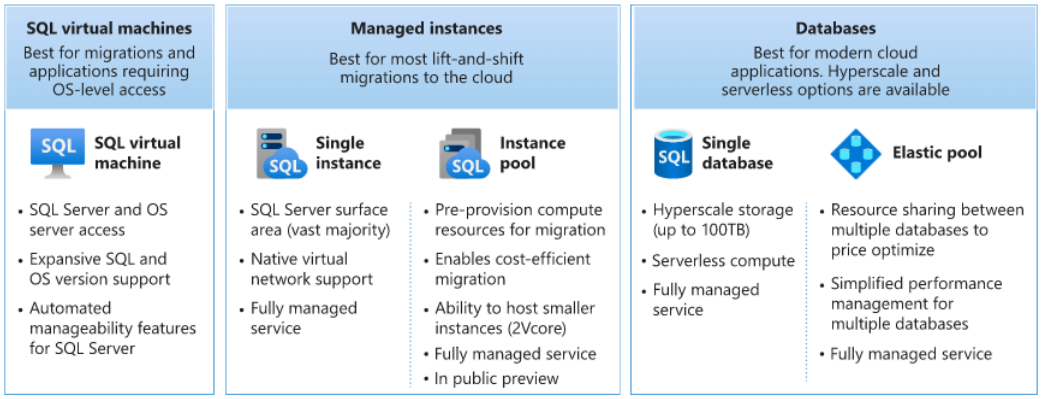
Managing Data in Azure SQL Database

- Introduction / Overview of SQL Database.
- Azure SQL Managed Instance
- Comparing SQL Azure Database to Azure / On-Premise SQL Server.
- Creating and Using SQL Server and SQL Database Services.
- Azure SQL Database Tools.
- Elastic Pools.
- Azure AD Authentication for Azure SQL Database server
- Dynamic Data Masking
- Configure Auditing
- Business Continuty and Data Recovery
- Active Geo Replication
- Partitioning

Azure SQL Database Introduction

- Azure gives three options to run SQL Server workloads
    1. SQL Database(Paas)
    2. SQL Managed Instance(Paas)
    3. SQL Virtual Machine(Iaas, SQL Server inside fully managed VM)



- **SQL Database** is a cloud-based relational database **service** is built on **SQL Server technologies** and abstracts both the OS and the SQL Server instance from user that. It supports T-SQL commands, tables, indexes, views, primary keys, stored procedures, triggers, roles, functions etc.
- SQL Database delivers predictable performance, scalability with no downtime, business continuity and data protection—all with **near-zero administration**. You do not need to architect a database installation for scalability, high availability, or disaster recovery as these features are provided automatically by the service and can focus on rapid app development and accelerating your time to market, rather than managing virtual machines and infrastructure.
- **Supports existing SQL Server tools(SSMS), libraries and APIs,** which makes it easier for you to move and extend to the cloud.
- Popular command-line interfaces like sqlcmd and bcp are supported with Azure SQL services.
- It is available in two purchasing models DTU and vCore.

Benefits of SQL Database

- **High Availability** - For each SQL database created on Windows Azure, there are **three** replicas of that database.
- **On Demand** – One can quickly provision the database when needed with a few mouse clicks.
- **Reduced management overhead** - It allows you to extend your business applications into the cloud by building on core SQL Server functionality while letting Microsoft Azure support staff handle the maintenance and patching tasks.

SQL Database top features:

- Tables, views, indexes, roles, stored procedures, triggers, and user defined functions
- Constraints
- Transactions
- Temp tables
- Basic functions (aggregates, math, string, date/time)
- Constants
- Cursors
- Index management and index rebuilding
- Local temporary tables
- Reserved keywords
- Statistics management
- Table variables

- Transact-SQL language elements such as create/drop databases, create/alter/drop tables, create/alter/drop users and logons

The following features of SQL Server are NOT SUPPORTED in SQL Database

- Windows Authentication (Azure AD Authentication is now Supported)
- Not all T-SQL Commands Supported
- Access to System Tables
- Common Language Runtime (CLR)
- Database file placement
- Database mirroring
- Distributed queries
- Distributed transactions
- Filegroup management
- Global temporary tables
- Support for SSIS (instead use Data Factory), SSAS (Separate Service), SSRS
- Support for Replication or SQL Server Service Broker

Azure SQL Database Purchasing Model

There are two purchasing models DTU and vCore

1. DTU:
    - DTU stands for *Database Transaction Unit*, and is a combined measure of compute, storage, and IO resources.
    - DTU based model is not supported for managed instance.
    - Available in Three service tiers:Basic,Standarad,Premium
2. vCores:
    - Allows you to independently select compute and storage resources, gives option to choose between generation of hardware,no of cores,memory and storage size.
    - GiveS you greater control over the compute and storage resources that you create and pay for.
    - Available in Three Service Tiers:General Purpose,Business Critical,Hyperscale.
    - In the vCore model, you pay for:
        - **Compute resources:** The service tier + the number of vCores and the amount of memory + the generation of hardware.
        - **Data and log storage:** The type and amount of data and log storage.
        - **Backup storage location:** Read-access geo-redundant storage (RA-GRS), Zone-redundant storage (ZRS), or locally redundant storage (LRS).

Azure SQL Managed Instance

1. It is a new deployment model of Azure SQL Database, providing near **100% compatibility** with the latest SQL Server on-premises (Enterprise Edition) Database Engine.

2. Classic on-prem application with complex environment and require SQL CLR,SQL Server Agent,Cross database queries can migrate to cloud with this model.
3. Ideal for customers who want to use **instance-scoped features** and want to move to Azure without rearchitecting their applications.
4. It provides a **native virtual network (VNet)** implementation that addresses common security concerns, and a business model favorable for on-premises SQL Server customers.
5. Managed Instance allows existing SQL Server customers to **lift and shift** their on-premises applications to the cloud with minimal application and database changes.
6. Managed Instance preserves all **PaaS capabilities** (automatic patching and version updates, automated backups, high-availability), that drastically reduces management overhead and administrator activities.

