endpoint for the deployed Managed Instance, I have expanded the option to connect through a Virtual Machine.

I will follow these steps by first using the provided PowerShell script and executing it from the Azure Cloud shell. I used the copy button , edited the script, to provide an admin password, and then pasted the script into an Azure Cloud Shell.

This virtual machine is deployed in the same virtual network as the Azure SQL Managed Instance. Therefore, you can RDP into this virtual machine and then use tools like SSMS (which is already installed in this VM) to connect to the Managed Instance. The screen shows you the DNS name to use to connect with SSMS (in my case, the server name to connect to is bwazuresqlmi.ef276e8e4194.database.windows.net).

Tip You can navigate to the newly created virtual machine by finding the Resource Group of the Managed Instance (which is bwazuresqlmirg in my example). The new VM is called **Jumpbox**. Navigate to the overview screen for this VM and select Connect and use the RDP file with a Remote Desktop Client. Once you use RDP to log in to the Windows Server, use SSMS to connect to the Managed Instance name with the SQL admin login and password you used during deployment.

The concept of using a virtual machine to connect to SQL Server for a Managed Instance is called a *jumpbox*. Jumpbox is the actual name of the VM created by the script provided as an example by Microsoft (but can be any name).

Figure 4-14 shows an example of SSMS connected to an Azure SQL Managed Instance.

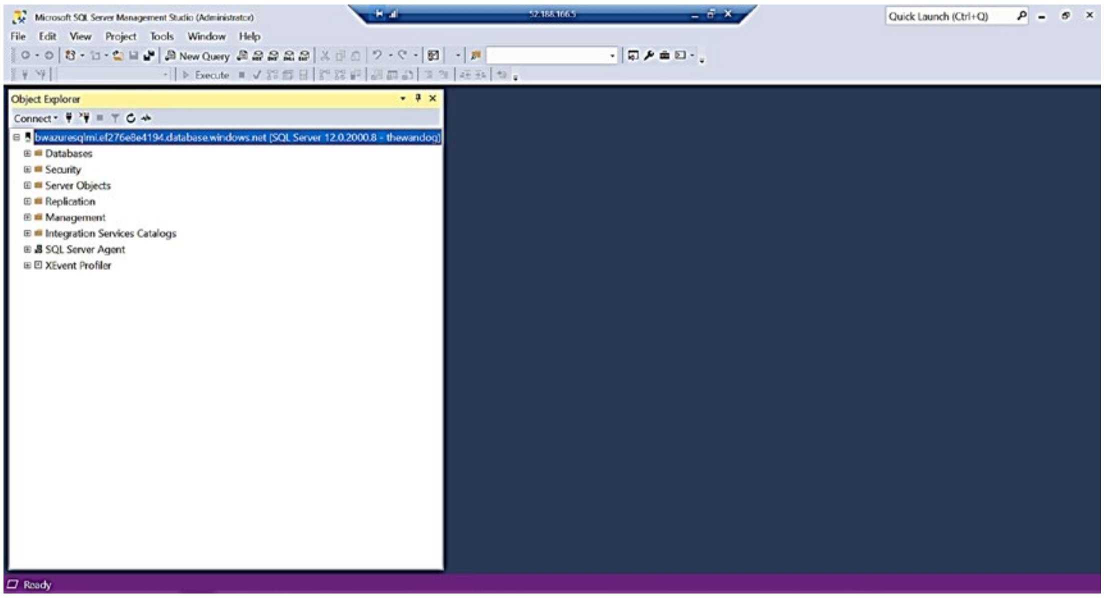

Microsoft SQL Server Management Studio (Administrator)
52.188.166.5
Quick Launch (Ctrl+Q)
File Edit View Project Tools Window Help
New Query
Execute
Object Explorer
Connect
bwazuresqlmi.ef276e8e4194.database.windows.net (SQL Server 12.0.2000.8 - thewandog)
Databases
Security
Server Objects
Replication
Management
Integration Services Catalogs
SQL Server Agent
XEvent Profiler
Ready

Figure 4-14. *SSMS connected to Azure SQL Managed Instance through a jumpbox*

If you want to connect from on-premises, you would need to either enable the public endpoint or connect your network to the Azure virtual network created.

Verify the Deployment

Notice that Object Explorer in SSMS looks almost identical to a SQL Server except for the Fully Qualified Domain Name (FQDN) for the server name.

To verify an installation of SQL Server, I often use a few techniques including running queries against system catalog views and DMVs and look at the ERRORLOG.

Examining the ERRORLOG

You do not have access to the filesystem for the virtual machine hosting the Managed Instance. To look at the ERRORLOG, we will need a tool like SSMS or T-SQL.

You can use Object Explorer in SSMS to view the ERRORLOG, but I prefer just T-SQL so can I execute `sp_readerrorlog` to look at the current ERRORLOG file. I must warn you that we dump all types of extra information in the ERRORLOG for a Managed Instance (yes even more than a SQL Server). My colleague Dimitri Furman wrote a blog post with some sample code to filter down the ERRORLOG for a Managed Instance. You can view this at https://techcommunity.microsoft.com/t5/datacat/azure-sql-db-managed-instance-sp-readmierrorlog/ba-p/305506.

There are a few key messages I look at startup in the ERRORLOG and found these in my Managed Instance:

```
SQL Server detected 1 sockets with 4 cores per socket and 8 logical
processors per socket, 8 total logical processors; using 8 logical
processors based on SQL Server licensing. This is an informational message;
no user action is required.
SQL Server is starting at normal priority base (=7). This is an
informational message only. No user action is required.
Detected 44645 MB of RAM. This is an informational message; no user action
is required.
```

I can see that eight logical processors were detected which is what I expect given I deployed an 8 vCore instance.

The memory detected is how much memory the SQL Server engine detects from the host or VM. You will see in several places in the book how Azure will use Windows Job Objects to limit the memory visible to SQL Server to enforce resource limits per service tier and vCores. You will find for this Managed Instance the job object will not allow SQL Server to access all the memory as shown here in the ERRORLOG. In fact, you should never rely on what the ERRORLOG shows but instead on the DMV sys.dm_os_job_object which I will show you how to use in the next section.

Verification Queries

Whenever I install a SQL Server, I typically use a few T-SQL queries as a *sanity check*. Let us look these and the results from the Managed Instance compared to SQL Server (I will not show the complete results of every query):

```
SELECT @@version
```

```
Microsoft SQL Azure (RTM) - 12.0.2000.8   May 15 2020 00:47:08   Copyright
(C) 2019 Microsoft Corporation
```

I explained in Chapter 1 of the book in describing the history of Azure SQL why v12 was a monumental moment for the service. Since that time, we have not changed the major version of 12.

Basically, the major version of Azure SQL Managed Instance has no meaning. It does not line up with any major version of SQL Server since Azure SQL Managed Instance is *versionless*. I will discuss more the concept of versionless in Chapter 5 of the book. Just know that Microsoft strives to keep instances and databases of Azure SQL up to date with all the right changes and fixes, so you do not worry about applying updates:

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

```
database_id  name        compatibility_level
1            master      150
2            tempdb      150
3            model       150
4            msdb        150
```

This looks normal (I have not created any user databases), except the mssqlsystemresource database is not listed as with SQL Server (it does exist as you can see it in the ERRORLOG). Notice the compatibility level is set to 150 which is the latest dbcompat level of SQL Server 2019:

```
SELECT name, object_id, type_desc FROM sys.objects
```

The results from this query are what you would normally think from a master database on a SQL Server with system tables at the top of the list. About 113 rows returned from this query in master which is very close to a SQL Server:

```
SELECT * FROM sys.dm_os_schedulers
```

Since we have deployed an 8 vCore Managed Instance, I would expect eight VISIBLE ONLINE schedulers which is the case. I also expect to see a few HIDDEN ONLINE schedulers which are there in the result of this query:

```
SELECT * FROM sys.dm_os_sys_info
```

This is a DMV that provides system information about SQL Server. I can see from the output of this DMV the number of CPUs detected, amount of memory detected, the target (max server memory), and total memory used by the engine and the actual worker thread max (1640 for 8 vCores), and that the conventional memory model is not used. I will talk more about locked pages in Chapter 5 of the book:

```
SELECT * FROM sys.dm_os_process_memory
```

This DMV shows OS-related memory information including whether locked or large pages are used (they are not for Azure SQL Managed Instance or Database). I normally just use this as a sanity check that enough memory is available for SQL Server in a VM or server:

```
SELECT * FROM sys.dm_exec_requests
```

This is one of the most common DMVs in the world to check the state of what is running on a SQL Server. I run this just to make sure all the normal background processes are running including LAZ WRITER, RECOVERY WRITER, LOCK MONITOR, and so on and that I can see active queries:

```
SELECT SERVERPROPERTY('EngineEdition')
```

Per the documentation at https://docs.microsoft.com/en-us/sql/t-sql/functions/serverproperty-transact-sql, the value of 8 is for a Managed Instance.

There are also two new DMVs specific to Azure not found in SQL Server:

```
SELECT * FROM sys.dm_user_db_resource_governance
```

This DMV is really intended to show you resource limits for a specific Azure SQL Database, but it also works for Managed Instance (you will see a row for all databases including system databases except for tempdb). You can view limits like memory, max storage, log rates, and so on. You can read the documentation for this DMV at https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-user-db-resource-governor-azure-sql-database?view=azuresqldb-current. Note that the docs say this is mostly for internal use, which means it might change in the future:

Note There is an undocumented DMV called **sys.dm_instance_resource_governance** which shows resource limits at the instance level.

```
SELECT * FROM sys.dm_os_job_object
```

This is a DMV specific to Azure (although I tested it and it works for a SQL Server, but you can't rely on the results there because it is not applicable) that shows resource limits Azure applies to a Managed Instance using Windows Job Objects. The specific column I look at is **memory_limit_mb** which shows me the true amount of memory the Managed Instance has access to. I talked about Windows Job Objects in the preceding section "Implementation Details."

You may be asking at this point what is so special about Managed Instance, since from the perspective of using a tool like SSMS, it feels like a SQL Server running in an Azure Virtual Machine. This is the point of a Managed Instance. We want you to have the feel of a SQL Server instance, but not worry about the details you might have to consider with a virtual machine. And since Azure SQL Managed Instance (MI) is a PaaS service, you will see the benefits of using MI, especially when it comes to a versionless SQL Server, predictable performance, and built-in high availability and disaster recovery.

Migrating to Azure SQL Managed Instance

As part of deploying an Azure SQL Managed Instance, you may be migrating existing databases. The process to migrate should include assessment and planning, the actual migration, application changes, and post-migration optimization.

Let us look at each of these aspects to the migration process.

Assessment and Planning

An assessment for migration includes analyzing any problems that might occur for your migration depending on what the source of your migration and your preferred Managed Instance deployment option. This is when you use the details I described with pre-deployment planning earlier in this chapter.

Most users migrating to Azure SQL Managed Instance are coming from an existing SQL Server installation. However, it is possible to migrate from other data platforms including Oracle. Check out this blog post on how to migrate from Oracle to Azure SQL Managed Instance, `https://techcommunity.microsoft.com/t5/microsoft-data-migration/migrate-your-oracle-database-to-azure-sql-database-managed/ba-p/368750`.

The key to getting an entire migration project started is to use **Azure Migrate** at `https://azure.microsoft.com/en-us/services/azure-migrate/`. From there, you can start an Azure Migrate project and use the Data Migration Assistant (DMA) tool (`https://docs.microsoft.com/en-us/sql/dma/dma-overview`) and Data Migration Service (DMS) (`https://azure.microsoft.com/en-us/services/database-migration/`).

I could show you a ton of details here and walk you through an example. But I have something better. My colleague Anna Thomas developed an entire workshop complete with slides and exercises at `https://github.com/microsoft/sqlworkshops-sqlg2c/tree/master/sqlgroundtocloud` (Modules 4 and 5). Anna walks you through everything to see how DMA and DMS work to perform an assessment and migration. You have the choice to migrate all the databases for the instance or one or more databases incrementally.

Note Two important things you should consider as you migrate are in the documentation, and I want to highlight them here.

When you are migrating a database protected by Transparent Data Encryption to a managed instance using native restore option, the corresponding certificate from the on-premises or Azure VM SQL Server needs to be migrated before database restore.

Restore of system databases is not supported. **To migrate instance-level objects** (stored in master or msdb databases), we recommend to script them out and run T-SQL scripts on the destination instance.

One last important point: consider using a database compatibility level that matches your current SQL Server installation and then move later to the latest compat level. Learn more about dbcompat at `https://aka.ms/dbcompat`.

Migration

To perform an actual migration of an existing SQL Server instance (the entire instance or just a database), Azure SQL Managed Instance provides a great capability to make this faster and better: the ability to restore a database from a backup of SQL Server.

This allows you to perform an *offline* migration (restore a full database backup) while the application is down and then connect back again to the Managed Instance when the restore finishes.

