Microsoft

Microsoft Azure Data Fundamentals

Exam Ref DP-900

Daniel A. Seara
Francesco Milano

Microsoft

Exam Ref DP-900 Microsoft Azure Data Fundamentals

Daniel A. Seara
Francesco Milano

NOTE **MIGRATING YOUR DATA WITH DMS**

Azure Database Migration Service (DMS) is a multifunctional assistant you can use to migrate on-premises databases to Azure relational workloads for any of the technologies and database services we describe in this chapter.

You can prepare your migration plan and start it when you are ready to change your environment. When you need a continuous migration path, you can use a Premium tier, which has an hourly cost, applied using four vCores. Using the Standard level is free, but that allows only offline migration, which means taking your database offline during the migration process.

The following is a partial list of the most important features covered by SQL-MI:

- Automatic software patching
- Built-in instance and database monitoring and metrics
- Latest database engine features
- Managed automated backups
- Multiple number of data files per the database
- VNet - Azure Resource Manager deployment

Two service tiers are available:

- **General purpose** Use High-Performance Azure Blob storage, with an 8 TB limit. The data and log files are stored directly in a blob repository.
- **Business Critical** Uses local SSD storage, up to 1 TB or 4 TB, depending on the server generation used, with Always On availability groups, read-only database replica, and OLTP in memory support.

EXAM TIP

Always On availability groups is a SQL Server solution implemented for high availability and recovery. It uses a Windows Server Failover Cluster and implements replicas between the members of the cluster. The replicas can be asynchronously committed when long distances must be covered, but usually the synchronous method is used.

You have a lot of metrics to consider when designing an Azure SQL Managed Instance implementation, and the resource limits vary over time, since new sizes, ranges, and functionality will be added progressively. We recommend that you measure the actual on-premises implementation and use Azure Calculator to evaluate the best combination for your needs. Once the SQL-MI is implemented, a frequent follow-up of resource usage can help refine the implementation.

Building a SQL-MI is a process that takes time. Several steps, each one with long-running processes, must take place. Table 2-11 gives you an idea about the tasks and their duration.

TABLE 2-11 Timetable for long processes in SQL-MI implementation

Action	Time
Virtual cluster creation	Up to 4 hours
Virtual cluster resizing (adding nodes)	Up to 3 hours
Instance compute scaling up/down	Up to 3 hours
Database seeding/Always On seeding	220 GB/hour

Other operations, such as attaching a database from Azure Storage, take only a few minutes. However, consider the time you may need to upload the files (or the data management system needed to upload the files).

SQL-MI is PaaS. That means the management of the hardware, software, updates, maintenance, and so forth is the responsibility of Microsoft. As such, the customer does not have direct access to the servers using RDP or any other protocol.

Of course, you will need to perform some "administrative" tasks, such as creating a new database or choosing a different storage space or computer combination. However, you will only be able to perform all those tasks by using the Azure portal or the other general management tools you will see later in this chapter, such as the CLI, PowerShell, or other mechanisms for process automation.

All the operations you issue must use the Tabular Data Stream (TDS) application layer protocol, which means using SQL statements to store or retrieve data. The connection between customer applications and the managed instance must be through the virtual network itself, via a virtual machine connected to the same virtual network, or through a VPN or Azure ExpressRoute connection. As you will see during the setup procedure, you can configure a connection endpoint, but it is only for data and cannot be used for management.

At any rate, all the communications are encrypted and the internal communication between the parts of the managed instance is encrypted and signed using certificates. The communication with Azure external services like Azure Key Vault and Azure Active Directory are encrypted and signed as well.

To create a SQL-MI in the Azure portal, type **SQL managed instances** in the Search box and select it from the results. Click **Add** to open a wizard with the following pages:

1. **Basics** Select your subscription and resource group. Then enter the instance name, select the region, configure the compute and storage, and enter the administrator credentials. The constraint rule for the administrator password in this case is stronger; it requires at least 16 characters.

