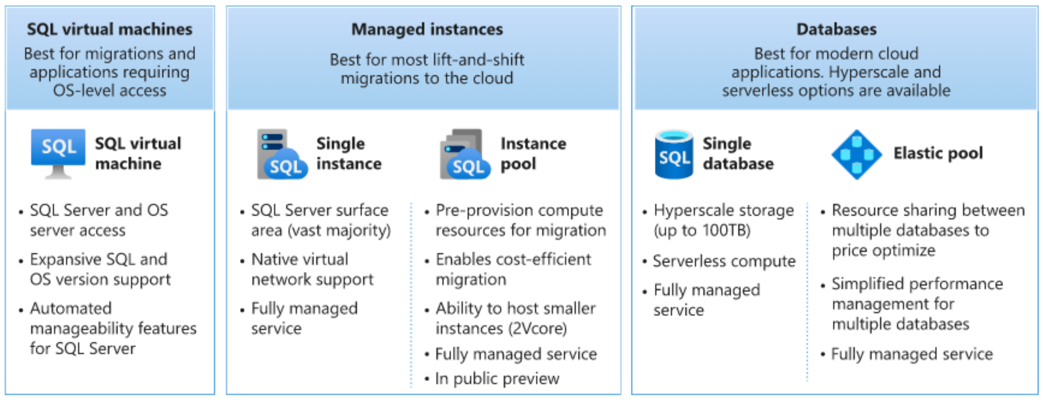
Managing Data in Azure SQL Database

- 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
- Export/Import

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 virtual machines
Best for migrations and applications requiring OS-level access
SQL virtual machine
• SQL Server and OS server access
• Expansive SQL and OS version support
• Automated manageability features for SQL Server
Managed instances
Best for most lift-and-shift migrations to the cloud
Single instance
• SQL Server surface area (vast majority)
• Native virtual network support
• Fully managed service
Instance pool
• Pre-provision compute resources for migration
• Enables cost-efficient migration
• Ability to host smaller instances (2Vcore)
• Fully managed service
• In public preview
Databases
Best for modern cloud applications. Hyperscale and serverless options are available
Single database
• Hyperscale storage (up to 100TB)
• Serverless compute
• Fully managed service
Elastic pool
• Resource sharing between multiple databases to price optimize
• Simplified performance management for multiple databases
• Fully managed service

- **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).

Refer:

https://learn.microsoft.com/en-us/azure/azure-sql/database/service-tiers-dtu?view=azuresql

https://learn.microsoft.com/en-us/azure/azure-sql/database/service-tiers-sql-database-vcore?view=azuresql

Choosing service Tier

The following table provides examples of the tiers best suited for different application workloads.

Service tier	Target workloads
Basic	Best suited for a small database, supporting typically **one single active operation** at a given time. Examples include databases used for development or testing, or small-scale infrequently used applications. For infrequent access and less demanding workloads.

Standard/General Purpose	Suited for most of the **generic ,budget oriented** workloads.The go-to option for cloud applications with low to medium IO performance requirements, supporting **multiple concurrent queries**. Examples include web applications.
Premium/Business Critical	Designed for high transactional volume with high IO performance requirements, supporting many concurrent users. Suited for applications that require **low-latency** responses from the underlying SSD storage (1-2 ms in average), **fast recovery** if the underlying infrastructure fails, or need reporting and read only analytic queries that can be redirected to the free-of-charge secondary read-only replica..Examples are databases supporting mission critical applications.
Hyperscale	Suited for large databases as it supports database size upto 100TB. Also suited for smaller databases, but require fast **vertical and horizontal compute scaling, high performance, instant backup, and fast database restore.**

Azure SQL Managed Instance

1. It is a deployment model of Azure SQL, 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

<u>Comparison</u>

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.
99.99 to 99.995% availability guaranteed based on service tier and zone redundancy	At least 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
  - Logon audit is done through SQL audit.
  - Server proxy account isn't applicable.

Creating SQL Database

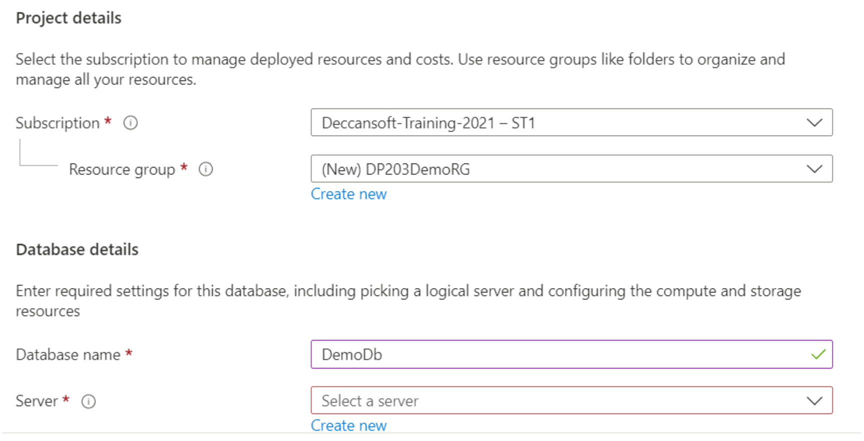
With Windows Azure SQL Database you can quickly create database solutions that are built on the SQL Server database engine. We can create a new SQL database in Windows Azure and then configure it later. We can decide whether to use an existing SQL database server or create a new one when you create your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

<u>Azure SQL logical server</u>

- 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

All services→Categories→Databases→Create SQL Database



Project details

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

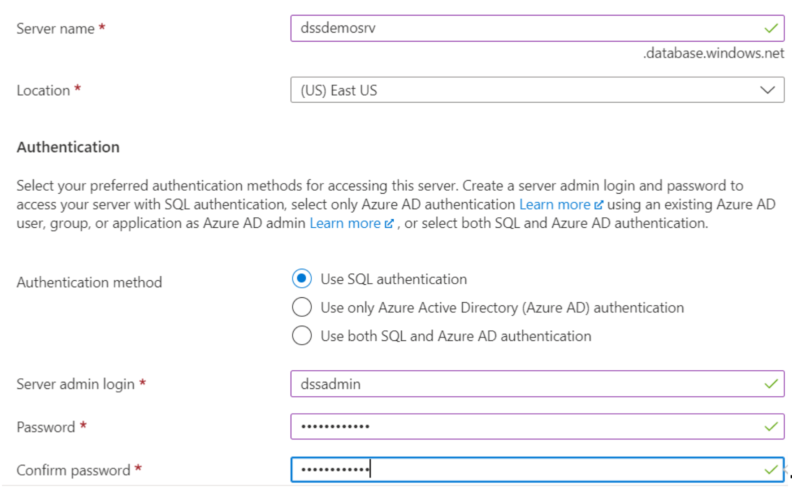
Subscription *	Deccansoft-Training-2021 – ST1
Resource group *	(New) DP203DemoRG Create new