Visit: Azure Portal → Create a resource → Azure SQL → Create

Comparison

Azure SQL Database (Logical server)	**SQL Managed Instance**	**SQL Server on VM**
PAAS Service	PAAS Service	IAAS Service
The most commonly used SQL Server features are available.	Near-100% compatibility with SQL Server. on-premises.	Fully compatible with on-premises physical and virtualized installations.
You can provision a *single database* in a dedicated, managed (logical) server; or you can use an *elastic pool* to share resources across multiple databases and take advantage of on-demand scalability.	Each managed instance can support multiple databases. Additionally, *instance pools* can be used to share resources efficiently across smaller instances.	SQL Server instances are installed in a virtual machine. Each instance can support multiple databases.
99.995% availability guaranteed.	99.99% availability guaranteed.	Up to 99.99% availability.
Latest stable Database Engine version.	Latest stable Database Engine version.	Fixed, well-known database engine version. All SQL Server Features are avaiable

Fully automated updates, backups, and recovery. Long-term backup retention for up To 10 years		You must manage all aspects of the server, including operating system and SQL Server updates, configuration, backups, and other maintenance tasks.
Migration from SQL Server might be hard.	Easy migration from SQL Server.	Easy migration from SQL Server on-premises.
Built-in High -Availability		You need to implement your own High-Availability solution.
Online change of resources (CPU/storage). **Scalability** lets you easily add more resources (CPU, memory, storage) without long provisioning.		There is a downtime while changing the resources (CPU/storage) because VM needs to resized and that restarts VM
Does not support SQL Server Agent.You can use Elastic Job Agent service in Azure to create and Schedule Jobs.	Supports SQL Server agent ,SQL Agent jobs are supported for T-SQL ans SSIS	
Ideal for customers want to build modern apps,with highest uptime and predictable performance	Ideal For Customers want t migrate to cloud ,remove management overhead but need Instance scoped fetaures	Ideal for customers want to migrate to cloud as fast as possible but maintain OS control and complete SQL Server functionality.

The Azure SQL Managed Instance and Azure SQL Database services restrict the following configurations:

- You can't stop or restart servers.
- You can't use:
  - Instant file initialization.
  - Locked pages in memory. We might configure Locked pages in some SLO deployments.
  - FILESTREAM and availability groups. We use availability groups internally.
  - Server collation. In SQL Managed Instance, you can select server collation during deployment but not change it.

- Startup parameters.
- Error reporting and customer feedback.
- ALTER SERVER CONFIGURATION.
- ERRORLOG configuration.
- "Mixed Mode" security is forced, though Azure Active Directory only authentication is in preview
- Logon audit is done through SQL audit.
- Server proxy account isn't applicable.

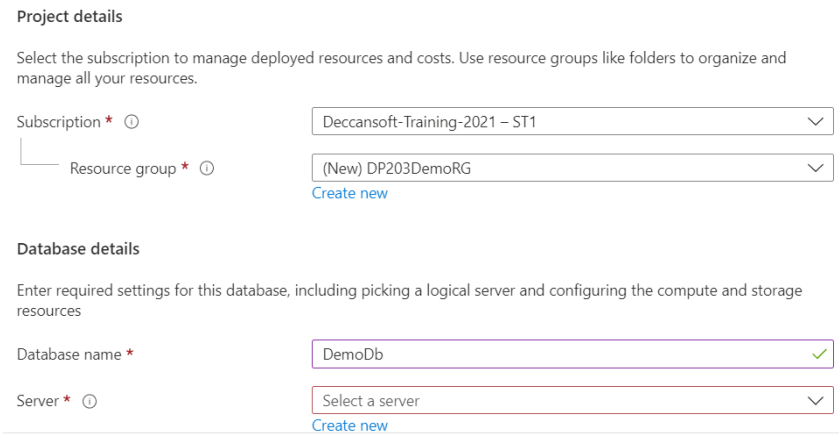
Creating SQL Database

With Windows Azure SQL Database you can quickly create database solutions that are built on the SQL Server database engine. We can create a new SQL database in Windows Azure and then configure it later. We can decide whether to use an existing SQL database server or create a new one when you create your new database. We can also import a saved database from Binary Large Object (BLOB) storage into SQL Database.

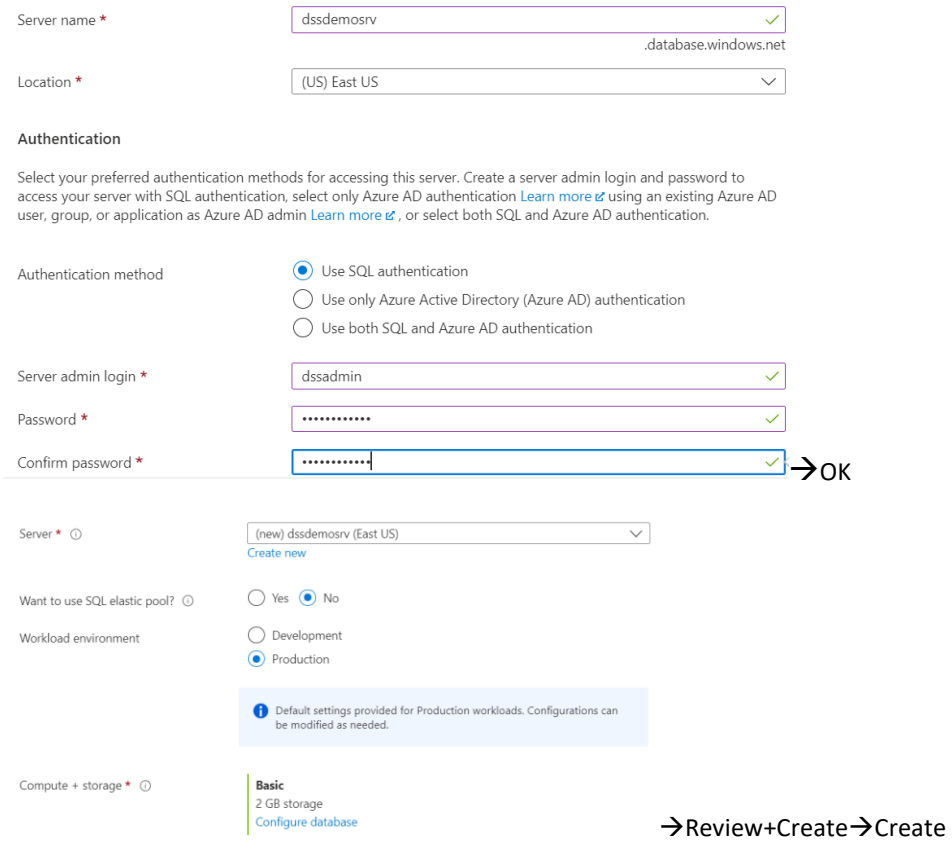
Azure SQL logical server

- For databases and elastic pools, an Azure SQL Database server is required
- The server name must be unique across all of Azure
- Consider Azure SQL logical server as, nothing but administrative container for your databases(SQL Database, Warehouse Database,pooled database).
- It enables you to group and manage certain permissions and configurations together,You can control logins, firewall rules,auditing rules and security policies through the logical server.
- You can also override these policies on each database within the logical server.

