Architecture

Last updated by | Vitor Tomaz | Aug 5, 2020 at 12:45 PM PDT

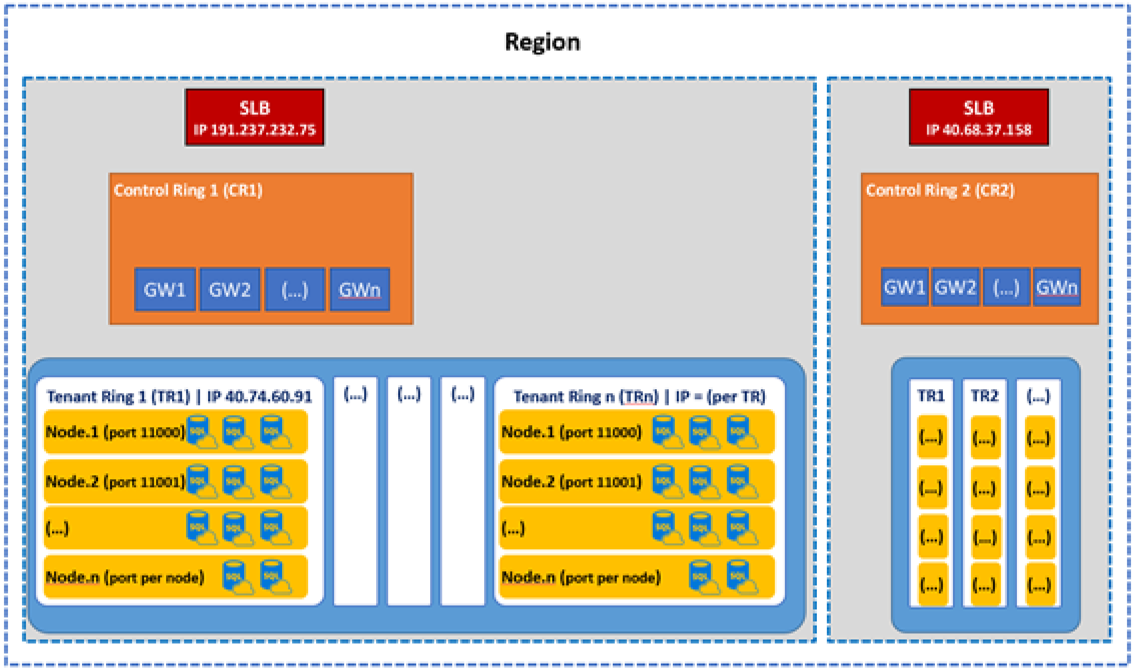
Contents

High-level architecture review of Azure SQL DB single/pools

To understand the differences in the Managed Instance architecture let's review the main concepts of Azure SQL DB Single/Elastic Pools:

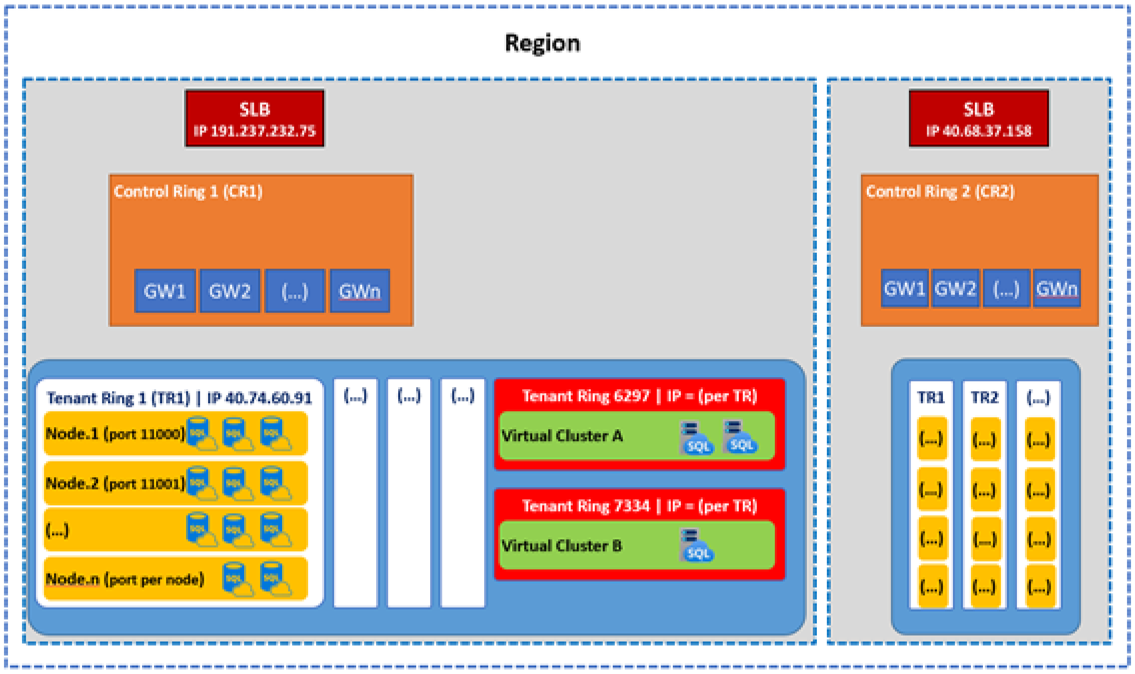
- Each region has multiple Control Rings (CR1, CR2, etc).
- Usually there are CRs dedicated for Control Plane and CRs dedicated to Data Plane.
- Each data plane control ring has a SLB (software load balancer) as entry point and multiple gateway nodes (GW.0, GW.1, etc.).
- Each data plane control ring takes care of a set of many Tenant Rings (TRs), 10's or 100's of them.
- Each individual TR has a public IP address.
- Each individual TR has many Nodes (DB.0, DB.1, etc.), 10's or 100's of them.
- Each node will have a port in the 11000-11999 range associated.
- Each node can have several databases running on them.
- Redirect connection policy will use the TR IP address and port associated with the Node to connect to your database directly after redirected from the Gateway.