2. **Networking** The SQL-MI requires a virtual network, and it offers to create a new one for you, or you can select one from your already created VNets. However, the creation process needs to modify the VNet configuration, asking if it should make the changes automatically or guide you in the process of doing it yourself. You must select a connection type (proxy or redirect), which affects how applications connect to the SQL-MI. Finally, you can opt to enable a public endpoint, which will be used only for data communication when you need it without using a VPN.
3. **Additional Settings** Here you can select the collation and the time zone for the SQL-MI, which cannot be changed after implementation. Also, you can add SQL-MI during the creation process as a secondary failover for an existing SQL-MI.
4. **Tags** As with any other resource, you can add your own tags here.
5. **Review + Create** After the validation, the page displays your configuration, the estimated cost, and the top limit for the creation process in hours.

After creation, virtual networks, network security groups, route tables, and the virtual clusters will be added to the resource group. Figure 2-13 shows the resource involved in a SQL-MI deployment.

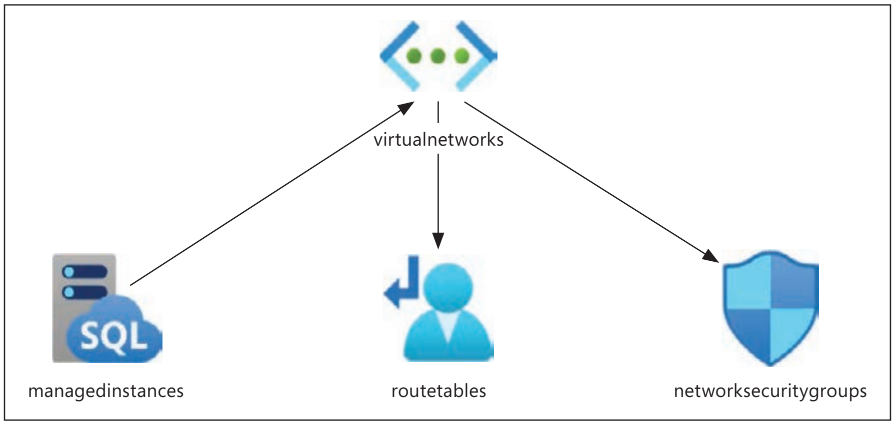

FIGURE 2-13 SQL-MI base resources

Skill 2.3: Identify basic management tasks for relational data

All the relational data services must be deployed, managed, and secured in order to perform most efficiently. And sometimes, problems will appear, and you will have to identify and fix them. Moreover, you will probably have to retrieve and update data.

This skill covers how to:

- Describe provisioning and deploying relational data services
- Describe method for deployment including ARM templates and Azure Portal
- Identify data security components (e.g., firewall, authentication)
- Identify basic connectivity issues (e.g., accessing from on-premises, access with Azure VNets, access from internet, authentication, firewalls)
- Identify query tools (e.g., Azure Data Studio, SQL Server Management Studio, sqlcmd utility, etc.)

Describe provisioning and deploying relational data services

Now that you have an idea of the various relational workloads Azure provides, let's look at the reasons that could make you decide to move to an Azure solution and how to select the option that best fits you.

Cost. This is probably the most important reason. Of course, if you make an investment in hardware and software, you want the best return on it, but what if you need to migrate or upgrade your implementation? If you consider hardware, software, and maintenance, security, redundancy, and reliability, you may find that moving to Azure is the best choice.

Think about the pros and cons of the available options. If you do not need your relational workload up and running all the time, IaaS will probably be a better choice than PaaS, since you can shut down your unused server and bring it back when needed. However, having SQL Server on a virtual machine will require more management effort from your team.

In this section, we analyze the measures and tiers used to estimate service costs, based on DTUs, vCores, and tier levels, among other factors. Note that there are other costs to consider. One is network traffic; all inbound traffic is free, but any outbound traffic over an initial free amount will be billed.

Also, the costs vary per region. Table 2-12 shows some of the differences, with prices as of June 2020.