Lab 1 :Creating an Azure SQL Database Instance by using the Azure Portal



→Create new



Azure SQL Database Tools

One of the advantages of SQL databases in Azure is the ability to use many monitoring tools that you use for on-premises databases.

A TDS(Tabular Data Stream) endpoint is exposed for each logical server in SQL Database. That means all drivers that normally work with SQL Server work with Azure SQL.This allows you to use SQL Server Management Studio with SQL Database in the same way you will use it with SQL Server standalone.

You can also use Azure Data Studio , provides a lightweight editor and other tools for interacting with Azure Data Services, such as SQL Server on-premises, Azure SQL, and Azure Database for PostgreSQL

Using SQL Server Management Studio:

1. Start SQL Server Management Studio locally
2. In Connect dialog, provide
    a. Server name= "**dssdemoserver.database.windows.net**"
    b. **Change Authentication = SQL Server Authentication**

c. Login = "DSSAdmin"

d. Password = "Password@123"

e. Connect

3. **This will give error**. From the error dialog note the IP address eg: 49.12.12.4
4. Configure firewall settings on SQL Server using the Azure Portal
    a. Azure Portal → Select Sql Server → Settings → **Firewall**
    b. Add Local IP OR

        Click Add Client IP**.**

        i. Rule Name = "Allowed IP".

        ii. Start IP = 49.12.12.0

        iii. End IP = 49.12.12.5

    c. Click Save
5. Return back to SSMS and try to connect again. (It might take upto 5 mins after allowing the IP in firewall)

To Restrict a given ip or a range of IP address for a particular database

```
-- Create database-level firewall setting for only IP 0.0.0.4
EXECUTE sp_set_database_firewall_rule N'Example DB Setting 1', '49.12.12.4', '49.12.12.4';


-- Update database-level firewall setting to create a range of allowed IP addresses
EXECUTE sp_set_database_firewall_rule N'Example DB Setting 1', '49.12.12.0', '49.12.12.100';
```

Note: If you specify an IP address range in the database-level IP firewall rule that's outside the range in the server-level IP firewall rule, only those clients that have IP addresses in the database-level range can access the database.

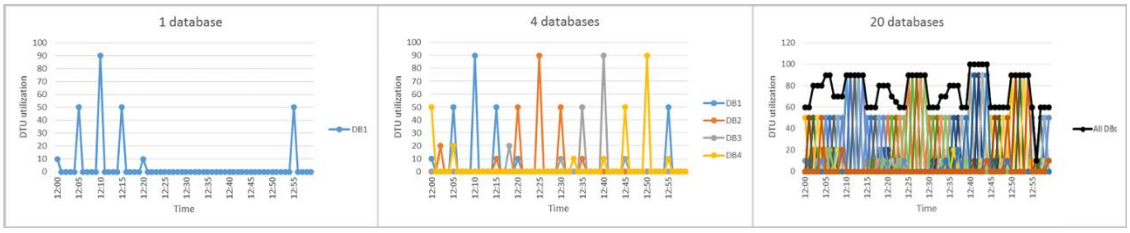
Elastic Pools

Elastic Pools:

- Elastic pools provide a **simple cost effective solution** to manage the performance goals for **multiple databases** (hosted on the same logical server) that have widely **varying and unpredictable** usage patterns.
- **elastic DTUs** (eDTUs) are used elastic databases in an elastic pool.
- A pool is given a set number of eDTUs, for a set price. Within the pool, individual databases are given the flexibility to auto-scale within set parameters.
- Provisioning resources for the entire pool rather than for single databases simplifies your management tasks.

- Under heavy load, a database can consume more eDTUs to meet demand. Databases under light loads consume less, and databases under no load consume no eDTUs.
- Additional eDTUs can be added to an existing pool with no **database downtime** or no impact on the databases in the elastic pool. Similarly, if extra eDTUs are no longer needed they can be removed from an existing pool at any point in time.
- You can add or subtract databases to the pool. If a database is predictably under-utilizing resources, move it out.

Which databases go in a pool?



- Databases that are great candidates for elastic pools typically have periods of activity and other periods of inactivity. In the example above you see the activity of a single database, 4 databases, and finally an elastic pool with 20 databases.
- Databases with varying activity over time are great candidates for elastic pools because they are not all active at the same time and can share eDTUs.
- Not all databases fit this pattern. Databases that have a more constant resource demand are better suited to the Basic, Standard, and Premium service tiers where resources are individually assigned.
- While the eDTU unit price for a pool is **1.5x greater than the DTU** unit price for a single database, **pool eDTUs can be shared by many databases and fewer total eDTUs are needed.**

Cost of Single Database = Database count * Cost of Each DTU * Number of DTU
Cost of Elastic Pool = Cost of eDTU * Number of eDTU **=** 1.5 * Cost of each DTU * Number of eDTU

Sizing an elastic pool:

The best size for a pool depends on the aggregate eDTUs and storage resources needed for all databases in the pool. This involves determining the larger of the following:

- Maximum DTUs utilized by all databases in the pool.
- Maximum storage bytes utilized by all databases in the pool.

SQL Database automatically evaluates the historical resource usage of databases in an existing SQL Database server and recommends the appropriate pool configuration in the Azure portal.

Creating a Pool and adding database to it.

1. Azure Portal → **SQL Servers** → Server blade → New Pool
2. Name = DemoPool
3. Pricing tier = Standard Pool (The pool's pricing tier determines the features available to the elastic databases in the pool, and the maximum number of eDTUs (eDTU MAX), and storage (GBs) available to each database.)
4. Configure the Pool → Specify Elastic database pool settings and in blade on top click **Add to Pool** to add database to the pool
5. Also Per database settings can be specified for eDTU max and min.

Creating a Pool and adding database to it.

1. Azure Portal→Create Resource→Elastic Pool→Select SQL Elastic pool→+New→
2. Elastic Pool Name:DemoPool
3. Pricing tier = Standard Pool (The pool's pricing tier determines the features available to the elastic databases in the pool, and the maximum number of eDTUs (eDTU MAX), and storage (GBs) available to each database.)
4. Configure the Pool → Specify Elastic database pool settings and in blade on top click **Add to Pool** to add database to the pool.
5. Also Per database settings can be specified for eDTU max and min.