In the picture below you can see the concepts listed before:



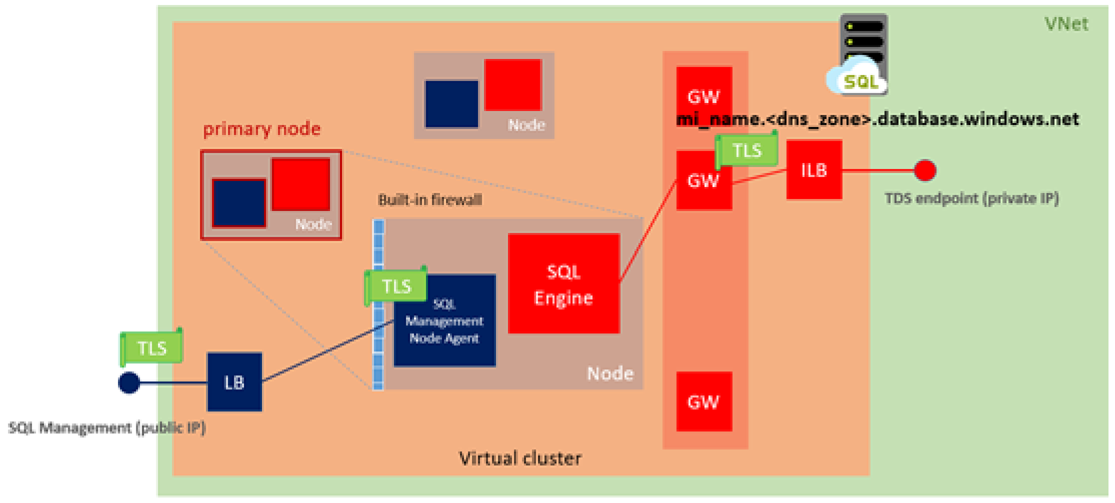
High-level architecture of Managed Instance

At a high level, a managed instance is a set of service components. These components are hosted on a dedicated set of isolated virtual machines that run inside the customer's virtual network subnet. These machines form a virtual cluster. A virtual cluster can host multiple managed instances. If needed, the cluster automatically expands or contracts when the customer changes the number of provisioned instances in the subnet. Because of this isolation, each subnet where managed instance is, will be associated with an entire Tenant Ring. These TRs will be way smaller when compared with the TRs for Single/Pools. Each Tenant Ring will have one Virtual Cluster. In case you have Managed Instances with different hardware generations, each generation will be in a different Virtual Cluster (and also a different TR).



Virtual cluster architecture

Let’s investigate virtual cluster architecture, the following diagram shows the conceptual layout of a virtual cluster.


VNet
SQL
GW
mi_name.<dns_zone>.database.windows.net
TLS
GW
ILB
TDS endpoint (private IP)
Node
primary node
Node
Built-in firewall
SQL Engine
TLS
SQL Management Node Agent
Node
TLS
LB
SQL Management (public IP)
GW
Virtual cluster

Clients connect to a managed instance by using a host name that has the form <mi_name>.<dns_zone>.database.windows.net.

This host name resolves to a private IP address. This private IP address belongs to the managed instance's internal load balancer. The load balancer directs traffic to the managed instance's gateways. The virtual cluster has several gateways.

Note that this architecture is different from SQL DB Single/Pools because they have dedicated gateways. In Single/Pools, there is a set of gateways shared for the entire control ring. Managed Instance customers don’t use the control ring gateways, they use gateways inside their virtual cluster.