The Database Migration Service (DMS) specifically allows for an online migration using technology based on SQL Server Log Shipping to restore a full backup and then a series of log backups until you are ready for the migration cutover. Anna's workshop goes through both options.

In addition, you can watch me on the Microsoft Mechanics channel at `https://youtu.be/P_4EaqVR5PI` and go through the entire migration process including a demo of migration to a Managed Instance. One of the aspects of using DMA I love is the SKU recommendation PowerShell script. This script analyzes your current SQL Server workload to guide you on Azure SQL Managed Instance deployment choices. Read more at `https://docs.microsoft.com/en-us/sql/dma/dma-sku-recommend-sql-db?view=sql-server-ver15`.

Note Even though I am the person interviewed for this video and demo, the real heroes behind the scenes are the migration team, including folks like Venkata Raj Pochiraju and Sreraman Narasimhan.

Application Changes

One of the best stories for migration is the minimal changes required by your application after migrating to Managed Instance. The most basic change for you to make is the connection string to use the new server name. You may also have to change the authentication method (SQL Authentication or Azure Active Directory). But other than that, provided you are using all the features supported by Managed Instance, these may be the only changes you need to get your application up and running.

Post Migration

After you have migrated your database(s) and made the necessary application changes, you may want to make some configuration changes or adjustments for security, performance, and availability.

Here is the good news. The rest of the book is devoted to these topics, so you have the resources you need to make any changes to go bigger with Azure.

Deploying an Azure SQL Database

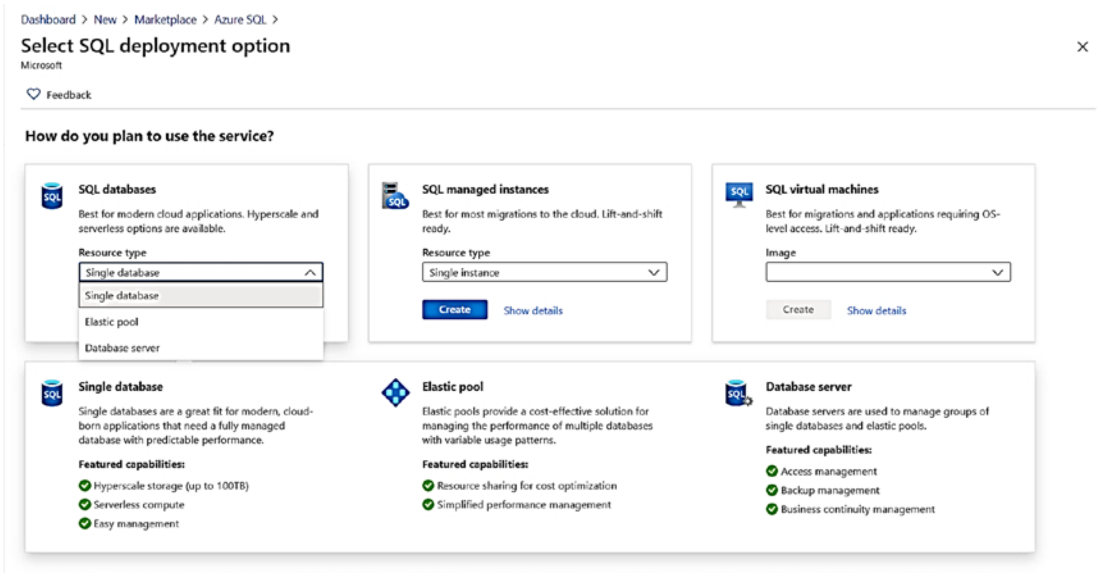
Deploying an Azure SQL Database is both different and like Azure SQL Managed Instance. The experience is similar as you will use the Azure Portal or CLI (or even T-SQL) to deploy a database but different since you are…well deploying a database, not an instance of SQL Server. Since you are deploying just a database, you get more options as I have described in this book, including Serverless and Hyperscale.

The basic process to deploy Azure SQL Database is

- Decide to deploy a single database or elastic pool.
- Choose a Resource Group and Region.
- Choose an existing or new logical database server.
- Choose your purchasing model, compute model, service tier, and size.
- Optionally supply other configuration choices.
- Deploy it!

Deployment and Options

To get started, I will use the **Azure SQL** screen to deploy by searching for Azure SQL in the marketplace. With a screen like in Figure 4-15, I can choose to deploy a Single database, Elastic pool, or Database server. I will show you how to deploy a Single database which will allow me to also deploy a Database server (and I will describe what and why you need a Database server).


Dashboard > New > Marketplace > Azure SQL >

Select SQL deployment option

Microsoft

Feedback

How do you plan to use the service?

SQL databases

Best for modern cloud applications. Hyperscale and serverless options are available.

Resource type: Single database

Single database

Elastic pool

Database server

SQL managed instances

Best for most migrations to the cloud. Lift-and-shift ready.

Resource type: Single instance

Create | Show details

SQL virtual machines

Best for migrations and applications requiring OS-level access. Lift-and-shift ready.

Image:

Create | Show details

Single database

Single databases are a great fit for modern, cloud-born applications that need a fully managed database with predictable performance.

Featured capabilities:
- Hyperscale storage (up to 100TB)
- Serverless compute
- Easy management

Elastic pool

Elastic pools provide a cost-effective solution for managing the performance of multiple databases with variable usage patterns.

Featured capabilities:
- Resource sharing for cost optimization
- Simplified performance management

Database server

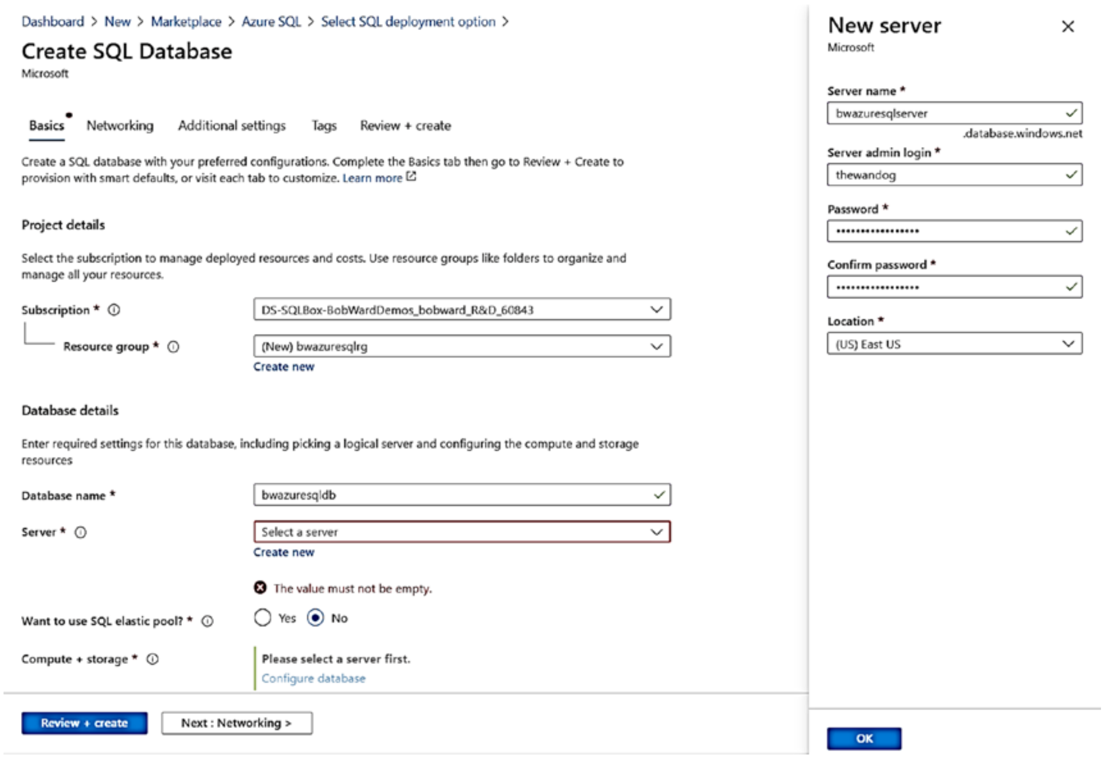
Database servers are used to manage groups of single databases and elastic pools.

Featured capabilities:
- Access management
- Backup management
- Business continuity management

Figure 4-15. *Using Azure SQL to choose to deploy an Azure SQL Database*

Basics

If you choose Single database, you are presented with a Basics screen like Azure SQL Managed Instance as seen in Figure 4-16.


Dashboard > New > Marketplace > Azure SQL > Select SQL deployment option >

Create SQL Database
Microsoft

Basics • Networking Additional settings Tags Review + create

Create a SQL database with your preferred configurations. Complete the Basics tab then go to Review + Create to provision with smart defaults, or visit each tab to customize. Learn more

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *: DS-SQLBox-BobWardDemos_bobward_R&D_60843
Resource group *: (New) bwazuresqlrg
Create new

Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name *: bwazuresqldb
Server *: Select a server
Create new
The value must not be empty.
Want to use SQL elastic pool? *: Yes [✓ No]
Compute + storage *: Please select a server first.
Configure database

Review + create | Next : Networking >

New server
Microsoft

Server name *: bwazuresqlserver .database.windows.net
Server admin login *: thewandog
Password *: ••••••••••••••••
Confirm password *: ••••••••••••••••
Location *: (US) East US

OK

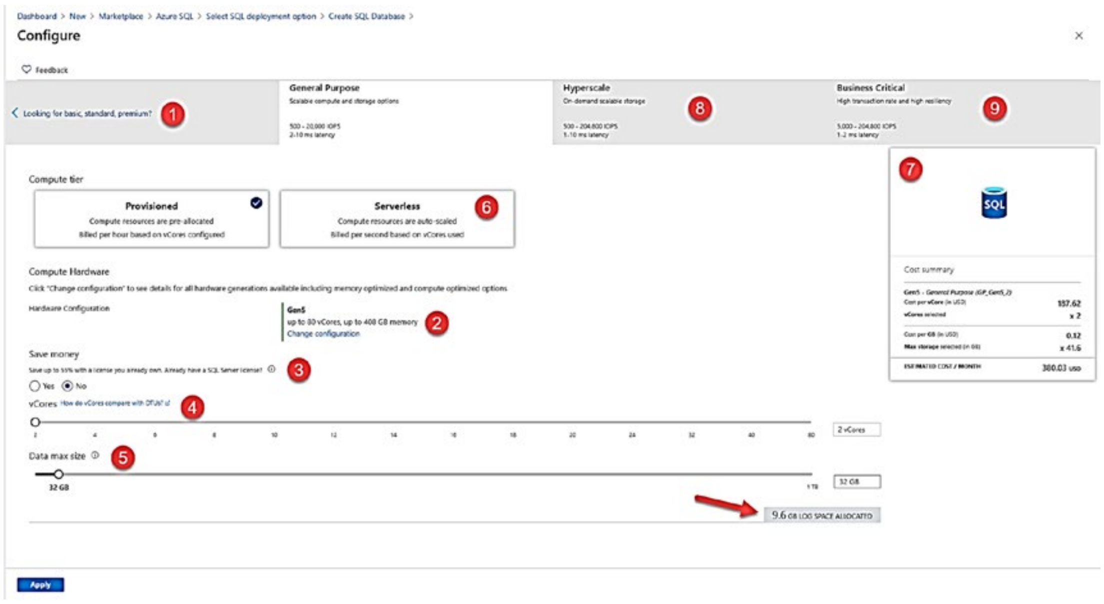
Figure 4-16. *Defining a Database server as part of Azure SQL deployment*

You can see from my screen that I have already created a new Resource Group, defined a Database name, and chose to create a new Database server. You may be wondering why you need a Database server when the promise of Azure SQL Database is "you own the database; Azure will manage everything else."

A **Database server**, also known as a *logical server*, is a collection of metadata stored in the Azure infrastructure used to organize one or more Azure SQL databases. It is not a single SQL Server instance on a physical server. A Database server contains a **logical master database** just like a true SQL Server instance. Notice the region is associated with the logical server, not the database. Any database created for the logical server will be hosted in the region of the server. All connectivity and networking will be associated with the logical server. In fact, you could create a logical server first, connect to that server, and use T-SQL CREATE DATABASE to create Azure SQL databases. The login and password you supply for the logical server becomes a login in the logical master database which is a server-level principal who is effectively a *server admin* for all databases. In Modules 6 and 7, I will show you how to integrate Azure Active Directory (AAD) for an admin login.

Once you click **OK** for the logical server (it will be created as part of the deployment), you can choose if you want to make this part of an elastic pool and select your options for purchasing model, service tier, and size. I will not spend a lot of time in the book discussing the details of how to create and manage an elastic pool. I recommend you use this documentation to learn and go further with using elastic pools, `https://docs.microsoft.com/en-us/azure/azure-sql/database/elastic-pool-overview`.

