

Domain 2: Implement a secure environment for a database service

Configure database authentication and authorization

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/configure-database-authentication-authorization/1-introduction>

Introduction

Completed

Azure SQL offerings provide several authentication and authorization options that differ from those in SQL Server. This is because Azure SQL Database and Azure SQL Managed Instance rely on Microsoft Entra ID instead of Windows Server Active Directory.

You'll explore the best practices for granting permissions and the various permissions available within a database. It also delves into the concept of *least privilege*. While built-in roles in SQL Server and other database engines offer broad security privileges, many applications require more granular security on database objects.

Learning objectives

In this module you'll learn about:

- Explain authentication options for Azure SQL offerings
- Describe security principals
- Explain object permissions
- Identify authentication and authorization failures

2. Describe authentication and identities

Describe authentication and identities

Completed

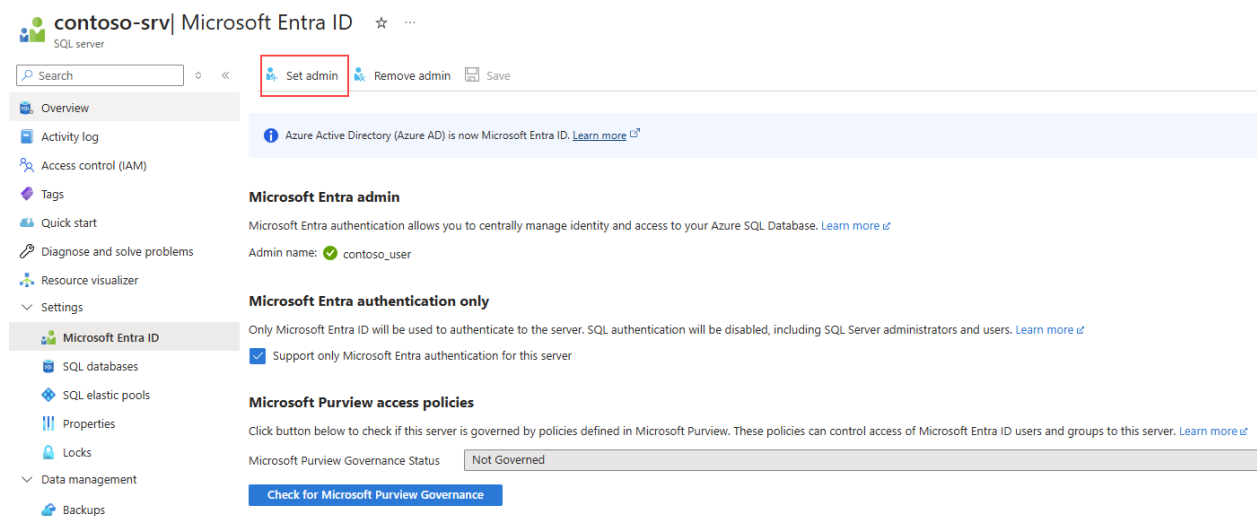
Both on-premises SQL Server and SQL Server in Azure VMs support SQL Server authentication and Windows Authentication. SQL Server authentication stores login details within SQL Server, while Windows Authentication uses Microsoft Entra ID (formerly Active Directory) accounts.

Microsoft Entra ID authentication is more secure and simplifies user management. If a user leaves, only the Microsoft Entra ID account needs to be locked.

Azure SQL Database also supports SQL Server authentication and Microsoft Entra authentication. Microsoft Entra authentication uses the same credentials for other resources like the Azure portal or Microsoft 365.

Microsoft Entra ID can sync with on-premises Active Directory, providing consistent credentials for both environments. It also supports multifactor authentication (MFA) for added security. MFA options include push notifications via the Microsoft Authenticator app, text messages, or access codes. Users with MFA must use the Universal Authentication with MFA option in SQL Server Management Studio.

You can set SQL admin permissions on an Azure SQL Database using the Azure portal.



It's a best practice to make this account a Microsoft Entra group, so access isn't dependent on a single login. The Microsoft Entra admin account grants special permissions and allows the account or group that holds that permission to have `sysadmin` like access to the server and all of the databases within the server. The admin account is only set using Azure Resource Manager and not at the database level. In order to change the account or group, you have to use the Azure portal, PowerShell, or Azure CLI.

Role-based access control (RBAC)

All Azure types of operations for Azure SQL Database are controlled through role-based access control (RBAC). RBAC is currently decoupled from Azure SQL security, but you can think of it as security rights outside of your database in SQL Database, with a scope that includes subscription, resource group, and resource. The rights apply to operations in the Azure portal, the Azure CLI, and Azure PowerShell. RBAC allows for separation of duties between deployment, management, and usage.

Built-in roles are available to reduce the need for higher-level RBAC roles, such as **Owner** or **Contributor**. Effectively, you can use these roles to have certain individuals deploy Azure SQL resources (or manage security policies) but grant other users actual access to use or manage the database. For example, a **SQL Server Contributor** could deploy a server and assign a user to be the admin of the server and databases. The built-in roles for Azure SQL Database include:

- **SQL DB Contributor**: Can create and manage databases but can't access the database (for example, can't connect and read data)
- **SQL Security Manager**: Can manage security policies for databases and instances (such as auditing) but can't access them
- **SQL Server Contributor**: Can manage servers and databases but can't access them.

3. Describe Security Principals

<https://learn.microsoft.com/en-us/training/modules/configure-database-authentication-authorization/4-describe-security-principals>

Describe Security Principals

Completed

Security principals are entities that can request SQL Server resources and to which you can (usually) grant permissions. There are several sets of security principals in SQL Server. Security principals exist at either the server level or the database level and can be either individuals or collections. Some sets have a membership controlled by the SQL Server administrators, and some have a fixed membership.

Note

New logins can be added by administrators on Azure SQL Database, but new server roles can't be created.

Schemas and securables

Before we look at the details of security principals, we need to understand the concepts of securables and schemas. SQL Server and Azure SQL Database have three scopes for securables. Securables are the resources within the database to which the authorization system manages access. For example, a table is a securable. To simplify access control, SQL Server contains securables in nested hierarchies called scopes. The three securable scopes are the server, the database, and the schema. A schema is a collection of objects within your database, which allows objects to be grouped into separate namespaces.

Every user has a default schema. If a user tries to access an object without specifying a schema name, as in: `SELECT name FROM customers`, it's assumed the object is in the user's default schema. If there's no such object in the default schema, SQL Server checks to see if the object is in the predefined `dbo` schema. If there's no object of the specified name in either the user's default schema, or in the `dbo` schema, the user receives an error message. It's a best practice to specify the schema name when accessing objects, so the previous select would be something like: `SELECT name FROM SalesSchema.customers`. If a user hasn't been given a default schema, their default schema is set to `dbo`.

By default, if no schema is specified when a user creates an object, SQL Server attempts to create it in the user's default schema. If the user hasn't been granted permission to create objects in their default schema, the object can't be created.

Logins and users

No matter the mode of authentication that is used, a login name used to access your SQL database is set up as a login within the instance. Those logins are set up at the instance level of SQL Server and stored in the master database. However, you can configure contained users, which are added at the database level. These users can be configured as SQL Server Authentication users and either Windows Authentication users or Microsoft Entra users (depending on which platform you're using). In order to create these users, the database must be configured for partial containment, which is configured by default in Azure SQL Database, and optionally configurable in SQL Server.

These users only have access to the database that the user is set up with. For the purposes of Azure SQL Database, it's a best practice to create users at the scope of the user database, and not in the master database.

```
CREATE USER [dba@contoso.com] FROM EXTERNAL PROVIDER;  
GO
```

The `CREATE USER` statement is executed in the context of the user database. In the example, the user is a Microsoft Entra user as indicated with the `FROM EXTERNAL PROVIDER` syntax.

If logins are created at the instance level in SQL Server, a user should then be created within the database, which maps the user to the server-based login as shown in the following example.

```
USE [master]
GO

CREATE LOGIN demo WITH PASSWORD = 'Pa55.w.rd'
GO

USE [WideWorldImporters]
GO

CREATE USER demo FROM LOGIN demo
GO
```

The login is first created in the master database, and then in the WideWorldImporters database a user is created to map to that login. Logins are used to access the SQL Server or the Azure SQL Database, but to do any work within a database, the login must be mapped to a username. The username is used for all authorization.

Logins and usernames are the most important security principals you need to be aware of, but the next sections describe some of the other concepts and terms when dealing with authorization.

Database roles

As you can imagine, database security can get complicated for applications with many users. In order to make it easier for both administrators and auditors, most database applications use role-based security. Roles are effectively security groups that share a common set of permissions. Combining permissions into a role allows a set of roles to be created for a given application. Some examples of roles would be administrators who had full access to all of the databases and servers, reporting users who only read the database, and an application account that had access to write data into the database. The roles can be defined when the application is designed, and then users can be assigned to those roles as they need access to the database. Role-based access control is a common architecture across computer systems and is how authorization is managed in Azure Resource Manager.