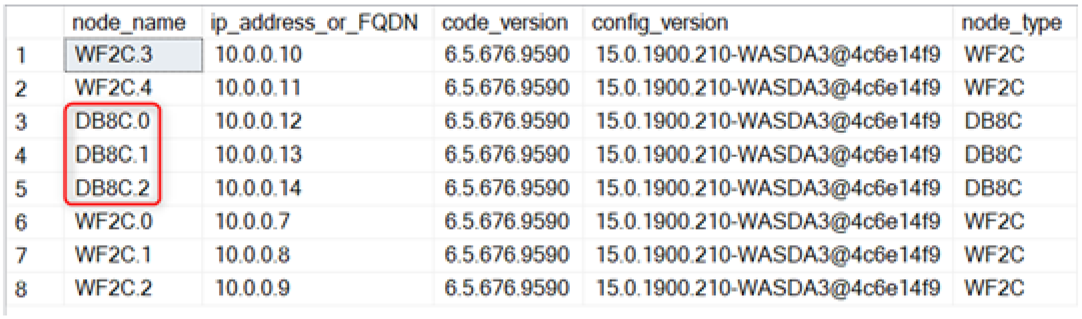
The zone-id is automatically generated when you create the cluster. If a newly created cluster hosts a secondary managed instance, it shares its zone ID with the primary cluster. Because multiple managed instances can run inside the same cluster, the gateway uses the managed instance's host name to redirect traffic to the correct SQL engine service. Gateways will redirect or proxy traffic to the instances where multiple nodes will be available for each Managed Instance. The number of nodes and their sizes is based on the offer (General Purpose or Business Critical) and the vCore size.

Each General Purpose will have 3 compute instances, one active and 2 spares in case something goes wrong. General Purpose availability model that is based on a separation of compute and storage. It relies on high availability and reliability of the remote storage tier. Going into mode detail, the General Purpose availability model includes two layers: A stateless compute layer that runs the sqlservr.exe process and contains only transient and cached data, such as TempDB, model databases on the attached SSD, and plan cache, buffer pool, and columnstore pool in memory. This stateless node is operated by Azure Service Fabric that initializes sqlservr.exe, controls health of the node, and performs failover to another node if necessary.

A stateful data layer with the database files (.mdf/.ldf) that are stored in Azure Blob storage. Azure blob storage has built-in data availability and redundancy feature. It guarantees that every record in the log file or page in the data file will be preserved even if SQL Server process crashes.

Whenever the database engine or the operating system is upgraded, or a failure is detected, Azure Service Fabric will move the stateless SQL Server process to another stateless compute node. Data in Azure Blob storage is not affected by the move, and the data/log files are attached to the newly initialized SQL Server process. This process guarantees 99.99% availability, but a heavy workload may experience some performance degradation during the transition since the new SQL Server instance starts with cold cache.

As mentioned before, Managed Instance has 3 compute nodes dedicated to this stateless compute layer when instance is General Purpose. You can see them below as DB8C nodes (WF nodes are Service Fabric nodes).

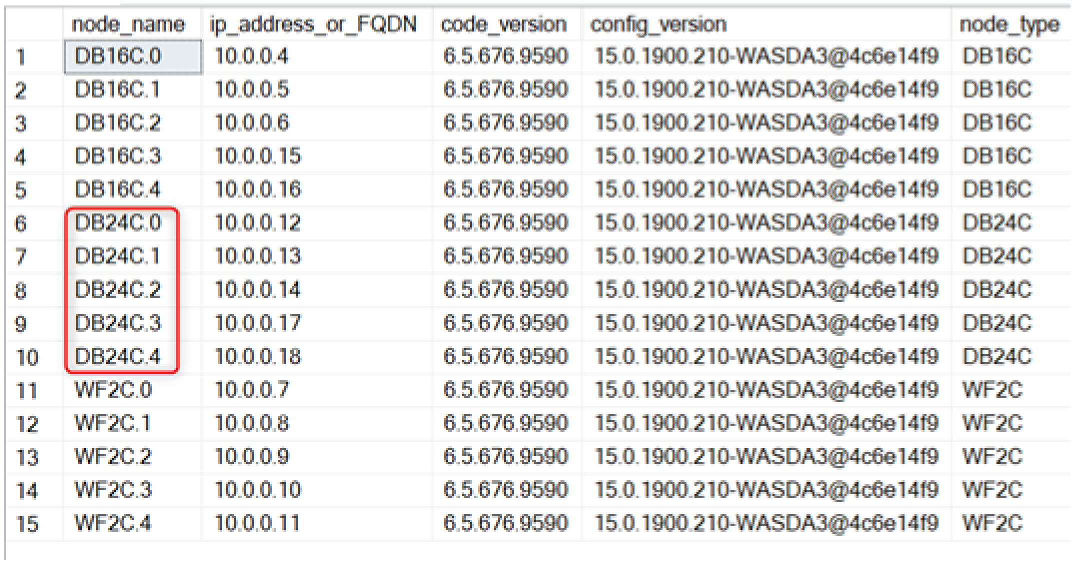


	node_name	ip_address_or_FQDN	code_version	config_version	node_type
1	WF2C.3	10.0.0.10	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	WF2C
2	WF2C.4	10.0.0.11	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	WF2C
3	DB8C.0	10.0.0.12	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	DB8C
4	DB8C.1	10.0.0.13	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	DB8C
5	DB8C.2	10.0.0.14	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	DB8C
6	WF2C.0	10.0.0.7	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	WF2C
7	WF2C.1	10.0.0.8	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	WF2C
8	WF2C.2	10.0.0.9	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	WF2C

Business Critical service tiers leverage the Premium availability model, which integrates compute resources (SQL Server Database Engine process) and storage (locally attached SSD) on a single node. High availability is achieved by replicating both compute and storage to additional nodes creating a three to four-node cluster.