Figure 4-17 shows an example of selecting **Configure database** to see your deployment options.



Dashboard > New > Marketplace > Azure SQL > Select SQL deployment option > Create SQL Database >

Configure

Looking for basic, standard, premium?

General Purpose
Scalable compute and storage options
500 - 20,000 IOPS
2-10 ms latency

Hyperscale
On-demand scalable storage
500 - 204,800 IOPS
1-10 ms latency

Business Critical
High transaction rate and high resiliency
5,000 - 204,800 IOPS
1-2 ms latency

Compute tier
[✓ Provisioned] Compute resources are pre-allocated. Billed per hour based on vCores configured
Serverless: Compute resources are auto-scaled. Billed per second based on vCores used

Compute Hardware
Click "Change configuration" to see details for all hardware generations available including memory optimized and compute optimized options.
Hardware Configuration: Gen5, up to 80 vCores, up to 408 GB memory

Save money
Save up to 55% with a license you already own. Already have a SQL Server license?
Yes [✓ No]

vCores: 2 vCores

Data max size: 32 GB

9.6 GB LOG SPACE ALLOCATED

Cost summary	
Gen5 - General Purpose (GP_Gen5_2)	
Cost per vCore (in USD)	187.62
vCores selected	x 2
Cost per GB (in USD)	0.12
Max storage selected (in GB)	x 41.6
ESTIMATED COST / MONTH	380.03 USD

Figure 4-17. *Azure SQL purchasing, compute, service tier, and size options*

Let us look at the various options on this screen, what they mean, and how they affect your deployment:

1. The default *purchasing model* is vCore. You can select on this screen the ability to choose the DTU purchase model. While I will not cover details about deploying the DTU model, you can read about this option at `https://docs.microsoft.com/en-us/azure/azure-sql/database/service-tiers-dtu`.

2. The default *hardware generation* for the vCore model is Gen5. As I described in pre-deployment planning, choosing Change configuration allows you to pick other hardware generations such as Fsv2- and M-series.
3. Just like a Managed Instance, you can choose to apply your existing SQL Server license with Azure Hybrid Benefit (AHB).
4. Just like a Managed Instance, you have a slider bar to choose the number of vCores for your deployment. You will notice that for Azure SQL Database, you have more granular choices for vCores.
5. Just like a Managed Instance, you have a slider bar for Data max size. The more vCores you choose, the larger the max size. This max size is the maximum size of the single database file hosting your database. Notice below the size is a number for LOG SPACE ALLOCATED. You are given for free a maximum transaction log size that is 30% over your data max size.

Note If you are worried this may not be enough, we continuously back up the transaction log and have Accelerated Database Recovery enabled by default (which means a long active transaction does not hold up log truncation). As much as I tried, I never ran out of log space in my testing with Azure SQL Database.

---

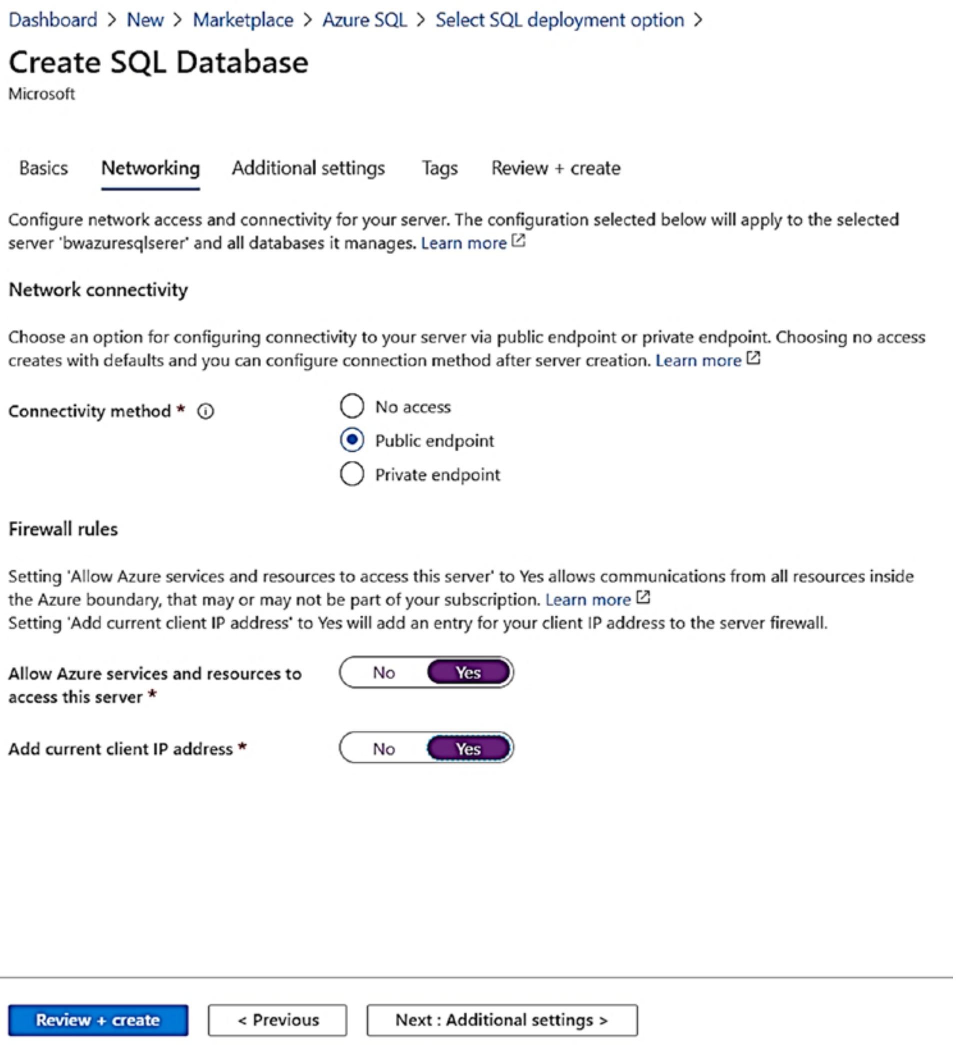
6. Provisioned is the default *Compute model*. For the General Purpose service tier, you have the choice of the *Serverless* model. I will show you Serverless options in the section titled “Deploying Serverless” later in this chapter.
7. As you make choices, the portal will show you estimated costs per month broken out by costs for vCores and storage.
8. Besides General Purpose, you also have the choice to deploy a Hyperscale service tier. I will show you the process of deploying Hyperscale in the following section titled “Deploying Hyperscale.”
9. You also have the choice to choose the Business Critical service tier. I will show you the process of deploying a Business Critical database in the following section titled “Deploying Business Critical.”

I know there are many choices which is one of the benefits of Azure SQL Database but does require some thinking for your requirements. Thankfully, changing these options or resizing your Azure SQL Database is flexible and fast (the exception is Hyperscale).

I will leave the default of General Purpose, 2 vCore, 32Gb max data size and click **Apply**.

Networking

Click **Next: Networking** > to see your choices for connectivity and network security as seen in Figure 4-18.



Configure network access and connectivity for your server. The configuration selected below will apply to the selected server 'bwazuresqlserer' and all databases it manages. Learn more

Network connectivity

Choose an option for configuring connectivity to your server via public endpoint or private endpoint. Choosing no access creates with defaults and you can configure connection method after server creation. Learn more

Connectivity method * : No access [✓ Public endpoint] Private endpoint

Firewall rules

Setting 'Allow Azure services and resources to access this server' to Yes allows communications from all resources inside the Azure boundary, that may or may not be part of your subscription. Learn more
Setting 'Add current client IP address' to Yes will add an entry for your client IP address to the server firewall.

Allow Azure services and resources to access this server * : No [✓ Yes]

Add current client IP address * : No [✓ Yes]

Figure 4-18. Networking choices when deploying an Azure SQL Database

Unlike Azure SQL Managed Instance, Azure SQL Database is not part of a virtual network. You have three choices during deployment:

No access – Deploy the database, but do not allow any connectivity until you are ready to make your choice.

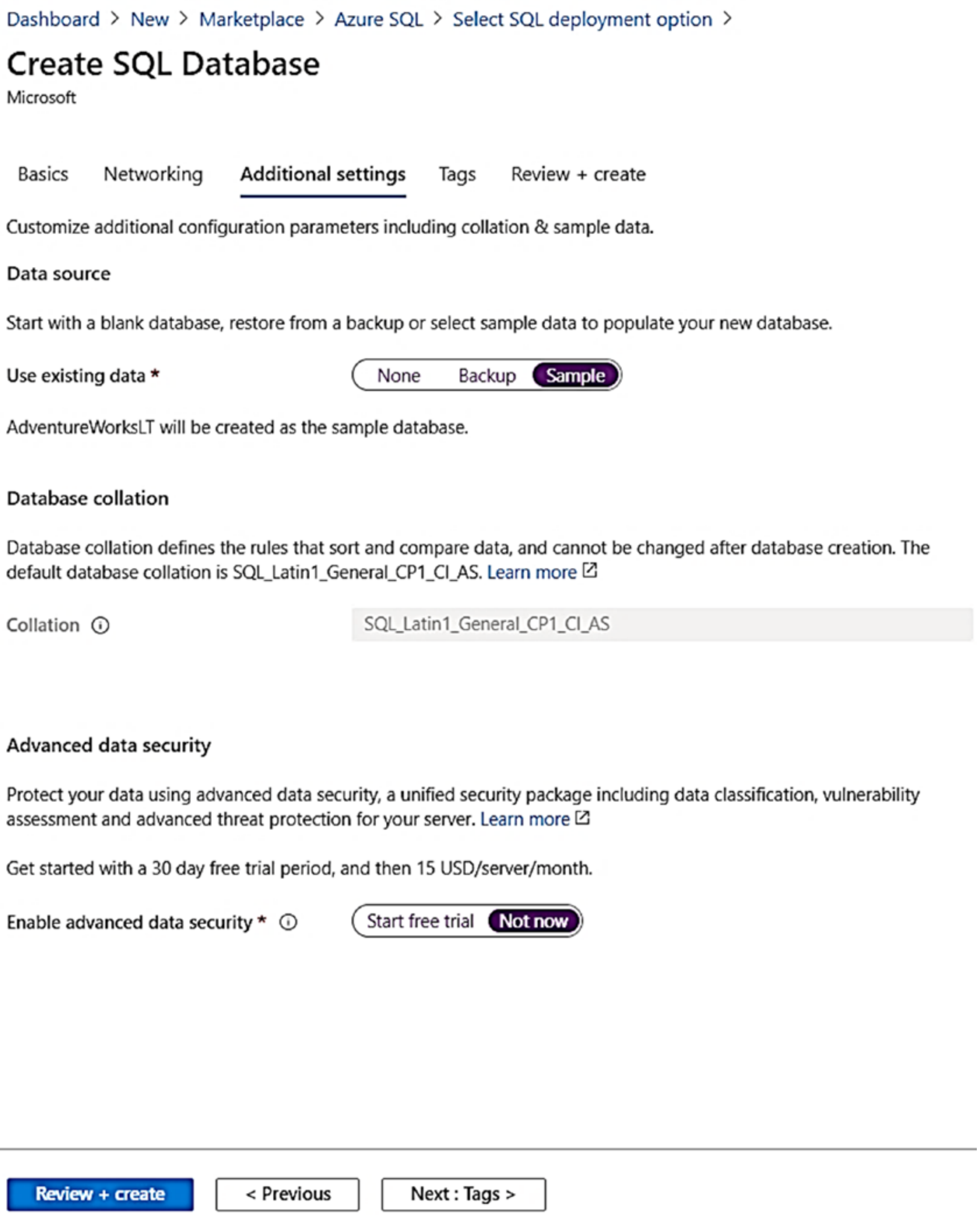
Public endpoint – Expose connectivity of the logical server and/or database to the public within Azure or to the Internet (or both).

Private endpoint – This is a new addition to Azure SQL Database to make it very secure. This allows you to deny public access to your server and/or database and only allows private connectivity within defined virtual networks in and outside of Azure.

For now, I will select Public endpoint and set **Allow Azure services and resources access this server** to Yes and **Add current client IP address** to Yes. This allows me to deploy an Azure Virtual Machine and connect to this database or to connect with SQL client tools on the client computer where I am currently deploying the browser. My client IP address will be added to a *firewall rule* to connect to the logical server associated with this database. I will show you how to tighten up the security of this model in Chapters 6 and 7 of the book.

Additional Settings

Click **Next: Additional settings** > to see more options for the deployment. Figure 4-19 shows these additional options.


Dashboard > New > Marketplace > Azure SQL > Select SQL deployment option >

Create SQL Database

Microsoft

Basics | Networking | **Additional settings** | Tags | Review + create

Customize additional configuration parameters including collation & sample data.

Data source

Start with a blank database, restore from a backup or select sample data to populate your new database.

Use existing data * None Backup [✓ Sample]

AdventureWorksLT will be created as the sample database.