Database details

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

Database name *	DemoDb
Server *	Select a server Create new

→Create new



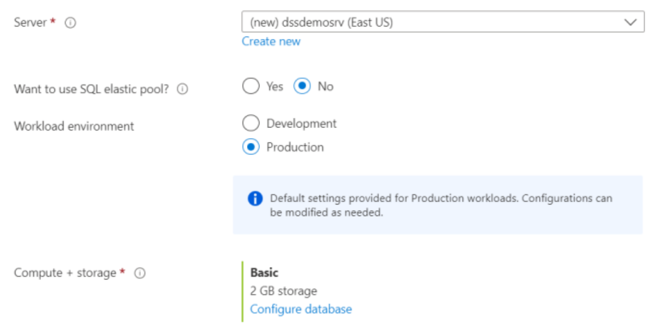
Server name *	dssdemosrv .database.windows.net
Location *	(US) East US

Authentication

Select your preferred authentication methods for accessing this server. Create a server admin login and password to access your server with SQL authentication, select only Azure AD authentication Learn more using an existing Azure AD user, group, or application as Azure AD admin Learn more , or select both SQL and Azure AD authentication.

Authentication method	[✓ Use SQL authentication] Use only Azure Active Directory (Azure AD) authentication Use both SQL and Azure AD authentication
Server admin login *	dssadmin
Password *	••••••••••••
Confirm password *	••••••••••••

→OK


Server * (new) dssdemosrv (East US)
Create new

Want to use SQL elastic pool? Yes [✓ No]

Workload environment Development [✓ Production]

Default settings provided for Production workloads. Configurations can be modified as needed.

Compute + storage * Basic 2 GB storage
Configure database

→Review+Create→Create

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

Lab 2: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 Database → Select your database→set server Firewall → **Firewall**
    b. **Public network access :**Select “Selected networks”
    c. **Firewall rules:**
    d. Click on Add your client Ipv4 address

        Click Add Client IP**.**

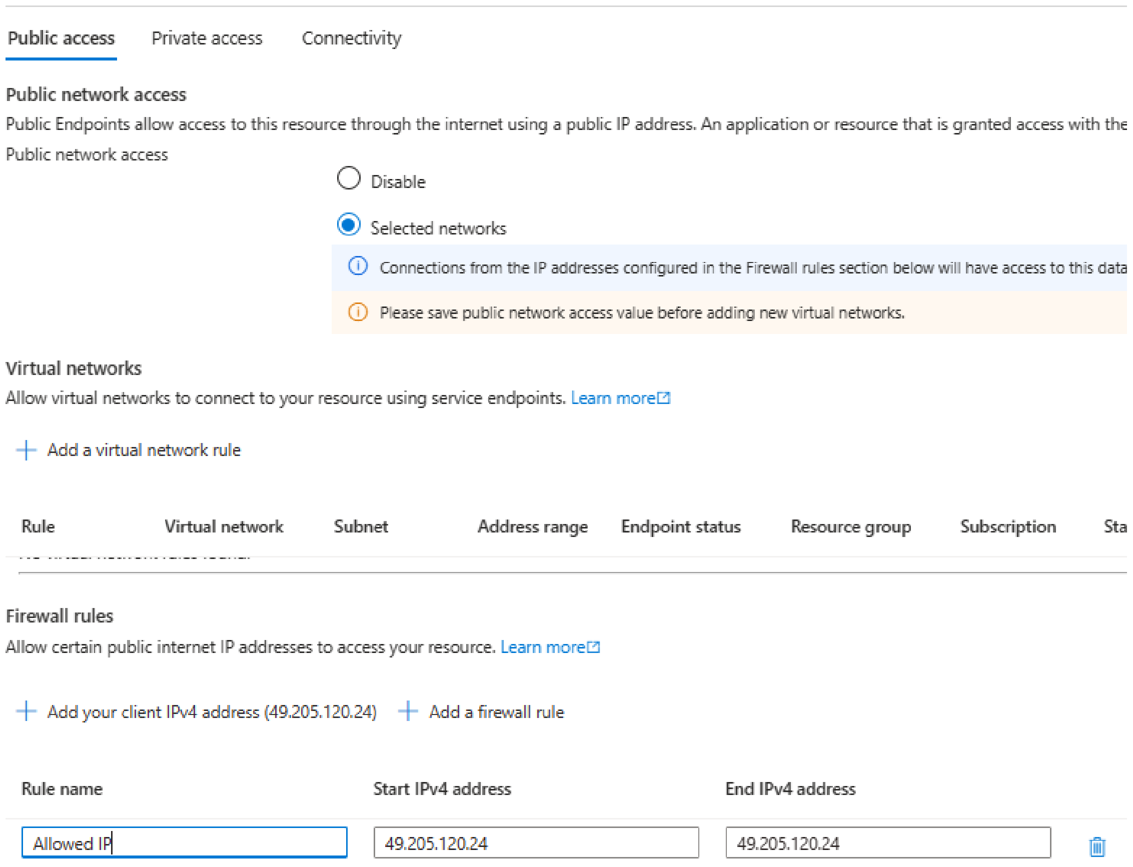
        i. Rule Name = "Allowed IP".

    ii. Start IP = 49.12.12.0

    iii. End IP = 49.12.12.5

  e. Click Save

5. Your screen will look similar to as shown below.



Public access | Private access | Connectivity

Public network access

Public Endpoints allow access to this resource through the internet using a public IP address. An application or resource that is granted access with the

Public network access

○ Disable

● Selected networks

Connections from the IP addresses configured in the Firewall rules section below will have access to this data

Please save public network access value before adding new virtual networks.

Virtual networks

Allow virtual networks to connect to your resource using service endpoints. Learn more

+ Add a virtual network rule

Rule	Virtual network	Subnet	Address range	Endpoint status	Resource group	Subscription	Sta

Firewall rules

Allow certain public internet IP addresses to access your resource. Learn more

+ Add your client IPv4 address (49.205.120.24) + Add a firewall rule

Rule name	Start IPv4 address	End IPv4 address
Allowed IP	49.205.120.24	49.205.120.24