The underlying database files (.mdf/.ldf) are placed on the attached SSD storage to provide very low latency IO to your workload. High availability is implemented using a technology similar to SQL Server Always On Availability Groups. The cluster includes a single primary replica (SQL Server process) that is accessible for read-write customer workloads, and up to three secondary replicas (compute and storage) containing copies of data. The primary node constantly pushes changes to the secondary nodes in order and ensures that the data is synchronized to at least one secondary replica before committing each transaction. This process guarantees that if the primary node crashes for any reason, there is always a fully synchronized node to fail over to. The failover is initiated by the Azure Service Fabric. Once the secondary replica becomes the new primary node, another secondary replica is created to ensure the cluster has enough nodes (quorum set). Once failover is complete, SQL connections are automatically redirected to the new primary node. As an extra benefit, the premium availability model includes the ability to redirect read-only SQL connections to one of the secondary replicas. This feature is called Read Scale-Out. It provides 100% additional compute capacity at no extra charge to off-load read-only operations, such as analytical workloads, from the primary replica.

Each Business Critical in Managed Instance will have 5 replicas deployed, not all used at the same time. You can see them below as DB24C nodes (WF nodes are Service Fabric nodes). The nodes with DB16C are related with another managed instance in the same Virtual Cluster.



	node_name	ip_address_or_FQDN	code_version	config_version	node_type
1	DB16C.0	10.0.0.4	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	DB16C
2	DB16C.1	10.0.0.5	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	DB16C
3	DB16C.2	10.0.0.6	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	DB16C
4	DB16C.3	10.0.0.15	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	DB16C
5	DB16C.4	10.0.0.16	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	DB16C
6	DB24C.0	10.0.0.12	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	DB24C
7	DB24C.1	10.0.0.13	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	DB24C
8	DB24C.2	10.0.0.14	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	DB24C
9	DB24C.3	10.0.0.17	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	DB24C
10	DB24C.4	10.0.0.18	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	DB24C
11	WF2C.0	10.0.0.7	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	WF2C
12	WF2C.1	10.0.0.8	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	WF2C
13	WF2C.2	10.0.0.9	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	WF2C
14	WF2C.3	10.0.0.10	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	WF2C
15	WF2C.4	10.0.0.11	6.5.676.9590	15.0.1900.210-WASDA3@4c6e14f9	WF2C

Management endpoint

In order to manage the Virtual Cluster and everything inside it, we need an entry point. SQL Management and deployment services connect to a managed instance by using a management endpoint that maps to an external load balancer. Traffic is routed to the nodes only if it's received on a predefined set of ports that only the managed instance's management components use. A built-in firewall on the nodes is set up to allow traffic only from Microsoft IP ranges. Certificates mutually authenticate all communication between management components and the management plane. To find the endpoint's IP address, see Determine the management endpoint's IP address. When connections start inside the managed instance (as with backups and audit logs), traffic appears to start from the management endpoint's public IP address. You can limit access to public services from a managed instance by setting firewall rules to allow only the managed instance's IP address. For more information, see Verify the managed instance's built-in firewall. Traffic that goes to Azure services that are inside the managed instance's region is optimized and for that reason not NATed to managed instance management endpoint public IP address. For that reason, if you need to use IP based firewall rules, most commonly for storage, service needs to be in a different region from managed instance.

Networking architecture

Managed Instance needs to be deployed in a dedicated subnet inside the virtual network. The subnet must have these characteristics:

- Dedicated subnet: The managed instance's subnet can't contain any other cloud service that's associated with it, and it can't be a gateway subnet. The subnet can't contain any resource but the managed instance, and you can't later add other types of resources in the subnet.
- Network security group (NSG): An NSG that's associated with the virtual network must define inbound security rules and outbound security rules before any other rules. You can use an NSG to control access to the managed instance's data endpoint by filtering traffic on port 1433 and ports 11000-11999 when managed instance is configured for redirect connections.
- User defined route (UDR) table: A UDR table that's associated with the virtual network must include specific entries.
- No service endpoints: No service endpoint should be associated with the managed instance's subnet. Make sure that the service endpoints option is disabled when you create the virtual network.
- Enough IP addresses: The managed instance subnet must have at least 16 IP addresses. The recommended minimum is 32 IP addresses. You can deploy managed instances in the existing network after you configure

it to satisfy the networking requirements for managed instances.

To improve customer experience and service availability, Microsoft applies a network intent policy on Azure virtual network infrastructure elements. The policy can affect how the managed instance works. This platform mechanism transparently communicates networking requirements to users. The policy's main goal is to prevent network misconfiguration and to ensure normal managed instance operations. When you delete a managed instance, the network intent policy is also removed.

To address customer security and manageability requirements Managed Instance is transitioning from manual to service-aided subnet configuration. With service-aided subnet configuration user is in full control of data (TDS) traffic while Managed Instance takes responsibility to ensure uninterrupted flow of management traffic in order to fulfill SLA. With service-aided subnet configuration, the managed instance's subnet needs to be delegated to Microsoft.Sql/managedInstances resource provider.

