Microsoft Azure Options for SQL Server Relational Databases

Microsoft

Microsoft Corporation

Final Version: April 2018

Authors: Kalen Delaney

Contributors: Sunil Agarwal, Ajay Kalhan, Morgan Oslake, Borko Novakovic, Kevin Farlee, Ajay Jagannathan, Mine Tokus; Conor Cunningham, Alain Dormehl

Reviewers: Alain Dormehl, Eric Hudson

For the latest documentation on Azure SQL Database, please see
https://azure.microsoft.com/en-us/services/sql-database/

Summary: Azure is Microsoft's collection of cloud services for developers and IT professionals. Microsoft Azure provides both IaaS and PaaS options for building, deploying and managing SQL Server database applications in the cloud. This paper will examine the various Azure options available for SQL Server database applications including: Azure SQL DB single database, elastic pools, the new Azure Managed Instances and SQL Server on an Azure VM. We will describe the details of how each option is managed internally, and the differences between the assorted options. With this information, we'll be able to provide recommendations for using each of the options and discuss how to choose between the options based on your requirements.

Disclaimer

The information contained in this document represents the current view of Microsoft Corporation regarding the issues discussed as of the date of publication. Because Microsoft is always responding to changing market conditions, this document should not be interpreted as a commitment on the part of Microsoft. Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

TABLE OF CONTENTS

1 INTRODUCTION

Cloud computing is a major industry trend that allows you to rent computers and software from someone else's data center instead of buying, so your main costs are the operational expenses (OPEX) as opposed to a capital expense outlay (CAPEX). This allows for greater efficiencies of scale, allows customers to scale-up and scale-down, and allows best-practices (security, patching, upgrades) to be included – all of this greatly reducing many of the ongoing costs of running your own data center and allowing you to focus on your business. Microsoft hosts a cloud computing service called Microsoft Azure or just Azure. Unlike pure Virtual Machine consolidation models of the past, there are now additional choices (PaaS/SaaS) which have the option of greater automation/savings but at some loss of control/compatibility depending on the customer need. This paper outlines the choices for purchasing relational database options (SQL Server) in cloud environments.

SQL Azure Database is Microsoft's cloud-based relational database service. It is sometimes referred to as SQL Azure DB or simply SQL Azure, and informally as Azure SQL. This introductory section will present you with the basic information about what SQL Azure is and what it is not and define general terminology for use in describing SQL Azure databases and applications.

1.1 Azure Overview

Many, if not most, cloud-based databases provide storage and compute resources using a Virtual Machine (VM) model, frequently referred to as IaaS, or Infrastructure as a Service When you purchase your subscription from the vendor and set up an account, you are provided with a VM hosted in a vendor managed datacenter. However, what you do with that VM is then entirely isolated from anything that goes on in any other VM in the data center. Although the VM may come with some specified applications pre-installed, in your VM you can install additional applications to provide for your own business needs, in your own personalized environment. With Microsoft Azure, you can also specify specific versions of SQL Server to be preinstalled in your VM, with or without Always On enabled. You can even remove the applications that were preinstalled if you choose. It's exactly as if you're working with your own computer. Even though your applications can run in isolation, the hosting company provides the physical infrastructure, and the performance of your applications is impacted by the load on the data center machines, from other VMs using the same CPU, memory, disk I/O and network resources.

Although the VM model is an option with Microsoft Azure, and there are many preconfigured VM choices which include SQL Server, there are other completely different options available that will be the primary focus of this paper. The Microsoft Azure data centers have installed large capacity SQL Servers on commodity hardware that are used to provide computing resources and data storage to databases created by subscribers. In addition to these resources, Azure provides services to manage the SQL Server databases.

1.2 What is in this document

The focus of this paper is to provide you will details regarding the various options available for creating and managing a SQL Server relational database in Azure with Azure SQL Database or Azure VMs. Azure SQL Database has three deployment models: single database, elastic pool, and Managed Instance. We will look

at the architecture of the various offerings and describe their differences and similarities to help you in deciding which of the relational database Azure offerings will be best for your and SQL Server environment.

1.3 What is not in this document

This document only covers the specified relational services: There will be no discussion of the BI Services, Machine Learning, document management, etc.

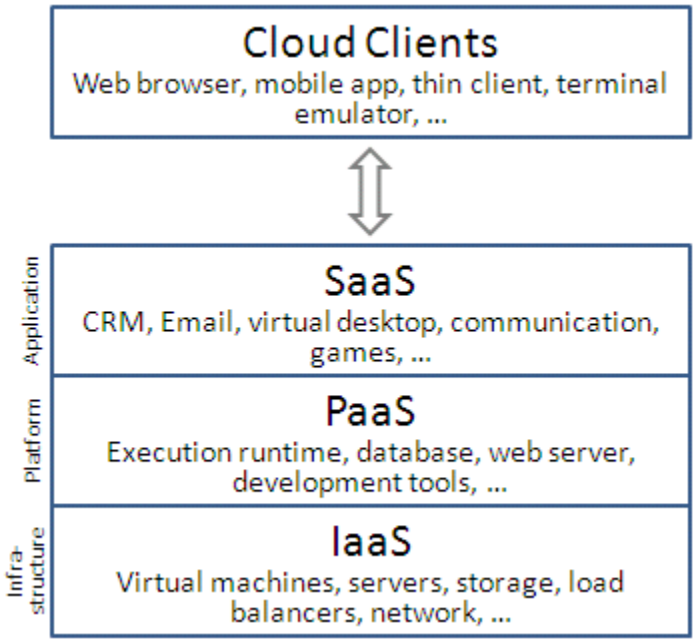
This document will not cover details of all the features that are supported and not supported in Azure SQL DB, compared to a SQL Server database that you create in a SQL Server instance on your own premises on your own hardware. It will describe some of the key differences, but it will not provide a complete list.

Finally, this document is not a HOW-TO guide to setting up your Azure subscription, using the Microsoft Azure Portal (https://portal.azure.com/), or creating your servers or databases. There will be only minimal information about connecting to your Azure databases.

1.4 Terminology

When using Microsoft Azure, your machine computer infrastructure and some or all of your SQL Server configuration and management is handled for you. The term used for systems and databases not in the cloud is on-premises, or on-prem.