6. Return back to SSMS and try to connect again. (It might take some time 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';
```

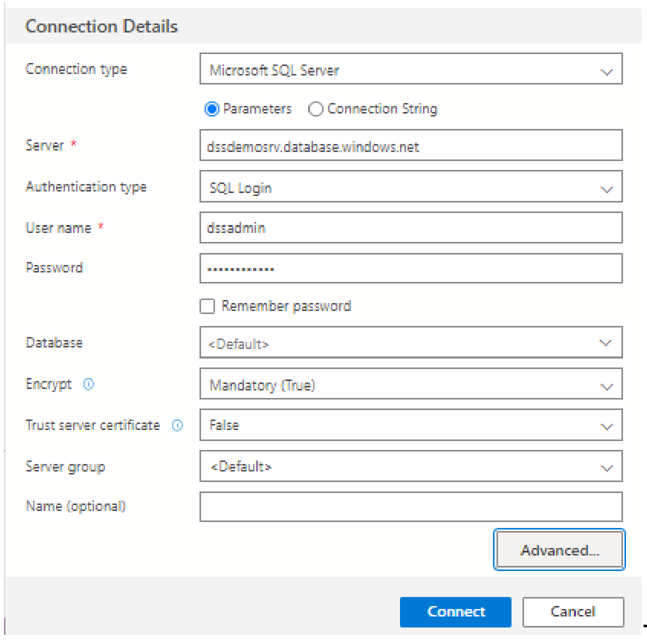
Note: If you specify an IP address range in the database-level IP firewall rule that's outside the range in the server-level IP firewall rule, only those clients that have IP addresses in the database-level range can access the database.

Lab3 :Using Azure Data Studio:

Azure Data Studio is a lightweight, cross-platform data management and development tool with connectivity to popular cloud and on-premises databases.

Refer: https://learn.microsoft.com/en-us/sql/azure-data-studio/what-is-azure-data-studio?view=sql-server-ver16

1. Start Start SQL Server Management Studio locally
2. Connections→Servers→



Connection Details

Connection type	Microsoft SQL Server
	(● Parameters ○ Connection String)
Server *	dssdemosrv.database.windows.net
Authentication type	SQL Login
User name *	dssadmin
Password	•••••••••••••
	☐ Remember password
Database	<Default>
Encrypt	Mandatory (True)
Trust server certificate	False
Server group	<Default>
Name (optional)	

→Connect

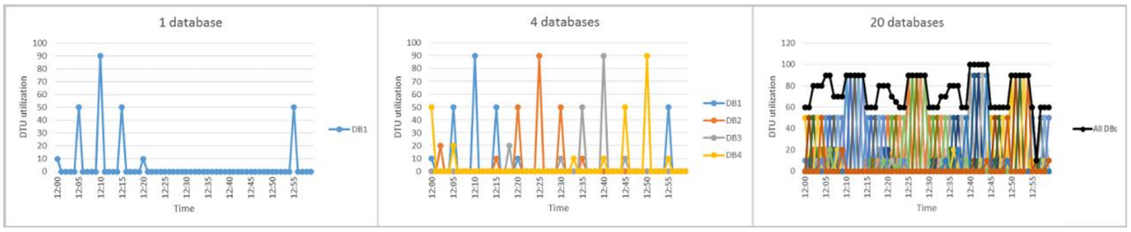
Elastic Pools

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:

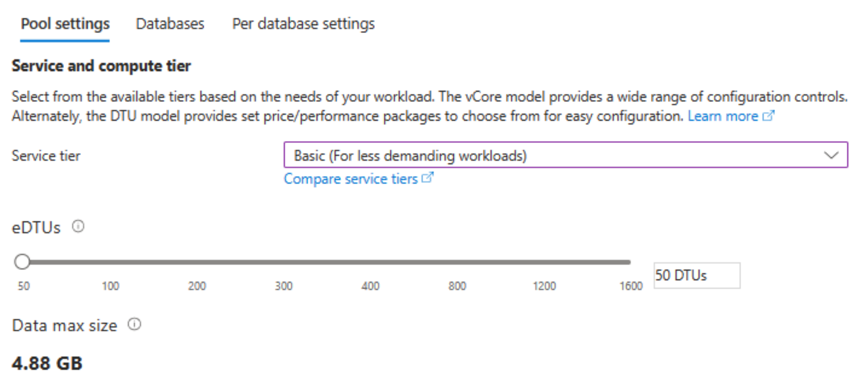
- Maximum DTUs utilized by all databases in the pool.
- Maximum storage bytes utilized by all databases in the pool.

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.

Lab4 :Creating a Pool and adding database to it.

1. Azure Portal → All services→Database→**SQL Servers** → →Select Your SQL Server→Server blade → +New Pool
2. Name = DemoPool

3. Compute+storage = Click on configure elastic 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
5. Pool settings



Service and compute tier

Select from the available tiers based on the needs of your workload. The vCore model provides a wide range of configuration controls. Alternately, the DTU model provides set price/performance packages to choose from for easy configuration. Learn more

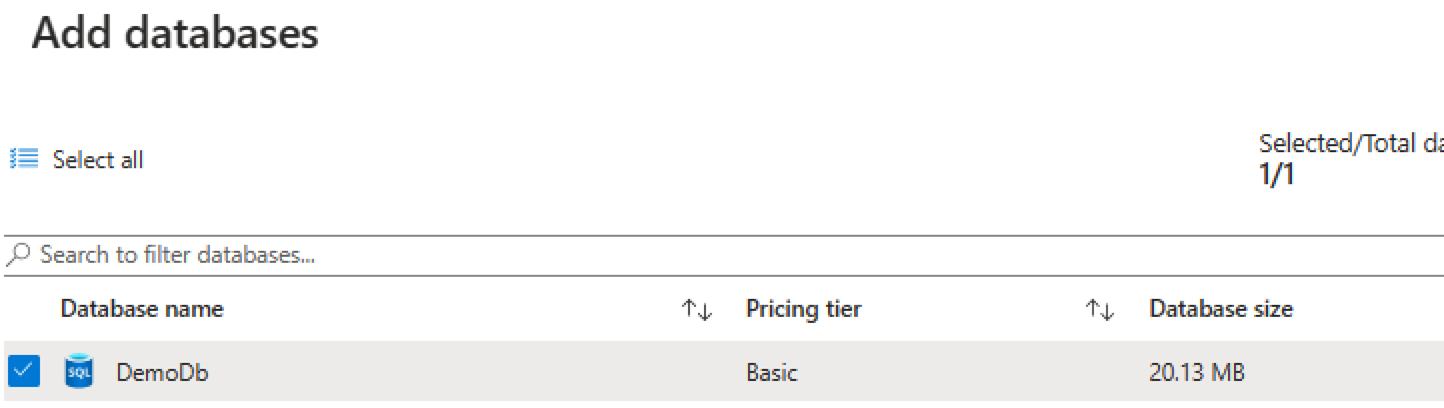
Service tier: Basic (For less demanding workloads)

50 DTUs

Data max size

4.88 GB

6. Databases→+Add Databse→Select your databse→



Add databases

Selected/Total da
1/1

	Database name	Pricing tier	Database size
✓	DemoDb	Basic	20.13 MB

→Apply

7. Per database settings can be specified for eDTU max and min →Apply

Creating a Pool and adding database to it.

1. Azure Portal→Serach for SQL **Elastic Pool**→Select SQL Elastic pool→+Create→
2. You can configure as shown in above example

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

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

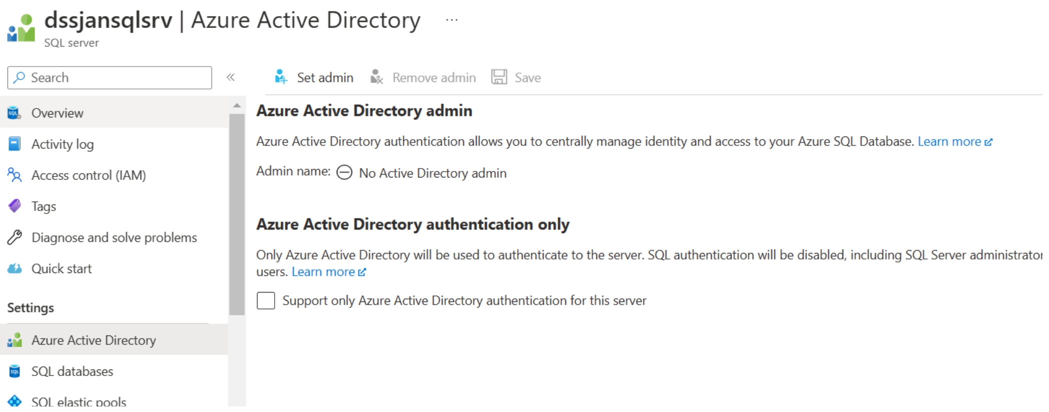
- Traditionally for users to have access to database, the master database must have a login that matches the connecting credentials and the login must be able to be mapped to a database user in the user database.
- connection to the user database has a dependency upon the login in the master database, 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.

Lab5 :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


dssjansqlsrv | Azure Active Directory

SQL server

Azure Active Directory admin

Azure Active Directory authentication allows you to centrally manage identity and access to your Azure SQL Database. Learn more

Admin name: No Active Directory admin

Azure Active Directory authentication only

Only Azure Active Directory will be used to authenticate to the server. SQL authentication will be disabled, including SQL Server administrator users. Learn more

☐ Support only Azure Active Directory authentication for this server

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

IV. Save

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** .Replace[abc@Xyz.com] with required user id