Note:If you select Vcore model under pool settings you can set Vcore and Data max size .Under Per datbase settings you can specify vCores.

You can add or remove database to pool by going to **configuration** section.

Refer:

https://docs.microsoft.com/en-us/azure/azure-sql/database/elastic-pool-overview

SQL Managed Instance pools :allow you to host multiple managed instances and share resources. You can pre-provision compute resources. Doing so can reduce overall deployment time to make migrations easier. You can also host smaller managed instances in an instance pool than you can in a single managed instance. This offer is currently in public preview.

Azure AD Authentication for Azure SQL Database server

- Azure AD Authentication is a mechanism of connecting to Azure SQL Database, managed instance by using identities in Azure Active Directory (Azure AD).

- Azure AD authentication uses contained database users to authenticate identities at the database level

Traditional Login and User Model

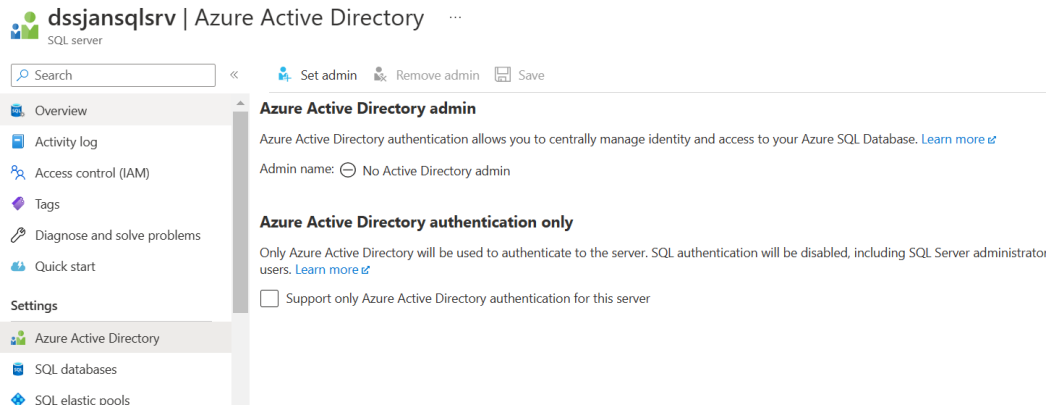
- Traditionally for users to have access to database, the master database must have a login that matches the connecting credentials and the login must be able to be mapped to a database user in the user database.
- connection to the user database has a dependency upon the login in the master database, and this limits the ability of the database to be moved to a different hosting SQL Server or Azure SQL Database server.

Contained Database User Model

- In the contained database user model ,the authentication process occurs at the user database, and the database user in the user database does not have an associated login in the master database.
- To connect as a contained database user, the connection string must always contain a parameter for the user database so that the Database Engine knows which database is responsible for managing the authentication process.
- Azure sql database support Azure Active Directory identities as contained database users.
- When using Azure Active Directory authentication, connections from SSMS can be made using Active Directory Universal Authentication.

Configure Azure AD authentication with SQL

1. Create an Azure AD and populate it with users and groups
2. **Create an Azure AD administrator for Azure SQL server**
   - I. Azure Portal → SQL Servers → Select the <Server>
   - II. On **SQL Server** page, Settings→ **Active Directory admin**→Set Admin



   - III. In the **Add admin** page, search for a user, select the user or group to be an administrator, and then select **Select**.

Note

- *The Active Directory admin page shows all members and groups of your Active Directory. Users or groups that are grayed out cannot be selected because they are not supported as Azure AD administrators. (See the list of supported admins in the **Azure AD Features and Limitations** section of Use Azure Active Directory Authentication for authentication with SQL Database or SQL Data Warehouse.)*
- *Removing the Azure Active Directory administrator for Azure SQL server **prevents any Azure AD authentication user** from connecting to the server. If necessary, unusable Azure AD users can be dropped manually by a SQL Database administrator.*

Grant access to other Azure AD users

1. **Create contained database users in your database mapped to Azure AD identities**

   Execute the following commands to create a contained **database user**

```
CREATE USER [abc@Xyz.com] FROM EXTERNAL PROVIDER;
```

   To create a contained database user representing an **Azure AD domain group**, provide the display name of a security group:

```
CREATE USER [ADGroup1] FROM EXTERNAL PROVIDER;
```

2. **Grant required permissions to user using Grant command**

```
GRANT SELECT,INSERT ON EMPLOYEE to abc@Xyz.com
```

Dynamic Data Masking

- Dynamic data masking (DDM) limits sensitive data exposure by masking it to non-privileged users. It can be used to greatly simplify the design and coding of security in your application.
- Dynamic data masking is a great feature for both on-premise SQL Server (from SQL Server 2016) and Azure SQL Database as well. This feature can help users to secure their critical data elements without making any change at physical level.
- All the unprivileged users can only see masked data and don't have access to actual values since masking rules are applied in the query results
- Dynamic data masking is easy to use with existing applications.
- As an example, a call center support person may identify callers by several digits of their social security number or credit card number. Social security numbers or credit card numbers should not be fully exposed to

the support person. A masking rule can be defined that masks all but the last four digits of any social security number or credit card number in the result set of any query

```
Create table DemoTable
        (ID Int, PersonName varchar (100),
        Age int,
        EmailAddress varchar(120),
        CreditCardNumber varchar(19),
        SocialSecurityNumber varchar(11))
INSERT INTO DemoTable Values (1, 'Sandeep Soni',43,'sandeep@abc.com','1234-5678-4321-8765','123-45-6789')
SELECT * FROM DemoTable --Result will not be masked
```

Masking Functions

SQL Server provides four built in functions to mask data in SQL tables. These functions are as follows:

1. default():Full masking according to the data types of field.

   For string data types, use XXXX or fewer Xs if the size of the field is less than 4 characters

   For numeric data types use a zero value

   For date and time data types use 01.01.1900

   Example:
   ```
   Alter Table DemoTable
       Alter Column PersonName varchar (100)  MASKED WITH (FUNCTION='default()')
   ```
2. email():**Masking method, which exposes the first letter and replaces the domain with XXX.com**

   Example: aXX@XXXX.com

   Example:
   ```
   Alter Table DemoTable
    Alter Column EmailAddress varchar (120)  MASKED WITH (FUNCTION='email()')
   ```
3. random(): **Masking method, which generates a random number** according to the selected boundaries.