Database collation

Database collation defines the rules that sort and compare data, and cannot be changed after database creation. The default database collation is SQL_Latin1_General_CP1_CI_AS. Learn more

Collation: SQL_Latin1_General_CP1_CI_AS

Advanced data security

Protect your data using advanced data security, a unified security package including data classification, vulnerability assessment and advanced threat protection for your server. Learn more

Get started with a 30 day free trial period, and then 15 USD/server/month.

Enable advanced data security * Start free trial [✓ Not now]

Review + create | < Previous | Next : Tags >

Figure 4-19. *Additional settings for the deployment of Azure SQL Database*

Your first choice is to either create a blank database or create a database based on a backup of a geo-replicated Azure SQL Database or from the sample AdventureWorksLT (LT stands for light). You can learn more about how to restore from a geo-replicated backup at `https://docs.microsoft.com/en-us/azure/azure-sql/database/recovery-using-backups#geo-restore`. I will choose the sample AdventureWorksLT database because I want to show some demonstrations of other capabilities using that database later in the book.

Your next choice is *Database collation*. Since I chose a sample database, the collation is already decided. For a new blank database, it is important to choose this during deployment because you cannot change it later.

The final choice is to enable *Advanced Data Security*. I will not enable this for now and show you more about this capability in Chapters 6 and 7 of the book.

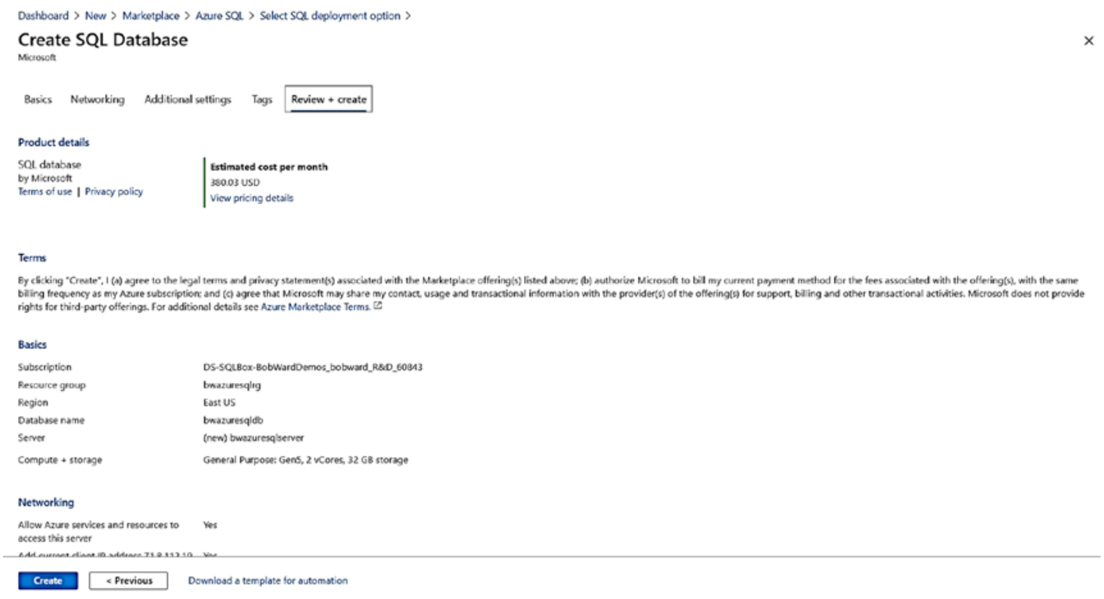
Tags

Click **Next: Tags** > to define a tag for the deployment. Just like with Azure Virtual Machine and Managed Instance, I will use a Name = Environment and Value = Development. In this case, I will leave the resources selected for both SQL Database and SQL database server.

Note After you deploy, you can now search for Tags for your description and see all your resources that are "for development purposes."

Deploy It!

Click **Next: Review + create** > to see the final validation screen as seen in Figure 4-20.


Dashboard > New > Marketplace > Azure SQL > Select SQL deployment option >
Create SQL Database
Microsoft
Basics
Networking
Additional settings
Tags
Review + create
Create
< Previous
Download a template for automation

Product details

SQL database
by Microsoft
Terms of use | Privacy policy

Estimated cost per month
380.03 USD
View pricing details

Terms

By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. For additional details see Azure Marketplace Terms.

Basics

Subscription	DS-SQLBox-BobWardDemos_bobward_R&D_60843
Resource group	bwazuresqlrg
Region	East US
Database name	bwazuresqldb
Server	(new) bwazuresqlserver
Compute + storage	General Purpose: Gen5, 2 vCores, 32 GB storage

Networking

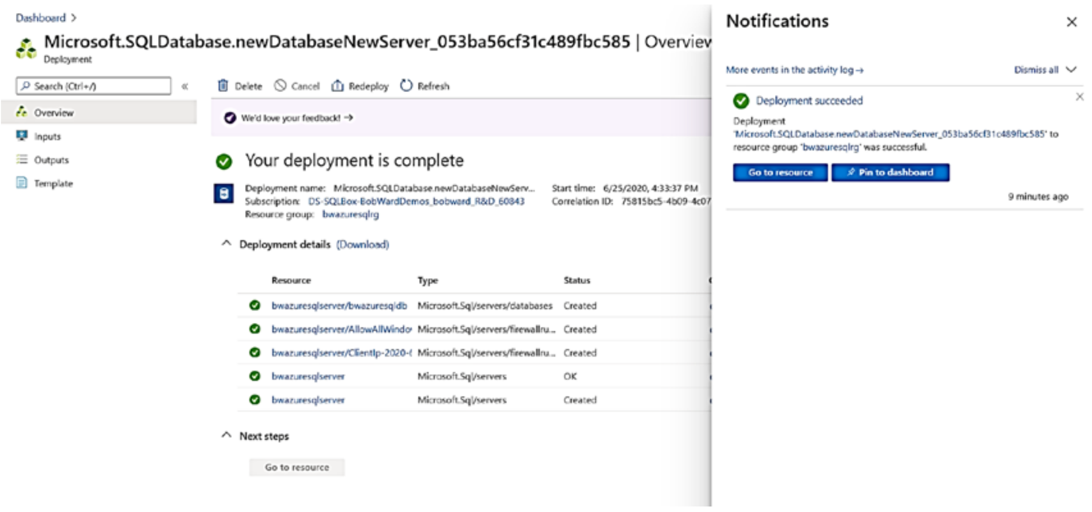
Allow Azure services and resources to access this server	Yes
Add current client IP address 71.9.112.10	Yes

Figure 4-20. *Final screen before deployment of Azure SQL Database*

Just like with Managed Instance, you can see estimated costs, Terms of Use, privacy policy, a review of your choices, and the ability to download an Azure template.

Click **Create** to see the deployment take off. Just like with Managed Instance, if you do not leave this screen, you will see a progress of your deployment in both the Notifications area of the portal and on your main screen.

There is no warning of time to deploy like with Managed Instance, because in most cases, the deployment should finish quickly. In my example, the deployment only took a matter of minutes as shown in Figure 4-21.



Microsoft.SQLDatabase.newDatabaseNewServer_053ba56cf31c489fbc585 | Overview
Deployment

Your deployment is complete

Deployment name: Microsoft.SQLDatabase.newDatabaseNewServ...
Subscription: DS-SQLBox-BobWardDemos_bobward_R&D_60843
Resource group: bwazuresqlrg
Start time: 6/25/2020, 4:33:37 PM
Correlation ID: 75815bc5-4b09-4c07

Deployment details (Download)

Resource	Type	Status
bwazuresqlserver/bwazuresqldb	Microsoft.Sql/servers/databases	Created
bwazuresqlserver/AllowAllWindo	Microsoft.Sql/servers/firewallru...	Created
bwazuresqlserver/ClientIp-2020-(	Microsoft.Sql/servers/firewallru...	Created
bwazuresqlserver	Microsoft.Sql/servers	OK
bwazuresqlserver	Microsoft.Sql/servers	Created

Deployment succeeded

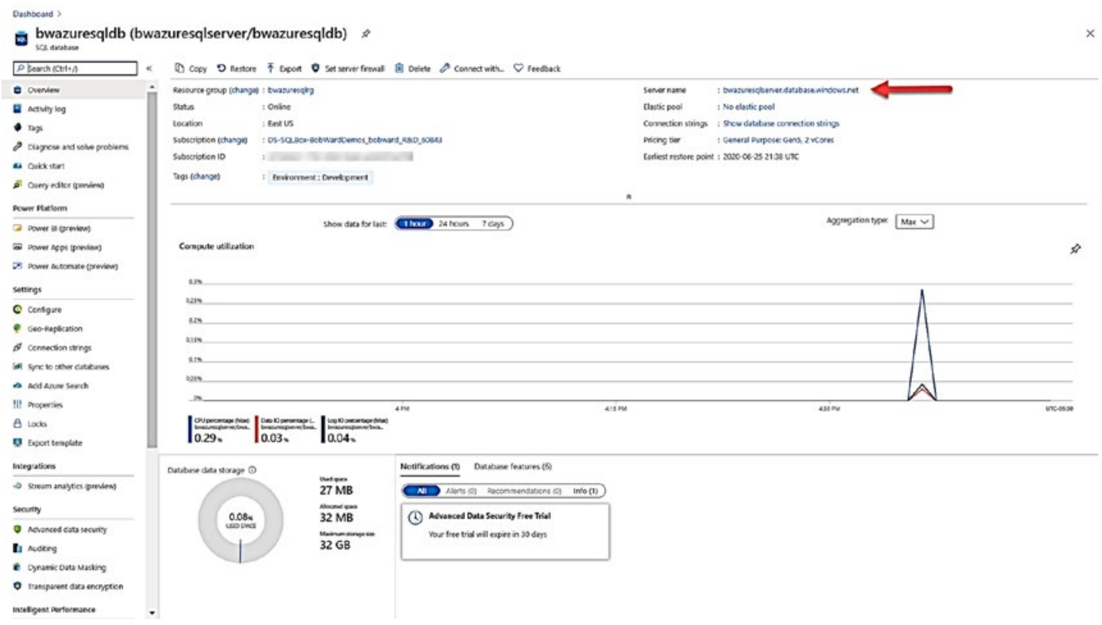
Deployment 'Microsoft.SQLDatabase.newDatabaseNewServer_053ba56cf31c489fbc585' to resource group 'bwazuresqlrg' was successful.

Figure 4-21. *A completed deployment of Azure SQL Database*

Tip I have found that I can connect to the logical server and use the database being deployed even before I get signaled the deployment is complete.

Just like a Managed Instance deployment, you can click More events in the activity log to see the sequence of deploying all resources.

Click Go to resource to see the Overview screen on the database like Figure 4-22.



Resource group (change) : bwazuresqlrg
Status : Online
Location : East US
Subscription (change) : DS-SQLBox-BobWardDemos_bobward_R&D_60843
Subscription ID : [redacted]
Tags (change) : Environment : Development

Server name : bwazuresqlserver.database.windows.net
Elastic pool : No elastic pool
Connection strings : Show database connection strings
Pricing tier : General Purpose: Gen5, 2 vCores
Earliest restore point : 2020-06-25 21:38 UTC

Used space 27 MB
Allocated space 32 MB
Maximum storage size 32 GB

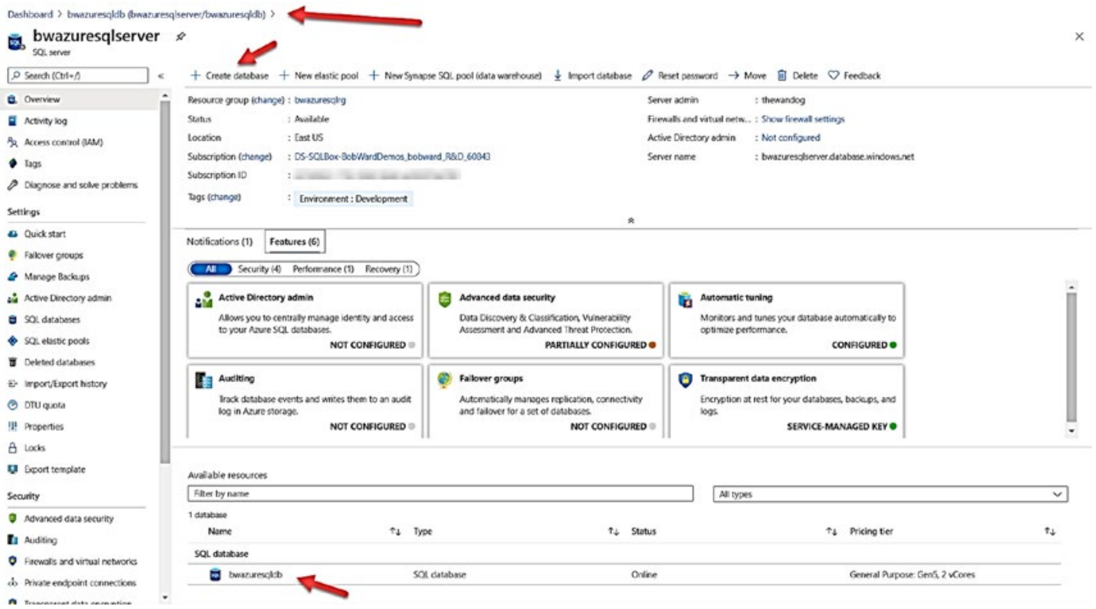
Advanced Data Security Free Trial
Your free trial will expire in 30 days

Figure 4-22. *The Overview screen of an Azure SQL Database*

Just like with a Virtual Machine and Managed Instance, the portal shows a Resource Menu, Command Bar, Working Pane, and Monitoring Pane. While this looks like other Azure resources, most of this information is specific to Azure SQL Database. We will use many of these options throughout the rest of the chapter as you explore security, performance, availability, and other features.