Why enough IP addresses?

Azure SQL Database Managed Instance must be deployed within an Azure virtual network (VNet). The number of Managed Instances that can be deployed in the subnet of VNet depends on the size of the subnet (subnet range).

When you create a Managed Instance, Azure allocates several virtual machines depending on the tier you selected during provisioning. Because these virtual machines are associated with your subnet, they require IP addresses. To ensure high availability during regular operations and service maintenance, Azure may allocate additional virtual machines. As a result, the number of required IP addresses in a subnet is larger than the number of Managed Instances in that subnet.

By design, a Managed Instance needs a minimum of 16 IP addresses in a subnet and may use up to 256 IP addresses. If you plan to deploy multiple Managed Instances inside the subnet and need to optimize on subnet size.

The first thing to consider is that Azure reserves 5 IP addresses within each subnet. These are x.x.x.0-x.x.x.3 and the last address of the subnet (x.x.x.1-x.x.x.3 is reserved in each subnet for Azure services):

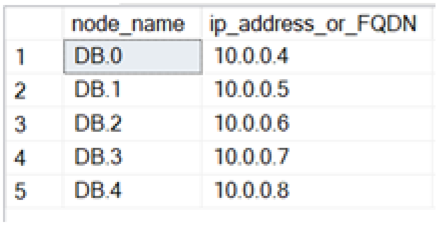
- x.x.x.0: Network address
- x.x.x.1: Reserved by Azure for the default gateway
- x.x.x.2, x.x.x.3: Reserved by Azure to map the Azure DNS IPs to the VNet space
- x.x.x.255: Network broadcast address

The second consideration is that each virtual cluster in the subnet will use 1 IP address because each one will have its own Load Balancer (ILB) and each ILB will have its own IP address. Running nslookup against the MI DNS record will reveal what is the IP address of the ILB. Usually the addresses for the ILB are given from the back: <max address="" in="" address="" range=""> - 1 As an example, in a subnet with range 10.0.0.0/24, the first ILB will have 10.0.0.254 and a second ILB will have 10.0.0.253.

As final step, we need to look inside the Virtual Cluster, different Managed Instance architectures (“G.M. minus“, “G.M. plus”) will have different requirements.

The “G.M. minus“ architecture can be seen as the first architecture we had, all instances based on Gen4 hardware may still have this architecture. The “G.M. plus“ architecture is the current one, most of the customers running on Gen5 should already have been migrated. Any new deployment on Gen5/6/7 should use this new internal architecture.

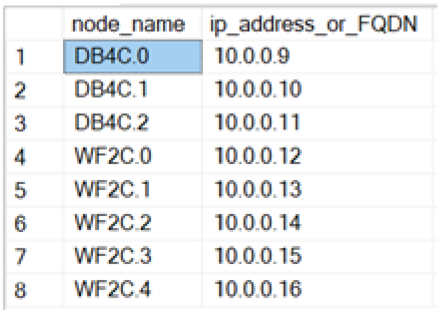
The main difference between both is related with the number and size of nodes we deploy on each. With the previous architecture we deploy several full-size nodes (starts with 5 nodes) and run Service Fabric and Instances on those nodes. You can see if they are related with the first generation if node names have the format DB.# like you can see in the following image:



	node_name	ip_address_or_FQDN
1	DB.0	10.0.0.4
2	DB.1	10.0.0.5
3	DB.2	10.0.0.6
4	DB.3	10.0.0.7
5	DB.4	10.0.0.8

With the more recent architecture, we deploy more nodes but likely smaller ones. We deploy nodes for Service Fabric and Nodes for Instances. Nodes for instances may not be full-size nodes, especially when smaller SLOs are used. For General Purpose, the size of the node will likely be similar to the size of the instance that customers selected. If it’s a Business Critical instance, you may see bigger (then the selected SLO) nodes being deployed do to the disk space, networking or other requirements that smaller SLOs cannot fulfill.

All this results in different IP Address requirements. You can see if the architecture is G.M.+ if the node names have the format DB# C.# or WF”C.# like you can see in the following image:



	node_name	ip_address_or_FQDN
1	DB4C.0	10.0.0.9
2	DB4C.1	10.0.0.10
3	DB4C.2	10.0.0.11
4	WF2C.0	10.0.0.12
5	WF2C.1	10.0.0.13
6	WF2C.2	10.0.0.14
7	WF2C.3	10.0.0.15
8	WF2C.4	10.0.0.16

The IP Address usage/requirements for the G.M.+ architecture are:

- Azure uses 5 IP addresses in the subnet for its own needs.
- Service Fabric will consume another 5 addresses.
- Each Virtual Cluster in the subnet will use 1 address.
- Each General Purpose instance needs 3 addresses.
- Each Business Critical instance needs 5 addresses.