   If the designated boundaries are equal, then the masking function is a constant number. Present as

   "**Random number"** in portal

   Example:
   ```
   Alter Table DemoTable
   Alter Column age int MASKED WITH (FUNCTION ='random(1,20)')
   ```
4. Partial(): Masking method that exposes the first and last letters and adds a custom padding string in the middle. If the original string is shorter than the exposed prefix and suffix, only the padding string is used.Present as "

   "Custom text" in portal.

   Example: prefix[padding]suffix:3[X-X-X-X]1 →123X-X-X-X9

   Example:
   ```
   Alter Table DemoTable
   Alter Column  SocialSecurityNumber varchar(11) MASKED WITH (FUNCTION ='partial(2,"XXXXX",3)')
   ```

5. Credit card: **Masking method, which exposes the last four digits of the designated fields** and adds a constant string as a prefix in the form of a credit card.
   Example: XXXX-XXXX-XXXX-1234

Note:Masking functions available in portal:Default,Creditcard,Email,Randum number,Custom text

Dropping Mask

ALTER TABLE DemoTable

ALTER COLUMN SocialSecurityNumber DROP MASKED;

To Mask Data using portal:

Azure Portal → Select Database → **Dynamic Data Masking** → + Add Mask (Look at Recommended fields to mask)

Now execute the command again:

```
SELECT * FROM DemoTable
```

We can see that data is still visible as inserted. There is no change in data behavior and the data doesn't mask. The reason for this behavior is user permission. In the current scenario, my ID has db_owner permission and has full access to the data.

To understand the behavior of mask functions and masked data, we will create a new database user TestMaskUser (without login) and will grant select permission on the TestDDM table to the newly created database user.

```
CREATE USER TestMaskUser WITHOUT LOGIN;
GRANT SELECT ON DemoTable TO TestMaskUser;
```

Now, we will change the context of the query execution and review the TestDDM data table.

```
EXECUTE AS USER = 'TestMaskUser';
SELECT * FROM DemoTable;
REVERT;
```

Grant and Revoke UNMASK Permission

UNMASK permission,when granted to a user, the user can see the original values in a table.

GRANT UNMASK TO TestMaskUser;

REVOKE UNMASK TO TestMaskUser;

Note: **Note: We can exclude the SQL users from masking and administrators are always excluded**

T-SQL Command:GRANT UNMASK TO TestMaskUser

More: https://docs.microsoft.com/en-us/sql/relational-databases/security/dynamic-data-masking?view=sql-server-2017

Configure Azure SQL Database Auditing

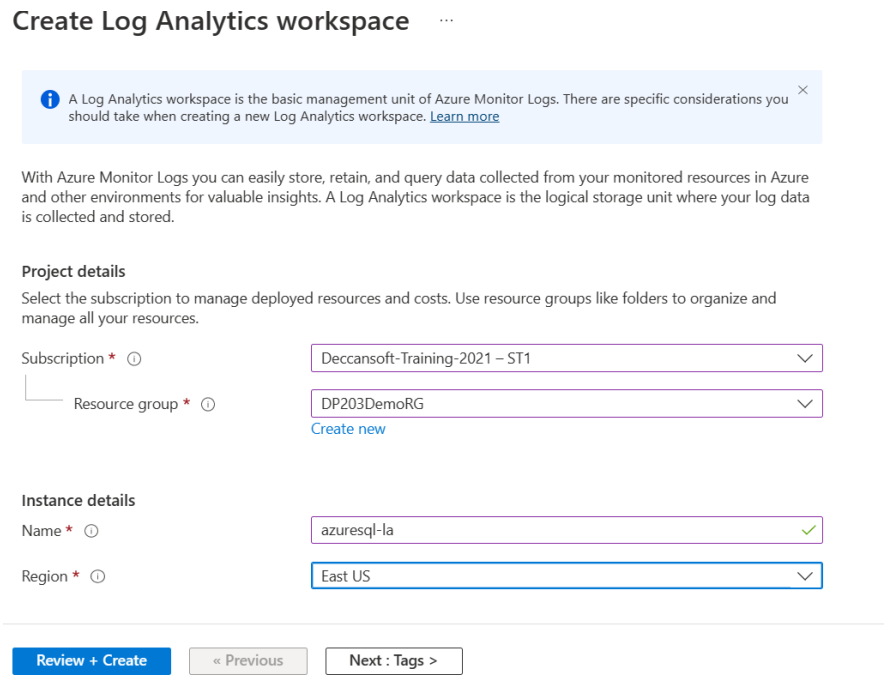
The auditing feature tracks database and server events and writes events to an audit log in either Azure storage or Azure Monitor logs, or to an Azure event hub. Auditing can help you maintain regulatory compliance, understand database activity, and gain insight into discrepancies and anomalies that could indicate potential security violations.

SQL Database Auditing allows you to:

- **Retain** an audit trail of selected events. You can define categories of database actions to be audited.
- **Report** on database activity. You can use preconfigured reports and a dashboard to get started quickly with activity and event reporting.
- **Analyze** reports. You can find suspicious events, unusual activity, and trends.