Let us see the experience of deploying an Azure SQL Database Business Critical, Hyperscale, and Serverless database on the *same logical server*. In the Working Pane, click the Server Name.

This is the Overview screen for the Database Server as seen in Figure 4-23.



Resource group (change) : bwazuresqlrg
Status : Available
Location : East US
Subscription (change) : DS-SQLBox-BobWardDemos_bobward_R&D_60043
Subscription ID : [redacted]
Tags (change) : Environment : Development

Server admin : thewandog
Firewalls and virtual netw... : Show firewall settings
Active Directory admin : Not configured
Server name : bwazuresqlserver.database.windows.net

Active Directory admin
Allows you to centrally manage identity and access to your Azure SQL databases.
NOT CONFIGURED

Advanced data security
Data Discovery & Classification, Vulnerability Assessment and Advanced Threat Protection.
PARTIALLY CONFIGURED

Automatic tuning
Monitors and tunes your database automatically to optimize performance.
CONFIGURED

Auditing
Track database events and writes them to an audit log in Azure storage.
NOT CONFIGURED

Failover groups
Automatically manages replication, connectivity and failover for a set of databases.
NOT CONFIGURED

Transparent data encryption
Encryption at rest for your databases, backups, and logs.
SERVICE-MANAGED KEY

1 database

Name	Type	Status	Pricing tier
SQL database			
bwazuresqldb	SQL database	Online	General Purpose: Gen5, 2 vCores

Figure 4-23. *Overview of an Azure SQL Database Server*

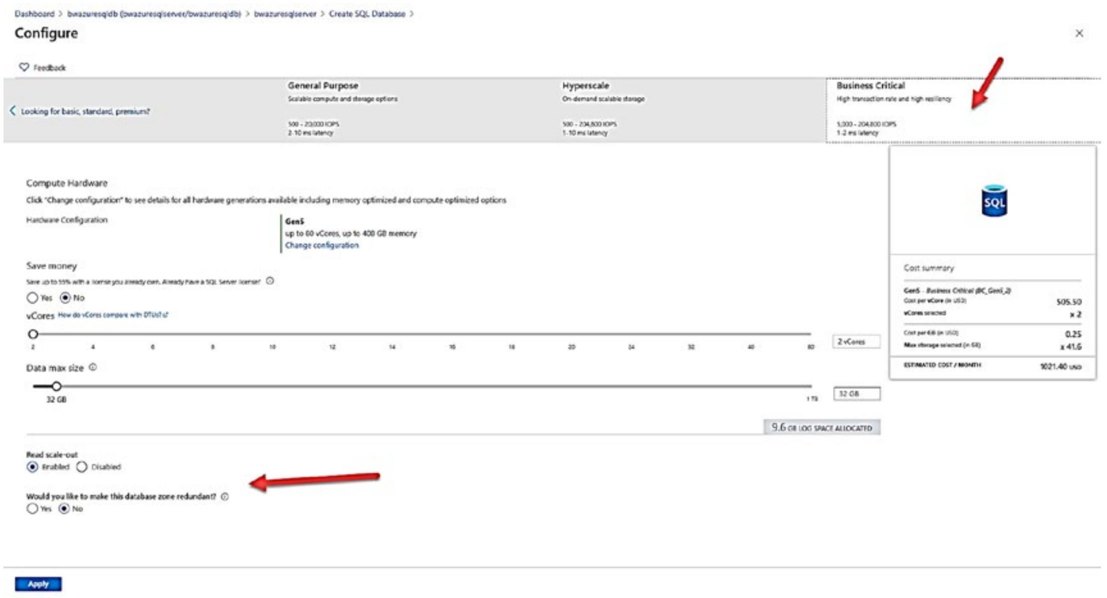
Notice at the bottom the database we just deployed. I also want to stop and point out a key feature of the portal called *breadcrumbs*. You were brought to the overview screen of the database server by selecting it from the overview screen of the database. The breadcrumbs "show you where you came from" and allow you to *navigate* to a specific resource in the portal.

Click **Create database** from the command bar.

Note The following sections for Business Critical, Serverless, and Hyperscale require costs if you leave these deployments active. You can choose to go through and deploy these or just follow along.

Deploying Business Critical

Notice now you are brought to a Basics screen to create a new database, but the resource group and server are already selected. Click configure database and choose the Business Critical service tier. Notice some new choices as seen in Figure 4-24.



Dashboard > bwazuresqldb (bwazuresqlserver/bwazuresqldb) > bwazuresqlserver > Create SQL Database >

Configure

Feedback

Looking for basic, standard, premium?

General Purpose
Scalable compute and storage options
500 - 20,000 IOPS
2-10 ms latency

Hyperscale
On-demand scalable storage
500 - 204,800 IOPS
1-10 ms latency

Business Critical
High transaction rate and high resiliency
5,000 - 204,800 IOPS
1-2 ms latency

Compute Hardware
Click "Change configuration" to see details for all hardware generations available including memory optimized and compute optimized options

Hardware Configuration: Gen5
up to 80 vCores, up to 408 GB memory
Change configuration

Save money
Save up to 55% with a license you already own. Already have a SQL Server license?
Yes [✓ No]

vCores How do vCores compare with DTUs?
2 4 6 8 10 12 14 16 18 20 24 32 40 80
2 vCores

Data max size
32 GB … 1 TB
32 GB

9.6 GB LOG SPACE ALLOCATED

Read scale-out
[✓ Enabled] Disabled

Would you like to make this database zone redundant?
Yes [✓ No]

Apply

Cost summary

Gen5 - Business Critical (BC_Gen5_2)	
Cost per vCore (in USD)	505.50
vCores selected	x 2
Cost per GB (in USD)	0.25
Max storage selected (in GB)	x 41.6
ESTIMATED COST / MONTH	1021.40 USD

Figure 4-24. *Azure SQL Business Critical service tier choices*

You still choose the number of vCores and Data max size. But you also choose whether this database will enable read scale-out and zone redundancy. I will discuss these high availability capabilities in Chapters 10 and 11 of the book. For now, I will leave Read scale-out Enabled (you should; it is free) and select Yes for zone redundancy.

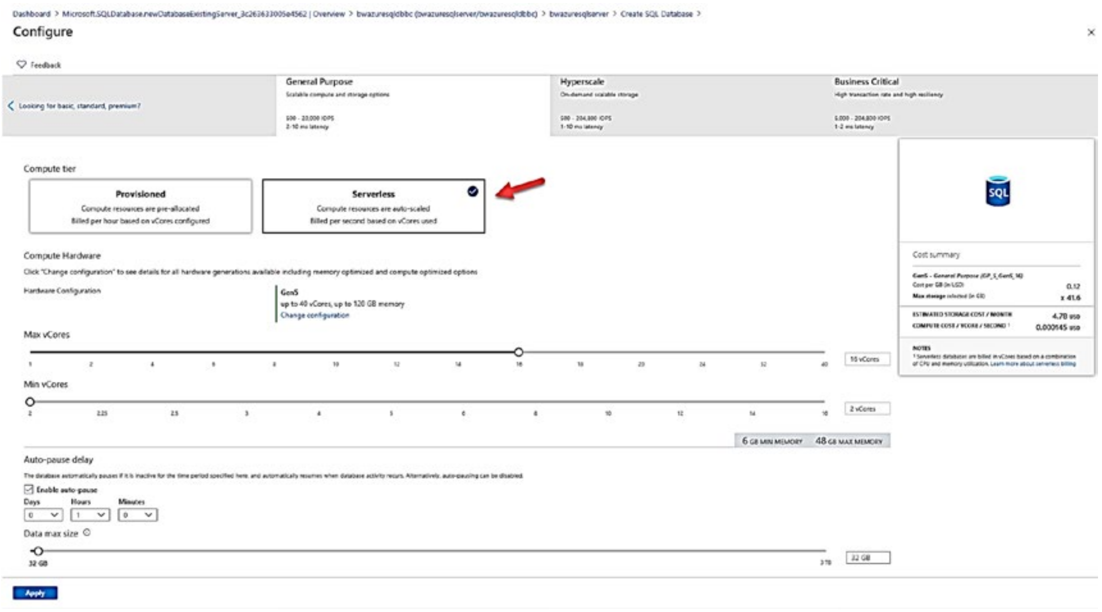
Click **Apply** and put in a database name. I will use a new name and call it **bwazuresqldbbc**. Instead of making additional choices, I will choose **Review + create** and then **Create**.

Note You could make other choices here including setting up a Private endpoint for your database different than the choice made for the first database on the server.

Once this deployment is complete (it should be fast), you can click **Go to resource** and see you now have a Business Critical database deployed. Click the **Server name** again and repeat the process to create a new database.

Deploying Serverless

Like the Business Critical Scenario, click **configure database**. Then select Serverless as seen in Figure 4-25 with new choices.



Dashboard > Microsoft.SQLDatabase.newDatabaseExistingServer_3c263633005e4562 | Overview > bwazuresqldbbc (bwazuresqlserver/bwazuresqldbbc) > bwazuresqlserver > Create SQL Database >

Configure

Feedback

Looking for basic, standard, premium?

General Purpose
Scalable compute and storage options
500 - 20,000 IOPS
2-10 ms latency

Hyperscale
On-demand scalable storage
500 - 204,800 IOPS
1-10 ms latency

Business Critical
High transaction rate and high resiliency
5,000 - 204,800 IOPS
1-2 ms latency

Compute tier

Provisioned
Compute resources are pre-allocated
Billed per hour based on vCores configured

[✓ Serverless
Compute resources are auto-scaled
Billed per second based on vCores used]

Compute Hardware

Click "Change configuration" to see details for all hardware generations available including memory optimized and compute optimized options

Hardware Configuration: Gen5
up to 40 vCores, up to 120 GB memory
Change configuration

Max vCores: 16 vCores

Min vCores: 2 vCores

6 GB MIN MEMORY 48 GB MAX MEMORY

Auto-pause delay

The database automatically pauses if it is inactive for the time period specified here, and automatically resumes when database activity recurs. Alternatively, auto-pausing can be disabled.

[✓] Enable auto-pause

Days: 0 Hours: 1 Minutes: 0

Data max size: 32 GB

Cost summary

Gen5 - General Purpose (GP_S_Gen5_16)	
Cost per GB (in USD)	0.12
Max storage selected (in GB)	x 41.6
ESTIMATED STORAGE COST / MONTH	4.78 USD
COMPUTE COST / VCORE / SECOND ¹	0.000145 USD

NOTES
¹ Serverless databases are billed in vCores based on a combination of CPU and memory utilization. Learn more about serverless billing

Apply

Figure 4-25. *Serverless compute options*

Instead of a slider to choose vCores, I can choose a min and max vCore. Serverless will *autoscale* based on the CPU needs of my workload. Notice the values for min and max memory. This means that for my Serverless database I can use a maximum of 48Gb, but when my usage is idle, my memory may be trimmed down to as low as 6Gb. Think of this like max and min server memory configuration values for SQL Server. The difference is that our Azure services will trim memory (almost like external memory pressure) if usage is idle.