Tenant Ring / Public endpoint IP address

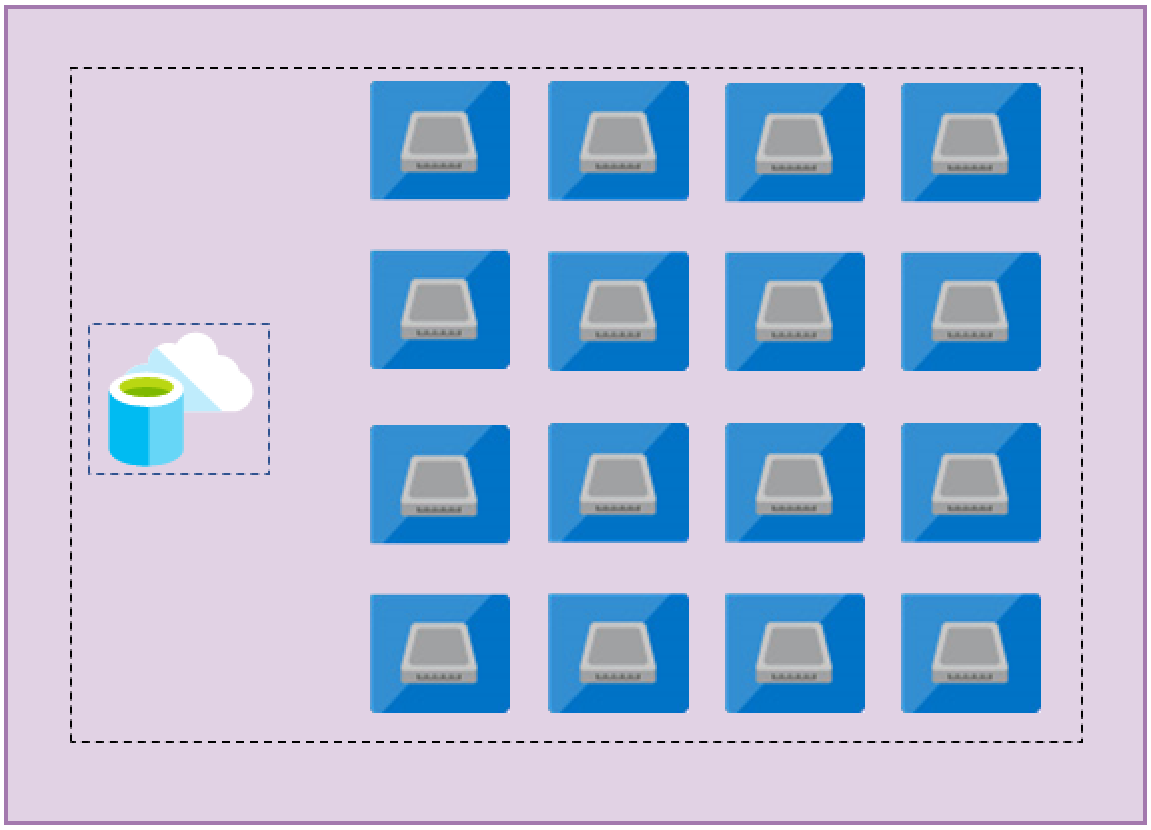
Please note that Tenant Rings for Managed Instance do not use IP addresses from Azure SQL DB IP Address ranges. The public endpoint will resolve to the Tenant Ring IP Address, as just mentioned, this address is not part of the pool of addresses for Azure SQL DB.

Storage layer in General Purpose Azure SQL Managed Instance

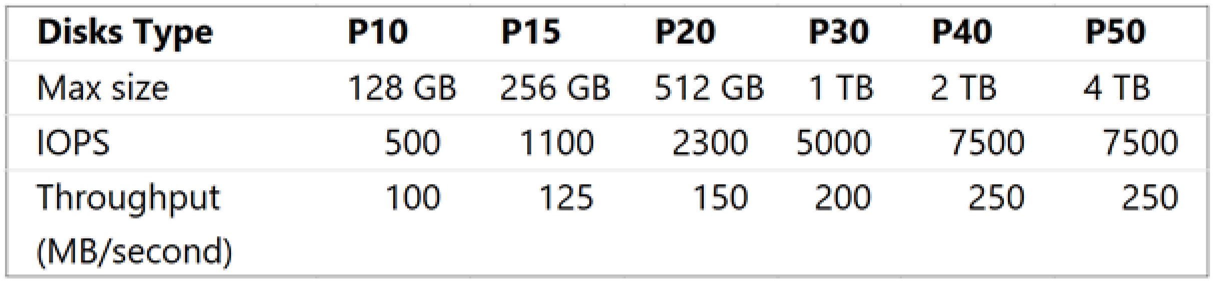
There are two layers in General Purpose architecture:

1. Compute layer that runs SQL Server Database Engine process and it is used for query processing and data caching. This is stateless compute VM that is re-initialized on another node if the process or underlying infrastructure crashes.

2. Data layer where all database files (.mdf, ldf, .ndf) files are placed. The data layer is implemented using Azure Premium Disk storage that has built-in replication and redundancy that guarantee that the data will be preserved.