Cloud service providers, including Microsoft, offer several different models of services depending on how much of the typical on-prem resources are provided and managed for you. The different models are frequently presented as layers in a stack, as shown in the figure below, taken from the Wikipedia article: https://en.wikipedia.org/wiki/Cloud_computing



- IaaS, or Infrastructure as a Service, provides a hardware infrastructure and allows the consumer to install and run any software, including the operating system of their choice. In the Microsoft Azure offerings, this service model is available when using an Azure VM.
- PaaS, or Platform as a Service, provides programming languages, libraries, services and tools, as well as the network, servers, operating system and storage. Specialized PaaS offerings are dPaaS (Data

Platform as a Service) or DBaaS (Database as a Service). Microsoft Azure's SQL DB and the new Managed Instance (MI) offerings are considered PaaS.

- SaaS, or Software as a Service model, cloud providers managed application software in the cloud and users access it from cloud clients. No installation of software on the users' computers is needed. Microsoft's Office 365 is a common example of SaaS. The Microsoft Dynamics 365 suite of services is a PaaS offering built on top of Azure SQL DB.

As mentioned above, there are four different options which will be discussed for creating and managing a SQL Server relational database on Microsoft Azure. Details of each of the models will be discussed later, but in this section, we'll look at the various terms used for the relational database offerings.

- SQL Server can be installed in an Azure Virtual machine, and sometimes this option is just referred to as SQL VM.
- Azure SQL Database or Azure SQL DB is an Azure PaaS offering that includes single databases, elastic pools and Managed Instances.
- A single database in Azure SQLDB is sometimes referred to as a singleton database. Each single database is associated with a logical server, which is a collection of databases which you can manage as a unit.
- An Elastic pool is a collection of single databases grouped for sharing of resources. There is no shortened name used. Each elastic pool is associated with a logical server.
- An Azure SQL Database Managed Instance (Managed Instance or MI), is the newest Azure relational database offering, as of March 2018. A MI allows you to have a collection of databases as if they're on a SQL Server instance on-prem and gives you the most parity with an on-premises SQL Server installation. You can perform cross-database operations, as well as control server level functionality such as managing SQLAgent alerts and jobs.

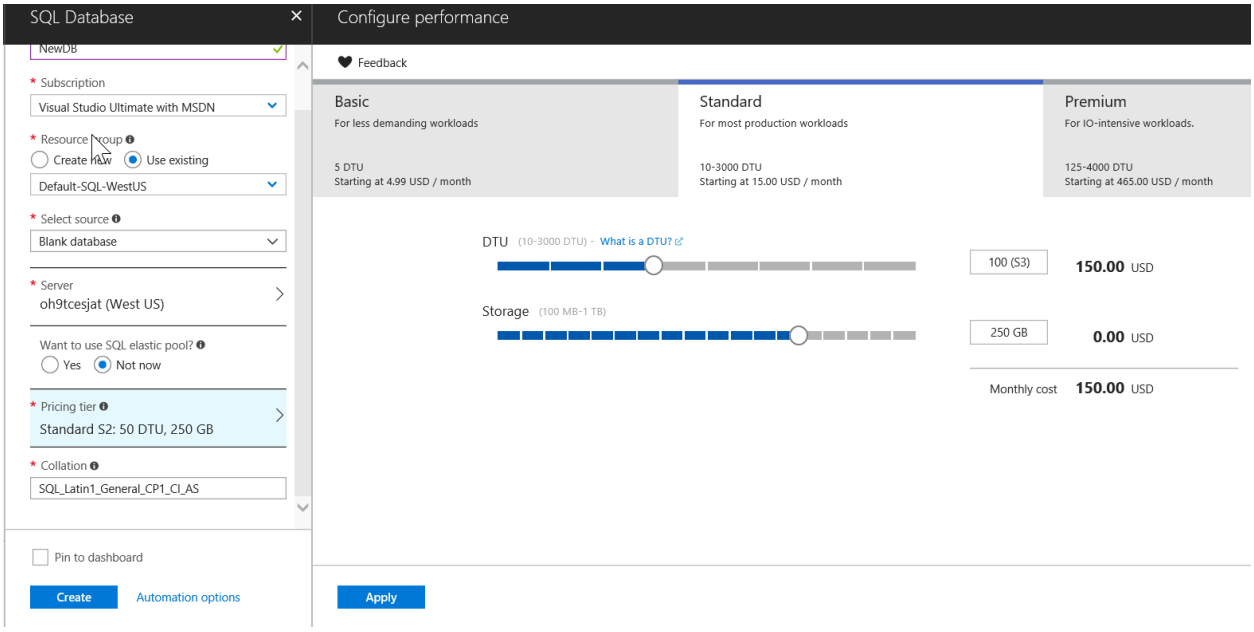
1.5 Pricing

Pricing for computing resources for Azure SQL Database has been based on a unit called DTU (Database Transaction Unit) when is a blended measure of CPU, memory and I/O, basically relating to the performance and throughput of your SQL Server system. When your workload exceeds the amount of any of these resources, your throughput is throttled - resulting in slower performance and timeouts.

When creating an Azure SQL DB, you need to specify details about the pricing tier, as the Azure Portal calls it. The most crucial decision, from a technical perspective, is the Service Tier, where you choose between Basic, Standard or Premium. You need to specify the amount of storage you require, as well as the computing resources (DTUs) needed. Within each service tier, you can specify a performance level which defines the number of DTUs within the service tier. For example, S3 is a performance level in the Standard tier for single databases which provides 100 DTUs.

For a single Azure SQL database at a specific performance level within a service tier, Microsoft guarantees a certain level of resources for that database, which is independent of any other database in the Azure cloud and provides a predictable level of performance. A fixed price per month is determined based on your requested Service Tier, the number of DTUs desired, and the amount of storage space you need. As you set up your Azure resources in the Portal, this cost will be clearly displayed. For example, in the graphic below, a new SQL DB is being created. When I click on the pricing tier option in the left pane, the information on the right becomes available. I can choose one of the three service tiers (Basic, Standard or Premium) and in the graphic, Standard has been selected. I have chosen 100 DTUs which costs $150 per month and comes with

250GB of storage for no extra cost. If I move the slider for additional storage to the right, there would be an additional monthly fee displayed. Note that costs can vary from region to region and the numbers shows here are just for my specific example.



If you'd like to get some sort of idea when planning your move to Azure SQL DB of how many DTUs (or eDTUs for elastic pools) you might need, you might want to take a look at the Azure SQL Database DTU Calculator, which can be downloaded from here: http://dtucalculator.azurewebsites.net/ This calculator recommendations regarding the minimum performance level and service tier that you need before you migrate to Azure SQL Database.