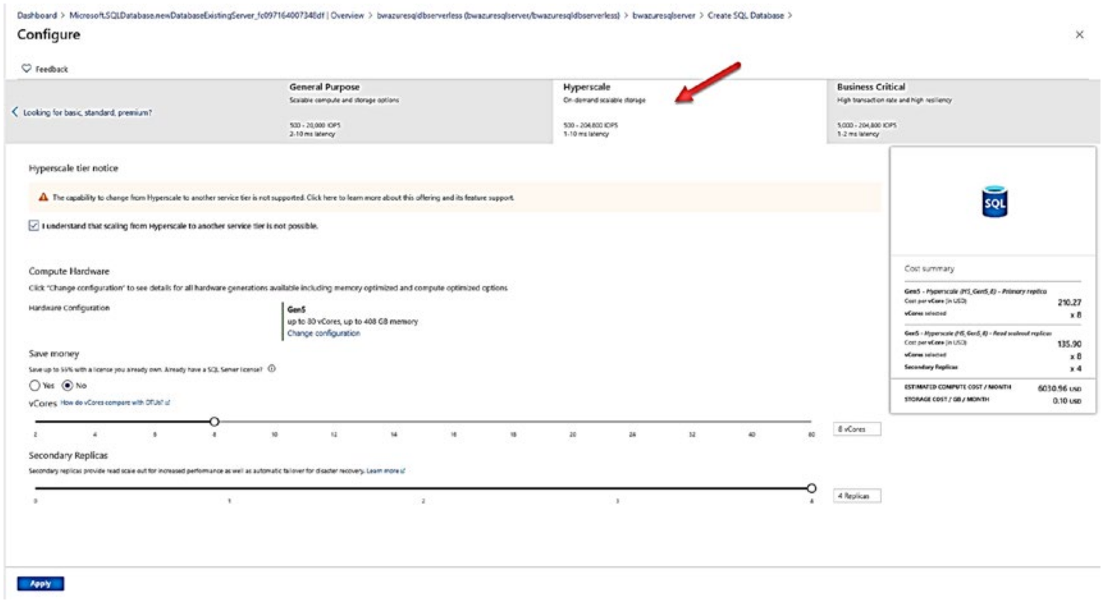
So, what defines idle? That is the choice for Auto-pause delay. If there is no usage for 1 hour, the compute for this database will be *paused* and memory resources reclaimed. Drill more into Serverless at `https://docs.microsoft.com/en-us/azure/azure-sql/database/serverless-tier-overview`.

Click Apply and put in a database name. I used **bwazuresqldbserverless**. Click **Review + create**. Notice on the validation screen the costs is listed as per second which is one of the great stories of saving costs with Serverless. Click **Create** and wait for the deployment to finish. When the deployment is finished, click **Go to resource**. Notice in the Monitoring Pane metrics for Compute utilization vs. App CPU billed. This helps you track billing per second of compute usage for a Serverless database.

Click the Server name again so you can create a Hyperscale database.

Deploying Hyperscale

Click Create Database from the command bar to see the screen again to create a new database on the same logical server. Click configure database and select the Hyperscale option as seen in Figure 4-26.



Dashboard > Microsoft.SQLDatabase.newDatabaseExistingServer_fc097164007348df | Overview > bwazuresqldbserverless (bwazuresqlserver/bwazuresqldbserverless) > bwazuresqlserver > Create SQL Database >

Configure

Feedback

Looking for basic, standard, premium?

General Purpose	Hyperscale	Business Critical
Scalable compute and storage options	On-demand scalable storage	High transaction rate and high resiliency
500 - 20,000 IOPS 2-10 ms latency	500 - 204,800 IOPS 1-10 ms latency	5,000 - 204,800 IOPS 1-2 ms latency

Hyperscale tier notice

The capability to change from Hyperscale to another service tier is not supported. Click here to learn more about this offering and its feature support.

[✓] I understand that scaling from Hyperscale to another service tier is not possible.

Compute Hardware

Click "Change configuration" to see details for all hardware generations available including memory optimized and compute optimized options

Hardware Configuration: Gen5 — up to 80 vCores, up to 408 GB memory — Change configuration

Save money

Save up to 55% with a license you already own. Already have a SQL Server license?

Yes [✓ No]

vCores How do vCores compare with DTUs?

2 4 6 8 10 12 14 16 18 20 24 32 40 80 — 8 vCores

Secondary Replicas

Secondary replicas provide read scale out for increased performance as well as automatic failover for disaster recovery. Learn more

0 1 2 3 4 — 4 Replicas

Cost summary

Gen5 - Hyperscale (HS_Gen5_8) - Primary replica	
Cost per vCore (in USD)	210.27
vCores selected	x 8
Gen5 - Hyperscale (HS_Gen5_8) - Read scaleout replicas	
Cost per vCore (in USD)	135.90
vCores selected	x 8
Secondary Replicas	x 4
ESTIMATED COMPUTE COST / MONTH	6030.96 USD
STORAGE COST / GB / MONTH	0.10 USD

Figure 4-26. *Hyperscale deployment options*

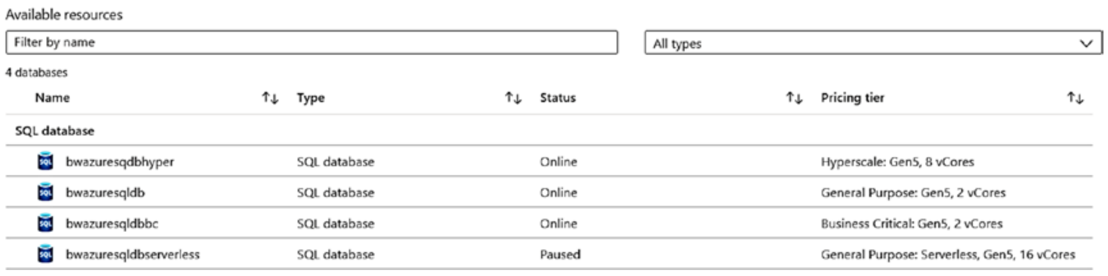
The first thing you may notice about this screen is that you must select the option that you understand that by choosing Hyperscale you cannot change the service tier (to General Purpose or Business Critical) once you deploy.

The slider for vCores looks just like General Purpose and Business Critical. But notice there is no Data max size. Hyperscale is theoretically limitless (although today we limit you to a 100TB database but a large transaction log). When you deploy a Hyperscale database, we create an initial size of ~40Gb and then just grow as you need space.

Hyperscale offers a unique choice for number of replicas. You will learn in Chapter 8 of this book about how Hyperscale provides unique availability. Allowing you to specify up to four replicas which can be used for read scale is one of them.

I will choose 8 vCores and four Secondary Replicas. Click Apply, put in a database name (I will use **bwazuresqldbhyper**), and click **Review + create**. I will click **Create** to start the deployment.

Once the deployment is complete, you can click **Go to resource** to bring up the Overview page of the database. If you select Server name, you can see now you have four databases spanning all the service and compute tiers as seen in Figure 4-27.


Available resources

Filter by name | All types

4 databases

Name	Type	Status	Pricing tier
SQL database			
bwazuresqdbhyper	SQL database	Online	Hyperscale: Gen5, 8 vCores
bwazuresqldb	SQL database	Online	General Purpose: Gen5, 2 vCores
bwazuresqldbbc	SQL database	Online	Business Critical: Gen5, 2 vCores
bwazuresqldbserverless	SQL database	Paused	General Purpose: Serverless, Gen5, 16 vCores

Figure 4-27. *Available databases from an Azure logical server*

Notice that my Serverless database is already paused because quite frankly I had paused for an hour before I deployed the Hyperscale database. Before we try to connect to these databases, let us explore more how to deploy with CLI tools and some implementation details of Azure SQL Database.

Deploying with a CLI

An Azure SQL Database can be deployed with command-line interfaces (CLI) through the **az sql db** (`https://docs.microsoft.com/en-us/cli/azure/sql/db?view=azure-cli-latest`) command interfaces or through **New-AzSQLDatabase** PowerShell cmdlet (`https://docs.microsoft.com/en-us/powershell/module/az.sql/New-AzSqlDatabase`).

Unlike Managed Instance, it is easier to use the az CLI for Azure SQL Database without an Azure template because I only have to create the logical server and then I can create the database (I don't have to create a virtual network and all the components first).

Here is an example of using az sql db to create a single database for a General Purpose, 2 vCore database:

```
az group create -name bwazuresqlrg2 -l eastus
az sql server create --location eastus --resource-group bwazuresqlrg2
--name bwazuresqlserver2 -u thewandog -p '<password>' --enable-public-
network true
az sql db create --resource-group bwazuresqlrg2 --server bwazuresqlserver2
--name bwazuresqldb2 --edition GeneralPurpose --family Gen5 --capacity 2
--sample-name AdventureWorksLT
```

Note The only option that cannot be done with the az CLI for database that I could in the portal is to set Allow Azure services to Yes and set the current Client IP address for a firewall rule. You can easily configure this after the deployment or use an Azure template.

PowerShell does give you all the options you need to deploy as with the portal. There is a good tutorial on using PowerShell at https://docs.microsoft.com/en-us/azure/azure-sql/database/scripts/create-and-configure-database-powershell?toc=/powershell/module/toc.json.

An Azure template is still a great idea to use and in fact is the best option to automate deployment of many databases. Read about how to use Azure templates and Azure SQL Database at https://docs.microsoft.com/en-us/azure/azure-sql/database/single-database-create-arm-template-quickstart.

Implementation Details

Note These implementation details may change over time as we change and improve the service. I offer up some of these details so you can understand how we build, manage, and run the service.

In Chapter 1 of this book, I covered the incredible history of how we have built an architecture to power millions of databases for Azure SQL Database. Let me give you a few more details about how we implement Azure SQL Database behind the scenes.

Note As I interviewed many people in the Microsoft engineering team behind the scenes about Azure SQL Database even as I write this chapter, we are looking into how to make our implementation more efficient. Therefore, it is possible even some of these details could be a bit outdated by the time you read this chapter. That is the *speed of the cloud* and an author's nightmare!

Dedicated Rings and Instances

Unlike Managed Instance, we pre-deploy rings dedicated to hosting Azure SQL Databases. With only a few exceptions, each database is hosted by a dedicated SQL Server instance (exceptions being "subcore" DTU options and elastic pools). This allows us to provide better isolation for a customer and keep the "just worry about the database" model while opening some instance-level surface area (e.g., DMVs and columnstore indexes). We may provision these instances on the same VM or node, but those details are abstracted from you, provided we keep to our SLA agreement and objectives for performance.

All the rings and instances are powered and managed using **Azure Service Fabric**. This is the same service fabric software that you can build your own microservices. The Azure Service Fabric architecture is well documented at https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-architecture.

The Logical Server

As I stated earlier in this chapter, a database or logical server is just a metadata concept. We provide an interface to a server and a master database. But, when you query various aspects of the master database, we may be pulling data from other stores or files within the service to show you the information. There is a good description of why you need a logical server for even one database at https://docs.microsoft.com/en-us/azure/azure-sql/database/logical-servers.

Storage, Compute, and Gateways

You will see in Chapter 8 for Availability more details on the architecture behind the scenes on how we implement High Availability (HA) for General Purpose, Business Critical, and Hyperscale service tiers. We achieve certain HA capabilities by either using Azure Storage or local storage with technologies like Always On Availability Groups.

In each of these cases, one of the key components for connectivity is a *gateway*. Gateways are nodes that basically route traffic to nodes hosting SQL Server databases. I mentioned the use of redirect vs. proxy connection types with a Managed Instance earlier in the chapter. The same concept will apply with an Azure SQL Database. Gateways are critical to connectivity in that they provide abstractions to an application for connectivity no matter where the node for the database lives which you will learn more about in Chapter 6 on Security and Chapter 8 on Availability.

Serverless

Serverless compute models involve several interesting technologies we implement within the standard deployment of a SQL Server. Many of these details are described in our documentation at `https://docs.microsoft.com/en-us/azure/azure-sql/database/serverless-tier-overview`.

Since storage and compute are separated for a Serverless deployment, *pausing* a database is not that difficult since no application is connected. We just need to keep around enough state information that when a new login is made, we can connect the database to an instance and "warm up" the application.

Autoscaling is more interesting. We need to scale up or down the database CPU resources without application disruption. And if we can scale on a node that can meet the new scaling demand, there is no disruption. However, if we cannot meet that demand, we may need to use an Azure Load Balance to keep the application connected if possible until a new node is found to meet demand, but there can be a disconnection when the new node is brought up.

Memory management is also different in that we must deploy memory policies to reclaim memory for the SQL Server instance when CPU or cache utilization is low. Think of this concept as though we are signaling SQL Server there is external memory pressure and lowering the target for memory. Autoscaling and Memory Management for Serverless is described more at `https://docs.microsoft.com/en-us/azure/azure-sql/database/serverless-tier-overview#autoscaling`.

Hyperscale

Hyperscale is a unique implementation for a database within the same architecture of databases (dedicated rings and nodes) implemented much differently than General Purpose or Business Critical service tiers.

A hyperscale deployment involves a series of nodes for compute, logging, and caching combined with Azure Storage. I talked about this architecture in Chapter 1 of the book, but it is worth showing you again as seen in Figure 4-28 (which is directly from the documentation at https://docs.microsoft.com/en-us/azure/azure-sql/database/service-tier-hyperscale#distributed-functions-architecture).