Every database file is placed on separate Azure Premium disk (one file per disk). Azure Premium disks that are used in storage layer have fixed sizes: 128 GB, 256 GB, 512 GB, 1 TB, 2 TB, and 4 TB, and Managed Instance uses minimal disk size that is required to fit the database file with some size. For example, database file with 200 GB will be placed on 256 GB disk, a database file with 800 GB will be placed on 1 TB disk, etc. Every disk automatically grows if the database file size reaches the disk storage limits. Important thing that you need to understand is that file performance depends on the disk size. Bigger disks have bigger throughput and IOPS, as shown in the following table:



Disks Type	P10	P15	P20	P30	P40	P50
Max size	128 GB	256 GB	512 GB	1 TB	2 TB	4 TB
IOPS	500	1100	2300	5000	7500	7500
Throughput (MB/second)	100	125	150	200	250	250

This might be surprising because the larger databases files are faster than the smaller ones. In some cases, you might get better performance just by increasing the file size. Also, if you have multiple files in database and your workload can be parallelized, you might get great performance because the database files do not share IOPS and throughput. However, if you have multiple small files and your workload touches the same file, you might experience performance issues because multiple files are smaller and smaller IOPS and throughput. Every Managed Instance has up to 35 TB of total internal storage reserved for Azure Premium disks storage layer. If you create many files and the total size of the underlying disks reaches 35 TB, you will get internal storage limit error. This means that once you provision Managed Instance you have two storage limits:

1. Managed instance user-defined storage is the Managed Instance max storage size for your database files that you choose on portal and you pay for this amount of storage.
2. Internal physically allocated azure premium disk storage that cannot exceed 35 TB.

As a result, you cannot have more than 280 files on the General Purpose instance because 280 files placed on the smallest 128GB disks will reach 35TB limit.

You can reach this 35 TB limit even with the smaller number of files (less than 280). For example, a Managed Instance could have one database file with 1.2 TB that is placed on a 4 TB disk, and 248 files each 1 GB size that are placed on separate 128 GB disks.

In this example, the total disk storage size is 1 x 4 TB + 248 x 128 GB = 35 TB. However, the total reserved space for database files on the instance is 1 x 1.2 TB + 248 x 1 GB = 1.4 TB. This illustrates that under certain circumstance, due to a very specific distribution of files, a Managed Instance might reach the 35 TB reserved for attached Azure Premium Disk when you might not expect it to.

In this example existing databases will continue to work and can grow without any problem as long as new files are not added. However new databases could not be created or restored because there is not enough space for new disk drives, even if the total size of all databases does not reach the instance size limit. The error that is returned in that case is not clear.

You can use Transact-SQL scripts that can help you to see are you reaching this internal 35 TB storage limit on General Purpose Managed Instance.

First, we will create a schema and view that wraps standard sys.master_files view and returns the allocated disk size for every file:

```sql
CREATE SCHEMA mi;
GO
CREATE OR ALTER VIEW mi.master_files
AS
WITH mi_master_files AS
( SELECT *, size_gb = CAST(size * 8. / 1024 / 1024 AS decimal(12,4))
FROM sys.master_files where physical_name LIKE 'https:%')
SELECT *, azure_disk_size_gb = IIF(
database_id <> 2,
CASE WHEN size_gb <= 128 THEN 128
WHEN size_gb > 128 AND size_gb <= 256 THEN 256
WHEN size_gb > 256 AND size_gb <= 512 THEN 512
WHEN size_gb > 512 AND size_gb <= 1024 THEN 1024
WHEN size_gb > 1024 AND size_gb <= 2048 THEN 2048
WHEN size_gb > 2048 AND size_gb <= 4096 THEN 4096
ELSE 8192
END, NULL)
FROM mi_master_files;
GO
```

Now we can see the size allocated for the underlying Azure Premium Disks for every database file:

```
SELECT db = db_name(database_id), name, size_gb, azure_disk_size_gb
FROM mi.master_files;
```

Sum of the azure disk sizes should not exceed 35 TB — otherwise you will reach the azure storage limit errors. You can check total allocated azure storage space using the following query:

```
SELECT storage_size_tb = SUM(azure_disk_size_gb) /1024.
FROM mi.master_files
```

Using this information, you can find out how many additional files you can add on a managed instance (assuming that new file will be smaller than 128GB):

```
SELECT remaining_number_of_128gb_files =
(35 - ROUND(SUM(azure_disk_size_gb) /1024,0)) * 8
FROM mi.master_files
```

This is important because if this count became zero, you will not be able to add more files of database on the instance.

Conclusion

Managed Instance is built on the same infrastructure that's been running millions of databases and billions of transactions daily in Azure SQL Database, over the last several years. The same mechanisms for automatic backups, high availability and security are used for Managed Instance. The key difference is that the new offering exposes entire SQL instances to customers, instead of individual databases. On the Managed Instance, all databases within the instance are located on the same SQL Server instance under the hood, just like on an on-premises SQL Server instance. This guarantees that all instance-scoped functionality will work the same way, such as global temp tables, cross-database queries, SQL Agent, etc. This database placement is kept through automatic failovers, and all server level objects, such as logins or SQL Agent logins, are properly replicated.