The new Managed Instance introduces a completely different pricing model called virtual cores, or vCores. A vCore represents the logical CPU with an option to choose between generations of hardware. For example, Gen 4 Logical CPUs are based on Intel E5-2673 v3 (Haswell) 2.4 GHz processors and Gen 5 Logical CPUs are based on Intel E5-2673 v4 (Broadwell) 2.3 GHz processors. The vCore-based model is designed to give customers flexibility, control, transparency and a straightforward way to translate on-premises workload requirements to the cloud. Although the pricing model refers only to the compute resources (the vCores), it will fully separate compute, storage and IO and allow subscribers to independently request these resources. Compute resources, as mentioned, will be specified as the number of vCores, storage will be specified in GB, as it is in the current DTU pricing model, and IO will be denominated in IOPs.

Although initially, only Managed Instance used the vCore pricing model, as of April 2018, vCore is an option for both single Azure SQL Databases and Elastic Pools. The DTU and vCore-based models will continue to exist side by side for single databases and elastic pools. Subscribers who do not want to worry about the underlying resources and prefer the simplicity of a preconfigured resource bundle with a known fixed cost each month, may find the DTU-based model more suitable for their needs. However, for those who need or want more transparency into the underlying resources or need to scale CPU, memory and I/O independently to achieve optimal performance at a fixed hourly cost, the vCore-based model will be the best choice (Note that for Managed Instance, vCore will be the only pricing option.)

When considering the best pricing option and comparing it to your costs for your on-premises SQL Servers, make sure you keep in mind both the capital expenses, such as your outlay for new hardware, and your operating expenses to manage your datacenter on an ongoing basis. Microsoft Azure's PaaS offering can significantly reduce both cost elements.

2 AZURE SQL DATABASE

2.1 Overview

Azure SQL Database includes the three PaaS offerings: single databases, elastics pool and managed instance and most of the technical details described in this section apply to all three. Single databases, elastics pool and managed instance are all different managed service deployment options of Azure SQL Database.

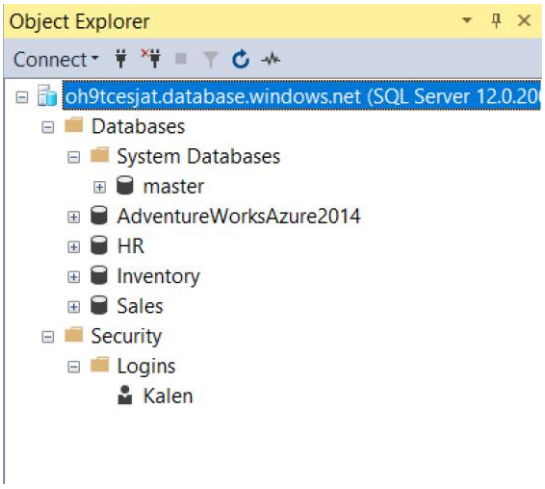
2.2 Architecture of Azure SQL Databases

From the subscriber's perspective, Azure SQL Database provides databases for application data storage. The data is presented to the subscriber through the logical database that abstracts the physical storage architecture and uses automatic load balancing and connection routing to access the data. The logical database that the subscriber creates and uses for their database storage is referred to as an Azure SQL Database.

When using a Premium Tier SQL DB, each of the databases hosted in the SQL Azure datacenter has at least three replicas: a primary and two secondaries, all of which use local storage for their database files. All reads and writes go through the primary replica and any changes are replicated to the secondaries synchronously. The replicas are the primary means of providing high availability for your Azure SQL DBs. The replicas are invisible to users, who will see just one access point to the Azure SQL Database. However, one of the secondaries can be used for read operations to enhance read performance.

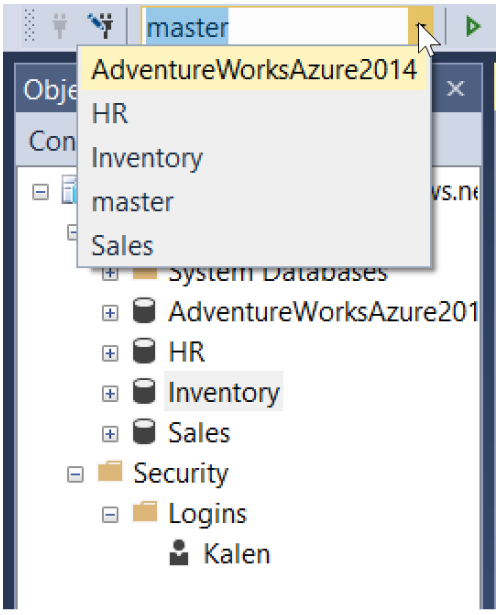
A Microsoft Azure Data Center will have hundreds or thousands of machines (depending on the region) with hundreds or thousands of actual SQL Servers to host the SQL Azure replicas. The primary replica of each Premium Tier SQL Database and each replica will have its own physical SQL Server, unless you are using elastic pools, which will be discussed below.

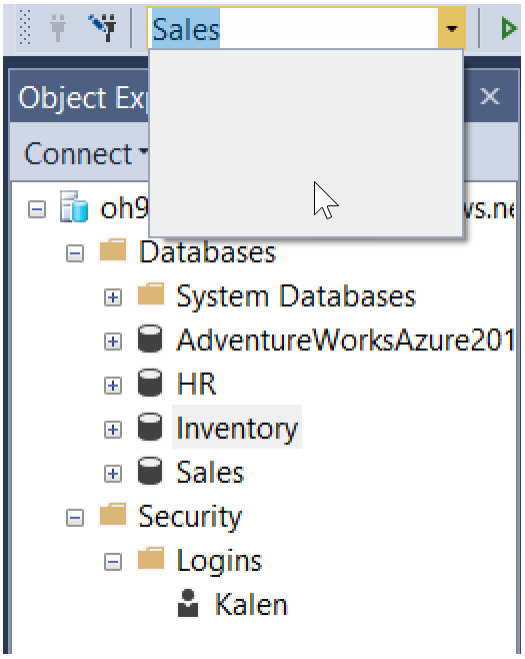
Each Azure SQL DB will be associated with an Azure logical server, which is basically just a logical container for managing a collection of databases or elastic pools belonging to the same subscriber. In addition, each logical server has a logical 'master' database, which is not the same as the *master* database you are familiar with in your on-prem SQL Server instances. When connecting to your logical SQL Server through a tool such as SQL Server Management Studio, it will look almost like you have a 'normal' SQL Server with a *master* database and possibly multiple user databases, as shown in the Figure below:



In the figure, *oh9tcesjat.database.windows.net* is the name of a logical server. It has four SQL DBs that were each created individually. Each of the databases, plus *master*, are stored completely separately. Each database, including what is shown as *master*, is stored as a separate user database on its own physical SQL Server instance in the data center. The *master* database contains information controlled by Azure, such as server-scoped credentials, and database and pool statistics. This data in master is readable, but not manageable by any users. Note that 'master' is the only system database accessible. Because the databases are created on a SQL Server instance, the other system databases (*tempdb*, *model* and *msdb)* exist, but they are never visible. The *tempdb* database, though invisible, is used for normal purposes, such as storing temporary tables, and providing space for sorting and the version store, if needed.