```
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:

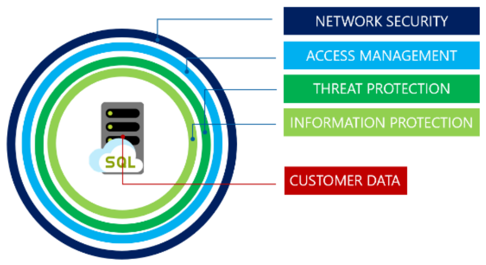
```
CREATE USER [ADGroup1] FROM EXTERNAL PROVIDER;
```

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

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

Azure SQL Database Security

Azure SQL Database has multiple layers of security as shown:



Network Security

- IP Firewall rule: Firewalls prevent network access to the database server.You need to grant access to set of IP addresses
- Virtual network firewall rules: Virtual network rules enable Azure SQL Database to only accept communications that are sent from selected subnets inside a virtual network.

Access Management

- Authentication :Process of validating identity of user.
- Azure SQL Database supports two types of authentication
  1. SQL Authentication :Authenticate the request using username and password
  2. **Azure Active Directory authentication**: It is a mechanism of connecting to Azure SQL Database by using identities in Azure Active Directory Azure AD authentication allows administrators to centrally manage the identities and permissions of database users along with other Microsoft services in one central location.

Authorization

- Authorization refers to the permissions assigned to a user within an Azure SQL Database, and determines what the user is allowed to do.
  - Permissions can be manged by using database roles or specifying object level permission.

Threat Protection

- Auditing:Database events can be captured to an audit log in a customer-owned Azure storage account. Auditing allows users to monitor ongoing database activities, as well as analyze and investigate historical activity to identify potential threats or suspected abuse and security violations

- Advanced Threat Detection: Advanced Threat Protection is analyzing your SQL Server logs to detect unusual behavior and potentially harmful attempts to access or exploit databases.

Information Protection:

- **TLS:** SQL Database secures customer data by encrypting data in motion with Transport Layer Security.
- **TDE:** It adds layer of security to protect data at rest from unauthorized or offline access to raw files or backups.Entire database in encrypted using encryption keys .encryption key is protected by a built-in server certificate. Certificate maintenance and rotation are managed by the service and requires no input from the user

Key management with Azure Key Vault

- Customers who prefer to take control of the encryption keys can manage the keys in Azure Key Vault.

Dynamic data Masking

- SQL Database dynamic data masking limits sensitive data exposure by masking it to non-privileged users.

Always Encrypted

- **Always Encrypted** is a feature designed to protect sensitive data(Example:Credit card number) stored in specific database columns from access.
- Sensitive Data is protected at rest on the server, during movement between client and server, and while the data is in use, ensuring that sensitive data never appears as plaintext inside the database system. This data will not be available for administrators also.
- Encrypted data is decrypted only for processing by client applications with access to the encryption key. The encryption key is never exposed to SQL and can be stored either in the Windows Certificate Store or in Azure Key Vault.

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

Lab6:Explore Dynamic Data Masking

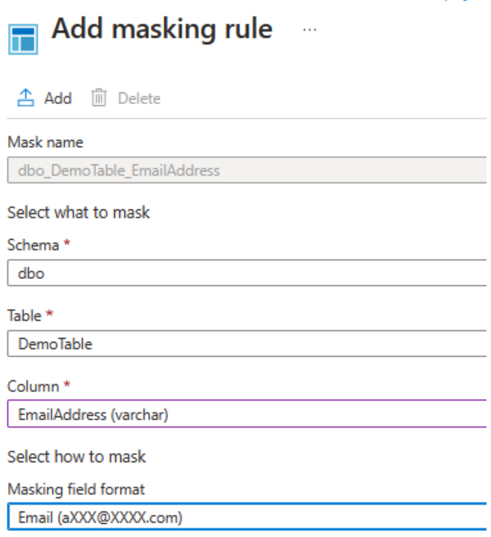
1. Create Following TABLE

```
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
```

2. **To Mask Data using portal**:

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

Example:Add mask for Email Address in DemoTable


Add masking rule

Add Delete

Mask name
dbo_DemoTable_EmailAddress

Select what to mask

Schema *
dbo

Table *
DemoTable

Column *
EmailAddress (varchar)

Select how to mask

Masking field format
Email (aXXX@XXXX.com)

→Add→Save

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

3. 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.