SQL Server and Azure SQL Database both include built-in roles that are defined by Microsoft, and also provide the option to create custom roles. Custom roles can be created at the server or database level. However, server roles can't be granted access to objects within a database directly. Server roles are only available in SQL Server and Azure SQL Managed Instance, not in Azure SQL Database.

Within a database, permissions can be granted to the users that exist within the database. If multiple users all need the same permissions, you can create a database role within the database and grant the needed permissions to this role. Users can be added as members of the database role. The member of the database role inherits the permissions of the database role.

```
CREATE USER [DP300User1] WITH PASSWORD = 'Pa55.w.rd'
GO

CREATE USER [DP300User2] WITH PASSWORD = 'Pa55.w.rd'
GO
```

```

CREATE ROLE [SalesReader]
GO

ALTER ROLE [SalesReader] ADD MEMBER [DP300User1]
GO

ALTER ROLE [SalesReader] ADD MEMBER [DP300User2]
GO

GRANT SELECT, EXECUTE ON SCHEMA::Sales TO [SalesReader]
GO

```

In the above example, you can see that two users are created, and then a role called SalesReader is created. The two new users are added to the newly created role, and then finally the role is granted `SELECT` and `EXECUTE` permissions on the *Sales* schema. Any user who is in that role can select from any object in the *Sales* schema, and execute any stored procedure in the schema.

Application roles

Application roles can also be created within a SQL Server database or Azure SQL Database. Unlike database roles, users aren't made members of an application role. An application role is activated by the user, by supplying the preconfigured password for the application role. Once the role is activated the permissions that are applied to the application role are applied to the user until that role is deactivated.

Built-in database roles

Microsoft SQL Server contains several fixed database roles within each database for which the permissions are predefined. Users can be added as members of one or more roles. These roles give their members a predefined set of permissions. These roles work the same within Azure SQL Database and SQL Server.

Database role	Definition
db_accessadmin	Allows users to create other users within the database. This role doesn't grant access to the schema of any of the tables, nor does it grant access to the data within the database.
db_backupoperator	Allows users to back up a database in a SQL Server or SQL Managed Instance. The role <i>db_backupoperator</i> doesn't confer any permissions in an Azure SQL Database.
db_datareader	Allows users to read from every table and view within the database.
db_datawriter	Allows users to <code>INSERT</code> , <code>UPDATE</code> , and <code>DELETE</code> data from every table and view within the database.
db_ddladmin	Allows users to create or modify objects within the database. Members of this role can change the definition of any object, of any type, but members of this role aren't

Database role	Definition
	granted access to read or write any data within the databases.
db_denydatareader	Users who need to be prevented from reading data from any object in the database, when those users have been granted rights through other roles or directly.
db_denydatawriter	Users who need to be prevented from writing data to any object in the database, when those users have been granted rights through other roles or directly.
db_securityadmin	Users who need to be able to grant access to other users within the database. Members of this role aren't granted access to the data within the database; however members of this role can grant themselves access to the tables within the database. Membership in this database role should be limited to only trusted users.
db_owner	Users who need administrative access to the database. Members of this role can perform any action within the database by default. However, unlike the actual database owner, who has the user name <i>dbo</i> , users in the db_owner role can be blocked from accessing data by placing them in other database roles, such as db_denydatareader , or by denying them access to objects. Membership in this database role should be limited to only trusted users.

All users within a database are automatically members of the **public** role. By default, this role has no permissions granted to it. Permissions can be granted to the public role, but you should consider carefully whether that is really something you want to do. Granting permissions to the public role would grant these permissions to any user, including the guest account, if the guest account was enabled.

The built-in database roles do meet the needs of many applications; however with applications that require more granular security (for example, when you only want to grant access to a specific subset of tables) a custom role is often a better choice.

Note

By default, users in roles like *db_owner* can always see all of the data in the database. Applications can take advantage of encryption options like **Always Encrypted** to protect sensitive data from privileged users.

For Azure SQL Database, there are a few nuances. You can have logins that exist in the virtual **master** database, database users, and even contained database users for Microsoft Entra accounts (recommended). Although the server admin for Azure SQL Database essentially has **sysadmin** rights, you can create more limited admins by using server or database level roles. Two database level roles are available for SQL Database that only exist in the virtual **master** database:

Database Role	Definition
loginmanager	Allows members to create logins for the database server. This role is the equivalent of the dbcreator fixed server role in an on-premises Microsoft SQL Server.

Database Role	Definition
dbmanager	Allows members to create and delete databases for the database server. This role is the equivalent of the securityadmin fixed server role in an on-premises Microsoft SQL Server.

Fixed server-level roles

In addition to database roles, SQL Server and Azure SQL Managed Instance both provide several fixed server-level roles. These roles assign permissions at the scope of the entire server. Server-level principals, which include SQL Server logins, Windows accounts, and Windows group can be added into fixed server-level roles. The permissions for fixed server-level roles are predefined, and no new server roles can be added.

Fixed server-level role	Definition
sysadmin	Allows its members to perform any activity on the server.
serveradmin	Allows its members to change server-wide configuration settings (for example Max Server Memory) and can shut down the server.
securityadmin	Allows its members to manage logins and their properties (for example, changing the password of a login). The members can also grant and revoke server and database level permissions. This role should be treated as being equivalent to the sysadmin role.
processadmin	Allows its members to kill processes running inside of SQL Server.
setupadmin	Allows its members to add and remove linked servers using T-SQL.
bulkadmin	Allows its members to run the BULK INSERT T-SQL statement.
diskadmin	Allows its members to have the ability to manage backup devices in SQL Server.
dbcreator	Allows its members to create, restore, alter, and drop any database.
public	Every SQL Server login belongs to the public user role. Unlike the other fixed server roles, permissions can be granted, denied, or revoked from the public role.

Here's a list of server-level roles in Azure SQL Database:

- **##MS_DatabaseConnector##**
- **##MS_DatabaseManager##**
- **##MS_DefinitionReader##**
- **##MS_LoginManager##**
- **##MS_SecurityDefinitionReader##**
- **##MS_ServerStateReader##**
- **##MS_ServerStateManager##**

4. Describe database and object permissions

<https://learn.microsoft.com/en-us/training/modules/configure-database-authentication-authorization/5-describe-database-object-permissions>

Describe database and object permissions

Completed

All Relational Database Management platforms have four basic permissions, which control data manipulation language (DML) operations. These permissions are `SELECT`, `INSERT`, `UPDATE`, and `DELETE`, and they apply to all SQL Server platforms. All of these permissions can be granted, revoked, or denied on tables and views. If a permission is granted using the `GRANT` statement, then the permission is given to the user or role referenced in the `GRANT` statement. Users can also be denied permissions using the `DENY` command. If a user is granted a permission and denied the same permission, the `DENY` always supersedes the grant, and the user will be denied access to the specific object.

```
GRANT SELECT ON dbo.Company to Demo
GO
DENY SELECT ON dbo.Company to Demo
GO
EXECUTE AS USER = 'Demo'
SELECT Name, Address FROM dbo.Company
```

Messages

Msg 229, Level 14, State 5, Line 17
The SELECT permission was denied on the object 'Company', database 'WideWorldImporters', schema 'dbo'.

Completion time: 2020-05-13T14:42:28.8361616-07:00

In the example, the user Demo is granted `SELECT` and then denied `SELECT` permissions on the `dbo.Company` table. When the user tries to execute a query that selects from the `dbo.Company` table, the user receives an error that `SELECT` permission was denied.

Table and view permissions

Tables and views represent the objects on which permissions can be granted within a database. Within those tables and views, you can additionally restrict the columns that are accessible to a given security principal (user or login). SQL Server and Azure SQL Database also include row-level security, which can be used to further restrict access.

Permission	Definition
SELECT	Allows the user to view the data within the object (table or view). When denied, the user is prevented from viewing the data within the object.
INSERT	Allows the user to insert data into the object. When denied, the user is prevented from inserting data into the object.
UPDATE	Allows the user the update data within the object. When denied, the user is prevented from updating data in the object.
DELETE	Allows the user to delete data within the object. When denied, the user is prevented from deleting data from the object.

Azure SQL Database and Microsoft SQL Server have other permissions, which can be granted, revoked, or denied as needed.

Permission	Definition
CONTROL	Grants all rights to the objects. It allows the user who has this permission to perform any action they wish against the object, including deleting the object.
REFERENCES	Grants the user the ability to view the foreign keys on the object.
TAKE OWNERSHIP	Allows the user the ability to take ownership of the object.
VIEW CHANGE TRACKING	Allows the user to view the change tracking setting for the object.
VIEW DEFINITION	Allows the user to view the definition of the object.

Function and stored procedure permissions

Like tables and views, functions and stored procedures have several permissions, which can be granted or denied.