The physical separation of databases that all are part of one logical Azure SQL server means that each connection is tied to a single database, not a single SQL Server instance (except in the case of Managed Instance). For a connection to issue a USE command would mean that the TDS connection might have to be rerouted to a completely different physical machine in the data center, so the USE command is not supported for Azure SQL DB connections. Most client tools will make a connection to the logical SQL Server and then specify which database they need to access. Unlike databases on physical SQL Servers, once you have connected to a database, you will not be able to access any of the other databases managed by that logical SQL Server, either through the USE command or through the graphical interface. The exception is the 'master' database, which will be your context when you first connect to the logical server. From master, you can choose to connect to any of the other databases, as shown on the left below. However, once you are in another database, you will not be able to change context to another database, even master, as shown on the right.





For SQL Databases created using a Standard Tier, the storage is managed differently. There are no replicas created on other physical SQL Servers in the data center. In addition, the storage for the database files is remote storage. For S0, S1 and S2 databases, Azure Standard Storage is and for performance levels S3 and above, the data is stored on Azure Premium Storage. Even though the storage is remote, it is generally stored in the same on a nearby data center. The database storage is then replicated for high availability at the storage level, but not at the SQL Server (compute) level.

2.3 Differences between Azure SQL Database and an on-premises SQL Server

Although there are certain SQL Server features and capabilities that are only available in an on-prem SQL Server instance, the number of such features is getting smaller all the time. Most of the features that are missing having to do with physical configuration or control of your databases, such as altering files, or managing cross database or cross server operations. As mentioned above, the USE command is also not allowed. You can refer to this page to see more details of T-SQL syntax not supported or only partially support in Azure: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-transact-sql-information .

On the other hand, there are aspects of an Azure SQLDB that are not available on-prem. These features include some of the catalog views and DMVs as well as monitoring and troubleshooting capabilities, which will be discussed below.

2.4 Special metadata

An Azure SQL DB has certain metadata objects that are necessary for managing a database in the cloud than are not needed for your databases in your own data center. Some of these objects allow you to keep track of your resource usage and some allow you to manage your firewalls and other connection details. You can find a list here of the Azure SQL specific catalog views: https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/azure-sql-database-catalog-views

Be sure and note on the page for each object what version of Azure SQL Database it is valid with. Some of the view are only available from the logical server's master database, and some can be shut off by Microsoft when needed.

There are also some Dynamic Management Objects available in an Azure SQL DB that are not found in on-prem databases. Some of these are just variations of objects available on-prem but scoped to a single database. For example, in an on-prem database we could look at the wait statistics for the entire SQL Server instance with *sys.dm_os_wait_stats* but in Azure we have *sys.dm_db_wait_stats*.

2.5 Feature Support

Your Azure SQL Databases may support features that are not supported on-prem. Microsoft introduces new functionality to Azure first, as Azure is much easier to upgrade. Some of these features that were introduced first to Azure SQL DBs, that are now included in the on-prem SQL Servers, are: resumable index build, and adaptive query processing. In addition, there are features that are only supported above certain levels in an Azure SQL DB. For example, although columnstore indexes are available in a standard tier SQL DB, it must be S3 or above. Check the documentation for the features you're interested in to see what versions of SQL Server and what service tiers of Azure SQL DB they are available in.

2.6 Management

The primary job of a DBA is typically to manage your regular backups and perform consistency checks. These tasks are now handled by Microsoft within the data centers. Full backups are usually performed every week on every database, and differentials are performed once or twice a day. Log backups are made about every five minutes, but this can be extended if you are performing long running operations that take more than five minutes. Each database is assigned some backup storage when it is created, and a user with sufficient privilege can choose to restore from these backups to any point in time contained within the backups. In addition, a feature called geo-restore allows a restore of the latest database backup into any Azure region. This allows you to restore a recent backup into a different region in case of unavailability of the database in the original hosting region.

DBAs for an Azure SQL DB do not have any responsibility for the physical management and placement of files and filegroups. The files themselves are completely unavailable. All physical aspects of dealing with your databases are handled in the data center. On the other hand, the DBA does need to be aware of the amount of data space being used, to make sure it doesn't grow beyond the amount of storage requested. DBAs are also responsible for index maintenance and monitoring fragmentation and its effect on performance, if any. In addition, although *tempdb* is not directly accessible, a DBA does need to make sure that temporary tables do not grow too large and that transactions that use *tempdb* space, those that use the version store with snapshot isolation, are committed in a timely manner. Metadata is available to help monitor tempdb usage. Consider monitoring the output of *tempdb.sys.dm_db_file_space_usage* and *sys.dm_tran_version_store_space_usage*.

Upgrades and patching are also typical DBA tasks, and these are handled in the data center, one database or one replica at a time. In fact, upgrades to Azure SQL DBs happen much more often than SQL Server upgrades. Currently, there is a new SQL Azure version about every month or month and a half. Upgrades are handled using the same reconfiguration process used when a node goes down, because in fact, the node is made unavailable while the upgrade is being performed. In addition to upgrades of the SQL DB itself, the hosting SQL Server, as well as the operating system software, will occasionally need upgrades or patches, necessitating bringing the node offline temporarily.

2.7 Troubleshooting

Since you have no control over the physical provisioning or configuration of your machines or of the database files, your troubleshooting requirements will be minimized. The main types of troubleshooting you may have to do will involve poorly performing queries, and concurrency problems, i.e. blocking. Because your SQL Azure database are by default running in READ COMMITTED SNAPSHOT isolation level, your problems due to readers and writers blocking each other will most likely be minimized, but there will still be blocking issues due to other kinds of waits. In general, most of your tuning efforts will most likely involve query performance.

Some of the tools you can consider for monitoring and tuning your Azure databases are the following:

Extended Events

Almost all the xEvents available for an on-prem SQL Server that have to do with query performance are available in any of the Azure database options. Extended events allow you to see the duration of queries, batches and programming modules and find the ones taking the most time or using the most resources.