4. **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;
```

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

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

6. **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;
```

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

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

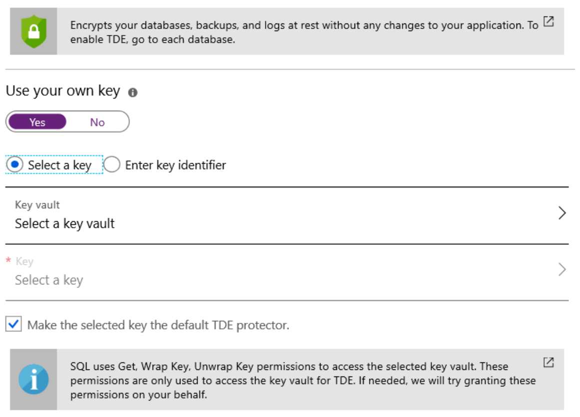
ALTER COLUMN SocialSecurityNumber DROP MASKED;

Transparent Data Encryption

TDE is to encrypt data at rest.

SQL Server→ Select Server → Transparent data encryption

Go to Database → Transparent data encryption → Enable On/Off



Encrypts your databases, backups, and logs at rest without any changes to your application. To enable TDE, go to each database.

Use your own key

[✓ Yes] No

[✓ Select a key] Enter key identifier

Key vault
Select a key vault

* Key
Select a key

✓ Make the selected key the default TDE protector.

SQL uses Get, Wrap Key, Unwrap Key permissions to access the selected key vault. These permissions are only used to access the key vault for TDE. If needed, we will try granting these permissions on your behalf.

Configure Azure SQL Database Auditing

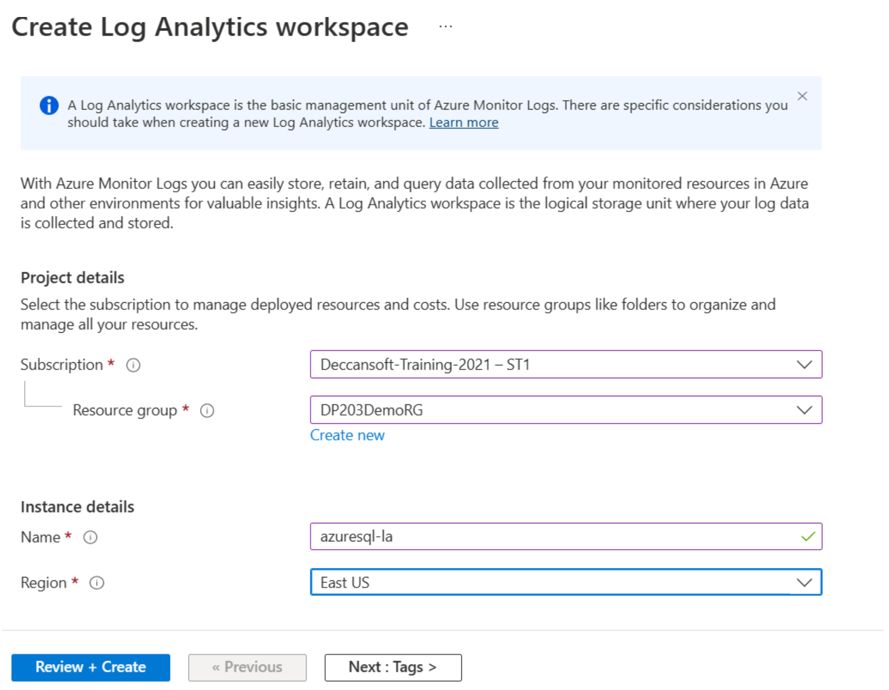
The auditing feature tracks database and server events and writes events to an audit log in either Azure storage or Azure Monitor logs, or to an Azure event hub. Auditing can help you maintain regulatory compliance, understand database activity, and gain insight into discrepancies and anomalies that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

Lab 7:Configure auditing:

1. Create Storage Account
2. Create Log Analytics Workspace


Create Log Analytics workspace

A Log Analytics workspace is the basic management unit of Azure Monitor Logs. There are specific considerations you should take when creating a new Log Analytics workspace. Learn more

With Azure Monitor Logs you can easily store, retain, and query data collected from your monitored resources in Azure and other environments for valuable insights. A Log Analytics workspace is the logical storage unit where your log data is collected and stored.

Project details

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

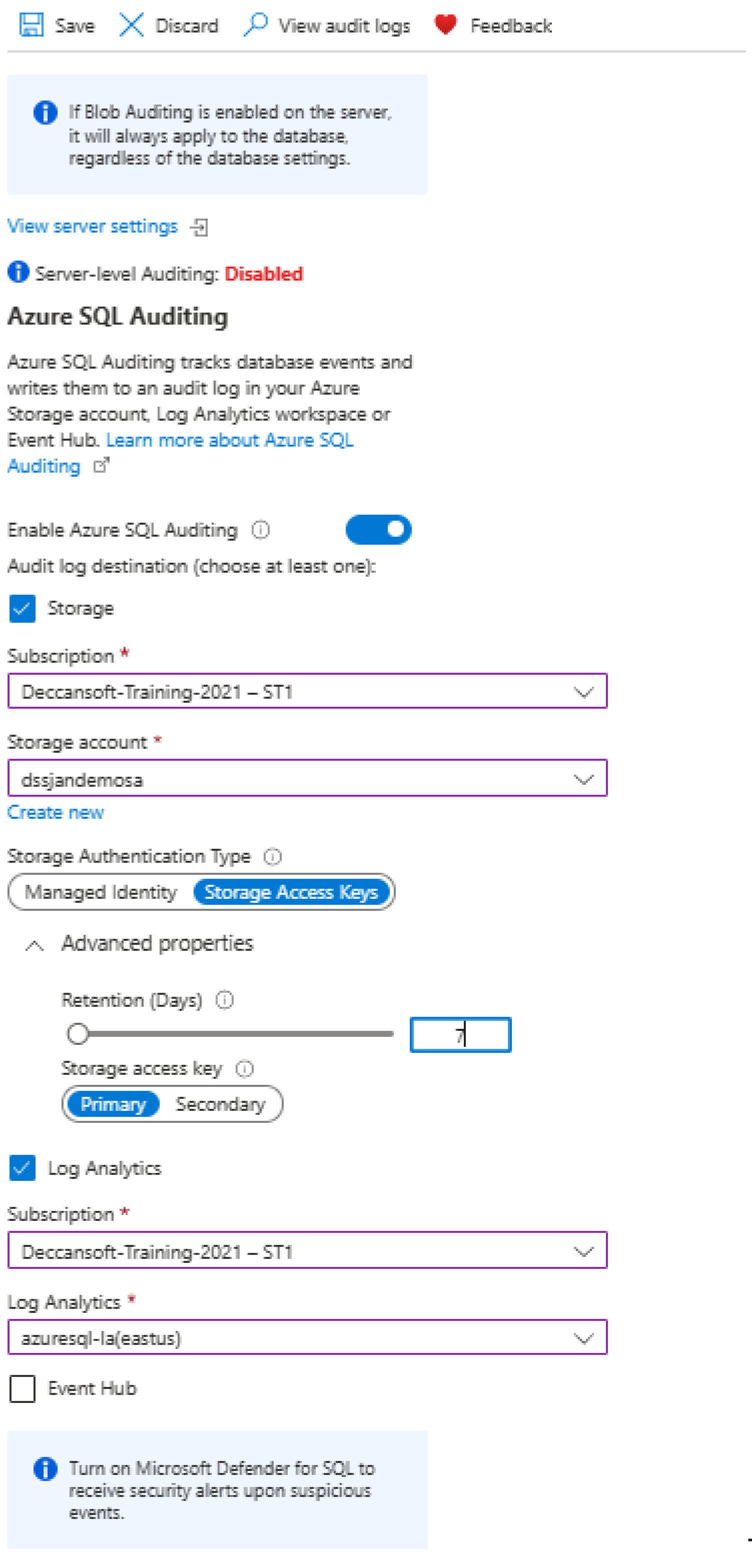
Subscription *	Deccansoft-Training-2021 – ST1
Resource group *	DP203DemoRG
	Create new