Permission	Definition
ALTER	Grants the user the ability to change the definition of the object.
CONTROL	Grants the user all rights to the object.
EXECUTE	Grants the user the ability to execute the object.
VIEW CHANGE TRACKING	Allows the user to view the change tracking setting for the object.
VIEW DEFINITION	Allows the user to view the definition of the object.

EXECUTE AS

The `EXECUTE AS [user name]`, or `EXECUTE AS [login name]` (only available in SQL Server and Azure SQL Managed Instance) commands allow for the user context to be changed. As subsequent commands and statements are executed using the new context with the permissions granted to that context.

If a user has a permission and the user no longer needs to have that permission, permissions can be removed (either grants or denies) using the `REVOKE` command. The revoke command removes any `GRANT` or `DENY` permissions for the right specified to the user specified.

Ownership Chains

A concept called chaining applies to permissions, which allow users to inherit permissions from other objects. The most common example of chaining is a function or stored procedure that accesses a table during its execution. If the procedure has the same owner as the table, the stored procedure is able to be executed and access the table, even though the user doesn't have rights to access the table directly. This access is available because the user inherits the rights to access the table from the stored procedure, but only during the execution of the stored procedure, and only within the context of the stored procedures execution.

In the example, run as a database owner or server administrator, a new user is created and added as a member of a new *SalesReader* role, which is then granted permission to select from any object and execute any procedure in the Sales schema. A stored procedure is then created in the Sales schema that accesses a table in the Production schema.

The example then changes content to be the new user and an attempt is made to select directly from the table in the Production schema.

```
USE AdventureWorks2016;
GO

CREATE USER [DP300User1] WITH PASSWORD = 'Pa55.w.rd';
GO

CREATE ROLE [SalesReader];
GO

ALTER ROLE [SalesReader] ADD MEMBER [DP300User1];
GO

GRANT SELECT, EXECUTE ON SCHEMA::Sales TO [SalesReader];
GO

CREATE OR ALTER PROCEDURE Sales.DemoProc
AS
SELECT P.Name,
       SUM(SOD.LineTotal) AS TotalSales,
       SOH.OrderDate
FROM Production.Product P
     INNER JOIN Sales.SalesOrderDetail SOD ON (SOD.ProductID = P.ProductID)
     INNER JOIN Sales.SalesOrderHeader SOH ON (SOH.SalesOrderID = SOD.SalesOrderID)
```

```

GROUP BY P.Name,
        SOH.OrderDate
ORDER BY TotalSales DESC;

GO

EXECUTE AS USER = 'DP300User1';

SELECT P.Name,
        SUM(SOD.LineTotal) AS TotalSales,
        SOH.OrderDate
FROM Production.Product P
        INNER JOIN Sales.SalesOrderDetail SOD ON (SOD.ProductID = P.ProductID)
        INNER JOIN Sales.SalesOrderHeader SOH ON (SOH.SalesOrderID = SOD.SalesOrderID)
GROUP BY P.Name,
        SOH.OrderDate
ORDER BY TotalSales DESC;

```

The query results in an error that the user *DP300User1* doesn't have `SELECT` permission, because the role that the user belongs to doesn't have any privileges in the `Production` schema. Now we can try to execute the stored procedure:

```

EXECUTE AS USER = 'DP300User1';

EXECUTE Sales.DemoProc;

```

The *DP300User1* user has `EXECUTE` permission on the stored procedure in the `Sales` schema, because the user's role has `EXECUTE` permission on the `Sales` schema. Because the table has the same owner as the procedure, we have an unbroken ownership chain, and the execution will succeed and results are returned.

Permission changes don't apply when dynamic SQL is being used within stored procedures. The reason that dynamic SQL breaks the permission chain is because the dynamic SQL is executed outside of the context of the calling stored procedure. You can see this behavior by changing the stored procedure to execute using dynamic SQL as follows.

```

CREATE OR ALTER PROCEDURE Sales.DemoProc
AS
DECLARE @sqlstring NVARCHAR(MAX)

SET @sqlstring = '
SELECT P.Name,
        SUM(SOD.LineTotal) AS TotalSales,
        SOH.OrderDate
FROM Production.Product P
        INNER JOIN Sales.SalesOrderDetail SOD ON (SOD.ProductID = P.ProductID)
        INNER JOIN Sales.SalesOrderHeader SOH ON (SOH.SalesOrderID = SOD.SalesOrderID)
GROUP BY P.Name, SOH.OrderDate'

EXECUTE sp_executesql @sqlstring
GO

```

```
--
```

```
EXECUTE AS USER = 'DP300User1'
```

```
EXECUTE Sales.DemoProc
```

The *DP300User1* user receives an error that the user doesn't have `SELECT` permission on the *Production.Product* table, just like the user tried to execute the query directly. Permission chains don't apply and the user account that is executing the dynamic SQL must have rights to the tables and views that are being used by the code within the dynamic SQL.

Principle of least privilege

The principle of least privilege is fairly simple. The basic idea behind the concept is that users and applications should only be given the permissions needed in order for them to complete the task. Applications should only have permissions that they need to do in order to complete the task at hand.

As an example, if an application accesses all data through stored procedures, then the application should only have the permission to execute the stored procedures, with no access to the tables.

Dynamic SQL

Dynamic SQL is a concept where a query is built programmatically. Dynamic SQL allows T-SQL statements to be generated within a stored procedure or a query itself.

```
SELECT 'BACKUP DATABASE ' + name + ' TO DISK = ''\\backup\sql1\' + name + '.bak'''  
FROM sys.databases
```

The statement generates a list of T-SQL statements to back up all of the database on the server. Typically, this generated T-SQL is executed using `sp_executesql` or passed to another program to execute.

5. Identify authentication and authorization failures

<https://learn.microsoft.com/en-us/training/modules/configure-database-authentication-authorization/6-identify-authentication-and-authorization>

Identify authentication and authorization failures

Completed

A connection failure can result from reconfiguration, firewall settings, connection timeouts, or incorrect login information. Furthermore, if some Azure SQL Database or SQL Managed Instance resources are over capacity, you won't be able to connect.

Transient fault

When heavy workloads increase in the SQL Database service, the Azure infrastructure is able to dynamically reconfigure servers, and the client application may lose connection to the database during this operation.

Transient faults occur during database reconfiguration of a planned event or an unplanned event. These events are brief and shouldn't take longer than 60 seconds to complete.

Here's a list of a few transient errors that applications may receive when connecting to Azure SQL Database:

- Can't open database "%. *ls" requested by the login. The login failed.
- Can't process request. Not enough resources to process request.
- Can't process request. Too many operations in progress for subscription "%ld".

Note

For a complete list of transient errors, see [Troubleshooting connectivity issues and other errors with Azure SQL Database and Azure SQL Managed Instance](#).

How to monitor transient connectivity errors

Error	Action
Login failures	Look for any outages during the time when the application reported the errors at Microsoft Azure Service Dashboard.
Database reaches resource limits	Monitor your database's compute and storage resources carefully, and take action when it reaches its resource limits to prevent transient failures.
Extended authentication failures	File an Azure support request through the Azure portal if your application encounters connectivity error for longer than 60 seconds or if it occurs more than once in a given day.

Retry logic

Application developers should anticipate periodic transient failures when integrating with cloud services, like Azure SQL Database, and implement a retry logic instead of displaying application errors to users. It's important to set a maximum number of retries before the program terminates.

We recommend waiting for 5 seconds at a minimum on your first retry. Each subsequential retry should increase the delay exponentially, up to a maximum of 60 seconds.

Note

If a `SELECT` statement fails with a transient error in SQL Database or SQL Managed Instance, avoid directly retrying it. Instead, retry the `SELECT` statement using a new connection.

Unable to log in to the server

When the error **Login failed for user '< User name >'** happens, the service administrator can follow the following steps:

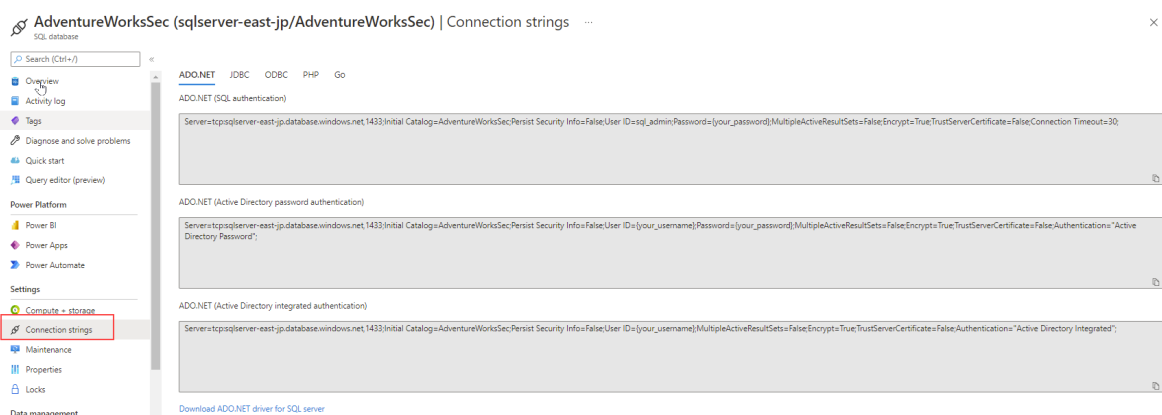
1. Check if the login is disabled by using the `sys.sql_logins` catalog view.
2. If the login is disabled, run `ALTER LOGIN <User name> ENABLE;` to enable it.
3. If the login doesn't exist, create it using the `CREATE LOGIN` statement.
4. Connect to the database you want to grant the user access to, and run the `CREATE USER` statement.
5. Either assign the user a role using the `ALTER ROLE` command, or grant the user access to one or more database objects using the `GRANT` command.