Dynamic Management Objects

Various DMVs allow you to monitor resource usage of your database connections and the queries that those connections are running. There are also DMVs that allow you watch all blocked or waiting processes. Many of SQL Server's and Azure SQL DB's monitoring tools are built on top of these DMVs. If you want to go beyond what the tools provide, you can access this information directly to pull out usage and performance data. For more information, see the following article: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-monitoring-with-dmvs

Query Store

Query Store was added to the boxed SQL Server product as of SQL Server 2016, but has been available in Azure SQL DB since November 2015. Although use of the feature is configurable, it is enabled by default in Azure databases. Query Store continuously collects and presents detailed historic information about all queries allowing you to detect changes in query performance, in particularly when they are due to a chance in the query plan. For more details about the benefits of Query Store, take a look at the following article: https://docs.microsoft.com/en-us/sql/relational-databases/performance/query-store-usage-scenarios For more information about using Query Store with your Azure SQL DBs, see the following: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-operate-query-store

Intelligent Insights

Intelligent Insights is available only in Azure SQL DBs because it is based on collected telemetry that only exists in Azure. This telemetry needs to be captured so that Microsoft can manage your SQL Server services and then this telemetry can be used by the users for additional analysis. The infrastructure to support Intelligent Insights does not exist in the on-prem SQL Server products. This page https://docs.microsoft.com/en-us/azure/sql-database/sql-database-intelligent-insights has this description of Intelligent Insights:

> *Intelligent Insights uses built-in intelligence to continuously monitor database usage through artificial intelligence and detect disruptive events that cause poor performance. Once detected, a detailed analysis is performed that generates a diagnostics log with an intelligent assessment of the issue. This assessment consists of a root cause analysis of the database performance issue and, where possible, recommendations for performance improvements.*

Intelligent Insights is configured and managed from Azure Portal. This article can get you started: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-metrics-diag-logging

2.8 Service Tiers

When you create an Azure SQL Database, you specify both a Service Tier and a performance level. Tiers are Standard and Premium. (It was mentioned earlier that there is also a Basic Tier, used for less demanding workloads.) Standard Tier is appropriate for most production workloads and Premium should be considered for IO-intensive operations. As mentioned above, part of the reason for this recommendation is that the storage for Premium Tier databases is local storage, and for Standard (and Basic) it is remote, and I/O latency can be higher. Within the service tier you can select a number for the performance level, which controls the number of DTUs and you also specify several gigabytes of storage requested.

For Standard, you can choose between S0, giving you 10 DTUs and S12, providing 3000 DTUs. You can choose anywhere between 100 MB and 1 TB of storage for your database, with different maximum values available for different DTU choices. The combination of these two choices determines your monthly cost. For Premium, you can choose between P1, for 125 DTUs and P15, providing 4000 DTUs. Storage options vary from 100 MB to 4 TB. Other resources are also dependent on the Performance level and Service Tier, including number of concurrent sessions and logins. Take a look at this article for details regarding the limits in the various Service and Performance Tiers: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-resource-limits. *Tempdb* resource limits are also dependent on the Tier being using and the article on the tempdb contains details on the tempdb limits for Azure SQL DB: https://docs.microsoft.com/en-us/sql/relational-databases/databases/tempdb-database#tempdb-database-in-sql-database

Performance tiers and levels can be changed, either within the Service Tier or to a different Service Tier. Keep in mind, however, that if you are changing to, from, or within Premium, the latency to complete the change is generally higher than to change within Standard or Basic. This is because Premium uses local storage on the physical server where the SQL Server is running. (This machine where the SQL Server is running is referred to as the compute instance.) Changing the storage amount or the number of DTUs generally involves moving to a new physical SQL Server, which requires copying the data from the source compute instance to the compute instance on a target machine. Database storage in the Basic/Standard tiers is provisioned remotely. Changing the performance level within Standard or Basic doesn't involve copying data and instead only needs to detach the remote database files for the database from the source compute instance and attach them to a target compute instance that has sufficient capacity to accommodate the change request.

Note that any performance level change involves a final step at the end of the operation (that is, after the data transfer is complete in the Premium tier case) where all connections from the source instance (using the previous performance level) are disconnected, to allow the service to transfer routing to the target instance (using the new performance level). During this brief period, usually on the order of a few seconds, the database is inaccessible. Once the transfer of the instance end points is complete, any new connections (retries) from the client will automatically be directed to the target instance.

2.9 High Availability

Hight available is managed automatically in Azure SQL Databases. If you're using a Premium Tier, server failures within the Azure data center are handled by just promoting one of the secondary replicas to become the primary replica. The replicas basically act as hot standbys. . For standard and basic tier, as mentioned above, the database storage is replicated for high availability at the storage level, but not at the SQL Server (compute) level. No matter the Service Tier, any active connections will need to reconnect, but all committed transactions are durable. It is also possible to configure geo-replication across data centers or across the globe. The details on all the options available is well beyond the scope of this document. Please see the document below for more information on using high availability features with Azure

https://docs.microsoft.com/en-us/azure/sql-database/sql-database-business-continuity

2.10 SQL Agent Jobs

With Azure SQL DB, as you only have access to a single database, there is no connection to the *msdb* database and no ability to create SQL Agent jobs. However, an Azure Automation Service is available that provides a consistent

management across your Azure and non-Azure environments, consisting of process automation, update management and configuration features.

For more details on Azure Automation, please see this document: https://docs.microsoft.com/en-us/azure/automation/

Another option for replacing SQL Agent jobs in Azure SQL DB is a feature called 'elastic jobs' (which are not just for elastic pools.) Elastic jobs provide similar functionality to SQL Server Agent although the scope of job targets is broader than SQL Server Agent. For example, a single elastic job agent can run jobs against single databases, elastic pools, or databases in pools across different logical servers in different data center regions.