Instance details

Name *	azuresql-la
Region *	East US

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

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


Save Discard View audit logs Feedback

If Blob Auditing is enabled on the server, it will always apply to the database, regardless of the database settings.

View server settings

Server-level Auditing: **Disabled**

Azure SQL Auditing

Azure SQL Auditing tracks database events and writes them to an audit log in your Azure Storage account, Log Analytics workspace or Event Hub. Learn more about Azure SQL Auditing

Enable Azure SQL Auditing: On

Audit log destination (choose at least one):

- [✓] Storage

Subscription *: Deccansoft-Training-2021 – ST1

Storage account *: dssjandemosa

Create new

Storage Authentication Type: Managed Identity [✓ Storage Access Keys]

Advanced properties

Retention (Days): 7

Storage access key: [✓ Primary] Secondary

- [✓] Log Analytics

Subscription *: Deccansoft-Training-2021 – ST1

Log Analytics *: azuresql-la(eastus)

- [ ] Event Hub

Turn on Microsoft Defender for SQL to receive security alerts upon suspicious events.

→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

Export and Import of Database using .bacpac

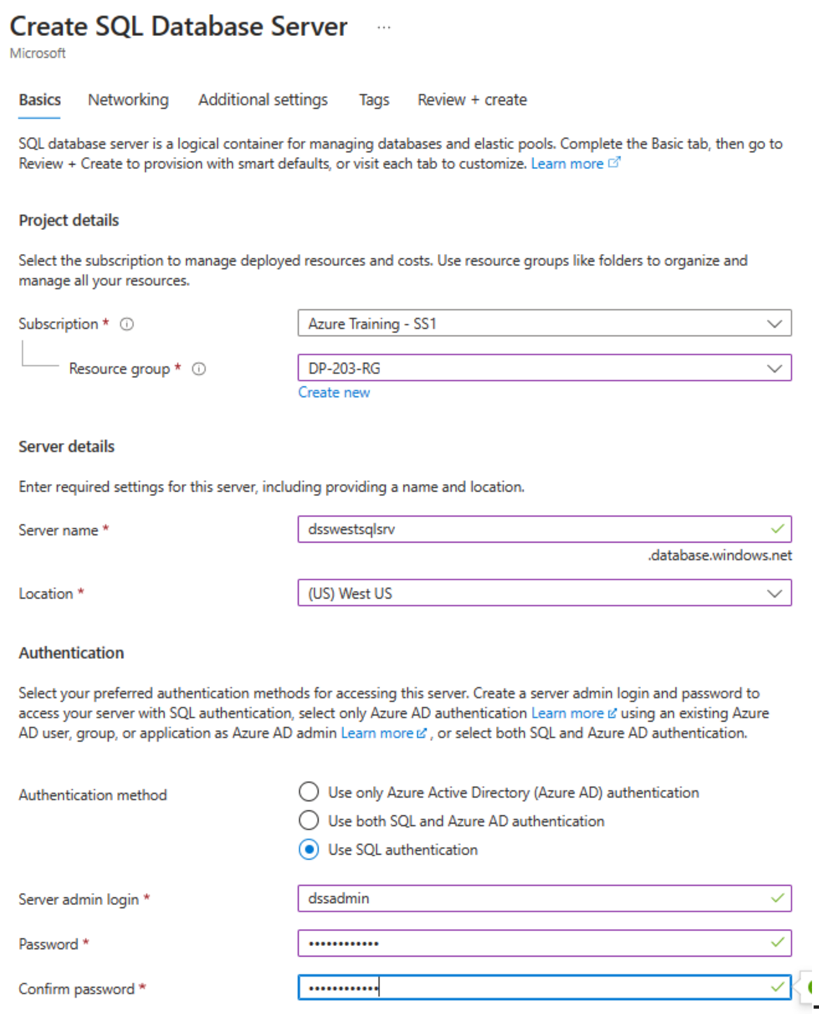
In Azure SQL Database, you **cannot** directly use the database and transaction log backup capabilities of SQL Server. Historically, this was remediated by periodically **exporting a copy** of each database that you want to protect, and storing the copy in a .**bacpac** file in a storage account. In 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.

Lab 8: Explore Copy and Export option

Prerequsite: Create Logical server in West-us Region



Create SQL Database Server ...
Microsoft

SQL database server is a logical container for managing databases and elastic pools. Complete the Basic tab, then go to Review + Create to provision with smart defaults, or visit each tab to customize. Learn more

Project details

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

Subscription *	Azure Training - SS1
Resource group *	DP-203-RG
	Create new

Server details

Enter required settings for this server, including providing a name and location.

Server name *	dsswestsqlsrv .database.windows.net
Location *	(US) West US

Authentication

Select your preferred authentication methods for accessing this server. Create a server admin login and password to access your server with SQL authentication, select only Azure AD authentication Learn more using an existing Azure AD user, group, or application as Azure AD admin Learn more , or select both SQL and Azure AD authentication.