Connection string

When you receive connectivity errors, it's a good practice to make sure your connection string is working properly. This is mostly important when provisioning a new database, or after making infrastructure changes to a database service.

The Azure portal allows you to retrieve the connection string you need to interact with Azure SQL Database.

1. From the Azure portal, select **All services**, and then **SQL databases**. Filter and select your database.
2. On the blade for your database, select **Connection strings**.



3. Copy and edit the connection string by including your password, or replacing the server name as needed.
4. Reference the connection string updated in the client application.

To learn more about connectivity errors for Azure SQL Database and Azure SQL Managed Instance, see [Troubleshooting connectivity issues and other errors with Azure SQL Database and Azure SQL Managed Instance](#).

6. Exercise: Authorize Access to Azure SQL Database with Microsoft Entra ID

<https://learn.microsoft.com/en-us/training/modules/configure-database-authentication-authorization/7-exercise-authorize-access-sql-database-azure-active-directory>

Exercise: Authorize Access to Azure SQL Database with Microsoft Entra ID

Completed

Now it's your chance to configure database authentication and authorization.

In this exercise, you'll learn how to authorize access to Azure SQL Database with Microsoft Entra.

Note

To complete this exercise, you'll need an [Azure subscription](#).

Launch the exercise and follow the instructions.

[Launch Exercise](#)

7. Module assessment

<https://learn.microsoft.com/en-us/training/modules/configure-database-authentication-authorization/8-knowledge-check>

Module assessment

Completed

8. Summary

<https://learn.microsoft.com/en-us/training/modules/configure-database-authentication-authorization/9-summary>

Summary

Completed

Azure SQL offerings provide several authentication and authorization options that differ from those in SQL Server. This is because Azure SQL Database and Azure SQL Managed Instance rely on Microsoft Entra ID instead of Windows Server Active Directory.

This module explores best practices for granting permissions and the various permissions available within a database. It also delves into the concept of *least privilege*. While built-in roles in SQL Server and other database engines offer broad security privileges, many applications require more granular security on database objects.

Now that you've reviewed this module, you should be able to:

- Explain authentication options for Azure SQL offerings
- Describe security principals
- Explain object permissions
- Identify authentication and authorization failures

Protect data in-transit and at rest

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/protect-data-transit-rest/1-introduction>

Introduction

Completed

Explore the encryption options available within Microsoft SQL Server, Azure SQL Database, and Azure SQL Managed Instance. Each of the various platforms supports different database encryption options. Students explore these data encryption options and how to configure them.

Consider the following three scenarios when evaluating data encryption methods.

Scenario	Definition
Data at rest	Encrypting it while it's on file storage.
Data in transit	Encrypting it while it travels through private or public network communication channels.
Data in use	Encrypting it while it's in RAM or CPU caches.

2. Explore Transparent Data Encryption

<https://learn.microsoft.com/en-us/training/modules/protect-data-transit-rest/2-explore-transparent-data-encryption>

Explore Transparent Data Encryption

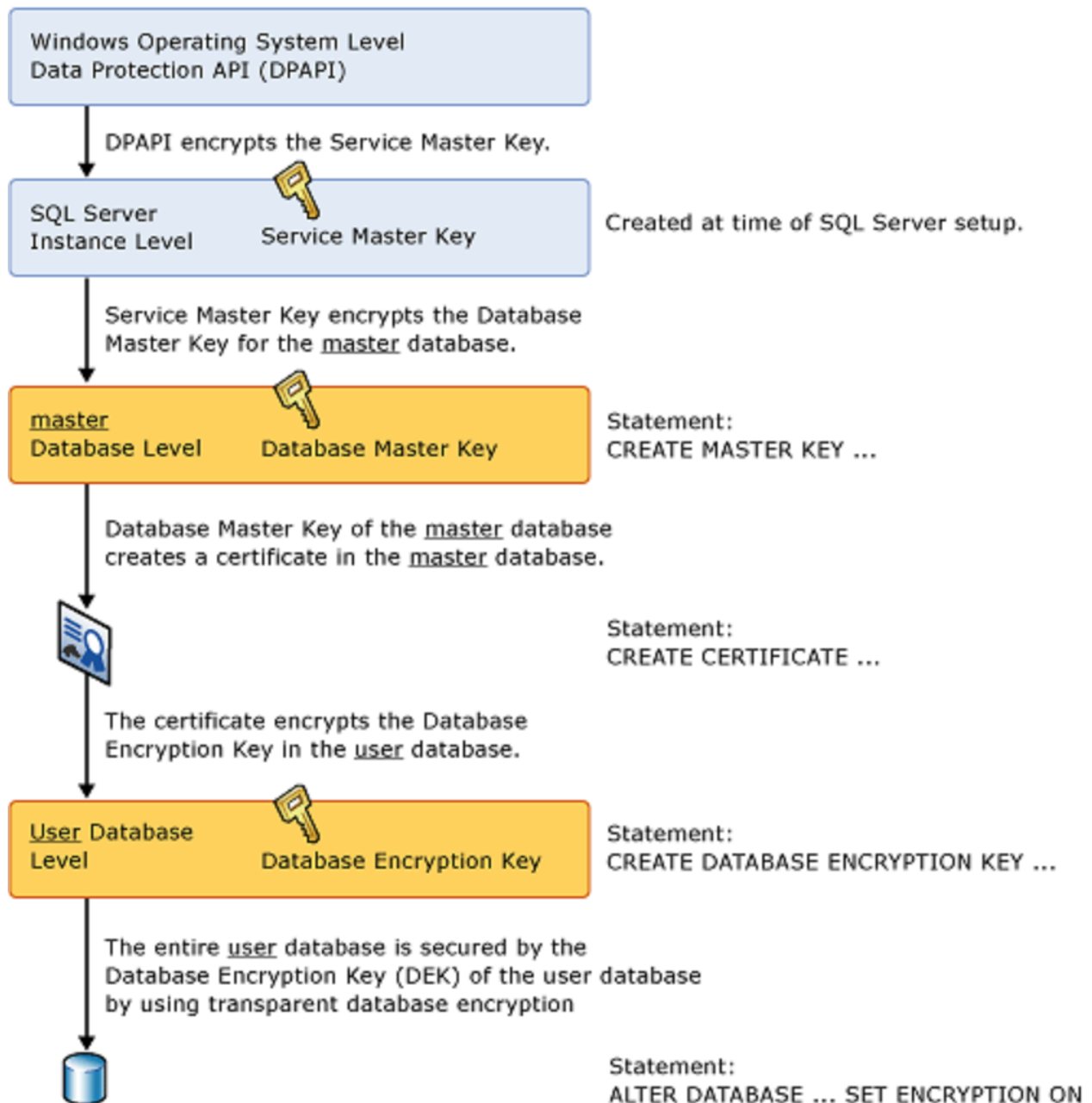
Completed

Microsoft SQL Server's Transparent Data Encryption (TDE) encrypts all data within a target database at the page level. Data is encrypted as it's written to the data page on disk and decrypted when read into memory, resulting in all data pages on disk being encrypted.

TDE doesn't encrypt data at the table or column level. Anyone with the appropriate permissions can read, copy, and share the data. Encryption at rest protects against restoring a backup to an unsecured server or copying database and transaction log files to another unsecured server. No decryption occurs during the backup operation.

TDE protects data at rest and complies with various industry laws, regulations, and guidelines. It allows software developers to encrypt data using AES and 3DES encryption algorithms without changing existing applications.

Transparent Database Encryption Architecture



Databases created in Azure SQL Database after May 2017 have TDE enabled automatically. Databases created before May 2017 need TDE to be manually enabled. For Azure SQL Managed Instance, TDE is

enabled by default for databases created after February 2019. Databases created before February 2019 need TDE to be manually enabled.

To enable TDE in an Azure SQL Database, edit the database in the Azure portal. From the **Transparent data encryption** pane, select to enable data encryption.