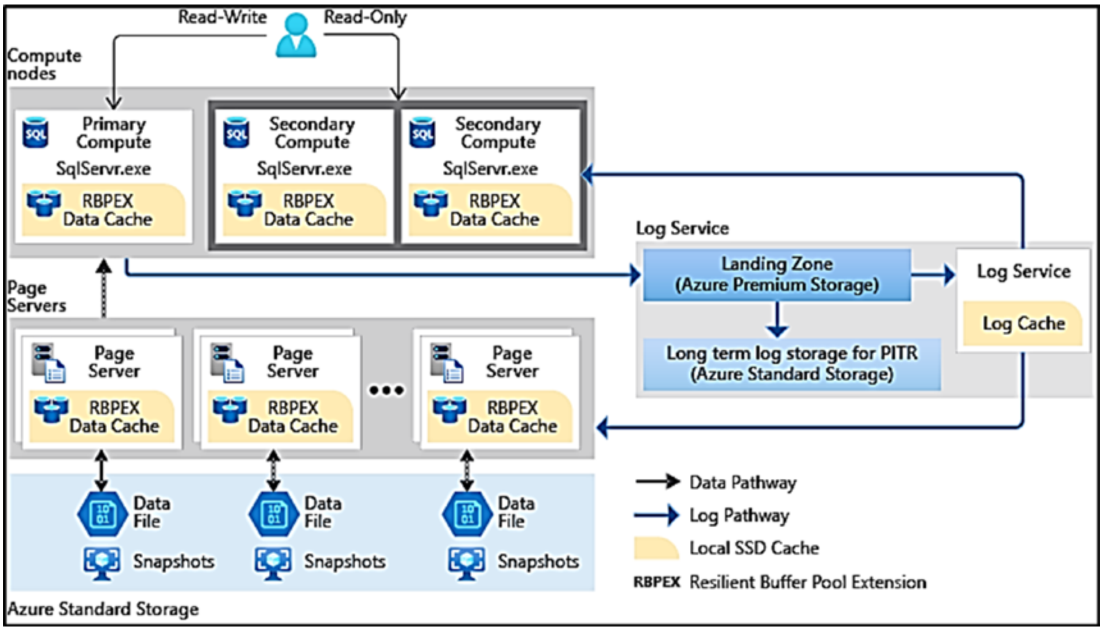


Figure 4-28. *The Hyperscale architecture*

I will show you more of the working parts of this architecture in Chapter 8 of this book, but let me stop and call out a few key components:

- Separate of compute and storage

  Just like General Purpose, we store database files on Azure storage. But notice here we use Azure Standard Storage. Speed to access the database files is not as important because of the caching system.

- The *caching* system

  We use a combination of page servers (actual nodes that host database pages) and buffer pool caches (think SSD drives that extend the buffer pool) on page servers and compute nodes.

- Log Service

  For Hyperscale, any logged changes are still in a log cache on the primary compute node. However, when log changes must be flushed to disk (a commit), these I/O requests are *redirected* to another node where a component called the Log Service runs (this is also called Xlog). The Log Service is responsible for ensuring changes are stored locally (called the *Landing Zone*) and are refreshed to the caching system, secondary replicas, and eventually to Azure storage.

- Decoupled Replicas

  In a way Hyperscale provides you the best of both the General Purpose and Business Critical tiers. The actual database and transaction log files are stored in Azure Storage (which has its own redundancy), but we also have replicas for extremely fast high availability.

  However, the secondary replica system does not use Always On Availability Groups. In fact, the primary and secondary replicas are not aware of each other. Secondary replicas use a log change methodology but are fed changes from the Log Service. Commits on the primary replica can proceed once the Log Service has hardened the changes not sending to a replica.

  High availability is even allowed with no secondary replicas. How? If the primary node has an issue, we can deploy a new primary replica on a new node and use page servers or even the database files on Azure Storage because it is decoupled. Having said this, RTO is much faster with the presence of secondary replicas. The secondary replica system (because you can have four of these) provides the best read-scale option for Azure SQL Database.

- *Hyperfast* Backup and Restore

  Because most of the data access comes from the caching system, reading database pages from the database files is rare with a *warm* system. This allows us to use snapshot backups for database files.

Snapshot backups are extremely fast since we just copy the files to another storage location. And the other amazing story is Restore. Restoring a database snapshot is crazy fast!

My colleague Kevin Farlee has an excellent video describing the Hyperscale architecture including the great story of restore at `https://youtu.be/Z9AFnKI7sfI`.

Resource Governance

To meet the SLA requirements for Azure SQL Database, we must put some resource limits on the usage of the database. I have described some of these limits in this chapter. Behind the scenes, we use these technologies to enforce these limits:

SQL Server Resource Governor

Azure SQL Managed Instance allows for user-defined workload groups and pools. Azure SQL Database uses Resource Governor behind the scenes to enforce certain limits. Moving to a dedicated SQL Server instance was a key driver in allowing us to use Resource Governor.

Engine enhancements

The engine has been enhanced in Azure to detect the generation of a certain size and rate of transaction log records and govern the application if necessary. The primary signal this governance is happening is seeing a wait type of LOG_RATE_GOVERNOR. You can read more about log rate governance at `https://docs.microsoft.com/en-us/azure/azure-sql/database/resource-limits-logical-server#transaction-log-rate-governance`.

Windows Job Objects

I have mentioned this technology before in this chapter. Windows Job Objects allow us to control resource usage on the SQL Server engine process to ensure we properly enforce resource limits like memory.

File Source Resource Manager (FSRM)

FSRM provides a mechanism so we can properly enforce storage maximum sizes outside of what we control through SQL Server file size limits.

We have created a great blog post talking about how we enforce resource limits using these technologies and why we use them at `https://azure.microsoft.com/en-us/blog/resource-governance-in-azure-sql-database/`.

Connecting and Verifying Deployment

Once you deploy an Azure SQL Databases, you are likely going to want to quickly connect and verify aspects of the deployment.

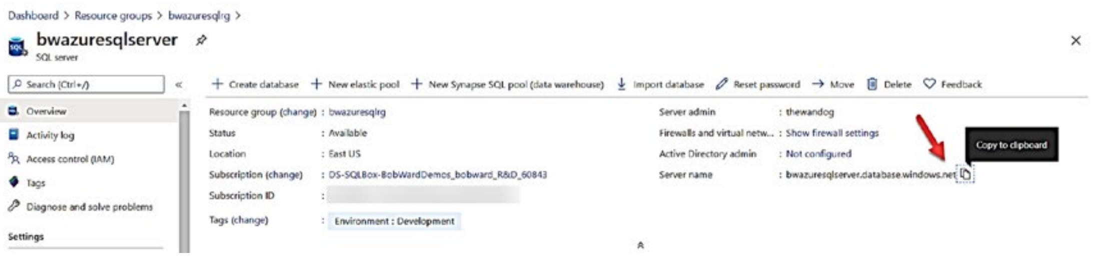
Connecting to Azure SQL Database

Earlier in this chapter, I deployed my logical server and Azure SQL Database allowing public endpoint access with options to access through Azure services and with a firewall rule for the client IP address where I deployed through the portal. I will show you how to connect using both techniques.

Because I used the option Allow Azure services and resources to access this server when I deployed the logical server for the database, I can use an Azure Virtual Machine or even sqlcmd from the Azure Cloud Shell to connect to this server and database.

To connect with Azure Cloud shell, you need to find the name of the logical server for the deployment. There are many ways to do this through the portal. You can simply look at your resource groups or resources from the home of the portal and find the server **bwazuresqlserver** (or your name).

Figure 4-29 shows the Working pane of the server with the Server name to use when connecting with a SQL Server tool or application.



Resource group (change) : bwazuresqlrg
Status : Available
Location : East US
Subscription (change) : DS-SQLBox-BobWardDemos_bobward_R&D_60843
Subscription ID : [redacted]
Tags (change) : Environment : Development

Server admin : thewandog
Firewalls and virtual netw... : Show firewall settings
Active Directory admin : Not configured
Server name : bwazuresqlserver.database.windows.net

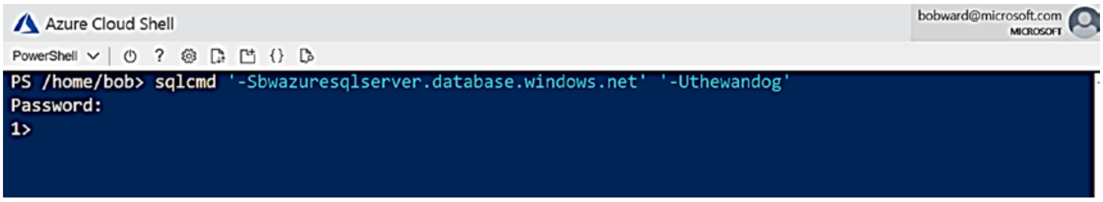
Figure 4-29. *Find the Server name to connect*

Notice I clicked next to the server name to copy the Fully Qualified Domain Name (FQDN) to the clipboard. You will find the FQDN is a combination of the logical server name and .database.windows.net.

Note Azure SQL Database also supports the concept of a DNS alias which you can read about at `https://docs.microsoft.com/en-us/azure/azure-sql/database/dns-alias-overview`.

I can now bring up the Azure Cloud Shell (you can use the home page of the portal, but I like to use `https://shell.azure.com`).

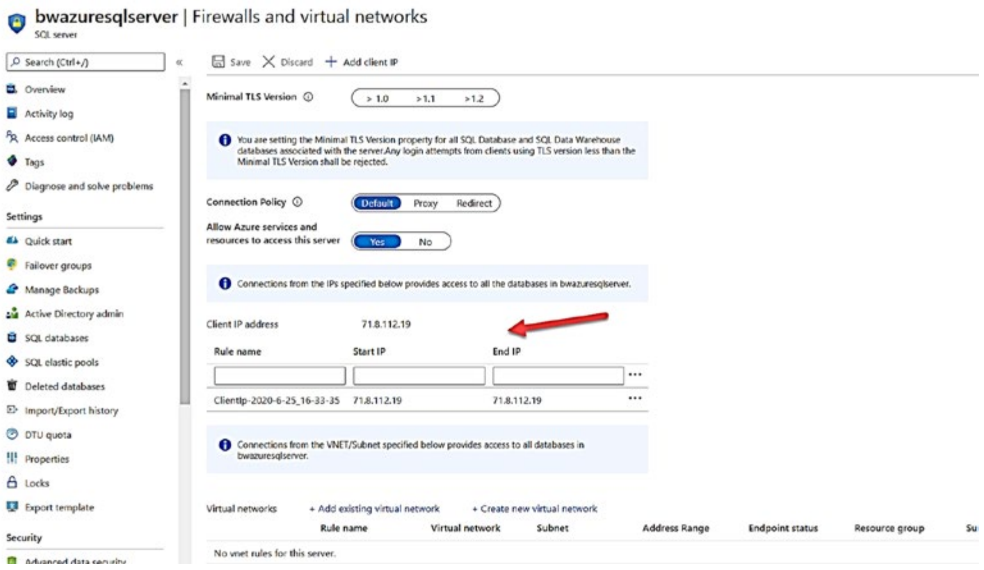
Since sqlcmd is installed with the cloud shell, I can use a syntax like Figure 4-30.



```
PS /home/bob> sqlcmd '-Sbwazuresqlserver.database.windows.net' '-Uthewandog'
Password:
1>
```

Figure 4-30. Using sqlcmd from the Azure Cloud Shell

I also configured the logical server for a firewall rule for the IP address of the computer when I was using the Azure Portal. In the working pane of the logical server, I can select **Show firewall settings** to see the following information. Figure 4-31 shows this firewall setting along with other network options.



Minimal TLS Version: > 1.0 | >1.1 | >1.2

You are setting the Minimal TLS Version property for all SQL Database and SQL Data Warehouse databases associated with the server.Any login attempts from clients using TLS version less than the Minimal TLS Version shall be rejected.

Connection Policy: [✓ Default] Proxy Redirect

Allow Azure services and resources to access this server: [✓ Yes] No

Connections from the IPs specified below provides access to all the databases in bwazuresqlserver.

Client IP address 71.8.112.19

Rule name	Start IP	End IP
ClientIp-2020-6-25_16-33-35	71.8.112.19	71.8.112.19

Connections from the VNET/Subnet specified below provides access to all databases in bwazuresqlserver.