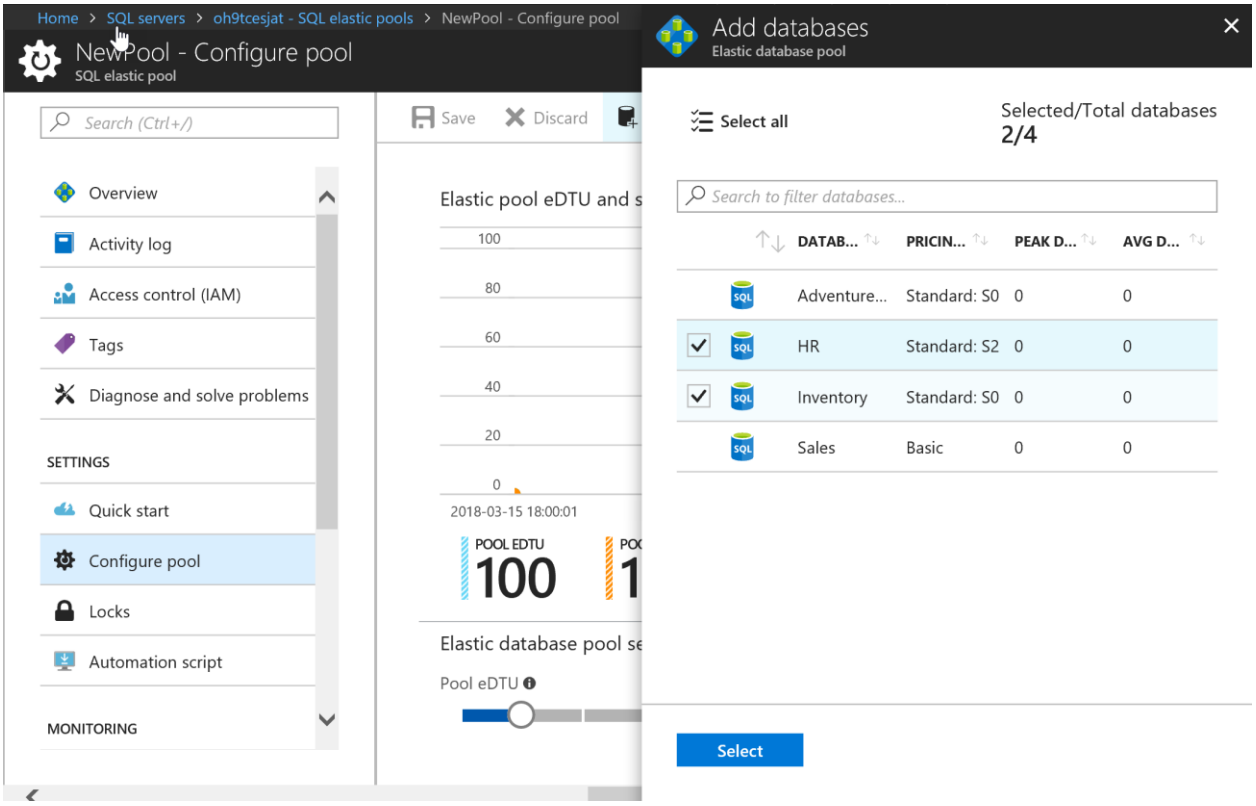
For more details on Elastic Jobs, please see this document: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-elastic-jobs-overview

The following chart provides a high-level comparison between the two SQL Agent job replacements:

	Pros	Cons
Elastic jobs in Azure SQL DB	• Native support for target database groups such as servers and pools. • Native semantics for job step execution, retries and guaranteed completion. • Familiar environment similar to SQL Agent.	• Can't perform management operations outside of SQL (restricted to T-SQL).
Azure Automation	• Integrated more Azure services than just SQL DB service. • Can implement management operations not available in T-SQL.	• Programming customization required to manage SQL DB targets. • PowerShell based – potentially requires ramp up.

3 ELASTIC POOLS

Elastic pools are another deployment option for Azure SQL DB when you have multiple databases to manage. SQL Database elastic pools are a simple, cost-effective solution for managing and scaling multiple databases that have varying and unpredictable usage demands. In fact, if you have already created multiple SQL Databases on the same logical server, you can combine two or more of them into an elastic pool through the Azure Portal. An elastic pool is created like any other resource, and you specify a service tier and a performance level which determines the number of special DTUs called eDTUs. Once the pool is created you can further configure the pool by selecting Add Database from the menu, and you will be shown the existing Azure SQL Databases on the logical server, as shown in the figure below. Alternatively, you can create new SQL databases and place them in the pool at creation time.



Elastic pools can allow you to share computing resources much more economically that having each SQL DB separate. Elastic pools do not provide any more options for sharing data and connection information between databases; their benefit is to allow sharing compute resources in terms of eDTUs and sharing storage and other resources on the SQL Server. All the databases in the elastic pool are hosted on the same physical SQL Server instance in the data center, so they share the same memory, the same I/O, the same *tempdb* and the same CPUs. Behind the scenes, SQL Server Resource Governor is managing many of the shared resources. It allows you to set a minimum and maximum eDTU threshold for all the member databases. This setting will help minimize the changes that one database could consume all the memory, CPU and I/O resources, and that each database will always have at least a minimum amount of these resources available. Note that there is one minimum eDTU value and one maximum eDTU value for the entire elastic pool, that applies to all SQL Databases in the pool. The settings do not guarantee sufficient resources always, because the memory provided by the eDTU choice only refers to the memory used by the buffer pool. Resource intensive queries that need large memory grants for operations such as sorting, or hashing will not be constrained by these limits and a workload in one database can starve workloads in other

databases. In addition, all SQL Databases in the pool share the same *tempdb* on the physical instance, and there is no configuration possible for *tempdb* usage. One SQL Database that makes excessive use of *tempdb*, through temp tables or version store usage, can drastically affect the *tempdb* resources available to other SQL Databases in the pool.

One of the most straightforward examples of the value of elastic pools is an application that needs to create a separate database for each client, but not all clients are using their resources at the same time. Consider an accounting firm's application that needs to keep each client's financial data strictly separated from all others.

When you create an elastic pool, the requested performance level specifies a total number of eDTUs for all databases in the pool combined. Elastic pool DTUs are more expensive that DTUs for singleton databases, but it frequently is worthwhile. So, here's an example. Let's assume DTUs for singleton databases are $1.50 per month each and eDTUs for elastic pools are $2.25 per month each. Suppose you have 100 clients, and none of them will ever need more than 100 DTUs. If each client has an S3 Azure SQL DB that can use up to 100 DTUs, it would cost $1.50 * 100 or $150/month each. But most of the time, the databases are quiet, and hardly use any DTUs. If at most 10 databases will be using their maximum DTUs concurrently, you could use an elastic pool with 100 SQL DBs, and purchase 1000 eDTUs. Although each eDTU is more expensive by about 50%, 1000 eDTUs at $2.25 are far cheaper than 100 separate databases using 100 DTUs each. The cost is $2,250 total for the pool containing all 100 databases as compared to $15,000 total if each database were provisioned in its own S3 performance level. .

4 MANAGED INSTANCES

The newest offering for your relational databases managed by Microsoft Azure is a Managed Instance, available in public preview in March 2018, in a limited number of regions.