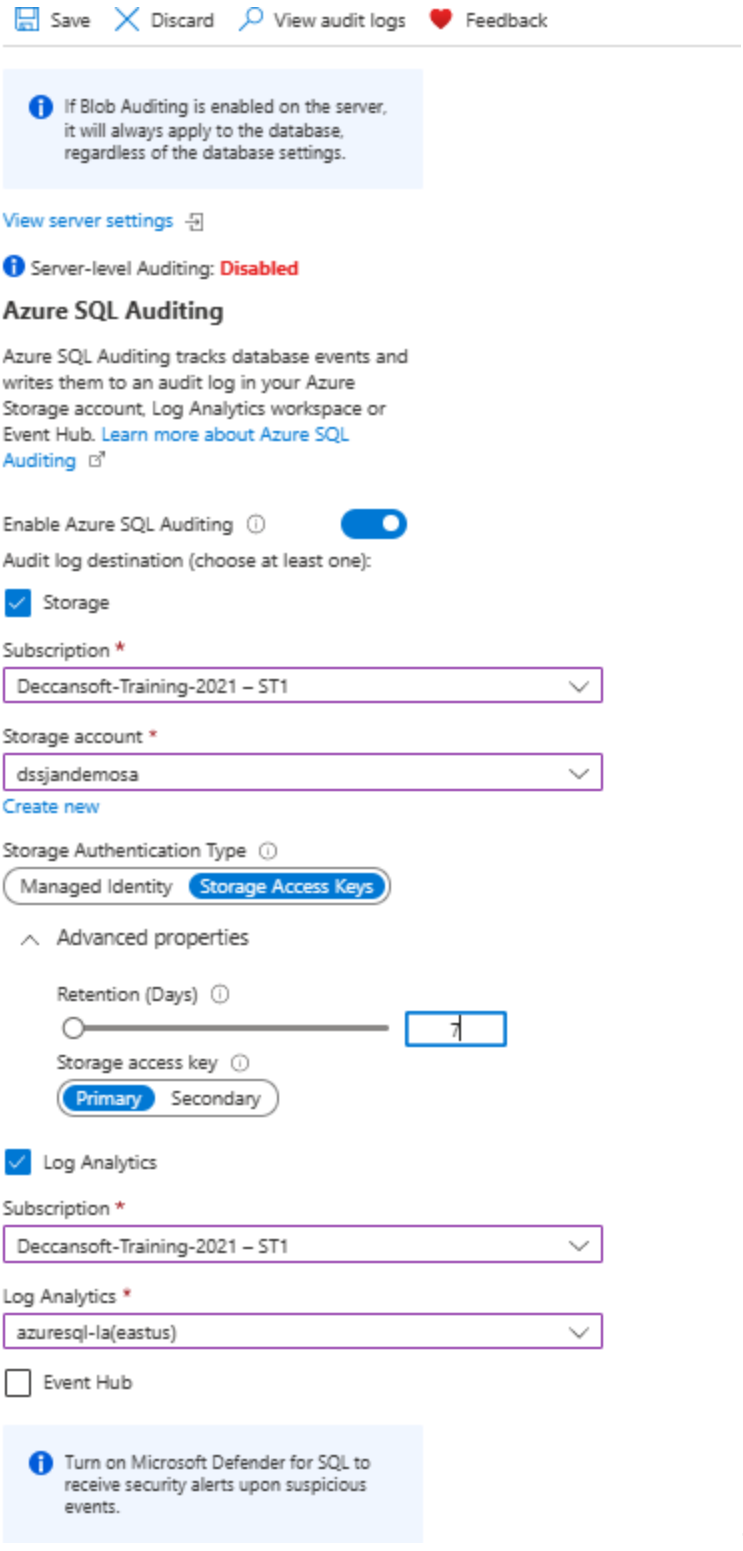
Configure auditing:

1. Create Storage Account
2. Create Log Analytics Workspace



3. Create AdventureWorks Database from the Sample Database provided by Microsoft

AdventureWorks→Security-Auditing→Configure auditing using storage account and log Analytics



→Save

The **selected** storage account will be used to collect XEvent log files, which are saved as a collection of blob files within a container named **sqldbauditlogs**.

4. View Audit Logs

   AdventureWorks→Setting→Auditing→View Audit Logs

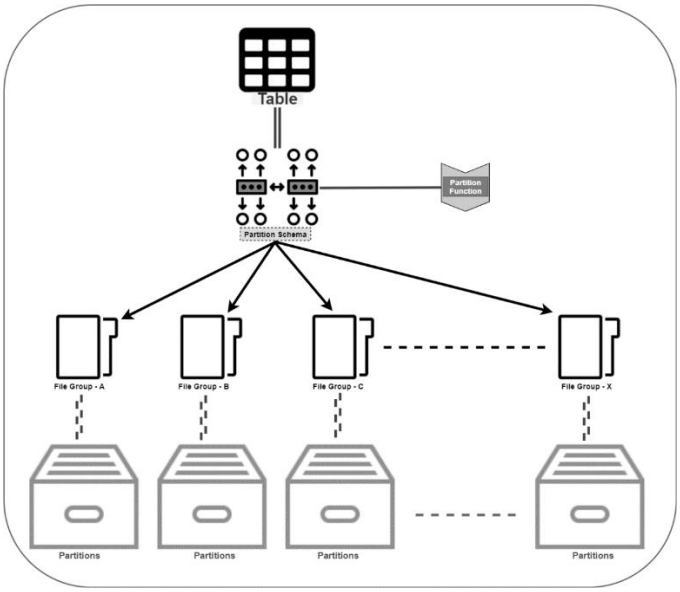
Partitioning

The table partitioning divides the large tables into multiple smaller logical tables. These smaller tables enable better data management and avoid the requirement of creating individual tables. These logical tables are unknown to end-users, and they can query the partitioned table like one logical table.

Benefits:

- You can transfer or access subsets of data quickly and efficiently.
- You can perform maintenance or data retention operations on one or more partitions more quickly. The operations are more efficient because they target only these data subsets, instead of the whole table
- You may improve query performance, based on the types of queries you frequently run.

Partitioning in SQL Server:



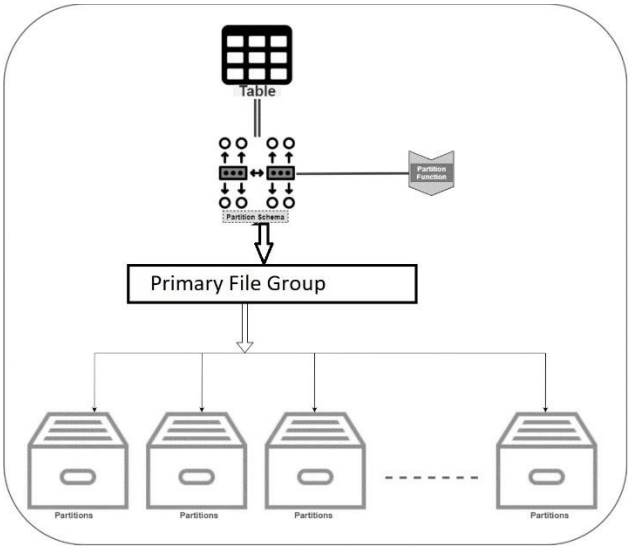
Partitioning Components:

- **Partition function:** The partitioning function defines the boundary for each partition. It maps the rows of a table or index into partitions based on the values of a specified column.
- **Partition Scheme:** The partitioning scheme maps the table partitions to one or different filegroups.
- **Partition column:** The partition function partitions data based on a column. You can use any data types except **ntext**, **text**, **image**, **xml**, **varchar(max)**, **nvarchar(max)**, or **varbinary(max).**

Azure SQL Database Partitioning

Azure SQL Database offers a PaaS solution for Microsoft SQL Server. Microsoft manages these databases; therefore, you have certain limitations in performing administrative activities.

With Azure SQL DB, we cannot add multiple data files and define the partition scheme to use different filegroups.Azure automatically optimized storage in Azure for all table partitions.