TABLE 2-12 Comparative costs in network traffic between regions

	Dollar per GB			
Ratio	**US**	**Europe**	**Korea**	**South Africa**
First 5 GB	Free	Free	Free	Free
5 GB - 10 TB	$ 0.087	$ 0.087	$ 0.120	$ 0.181
10 - 50 TB	$ 0.083	$ 0.083	$ 0.085	$ 0.175
50 - 150 TB	$ 0.070	$ 0.070	$ 0.082	$ 0.170
150 - 500 TB	$ 0.050	$ 0.050	$ 0.080	$ 0.160
Over 500 TB	Ask	Ask	Ask	Ask

Service level. Having a reliable platform is cost intensive. Having your databases in PaaS gives you 99.99 percent SLA, whereas IaaS gives you 99.95 percent. The difference between IaaS and PaaS can be covered if you perform some additional tasks, such as adding a second virtual machine instance or implementing SQL Always On to ensure availability.

Administration. PaaS will reduce the time your team dedicates to manage your relational infrastructure, since most of the work is performed by Azure management. On the other hand, you must consider other possible issues using some of the services.

Let's look at a probable case: if you have some CLR procedures, they are not available in PaaS at all. If you must keep them implemented, your choice must be IaaS (or SQL-MI).

Something similar happens with OPENROWSET, OPENQUERY, file streams, cross database queries, and so forth.

Migration path. This is about opportunity as well as procedure. You must select the best time to perform a migration, depending on your needs and the way you do it.

Let's see some examples:

- You have a web application using information of a database, or more than one, not interconnected. You can easily change the connection string in the web application.
- Moving to PaaS would probably be the best choice, using Data Migration Assistant or another tool to move data quickly and change the connection string only at the end of the movement.
- Your application uses different databases, where you need to update information in more than one of them at the same time, and uses cross-database references, using *fully qualified object names (<database>.<schema>.<object>* nomenclature) to perform the queries. Or your database uses external binary storage to enhance document management, allowing you to use the binary storage from outside the database.

In these cases, you are limited to using IaaS to keep it running. There will be an initial platform preparation, where you define and create one or more virtual machines in IaaS, and then plan and execute the migration, exactly in the same way you can use migration to a new on-premises server.

Which SQL Server flavor to use

When you move your data to the cloud, it is important to choose the best implementation for your needs. Here are some guidelines to help you decide which is the best option in different cases.

Azure SQL Database. As PaaS service, Azure SQL Database frees you of the administrative and maintenance tasks, ensures you 99.99 percent SLA, automates backup procedures, and can grow vertically on demand.

You can define a *single database*, which is a concept similar to the contained database used in SQL Server 2012 version. The idea is to have fewer dependencies on the underlying server. The metadata, the user access security, and statistics are isolated from the server.

The authentication in this kind of database, including Windows Authentication, is managed by the database without server participation.

Of course, logins from SQL Server can be allowed to reach the database, but this diminishes the "containment" of the database itself.

If you want reduce costs, you can choose a serverless implementation, with more fine-grained cost-per-use billing and the ability to automatically stop the resource usage when you are not using the database. Alternatively, you could use Hyperscale for higher performance; a large database, up to 100 TB; an almost instantaneous backup; quick restore; and rapid scaling (out and up), all applied to a single database. A scenario for this could be an implementation where you have just one database requiring all these special abilities and other databases with significantly fewer requirements. You can define each one as a single database and refine the configuration for each one.

Azure SQL Managed Instance. As we discussed earlier, for big servers with several databases, datacenters, and ISV providers, SQL-MI can be a good choice to automate movement from on-premises to the cloud. A set of SQL servers with high availability, replication (local and geographically dispersed), and clustering based on Always On features gives you the reliability and availability you need. If you will have cross-databases queries, that is another reason to use SQL-MI.

SQL Server on Azure VM. This IaaS service is your choice when you need fine-grained control over service configuration, maintenance, and patching; you have to move the SQL server from on-premises to the cloud without making any changes in your database or application; you have CLR code inside your database; or you will be using inter-database queries or linked views. Another scenario is when you are preparing a test or development environment for database design before moving it to a production environment. In a three-layer implementation, you can provide a SQL Server on a virtual machine to your development team and have a staging and production environment using other Azure SQL Database options.