 Save  Discard  Feedback

Transparent data encryption encrypts your databases, backups, and logs at rest without any changes to your application. To enable encryption, go to each database. [Learn more](#)

Transparent data encryption ⓘ

☒ Service-managed key

☐ Customer-managed key

By default, databases in Azure SQL Database are encrypted using a Microsoft-provided certificate (service-managed key). Azure also offers a Bring Your Own Key (BYOK) option, allowing you to use a customer-managed key created by your company and uploaded to Azure Key Vault. If the customer-managed key is removed from Azure, database connections are closed, and access to the database are denied.

Enabling TDE within a Microsoft SQL Server database is an easy process as only a few T-SQL commands are required. This process involves the following steps:

1. Set a master key within the master database using the `CREATE MASTER KEY ENCRYPTION` command.
2. Create a certificate in the master database using the `CREATE CERTIFICATE` command.
3. Create a database encryption key within the database using the `CREATE DATABASE ENCRYPTION KEY` command.
4. Enable the encryption key using the `ALTER DATABASE` command.

```
USE master;
GO

CREATE MASTER KEY ENCRYPTION BY PASSWORD = '<your-pwd>';
GO

CREATE CERTIFICATE MyServerCert
    WITH SUBJECT = 'TDEDemo_Certificate';
GO

USE [TDE_Demo];
GO

CREATE DATABASE ENCRYPTION KEY
    WITH ALGORITHM = AES_256 ENCRYPTION BY SERVER CERTIFICATE MyServerCert;
GO

ALTER DATABASE TDE_Demo SET ENCRYPTION ON;
GO
```

Once TDE is enabled, it takes time to encrypt the database as each page must be read, encrypted, and written back to disk. The larger the database, the longer this process takes. This background process runs at a low priority to avoid overloading the system's I/O or CPU.

The certificate used by TDE must be manually backed up and stored securely. SQL Server integrates with Enterprise Key Managers (EKMs) like Azure Key Vault to manage encryption keys. Managing the certificate is crucial because if it's lost and the database needs to be restored from a backup, the restore fails as the database can't be read.

Note

To use TDE with databases in an Always On Availability Group, the certificate used to encrypt the database must be backed up and restored to the other servers within the Availability Group that will be hosting copies of the database.

Customer-managed keys

You can alternately use BYOK and take advantage of an Azure key vault. The advantages of using customer-managed keys are:

- Full and granular control over usage and management of the TDE protector
- Transparency of the TDE protector usage
- Ability to implement separation of duties in the management of keys and data within the organization
- The key vault administrator can revoke key access permissions to make encrypted database inaccessible
- Central management of keys in AKV
- Greater trust from your end customers because AKV is designed so that Microsoft can't see or extract encryption keys

You can also take advantage of using a [user-assigned managed identity \(UMI\)](#) with customer-managed keys for TDE, which:

- Enables the ability to preauthorize key vault access for Azure SQL logical servers by creating a user-assigned managed identity and granting it access to key vault, even before the server or database are created.
- Allows creation of an Azure SQL logical server with TDE and CMK enabled.
- Enables the same user-assigned managed identity to be assigned to multiple servers, eliminating the need to individually turn on system-assigned managed identity for each Azure SQL logical server and providing it access to key vault.
- Provides the capability to enforce CMK at server creation time with an available built-in Azure policy.

[Automatic key rotation](#) are introduced for customer-managed keys using TDE. When enabled, the server continuously checks the key vault for any new versions of the key being used as the TDE

protector. If a new version of the key is detected, the TDE protector on the server is automatically rotated to the latest key version within 60 minutes.

Azure disk encryption

In addition to these SQL Server security features, Azure VMs include an extra layer of security, Azure Disk Encryption—a feature that helps protect and safeguard data and meet organization and compliance commitments. If you're using TDE, your data is protected by multiple layers of encryption with Azure Disk Encryption.

3. Configure server and database firewall rules

<https://learn.microsoft.com/en-us/training/modules/protect-data-transit-rest/3-configure-server-and-database-firewall>

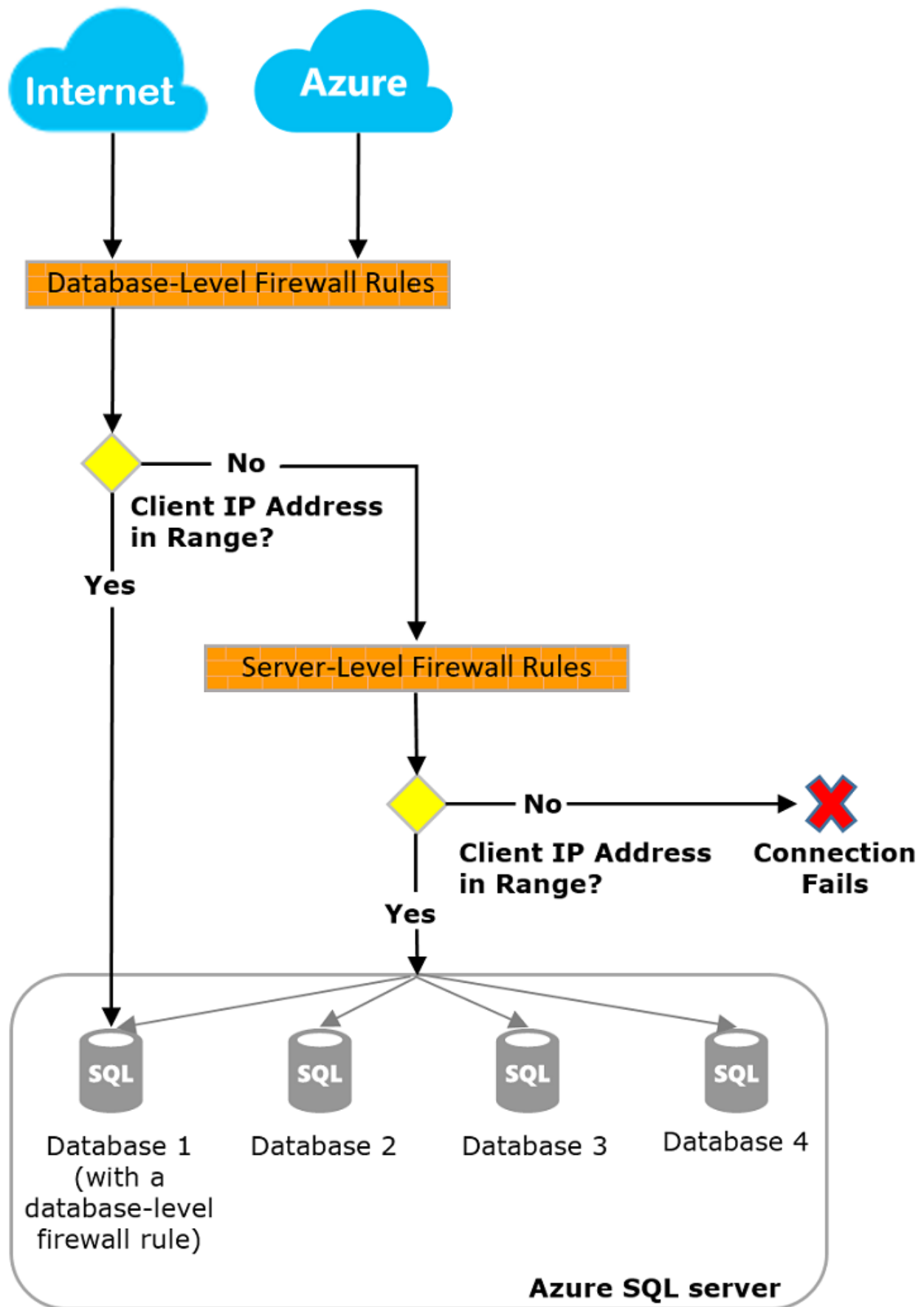
Configure server and database firewall rules

Completed

Firewalls prevent unauthorized users from accessing protected resources. Each Azure SQL Database maps to a public IP address hosted by Microsoft. In each Azure region, one or more public IP addresses allow you to reach your database gateway, which then directs you to your database.

How firewall works

As shown in the following diagram, connection attempts coming from the internet and Azure must go through the firewall before they reach your server or database.



Azure provides built-in firewalls to limit access in order to protect your database and your data. In Azure SQL Database there are two distinct sets of firewall rules: server-level firewall rules and database-level firewall rules.

Server-level firewall rules

Both server and database level firewalls use IP Address rules instead of SQL Server Logins, and allow all users at the same public IP Address to access the SQL Server. For most companies, this is their outbound IP address.

Server-level firewalls are configured to allow users to connect to all databases on the server. Database level firewalls are used to grant or block specific IP Addresses from accessing specific databases.

Server level firewall rules can be configured using the Azure portal or using the `sp_set_firewall_rule` stored procedure from within the *master* database.