Example:

```
CREATE PARTITION FUNCTION myRangePF1 (datetime2(0))
   AS RANGE RIGHT FOR VALUES ('2022-04-01', '2022-05-01', '2022-06-01') ;
GO
CREATE PARTITION SCHEME myRangePS1
   AS PARTITION myRangePF1
   ALL TO ('PRIMARY') ;
GO
CREATE TABLE dbo.PartitionTable (col1 datetime2(0) PRIMARY KEY, col2 char(10))
   ON myRangePS1 (col1) ;
GO
```

Partition Function:

A range type (either LEFT or RIGHT), specifies how the boundary values of the partition function will be put into the resulting partitions:

Example:

CREATE PARTITION FUNCTION YearPartitions (date)AS RANGE RIGHT FOR VALUES ('2010–01–01', '2015–01–01','2020–01–01')

Range Right: the lowest bounding value will be included in each partition. Partition 1: <'2010–01–01' Partition 2: >='2010–01–01'AND <'2015–01–01' Partition 3:>= '2015–01–01'AND <'2020–01–01'

Range Left: The highest bounding value will be included within a partition. Partition 1: <='2010–01–01' Partition 2: >'2010–01–01'AND <='2015–01–01' Partition 3:>= '2015–01–01'AND <='2020–01–01'

Business Continuity – Data Recovery (DR)

- SQL Database **automatically** performs a combination of **full database backups weekly**, **differential database every 12-24 hours,** and **transaction log backups every 5-10 minutes**.
- The full and differential database backups are also **replicated** to a **paired data center** for protection against a data center outage and uses Azure **read-access geo-redundant storage (RA-GRS)** to provide geo-redundancy. However, you can choose to alternatively have your backups in zone-redundant storage (ZRS) or locally-redundant storage (LRS)
- SQL Database provides up to 200% of your maximum provisioned database storage as backup storage at no additional cost. If your database exceeds the provided backup storage, you can choose to **reduce the retention period of PITR**. Another option is to pay for extra backup storage that is billed at the standard Read-Access Geographically Redundant Storage (RA-GRS) rate.
- Each SQL Database backup has a retention period that is based on the service-tier of the database. The retention period for a database in the:
    - Basic service tier is 7 days.
    - Standard service tier is 35 days.
    - Premium service tier is 35 days.

Customizing PITR and Long Term Backup:

SQL Server → Data Management→ Backups → Retention policies→**Select the database** → **Configure Policies**

Recover an Azure SQL Database:

SQL Database provides three options for database recovery using the automated database backups.

1. A new database on the **same logical server** recovered to a specified **point in time** within the retention period. **This will not work in case of Data Center outrage**.
2. **In case of data center outrage:** A **new database** on **any logical server** in any region recovered to the point of the most recent daily **backups in geo-replicated** blob storage (RA-GRS).
3. A database on the **same logical server** recovered to the deletion time for a **deleted database**.

1. Point-in-time restore (PITR) to same Logical Server (Works only if Primary Data Center is Up and running)

You can use the **automated backups** to recover a copy of your database to a known good point in time, provided that time is within the database retention period.

After the database is restored, you can either replace the original database with the restored database or copy the needed data from the restored data into the original database.

a) **Azure Portal → SQL databases** → Select database you want to restore → At the top of your database's blade, select **Restore**

b) Provide the required details specially the Restore Point → OK

2. Restoring Database from **Long Term Back to same Logical Server (Works only if Primary Data Center is Up and running)**

   Azure Portal → SQL databases → Select database you want to restore → At the top of your database's blade, select **Restore** → Change dropdown value to select source = **Long Term Back retention**

Using PowerShell:

```
$Database = Get-AzureRmSqlDatabase -ResourceGroupName "DemoRG" -ServerName "Server01" -DatabaseName "Database01"

Restore-AzureRmSqlDatabase -FromPointInTimeBackup -PointInTime UTCDateTime -ResourceGroupName $Database.ResourceGroupName -ServerName $Database.ServerName -TargetDatabaseName "RestoredDatabase" -ResourceId $Database.ResourceID -Edition "Standard" -ServiceObjectiveName "S2"
```

3. Restore Database to same/diferrent Logical Server (Geo Restore) (To restore from the paired data center of primary location of database)

If your application's downtime does not result in business liability you can use Geo-Restore as a method to recover your application database(s). It creates a copy of the database from its latest geo-redundant backup. **Only automatic hourly backup data can be recovered.**

i. Azure Portal → SQL Databases → **+Add**

ii. Provide basic details → Next

iii. Additional Settings Tab, Provide required details specially

   a. Use existing data = "**Backup**"

   b. Backup = Backup copy of Original Database

iv. Create

Using PowerShell:

```
$GeoBackup = Get-AzureRmSqlDatabaseGeoBackup -ResourceGroupName "ResourceGroup01" -ServerName "Server01" -DatabaseName "Database01"

Restore-AzureRmSqlDatabase -FromGeoBackup -ResourceGroupName "TargetResourceGroup" -ServerName "TargetServer" -TargetDatabaseName "RestoredDatabase" -ResourceId $GeoBackup.ResourceID -Edition "Standard" -RequestedServiceObjectiveName "S2"
```

4. Restore a deleted database to same logical server

If the database is deleted but the logical server has not been deleted, you can restore the deleted database to the point at which it was deleted. This restores a database backup to the same logical SQL server from which it was deleted. You can restore it using the original name or provide a new name or the restored database.

Azure Portal → SQL Servers → Select the Logical Server → In the Summary Blade scroll and go to Operations → **Deleted Databases** → Select the database to restore

```
$DeletedDatabase = Get-AzureRmSqlDeletedDatabaseBackup -ResourceGroupName "ResourceGroup01" -ServerName "Server01" -DatabaseName "Database01"

Restore-AzureRmSqlDatabase -FromDeletedDatabaseBackup -DeletionDate $DeletedDatabase.DeletionDate -ResourceGroupName $DeletedDatabase.ResourceGroupName -ServerName $DeletedDatabase.ServerName -TargetDatabaseName "RestoredDatabase" -ResourceId $DeletedDatabase.ResourceID -Edition "Standard" -ServiceObjectiveName "S2"
```

Use automated backups as your business continuity and recovery mechanism ONLY if your application:

- Is not considered mission critical.
- Doesn't have a binding SLA therefore the downtime of 24 hours or longer will not result in financial liability.
- Has a low rate of data change (low transactions per hour) **and losing up to an hour of change is an acceptable data loss**.

- Is cost sensitive.