4.1 Overview

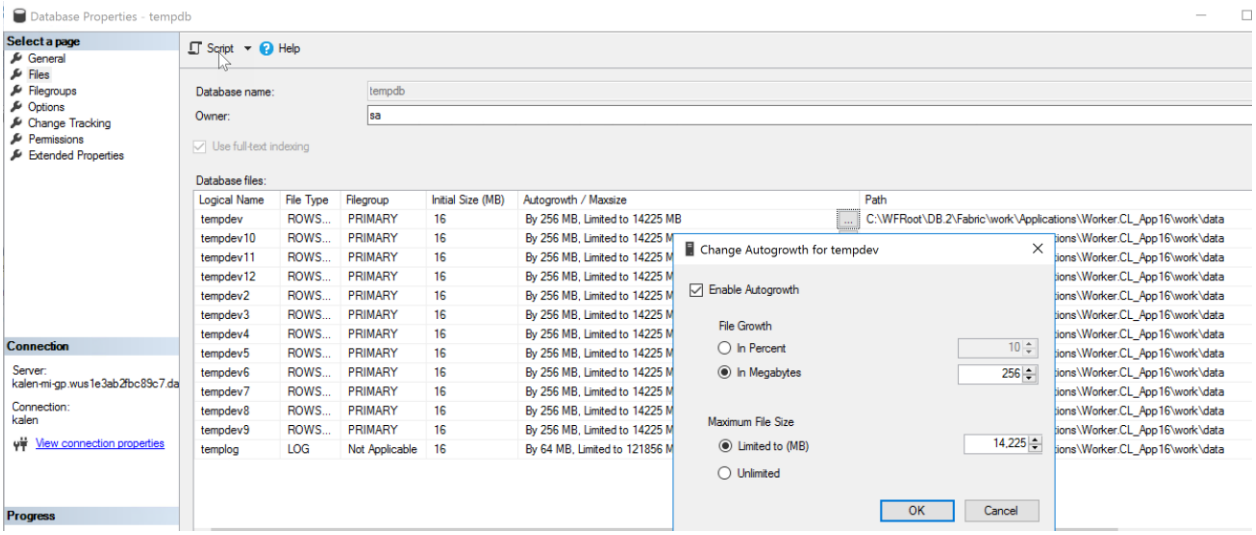
Azure SQL Database Managed Instance provides close to full compatibility with SQL Server on-premises, using a native Virtual Network (VNet) implementation to address common security concerns and a pricing model favorable for on-premises SQL Server customers. Managed Instance allows existing SQL Server customers to lift and shift their on-premises applications to the cloud with minimal, or even zero, application and database changes. At the same time, Managed Instance preserves all PaaS capabilities (automatic patching and version updates, backup, high-availability), that drastically reduces management overhead and ownership costs.

4.2 Capabilities of Managed Instance

Initially, Managed Instance is only available in a single service tier called General Purpose (GP). With this tier you can request up to 8 TB of Azure Premium Storage, and you can create up to 100 databases. With general availability, you will also have the option to create a managed instance using Business Critical (BC)tier. Like Premium Tier, the BC Tier for managed instance using local storage on fast SSDs.

With Managed Instance, each instance has its own storage account and its own physical SQL Server in the datacenter. Microsoft Azure manages redundancy to provide high availability as it does for the other PaaS offerings.

Because you have access to the full instance, you have access to *tempdb*. You can configure certain properties, such as the number and maximum size of the files. Note, that because MI is not IaaS, you don't have access the operating system and the physical placement of the files. *Tempdb* in a MI currently creates 12 data files with a maximum size of 14GB each. You can use either the ALTER DATABASE/MODIFY FILE command to increase that maximum size, or you can use SQL Server Management Studio, as shown below:



One thing to note in the public preview is that any changes you make to the settings are not preserved after a fail-over, when the SQL Server is restarted and *tempdb* is rebuilt. However, since SQL Agent is available in a MI, you can create a job that runs a script to configure the *tempdb* upon restart.

Having SQL Agent and *msdb* available means jobs and alerts can be created to manage and monitor all databases on the instance. Having access to the entire instance means you can use the USE command and run cross-database operations if needed. Other features available in Managed Instance include

- Native restore (from URL)
- Control of database file layout (add/modify file, file groups)
- CLR
- Service Broker
- Change Data Capture
- Transactional Replication (coming soon)

Managed Instance provides built-in high availability and recovery functionality, just like Azure SQL Database. Managed Instance uses a native VNet implementation and private IP addresses to guarantee full isolation and security. In addition, the following security features are available with MI:

- Auditing
- Always Encrypted
- Dynamic Data Masking
- Row-level security
- Threat Detection

- By General Availability, Managed Instance aims to deliver close to 100% surface area compatibility with the latest on-premises SQL Server version through a staged release plan. At present, there are some gaps between what is available on-prem and what is available in MI, and most of the differences have to do with statements that interact with the physical machine or the operating system. Check out https://docs.microsoft.com/en-us/azure/sql-database/sql-database-managed-instance-transact-sql-information for details

For more information about Managed Instance in Azure SQL DB, please see the following articles:

What is a Managed Instance (preview)?
Azure SQL Database pricing
Azure Hybrid Use Benefit (AHUB)
Azure SQL Database Managed Instance T-SQL differences from SQL Server
Create Managed Instance - Tutorial

5 MIGRATION

No matter what PaaS option you choose, there are tools and documentation available to help you migrate your databases. Migration can mean moving one or many databases from your datacenter to the Azure, or it can mean moving databases already in Azure using IaaS to a PaaS environment.

5.1 Data Migration Assistant

The Data Migration Assistant (DMA) can allow you to easily move any SQL Server database to an Azure SQL DB, by just creating an empty Azure SQLDB and using this tool to move all your data and objects. This method of migration can be quite simple but might require substantial downtime. As an alternative to DMA, you could also use a BACPAC file to import the database. This article provides more details: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-import .

5.2 Transactional Replication

Another method of migrating your database which could involve considerably less downtime is to use SQL Server Transaction Replication to migrate your data. The Azure SQL DB would be configured as the subscriber and all changes to data and schema would show up in the Azure SQL DB during the synchronization process. For details on both these first two methods (using DMA or replication), see https://docs.microsoft.com/en-us/azure/sql-database/sql-database-cloud-migrate

5.3 Data Migration Service

The new Azure Database Migration Service is a fully managed service designed to enable seamless migrations from multiple database sources to Azure Data platforms with a minimum of downtime. This service streamlines the tasks required to move your SQL Server databases to any of the Azure PaaS or IaaS offerings. The service utilizes the Data Migration Assistant (DMA) which can generate reports of recommendations as well as usage patterns that might block your migration. You will need to perform any remediation to make sure your database is ready for migration. Once you start, the Azure Database Migration Service performs all the necessary steps, so you can just fire it up and forget about it until the migration is complete. https://docs.microsoft.com/en-us/azure/dms/dms-overview

For more information about migrating to Azure SQL Database, please see the following articles:

Choosing your database migration path to Azure
Azure Migration Center

6 SQL SERVER IN AN AZURE VM

6.1 Description

A final option for hosting your SQL Server database and operations in the Azure cloud is to use an Azure Virtual Machine. In this case you are using IaaS, and you have complete control over the operating system. The storage, network and machine resources are managed by Microsoft, but everything else is up to you. You can choose to have a specific operating system preinstalled when the VM is deployed, but that is not required. And you can later change to a different operating system at a later time, if desired.