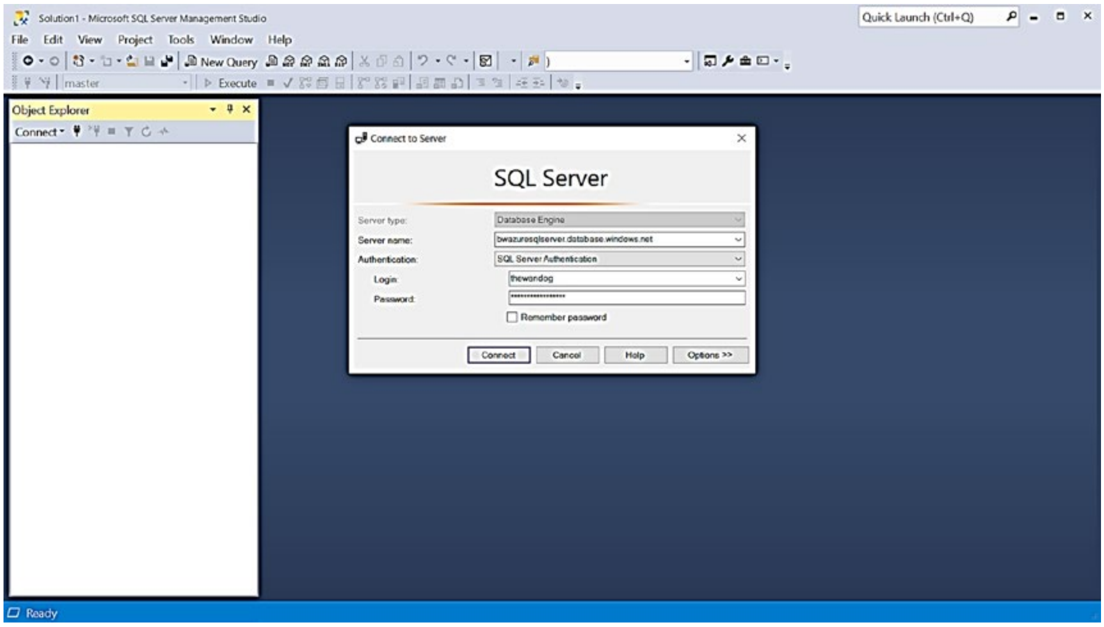
Virtual networks

Rule name	Virtual network	Subnet	Address Range	Endpoint status	Resource group	Su
No vnet rules for this server.						

Figure 4-31. Configure firewall rules for Azure SQL Database

A firewall rule is very much like a firewall rule you would configure in Windows Server or Linux. If you have used SQL Server before you know that by default, we do not open the firewall rules in the OS for port 1433 by default. The firewall rule earlier is opening access to the gateway for this logical server for this specific IP address. Firewall rules can be specified at the logical server or database level. You can read more about firewall rules for Azure SQL Database at `https://docs.microsoft.com/en-us/azure/azure-sql/database/firewall-configure`.

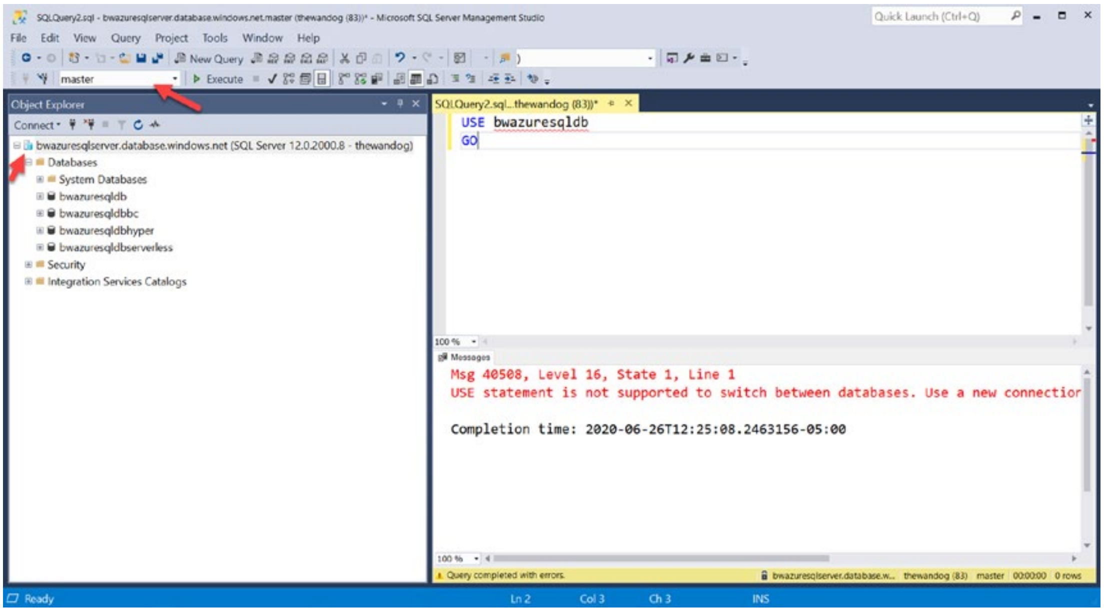
Since my client IP address is in the firewall rule, I can use a tool like SQL Server Management Studio to connect to the logical server as seen in Figure 4-32.



Server type: Database Engine
Server name: bwazuresqlserver.database.windows.net
Authentication: SQL Server Authentication
Login: thewandog
Password: ****************
☐ Remember password

Figure 4-32. Connecting to Azure SQL Database with SSMS

Once I hit Connect, I get the familiar Object Explorer, and expanding the list of databases, I see all the databases I have deployed in this chapter. I can right-click the server, select New Query, and try to switch database context to one of my databases as seen in Figure 4-33.


SQLQuery2.sql - bwazuresqlserver.database.windows.net.master (thewandog (83))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Tools Window Help
New Query
master
Execute
Object Explorer
Connect
bwazuresqlserver.database.windows.net (SQL Server 12.0.2000.8 - thewandog)
Databases
System Databases
bwazuresqldb
bwazuresqldbbc
bwazuresqldbhyper
bwazuresqldbserverless
Security
Integration Services Catalogs
SQLQuery2.sql...thewandog (83))*
100 %
Messages
Query completed with errors.
bwazuresqlserver.database.w... thewandog (83) master 00:00:00 0 rows
Ready
Ln 2 Col 3 Ch 3 INS

```
USE bwazuresqldb
GO
```

```
Msg 40508, Level 16, State 1, Line 1
USE statement is not supported to switch between databases. Use a new connectior

Completion time: 2020-06-26T12:25:08.2463156-05:00
```

Figure 4-33. *Trying to switch database context for Azure SQL Database*

You can first see some differences for Object Explorer (OE) including the color of the icon (Azure blue) for the server name. Notice also there are not as many choices in OE as there are with SQL Server or Managed Instance (because this is not a full SQL Server instance).

When I connected with SSMS specifying no options, I was put in the context of the logical master for the logical server. I tried to change database context with the familiar T-SQL USE statement, but if fails. Why?

If you think about T-SQL USE on a SQL Server, the engine switches context to a database stored on the instance. For an Azure SQL Database logical server, the databases are on separate SQL Server instances that can be spread across various rings in the Azure region. The USE statement is not built to redirect connections to different server.

Therefore, for SSMS, you can either specify the database to connect to before you hit the Connect button (use the Options button) or switch database context using the drop-down box (which does change the connection context).

In Chapter 5 of the book, I will show you how to configure the connection type for the Azure SQL Database to use redirect instead of proxy. In Chapter 6 of the book, I will show you how to make the connection to the database more secure.

Verifying Deployment

While you can use the Activity Log to verify the deployment of the database, there is no method to see the ERRORLOG behind the SQL Server instance. Therefore, you can use several T-SQL queries to examine the deployment:

Note Even though there are some queries that make sense to run in the context of the logical master, I ran these queries in the context of my database bwazuresqldb.

```
SELECT @@version
```

```
Microsoft SQL Azure (RTM) - 12.0.2000.8   May 15 2020 00:47:08   Copyright
(C) 2019 Microsoft Corporation
```

This is the exact same result as with Azure SQL Managed Instance indicating a versionless SQL Server:

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

```
database_id  name          compatibility_level
1            master        150
5            bwazuresqldb  150
```

You will always see master and only your database from the context of a user database. But note that this is still the logical master, not the physical master on the SQL Server instance hosting the database:

```
SELECT name, object_id, type_desc FROM sys.objects
```

Since this database was built on the sample AdventureWorksLT, I have ~204 rows from this catalog view including system tables:

```
SELECT * FROM sys.dm_os_schedulers
```

This is one of the DMVs we can expose since we are running on a dedicated SQL Server instance. I deployed a 2 vCore General Purpose database so as I would expect I get two ONLINE schedulers:

```
SELECT * FROM sys.dm_os_sys_info
```

Just like with a Managed Instance, I can use this to look at CPU and memory information for the database deployment. Keep in mind though that true limits must be observed with other DMVs as described as follows:

Note I used **sys.dm_os_process_memory** with Managed Instance, but that is not supported with Azure SQL Database.

```
SELECT * FROM sys.dm_exec_requests
```

This is one of the most common DMVs in the world to check the state of what is running on a SQL Server. I run this just to make sure all the normal background processes are running, including LAZ WRITER, RECOVERY WRITER, LOCK MONITOR, and so on.

Here is the interesting twist on this DMV for Azure SQL Database. This will show you requests for the instance for your database, not requests for other databases on your logical server (because they are deployed on other instances):

```
SELECT SERVERPROPERTY('EngineEdition')
```

Per the documentation at https://docs.microsoft.com/en-us/sql/t-sql/functions/serverproperty-transact-sql, the value of 5 is a SQL Database.

There are also two new DMVs specific to Azure not found in SQL Server:

```
SELECT * FROM sys.dm_user_db_resource_governance
```

This DMV is really intended to show you resource limits for a specific Azure SQL Database. You can view limits like memory, max storage, log rates, and so on. You can read the documentation for this DMV at https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-user-db-resource-governor-azure-sql-database?view=azuresqldb-current. Note that the docs say this is mostly for internal use, which means it might change in the future:

```
SELECT * FROM sys.dm_os_job_object
```

This is a DMV specific to Azure (although I tested it and it works for a SQL Server, but you can't rely on the results there because it is not applicable) that shows resource limits Azure applies to the Database using Windows Job Objects. The specific column I look at is **memory_limit_mb** which shows me the true amount of memory the database has access to. I talked about Windows Job Objects in the section "Implementation Details."

Note I do not recommend you rely on any results of running these queries in the context of the logical master. Even though you might see results, they do not mean anything since the logical master is not a true physical master database. There are a few queries that make sense to run in the logical master which you will see later in the book.

Migrating to Azure SQL Database

Migrating to Azure SQL Database involves the same process as with a Managed Instance of assessment and planning, migration, application changes, and post migration.

While the steps are the same, you will find several differences that are important:

- Azure SQL Database has more restrictions on features, so you may find your assessment is going to find more problems you need to take care before migrating. An example is a feature like Service Broker which is supported in Managed Instance but not in Azure SQL Database.
- Let me give you a simple example. When I first tried to migrate the example WideWorldImporters (`https://github.com/Microsoft/sql-server-samples/releases/tag/wide-world-importers-v1.0`) to Azure SQL Database, I ran into a bunch of problems because certain features used in the sample didn't work in Azure SQL Database. Therefore, I needed to use the Standard WideWorldImporters found at `https://github.com/Microsoft/sql-server-samples/releases/download/wide-world-importers-v1.0/WideWorldImporters-Standard.bacpac`.

- Migrating to Azure SQL Database involves migrating your schema (all your definitions) first and then the data. You can use DMS and DMA to do this. Read more about how to do this at https://docs.microsoft.com/en-us/azure/dms/tutorial-sql-server-to-azure-sql. Like Managed Instance, consider using a database compatibility level that matches your current SQL Server installation and then move later to the latest compat level. Learn more about dbcompat at https://aka.ms/dbcompat.
- You can also load your data into Azure SQL Database using SSIS packages, Azure Data Factor, bcp, or a BACPAC file. Remember that minimal logging for bulk import is not supported in Azure SQL Database.
- Even though the Microsoft Mechanics video I did on migration focuses demonstrations more on Azure Managed Instance, it is still worth watching for tips on migration to Azure SQL Database. Watch the video at https://youtu.be/P_4EaqVR5PI. One of the aspects of using DMA I love is the SKU recommendation PowerShell script. This script analyzes your current SQL Server workload to guide you on Azure SQL Database deployment choices. Read more at https://docs.microsoft.com/en-us/sql/dma/dma-sku-recommend-sql-db?view=sql-server-ver15.

Like Managed Instance, the application needs to change the connection string and possibly authentication method. Depending on what T-SQL features and language constructs are used, further application changes may be needed. The DMA tool does a good job of finding these based on database compatibility. To be more thorough, please look over this documentation for T-SQL differences at https://docs.microsoft.com/en-us/azure/azure-sql/database/transact-sql-tsql-differences-sql-server.

Use the rest of the chapters in the book to guide you on any post-migration changes you need to make to fully take advantage of security, performance, and availability in Azure.

Summary

In this chapter, you learned how to go through a pre-deployment exercise to make the best choices possible to deploy Azure SQL Managed Instance or Database. You learned the details of deployment for both a Managed Instance and Database along with some interesting implementation details.

You learned how to connect and run verification queries against a Managed Instance and Database deployment. You also learn migration techniques and tools to migrate an existing SQL Server to Azure SQL Managed Instance and Database.

Now that you have deployed, learn more in the next chapter about how to make configuration choices and compare these choices to configuring a SQL Server instance or database.