Note

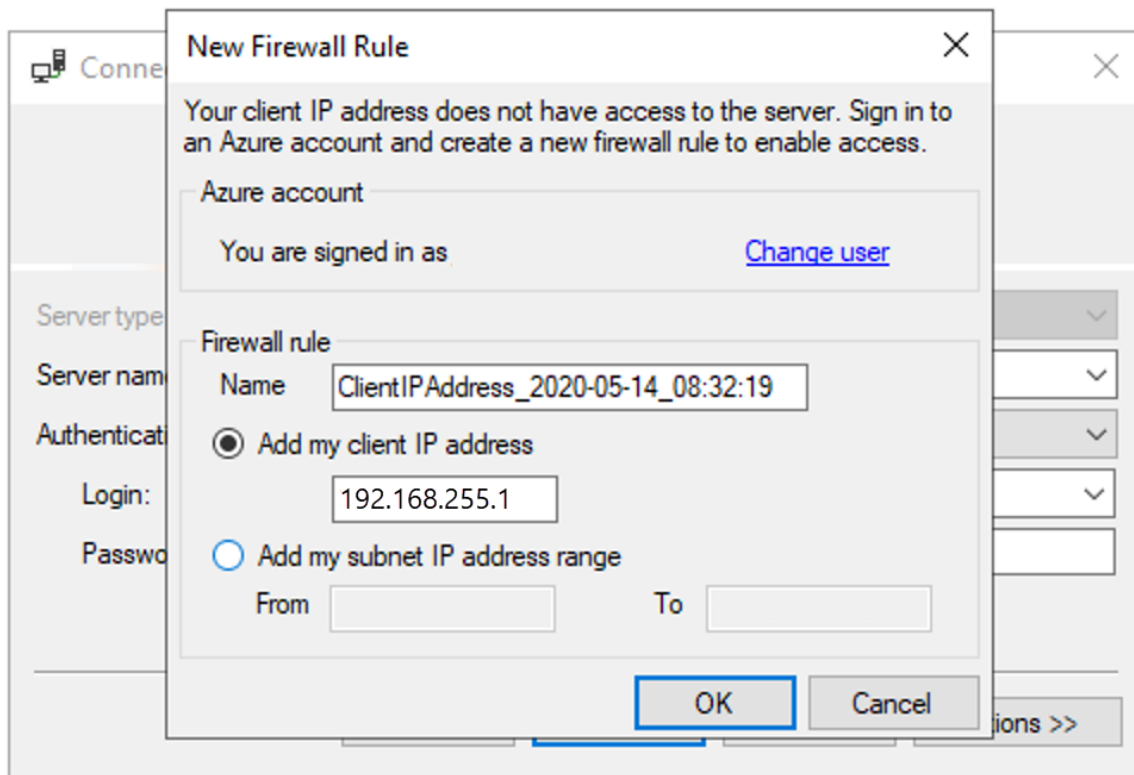
The **Allow Azure Services and resources to access this server** server setting counts as a single firewall rule when enabled.

Database-level firewall rules

Database-level firewall rules are configured through T-SQL only using the `sp_set_database_firewall_rule` stored procedure from within the user database.

Upon connection, Azure SQL Database first checks for a database-level firewall rule corresponding to the database name specified in the connection string. If no such rule exists, the firewall then checks the server-level IP firewall rules. Server-level IP firewall rules apply to all databases on the server. If a matching rule is found at either level, the connection is established.

If neither exist and the user is connecting through SQL Server Management Studio, they'll be prompted to create a firewall rule.



Virtual network endpoints

Virtual network endpoints allow traffic from a specific Azure Virtual Network. These rules apply at the server level, not just the database level.

Additionally, the service endpoint applies to only one region, which is the underlying endpoint's region.

Another important consideration is that the virtual network connecting to the Azure SQL Database must have outbound access to the database's public IP address. This can be configured using service tags for Azure SQL Database.

Private link

The Private Link feature allows you to connect to Azure SQL Database and other PaaS offerings using a private endpoint.

A private endpoint allows for a connection to your Azure SQL Database to go entirely over the Azure backbone network and not over the public internet.

This feature provides a private IP address on your Virtual Network. Another feature of private link is that it allows for Azure Express Route connections through that circuit.

Private link offers several benefits including cross-region private connectivity and protection against data leakage by only allowing connections to specific resources.

4. Explain object encryption and secure enclaves

<https://learn.microsoft.com/en-us/training/modules/protect-data-transit-rest/4-explain-object-encryption-secure-enclaves>

Explain object encryption and secure enclaves

Completed

In addition to supporting encryption at rest, SQL Server supports encrypting data within columns using Always Encrypted. Once data is encrypted, the application accessing the database must have the correct certificate in order to view the plain text values of the data.

Always Encrypted

Always Encrypted allows for the encryption of data within the client application, protecting sensitive data from malware and high-privileged users, such as Database Administrators (DBAs), server admins, cloud admins, or those who manage the data but should have no access. This encryption happens automatically based on the settings within the Microsoft SQL Server database, which tell the application what the encryption settings on the database column are.

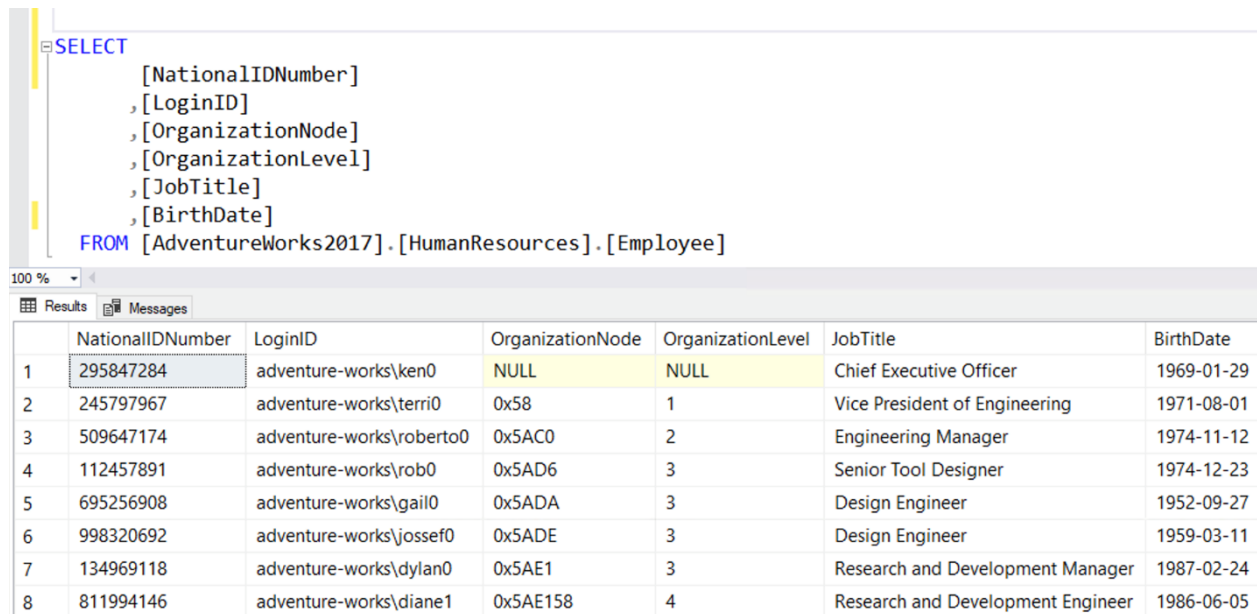
The following table provides some scenarios for Always Encrypted usage:

Scenario	Definition
Client and data on-premises	For scenarios where you need to protect your on-premises database from high-privileged users, for example, external vendors managing SQL Server.
Client on-premises with data in Azure	In this scenario, to ensure Microsoft cloud administrators have no access to the data, Always Encrypted keys are stored in key store hosted on-premises, for SQL Database or SQL Server running in a virtual machine on Microsoft Azure.
Client and Data in Azure	In this scenario, the environment is fully hosted on Azure. While Always Encrypted doesn't completely isolate data from cloud administrators, the customer still benefits from the fact that the data is encrypted in the database.

Always Encrypted is based on a master encryption key and a column encryption key. Having both keys allows each column to be encrypted using a different encryption key for maximum data

protection. Always Encrypted has various key stores that can store the certificate used by encryption.

Here's an example of enabling Always Encrypted. You can see that *NationalIDNumber* and *BirthDate* columns are both in plain text.