When running SQL Server on an Azure VM, you would be responsible for any upgrades and patches, as well as for making and managing the SQL Server database backups. You would also be responsible for configuring and managing whatever High Availability and Disaster Recovery (HA/DR) solutions you wished to implement. All tuning of the platform and the database would need to be done by you. However, even though patching, backups and HA/DR activities are not automatically provided when running SQL Server in an Azure VM, these are add-on features you can request through the Portal.

To request automated patching, see this document: https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sql/virtual-machines-windows-sql-automated-patching

To request automated backups, see this document: https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sql/virtual-machines-windows-sql-automated-backup

You can also configure azure key vault integration for SQL Server on Azure Virtual Machines. Take a look here: https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sql/virtual-machines-windows-ps-sql-keyvault

Finally, there are templates available for deploying Always On availability groups within an Azure VM. You can get details here: https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sql/virtual-machines-windows-portal-sql-alwayson-availability-groups

6.2 Best practices

You can consider using an Azure VM as a starting point, to get your feet wet, so to speak, in the Azure world. An Azure VM might be the most straightforward solution to just a lift and shift operation of moving your entire database solution and all supporting files to a cloud location. Once you get comfortable with working with an Azure-based solution, you could start moving to one of the other options, based on your requirements.

You might also consider running your SQL Server solution in an Azure VM if you need a specific version of SQL Server that isn't supported in any of the Azure SQL DB or Managed Instance options. Azure VM might also be a worthwhile option if you need to have additional services running on the same box as SQL Server, such as SSIS, SSAS or SSRS. Some of these additional services have a corresponding Azure service, but they are not all 100% compatible. Similarly, if you have third-party monitoring tools that work best when run on the same infrastructure as SQL Server, the VM option might be considered.

Finally, you might find Azure VMs the best option for you if need more resources than the other options can provide. A Managed Instance can support between 8 and 32 processors and a maximum storage size of 35 TB. An Azure VM can support up to 128 processors and up to 256 TB of storage.

7 AZURE OPTIONS FOR SQL SERVER RELATIONAL DATABASE WORKLOADS

The following chart provides usage scenarios for each of the deployment options discussed in this document.

Deployment Option	Usage scenarios and considerations

Azure SQL Database	Single Databases: For new applications, whose workload is stable and predictable, consider single databases. Benefits of this model are:
	• A simplified database-centric programming model
	• Predictable performance for each database
	• Three Service Tiers are available:
	  ○ Basic – for apps with light workloads
	  ○ Standard - for mid-level performance ad business continuity
	  ○ Premium - for low I/O latency and highest business continuity
	
	Elastic Pools: For developing new SaaS multi-tenant applications or intentionally transforming existing on-premises apps into a SaaS multi-tenant application, consider elastic pools. Additionally, elastic pools are an appropriate choice for server consolidation of multiple databases where only a small fraction have concurrent resource demands. Benefits of this model are:
	• Conversion of the business model from selling licenses to selling service subscriptions (for ISVs)
	• Easy and bullet-proof tenant isolation
	• A simplified database-centric programming model
	• Resource sharing across databases of existing Line-of-Business apps for higher efficiency
	• Same Service Tiers as for Single Databases
	
	Managed Instances: For modernizing many existing SQL Server applications from on-premises or IaaS, with the lowest migration friction. Using the fully automated Data Migration Service (DMS) in Azure, lift and shift on-premises SQL Server to a Managed Instance that offers compatibility with SQL Server on-premises and complete isolation of customer instances with native VNET support.
	• Two Service Tiers are available:
	  ○ General Purpose
	  ○ Business Critical (in public preview)

SQL IaaS (VMs)	For customizing the operating system or the database server, as well as satisfying specific requirements in terms of running third-party apps by side with SQL Server (on the same VM), consider SQL VMs / IaaS as the optimal solution.

8 COMPATIBILITY

Through SQL Server 2016, applications running on SQL Server could be 'certified' to work with a SQL Server version, and Microsoft provided guidance and even tools to help verify which version of SQL Server your applications were guaranteed to be compatible with. This is particularly important for ISVs selling applications that work with the latest versions of the Microsoft SQL Server product. Here's an example of the tool verify compatibility with SQL Server 2016: https://www.microsoft.com/en-us/download/details.aspx?id=54430

However, because Azure SQL Databases and now Managed Instances, are changing much more rapidly that the on-prem versions of SQL Server, certifying that your application works on Azure was becoming problematic. Going forward, Microsoft will no longer be certifying applications to work for a version, but rather at a compatibility level. If you are using an application that is certified for a compatibility level, you can just make sure that the Azure SQL Database that you are using is set to that level. Your database's compatibility level can be changed with the ALTER DATABASE command described here: https://docs.microsoft.com/en-us/sql/t-sql/statements/alter-database-transact-sql-compatibility-level

Using a specific compatibility level means that your database has the functionality of the associated version of SQL Server. However, there is no specific guarantee that the performance of queries will be the same as when running on the associated version. The following table shows the relationship between compatibility level and SQL Server version:

Compatibility Level	SQL Server version
140	SQL Server 2017
130	SQL Server 2016
120	SQL Server 2014
110	SQL Server 2012
100	SQL Server 2008

9 CONCLUSION

Microsoft Azure provides both IaaS and PaaS options for building, deploying and managing SQL Server database applications in the cloud. Azure SQL Database is a PaaS option with three deployment models: single database, elastic pools and managed instance. These three options provide efficient cost effect access to your data in the cloud, with many DBA services automatically managed such as backups and high availability. Azure VM is the IaaS offering that gives you full control over your operating system and database platform. You can choose to employ additional Azure services that perform management operations.

Summarize the key points from your document. At the end of your conclusion, place the following More Information and Feedback sections:

For more information:

SQL Server

Azure SQL Database

New DBA in the cloud – Managing your database in Azure SQL Database

SQL Server Documentation

Azure SQL Database Documentation

Did this paper help you? Please give us your feedback. Tell us on a scale of 1 (poor) to 5 (excellent), how would you rate this paper and why have you given it this rating? For example:

- Are you rating it high due to having good examples, excellent screen shots, clear writing, or another reason?
- Are you rating it low due to poor examples, fuzzy screen shots, or unclear writing?

This feedback will help us improve the quality of white papers we release.

Send feedback