Active Geo-Replication

- Active Geo-Replication enables you to configure up to **four readable secondary databases** in the same or different data center locations (regions). Secondary databases are available for **querying and for failover** in the case of a data center outage or the inability to connect to the primary database.
- If the primary database goes offline unexpectedly or you need to take it offline for maintenance activities, you can quickly promote a secondary to become the primary **(also called a failover)** and configure applications (change the connection string) to connect to the newly promoted primary. With a planned failover, there is no data loss. With an unplanned failover, there may be some **small amount of data loss** for very recent transactions due to the nature of asynchronous replication. After a failover, you can later failback - either according to a plan or when the data center comes back online. In all cases, users experience a small amount of downtime and need to reconnect.
- It is used to reduce recovery time and limit data loss associated with a recovery.
- The secondary database must be in the **same service tier** as the primary, so migrating your primary database to a different service tier requires you to either terminate the geo-replication link and rename the secondary database, or simply drop it.

Active Geo-Replication capabilities

1. Automatic Asynchronous Replication.
2. Multiple Secondary databases.
3. Readable secondary databases.
4. Active geo-replication of elastic pool database.
5. Secondary database can have lower performance level (DTU) than primary.
6. User-controlled failover and failback.
7. Keeping credentials and firewall rules in sync

Use Active Geo-Replication if your application meets any of these criteria:

- Is mission critical?
- Has a service level agreement (SLA) that does not allow for 24 hours or more of downtime.
- Downtime will result in financial liability.
- Has a high rate of data change is high and losing an hour of data is not acceptable?
- The additional cost of active geo-replication is lower than the potential financial liability and associated loss of business.

- Key benefit is that the secondary databases are readable and can be used to offload read-only workloads such as reporting jobs.
- **Database migration**: You can use Active Geo-Replication to migrate a database from one server to another online with minimum downtime.
- **Application upgrades**: You can create an extra secondary as a fail back copy during application upgrades.

Enable Geo Replication Backup:

1. Azure Portal → SQL databases → Select database → Scroll database blade → **Configure Geo Replication**
2. Select Target Region → Provide required details → OK

Note: The non-readable secondary type will be retired and existing non-readable databases will automatically be upgraded to readable secondaries.

To failover to a secondary and promote as primary:

1. Primary Database → On the SQL Database blade, select All settings → Geo-Replication.
2. In the **SECONDARIES** list, select the database you want to become the new primary and click **Failover**.

Note: There is a short period during which both databases are unavailable (on the order of 0 to 25 seconds) while the roles are switched. If the primary database has multiple secondary databases, the command automatically reconfigures the other secondaries to connect to the new primary. The entire operation should take less than a minute to complete under normal circumstances

Powershell:

Cmdlet	Description
Get-AzureRmSqlDatabase	Gets one or more databases.
New-AzureRmSqlDatabaseSecondary	Creates a secondary database for an existing database and starts data replication.
Set-AzureRmSqlDatabaseSecondary	Switches a secondary database to be primary to initiate failover.
Remove-AzureRmSqlDatabaseSecondary	Terminates data replication between a SQL Database and the specified secondary database.
Get-AzureRmSqlDatabaseReplicationLink	Gets the geo-replication links between an Azure SQL Database and a resource group or SQL Server.

Summary / Comparison table

The following table compares the ERT and RPO for the three most common scenarios.

Capability	Basic tier	Standard tier	Premium tier

Point in Time Restore from backup	Any restore point within 7 days	Any restore point within 35 days	Any restore point within 35 days
Restore from Long Term Backup	ERT < 12h, RPO < 1 wk	ERT < 12h, RPO < 1 wk	ERT < 12h, RPO < 1 wk
Automatic Geo-Restore from geo-replicated backups	ERT < 12h, RPO < 1h	ERT < 12h, RPO < 1h	ERT < 12h, RPO < 1h
Active Geo-Replication	ERT < 30s, RPO < 5s	ERT < 30s, RPO < 5s	ERT < 30s, RPO < 5s

ERT = Estimated Recovery Time

RPO = Recovery Point Objective is the maximum amount of recent data updates (time interval) the application can tolerate losing when recovering after the disruptive event.

Export and Import of Database using .bacpac

In Azure SQL Database, you **cannot** directly use the database and transaction log backup capabilities of SQL Server. Historically, this was remediated by periodically **exporting a copy** of each database that you want to protect, and storing the copy in a .**bacpac** file in a storage account. In the event of a SQL database or server failure, you could then create a new SQL database server, if necessary, and **import the copy** of the database from the exported file.

Export of Database:

- When you need to export a database for archiving or for moving to another platform, you can export the database schema and data to a BACPAC file.
- A BACPAC file is a ZIP file with an extension of BACPAC containing the metadata and data from a SQL Server database.
- A BACPAC file can be stored in Azure blob storage or in local storage in an on-premises location and later imported back into Azure SQL Database or into a SQL Server on-premises installation.
- If you are exporting to blob storage, **the maximum size of a BACPAC file is 200 GB**. To archive a larger BACPAC file, export to local storage.
- For an export to be transactionally consistent, you must ensure either **that no write activity** is occurring during the export, or that you are exporting from a transactionally consistent **copy** of your Azure SQL database.

Steps to Export:

1. Azure Portal → SQL databases → Select the Database
2. **Copy Database:** Azure Portal → SQL databases → Select the Database → Click Copy in database blade → Provide the required details → OK.

a. Can be either of same or different server

b. Service Tier can be changed.

3. Goto to **Copy of database** → Click **Export** in database blade → Provide the required details including Storage Account, Server Admin Login/Password → OK

Note: The length of time the export will take depends on the size and complexity of your database, and your service level. You will receive a notification on completion.

4. **Monitor the progress of the export operation**

   Azure Portal → Click SQL servers → click the server containing the original (source) database you just archived → Scroll down to Operations → click Import/Export history**:**

Note: The newest versions (v17 / 2017) of SQL Server Management Studio also provide a wizard to export an Azure SQL Database to a bacpac file.(Database→RC→Tasks→Export Data Tier Application)

Import a BACPAC file to create an Azure SQL database

1. Azure Portal → **SQL Servers** → In SQL Server blade → **Import database**
2. Click **Storage** and select your storage account, blob container, and .bacpac file and click **OK**
3. Select the pricing tier for the new database and click **Select**
4. Enter a **Database Name** for the database you are creating from the BACPAC file.
5. Choose the authentication type and then provide the authentication information for the server.
6. Click **Create** to create the database from the BACPAC.