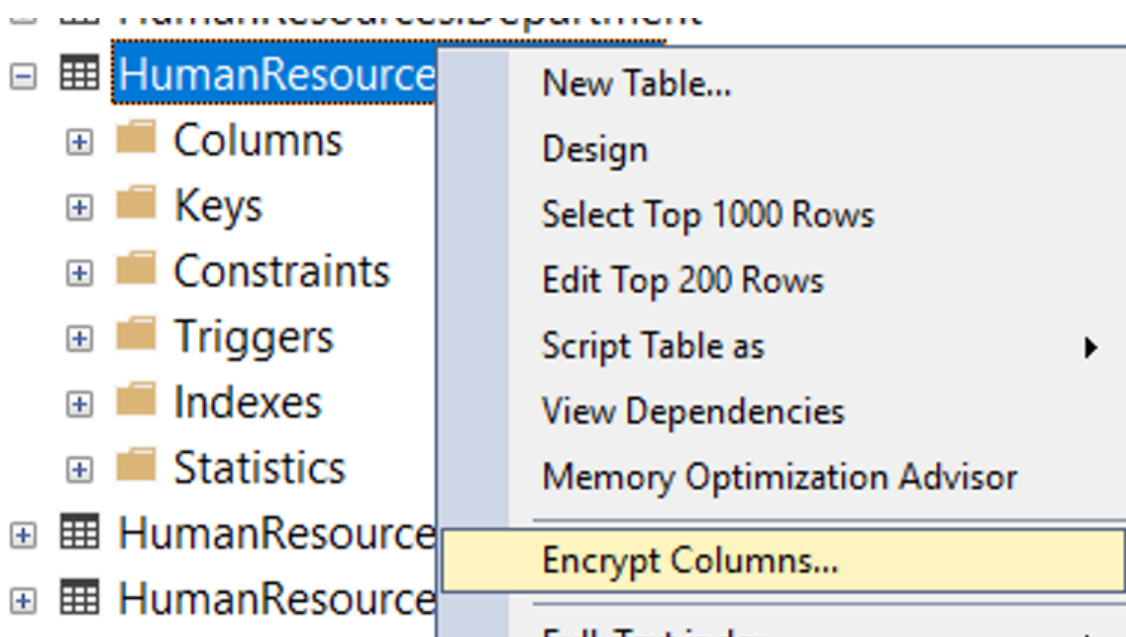


The screenshot shows a SQL query in the query editor and its results in the Results pane. The query selects columns from the Employee table in the HumanResources schema of the AdventureWorks2017 database. The results pane shows 8 rows of data. The columns are NationalIDNumber, LoginID, OrganizationNode, OrganizationLevel, JobTitle, and BirthDate. The first row shows a NationalIDNumber of 295847284 and a BirthDate of 1969-01-29.

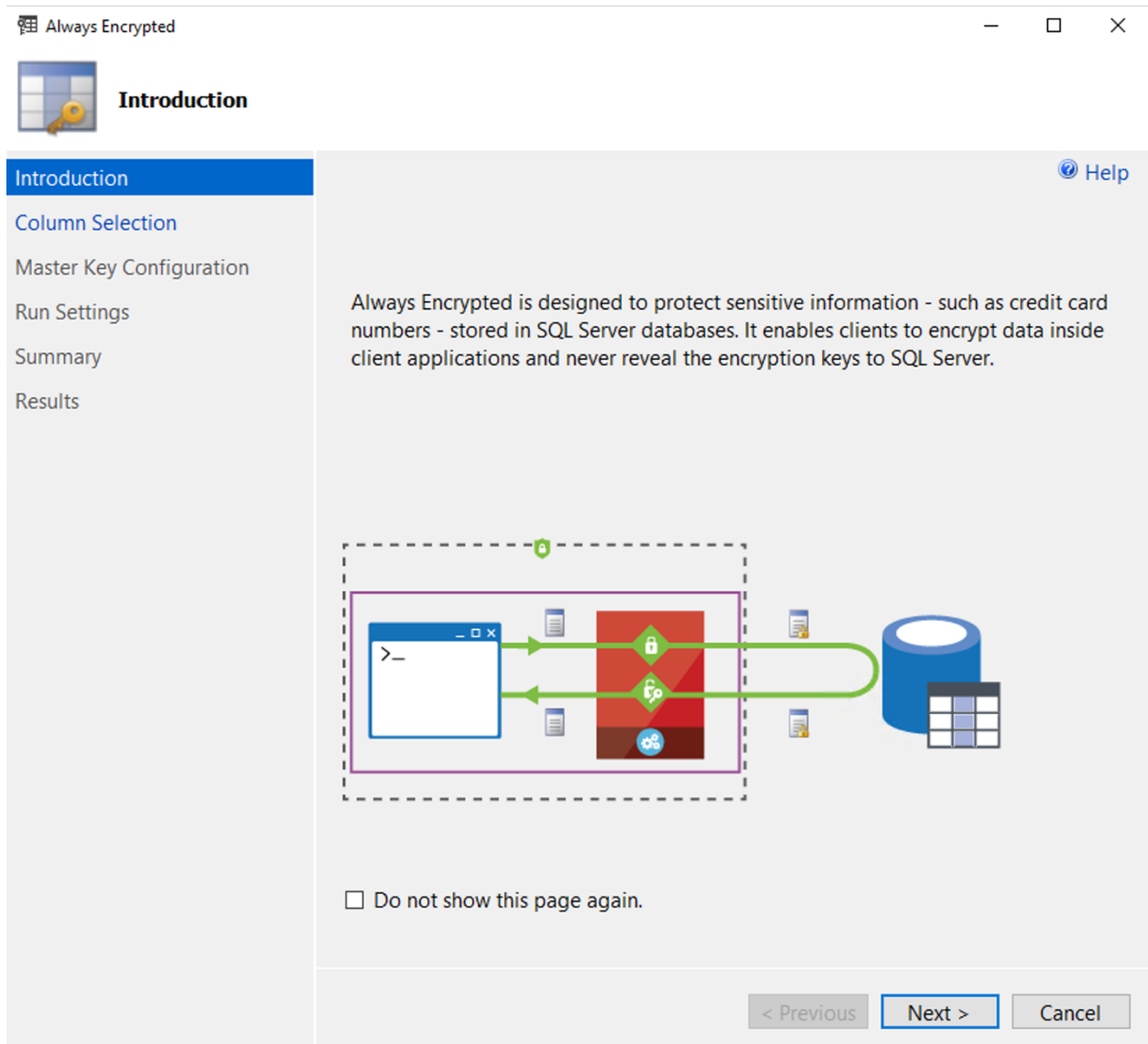
```
SELECT
    [NationalIDNumber]
  , [LoginID]
  , [OrganizationNode]
  , [OrganizationLevel]
  , [JobTitle]
  , [BirthDate]
FROM [AdventureWorks2017].[HumanResources].[Employee]
```

	NationalIDNumber	LoginID	OrganizationNode	OrganizationLevel	JobTitle	BirthDate
1	295847284	adventure-works\ken0	NULL	NULL	Chief Executive Officer	1969-01-29
2	245797967	adventure-works\terri0	0x58	1	Vice President of Engineering	1971-08-01
3	509647174	adventure-works\roberto0	0x5AC0	2	Engineering Manager	1974-11-12
4	112457891	adventure-works\rob0	0x5AD6	3	Senior Tool Designer	1974-12-23
5	695256908	adventure-works\gail0	0x5ADA	3	Design Engineer	1952-09-27
6	998320692	adventure-works\jossef0	0x5ADE	3	Design Engineer	1959-03-11
7	134969118	adventure-works\dylan0	0x5AE1	3	Research and Development Manager	1987-02-24
8	811994146	adventure-works\diane1	0x5AE158	4	Research and Development Engineer	1986-06-05