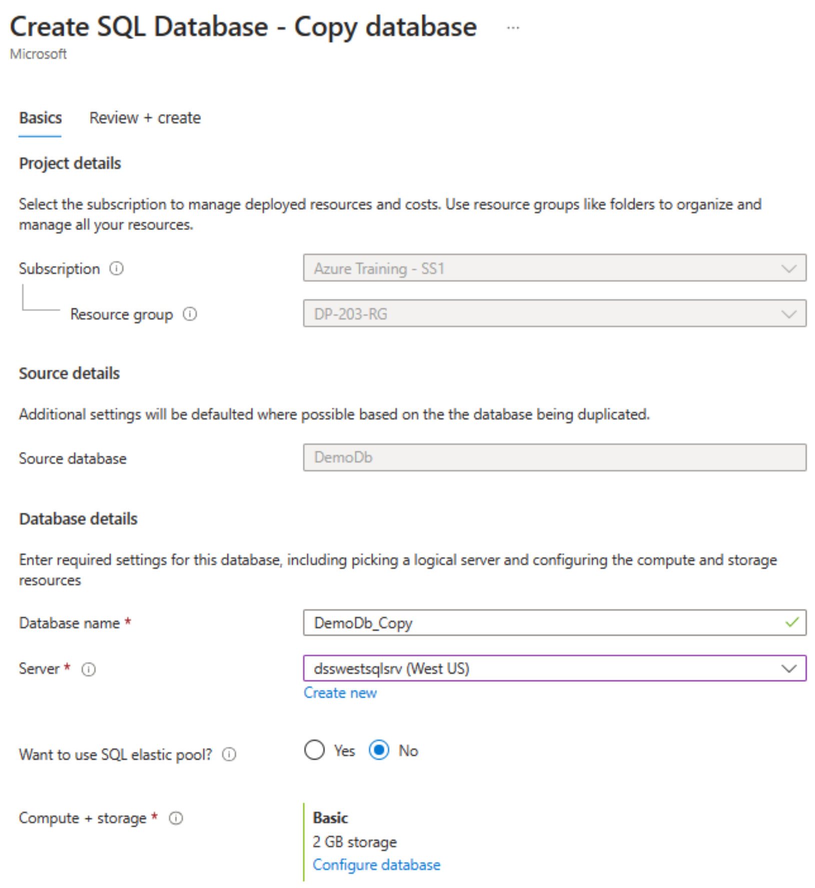
Authentication method: Use only Azure Active Directory (Azure AD) authentication / Use both SQL and Azure AD authentication / [✓ Use SQL authentication]

Server admin login *	dssadmin
Password *	············
Confirm password *	············

→**Review+Create→Create**

Steps to copy Database to different server and then Export:

1. Azure Portal → SQL database→Select Your Database(DemoDb) → **Copy**
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.


Create SQL Database - Copy database ...

Microsoft

Basics | Review + create

Project details

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

Subscription ⓘ	Azure Training - SS1
Resource group ⓘ	DP-203-RG

Source details

Additional settings will be defaulted where possible based on the the database being duplicated.

Source database	DemoDb

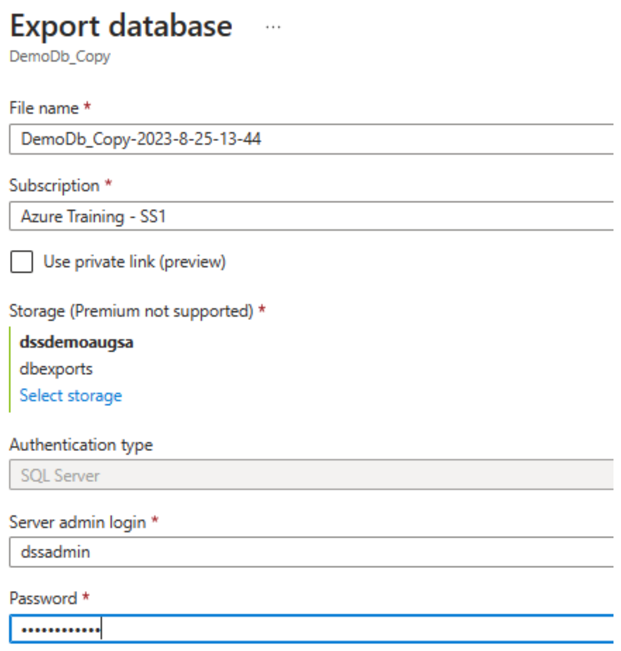
Database details

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

Database name *	DemoDb_Copy
Server * ⓘ	dsswestsqlsrv (West US) Create new
Want to use SQL elastic pool? ⓘ	Yes [✓ No]
Compute + storage * ⓘ	**Basic** 2 GB storage Configure database

→Review+Create→Create

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


Export database ...

DemoDb_Copy

File name *

DemoDb_Copy-2023-8-25-13-44

Subscription *

Azure Training - SS1

☐ Use private link (preview)

Storage (Premium not supported) *

dssdemoaugsa

dbexports

Select storage

Authentication type

SQL Server

Server admin login *

dssadmin

Password *

••••••••••••

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**

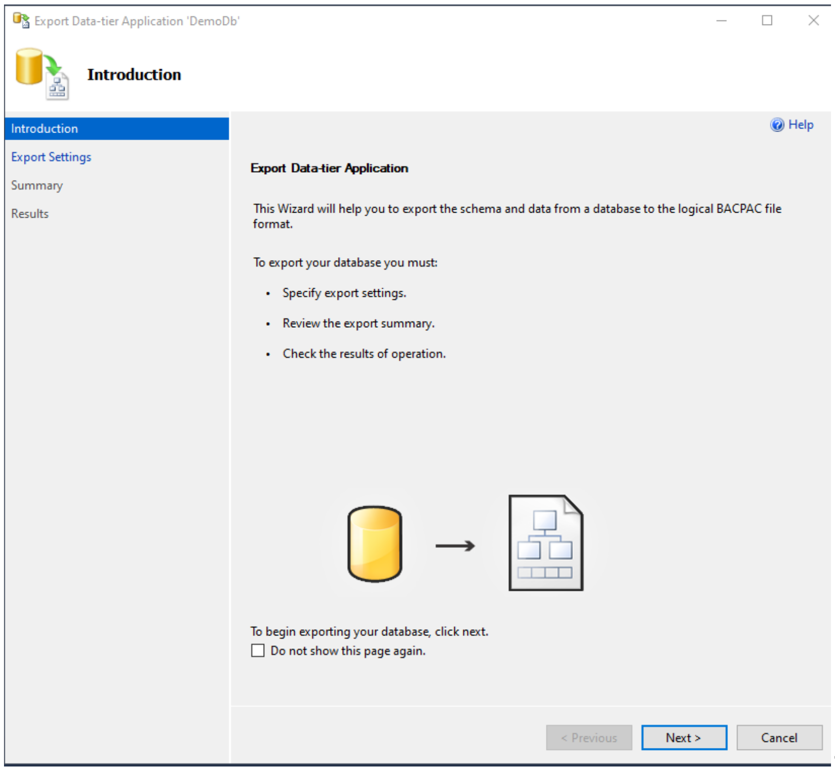
   Azure Portal → Click **SQL servers** → click the server(dsswestsqlsrv) containing the original (source) database you just archived →Scroll down to Data Management→ click **Import/Export history**:

Lab 9 :Export Database using SSMS

The newest versions (v17 / 2017) of SQL Server Management Studio also provide a wizard to export an Azure SQL Database to a bacpac file.

Steps:

1. (Databases→Select your Database(Demodb)→Tasks→Export Data Tier Application)



Export Data-tier Application

This Wizard will help you to export the schema and data from a database to the logical BACPAC file format.