Multiple Managed Instances can be placed into a so-called virtual cluster, which can then be placed into the customer's VNET, as a customer specified subnet, and sealed off from the public internet. The virtual clusters enable scenarios such as cross-instance queries (also known as linked servers), and Service Broker messaging between different instances. Both the virtual clusters and the instances within them are dedicated to a particular customer, and are isolated from other customers, which greatly helps relax some of the common public cloud concerns. "An instance of the Database Engine is a copy of the sqlservr.exe executable that runs as an operating system service. Each instance manages several system databases and one or more user databases. Each computer can run multiple instances of the Database Engine. Applications connect to the instance in order to perform work in a database managed by the instance. " SQL DB – Managed Instance is an instance of the database engine running in the cloud, within Azure SQL Database cloud service. To customers, it will look like auto-managed SQL Server instance in the cloud.

Is Managed Instance IaaS or PaaS?

It is a PaaS offering. SQL MI is part of Azure SQL DB, Microsoft's fully managed cloud database service, and thus it inherits all the built-in PaaS features. SQL MI introduces the managed SQL instance model within SQL DB, thus providing higher surface area and feature compatibility with on-premises SQL Server.

MI customers have sysadmin access to the managed SQL instance, but they will not have remote access to the operating system hosting SQL Server.

Cloud Lifter offers the same on-premises SQL Server on the Cloud as PaaS. Too many SQL workloads are encountering too much friction in moving to Azure SQL DB. To address this, SQL DB service will be extended with the CL offering.
CL targets the following objectives

1. On premise customers looking to migrate their applications to PaaS without design changes.
2. IaaS customers looking to migrate their SQL VM applications to PaaS without design changes.
3. SQL Database customers looking to expand the cloud application with the functionality available on premise without re-designing it.
4. SQL ISVs who want to enable their customers to migrate to the cloud New SQL DB flavor, that will enable easy SQL Server app migration from on-premises to a fully managed PaaS

Who is it for?

1. Customers who want to mass migrate their existing on-premise apps to cloud, with low migration effort & cost being a crucial factor
2. Customers that previously faced issues with migration to SQL DB

- Security concerns (lack of isolation mechanisms such as VNET)
- App compatibility (missing features in SQL DB imposed expensive re-designs)

We offer CL in diver sizes (SLOs) including Big Instances (RBI) that can handle tier 1 enterprise workloads. CL instances

- CL includes several service tiers for instances, like service tiers for Standard and Premier series databases
- Each service tier includes a fixed set of compute resources, storage and IO. Additional storage and IO can be purchased.
- No extra cost when individual databases are provisioned and no automatic RG. All databases on the instance share the same instance resources.
- standalone instance is provisioned, the customer will be billed for the cluster’s total DTU total throughput capacity (DTU). Each DTU tier has some amount of included storage and some included IO (sufficient to reach DTU rating). Additional Storage and IO can be purchased.
- If an instance is provisioned in a private cluster, there is no additional charge.

Managed Instance aims to deliver close to 100% surface area compatibility through a staged release plan, so it’ won’t be 100% compatible and not in v1.

It’s also a PaaS database service that share lot in common with SQL DB so that aspect needs to be honored.

There are specific cases when tools need to recognize that a particular feature works in a slightly different way or that service is not running in an environment fully controlled by the customer:

- HA is built-in / pre-configured => Always On is not exposed as it is on-prem
- User-initiated backup / restore options are limited (only “copy-only” backup and restore from URL will work), because there’s automated backup / PITR

- Managed Instance will not allow specifying full physical paths so all corresponding scenarios have to be supported differently: RESTORE DB does not support WITH MOVE, CREATE DB doesn't allow physical paths, BULK INSERT works with Azure Blobs only, etc..
- Managed Instance supports Azure AD auth as cloud alternative to Windows Auth. Any scenario that assumes Windows AD need to be supported through AAD
- Managed Instance do not support natively BI services (SSIS, SSRS, SSAS). Related features (such as Maintenance Plans) have to be supported differently (through integration with SSIS PaaS).
- In-memory OLTP will work only in perf-optimized SKU ("premium")

When customers should use VM instead of CL

- A highly-customized management infrastructure that requires full SQL and Windows Server system admin rights and full access to the server OS. This might be required when different monitoring and management components must be co-located on the same virtual machine.
- a 3rd party tool is used to manage SQL server instances.
- SQL server is part of an ISV application that is not certified for SQL 2016 and Managed SQL Server.
- On premise database application uses SQL Server 2012 or earlier and cannot migrate to SQL 2016
- Applications that need to be closely co-located with the databases (less than the Accelerated Networking round trip Azure latency, 300us) should target SQL VM. When customers should use SQLDB instead of CL The following scenarios are better fit for SQLDB:
- Application that need more aggregate DTU than single CL instance provides should use singleton/pooled DB.
- Customer instances contain all sorts of comingled functionality. Eventually a single instance does not scale to cover the application desired state, and they need to shred to use many Azure services.
- Cloud-born and new apps should use SQL DB (once we add R, SQLCLR). </max></dns_zone></mi_name>

How good have you found this content?