To export your database you must:

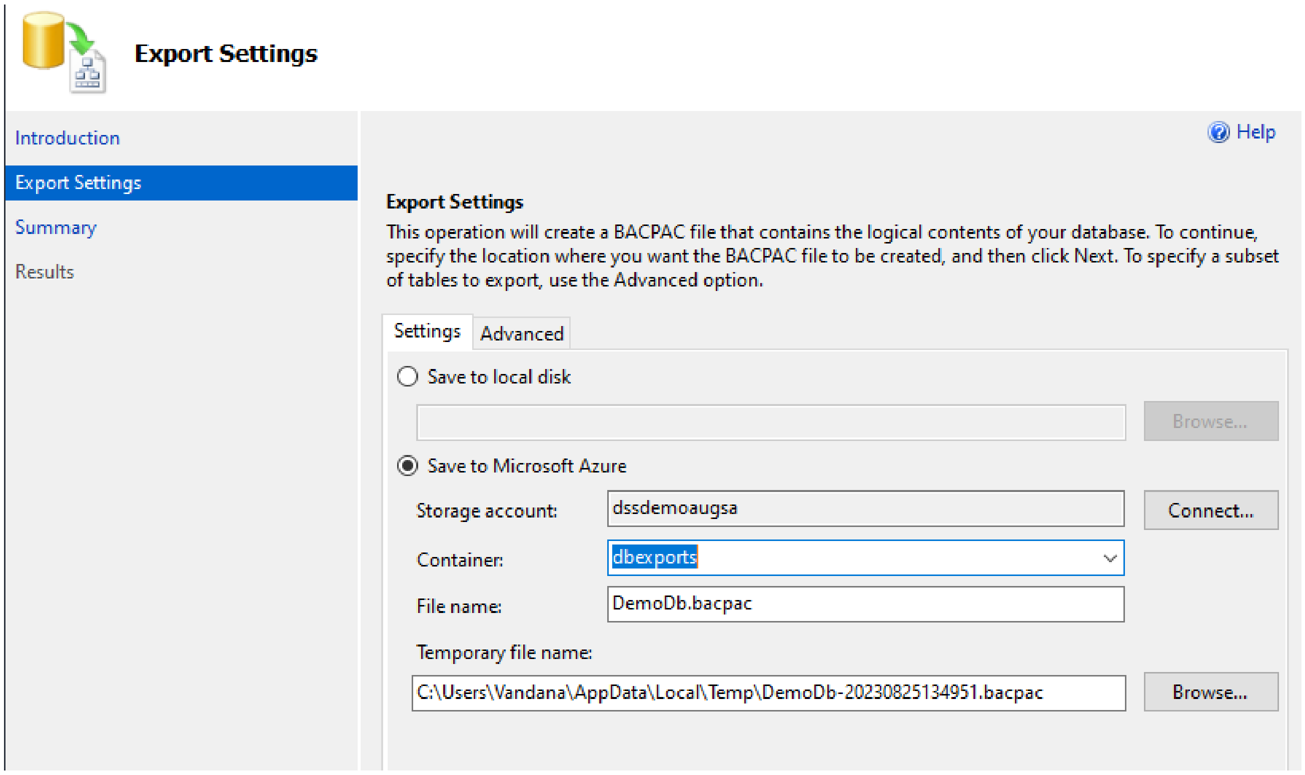
- Specify export settings.
- Review the export summary.
- Check the results of operation.

To begin exporting your database, click next.

☐ Do not show this page again.

→Next→

2. Save bacpac file to Storage Account(You need to sign in to your account) and configure


Export Settings

Introduction
Export Settings
Summary
Results

Help

Export Settings
This operation will create a BACPAC file that contains the logical contents of your database. To continue, specify the location where you want the BACPAC file to be created, and then click Next. To specify a subset of tables to export, use the Advanced option.

Settings | Advanced

○ Save to local disk

● Save to Microsoft Azure

Storage account: dssdemoaugsa

Container: dbexports

File name: DemoDb.bacpac

Temporary file name: C:\Users\Vandana\AppData\Local\Temp\DemoDb-20230825134951.bacpac

→Next→Finish

Lab 10:Import a BACPAC file to create an Azure SQL database

1. Azure Portal → **SQL Servers** → In SQL Server blade → **Import database**

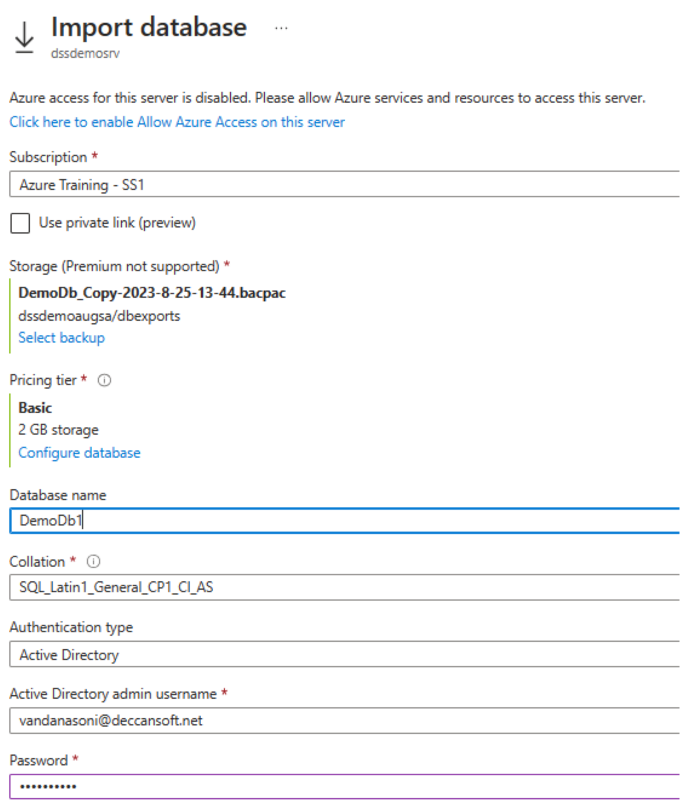
2. Click **Storage** and select your storage account, blob container, and .bacpac file and click **OK**

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.



Import database
dssdemosrv

Azure access for this server is disabled. Please allow Azure services and resources to access this server.
Click here to enable Allow Azure Access on this server

Subscription *
Azure Training - SS1

Use private link (preview)

Storage (Premium not supported) *
DemoDb_Copy-2023-8-25-13-44.bacpac
dssdemoaugsa/dbexports
Select backup

Pricing tier *
Basic
2 GB storage
Configure database

Database name
DemoDb1

Collation *
SQL_Latin1_General_CP1_CI_AS

Authentication type
Active Directory

Active Directory admin username *
vandanasoni@deccansoft.net

Password *
••••••••••

→OK