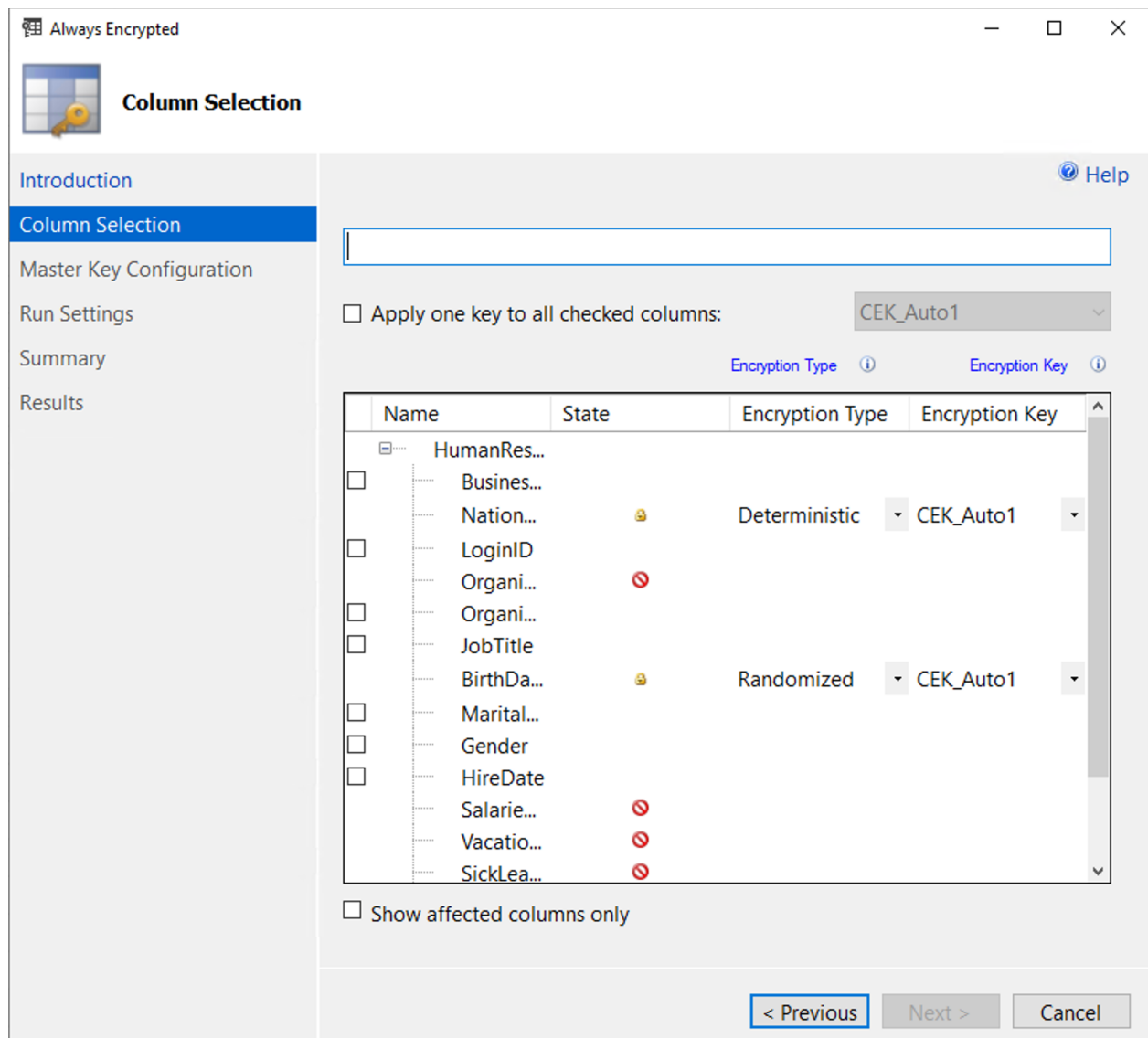
The next few images show you how we can encrypt both of these columns using Always Encrypted. The encryption could be done using T-SQL, but in this example, we use the wizard from SQL Server Management Studio. You can reach the wizard by right-clicking on the table name in Object Explorer as shown below.



When you select **Encrypt Columns...**, the wizard launches.



Select **Next** to choose the columns you want to encrypt.



There are two different types of encryption specified. The *NationalIDNumber* column is encrypted with **Deterministic** encryption, and the *BirthDate* column is encrypted using **Randomized** encryption.

Randomized encryption is more secure than deterministic encryption but comes with limitations. Once a column is created, you can't change its encryption type. It's recommended to use randomized encryption for columns with a few well-known distinct values that could be guessed by someone with access to the encrypted data, such as a three-digit credit card verification code.

Always Encrypted with randomized encryption is limited because the same value is encrypted differently each time. This means you can only return these columns in your results. In contrast, deterministic encryption always encodes a value the same way, allowing for comparisons using equality and inequality operators, as well as joins, grouping, and indexing.

Also, the wizard generates a column encryption key, which performs the data encryption. Each encrypted column can have its own key, or you can use the same key for multiple columns.

After identifying the columns you're encrypting, you can select **Next** and you'll see the **Master Key Configuration** screen:

Always Encrypted

Master Key Configuration

Introduction
Column Selection
Master Key Configuration
Run Settings
Summary
Results

Help

To generate a new column encryption key, a column master key must be selected to protect it. The column master key is stored outside of the database.

Select column master key:

Auto generate column master key

Select the key store provider

☐ Windows certificate store

☒ Azure Key Vault

You are signed in as user@contoso.com [Change user](#)

Select a subscription to use:

Contoso Ltd

Select an Azure Key Vault:

< Previous Next > Cancel

In this screen, you create the column master key, which is used to encrypt the column encryption keys. You can supply your own key, if you're using T-SQL to encrypt the columns. This key must be stored in a key store such as the Windows Certificate Store, Azure Key Vault, or a hardware security module. The database engine never stores the column master key, and only contains the metadata about where it's stored. Not storing the master key protects data access from users who have full access to the database.

For the highest level of security, the key should be stored within a third-party key store such as Azure Key Vault. Never generate the keys on the server hosting your database, as the key could potentially be extracted from memory on that server.

In the example, the key is being stored in Azure Key Vault. On the next screen, the wizard will provide you with the option to either finish the encryption process now, or to generate a PowerShell script. Once you complete the process, the data appears as encrypted to anyone querying the data without the key.

select * from HumanResources.Employee

Results	Messages	BusinessEntityID	NationalIDNumber	LoginID	OrganizationNode	OrganizationLevel	JobTitle	BirthDate
1		1	0x01C3979CCA373497E69A372F30A091E096D1A8C60C5738E...	adventure-works\ken0	NULL	NULL	Chief Executive Officer	0x01C395FEDE5C8965A81D512F3C5EA6E6D97AF13840AA6CF...
2		2	0x01B8850F9C9599EAE1B0653F01FCE7321671F6F0FA0822D3...	adventure-works\terri0	0x58	1	Vice President of Engineering	0x014DE4461EA605D07B955A82832523A57206970670C7D...
3		3	0x01BDD49E6DF84DFB95F5C42EC511A86A6C5FCA4588C63F...	adventure-works\roberto0	0x5AC0	2	Engineering Manager	0x01CE6DAD98D6B28AF76EA565565615A662A025F161858CD...
4		4	0x01E8634C71B297A23C9AC247F682C28E00976CFA5265F836...	adventure-works\robo0	0x5AD6	3	Senior Tool Designer	0x0117B23F362D932E0380FAF28A9EBF7B59C68AA21840A8725...
5		5	0x01B49F37A7628AB9093EC44AA071FFAF3193B2F820A5B5C5...	adventure-works\gail0	0x5ADA	3	Design Engineer	0x01DFA48EB32386DA7F39C8BEA1CF34CCF71A2F38F3185519...
6		6	0x01D2760AEC79ED9659F70DD3AC8E26232D64584D3B2E1F...	adventure-works\josse0	0x5ADE	3	Design Engineer	0x0186FF5BAD106F4ACDFF47194FB0484615A7D812959B5D492...
7		7	0x019FD96C3C09F255FAA6F789D105CA8D41E49D3B972E0E3...	adventure-works\dylan0	0x5AE1	3	Research and Development Manager	0x0134262E465846A90A7B2E134A22092AC1A582829D3A19F38...
8		8	0x019280287AAAD3386A78AA9071FD9F8395C68B2538320F03...	adventure-works\diane1	0x5AE158	4	Research and Development Engineer	0x0153027957F29425CF8949093C8A087428128C871E018C6C1...
9		9	0x01B2A71B39816D3BE16023B774F92B8C72060A69E9E00E1E4...	adventure-works\gigi0	0x5AE168	4	Research and Development Engineer	0x01F458C1E6E5C0D493F13AD2617844C2BEDAE057E074E5713...
10		10	0x01975B145982F73ED3E8A4253D805CF0AA48DA3613083CC4...	adventure-works\michael6	0x5AE178	4	Research and Development Manager	0x01FA2B454898604F48B6F2985B4D9D022305E38CF4FA6B4C2...
11		11	0x01553EF331044852447480DA345E586285ACDF7FED163E25F...	adventure-works\ovidio0	0x5AE3	3	Senior Tool Designer	0x01EC391F0168F8B338026F380643B3CED587AA0CA23F65BA...
12		12	0x01041122F34858C6171448A3FBEEF2BA97EC148B8CE3E119E...	adventure-works\thierry0	0x5AE358	4	Tool Designer	0x01D946A0CF88619E8C2128B8C9ED95DDE8B7758DFD0467...
13		13	0x01C53D73AC0DFCE594D1CF72DE9837A0F11A9A317405862...	adventure-works\janice0	0x5AE368	4	Tool Designer	0x018D56101A359B7680CCD9E00FD13013C9473746CFD98C7...
14		14	0x015EADCC6AC88F6C12844545074303726F10A426DD95C90B...	adventure-works\michael8	0x5AE5	3	Senior Design Engineer	0x0101929CAC7B3F4A8A7F3FE1C729C034EBF582D5D357DF11...
15		15	0x0120328DDE84AC518B081396151CE1191F473EE04DF36AF1...	adventure-works\sharon0	0x5AE7	3	Design Engineer	0x018761024DFE8DE4CABDF0E0449A2707B0413E2786287...

In order to decrypt data from an Always Encrypted column, your application needs an Always Encrypted driver to connect to the database, followed by the following actions:

1. The application has access to the key store where the Always Encrypted keys are stored
2. The application then retrieves the data
3. Data that is written back to the database is encrypted at the client through the driver

In addition to the driver, the application's connection string needs to have the setting **Column Encryption Setting=enabled** provided. This setting causes a metadata lookup to be made for each column that is used by the application.

Note

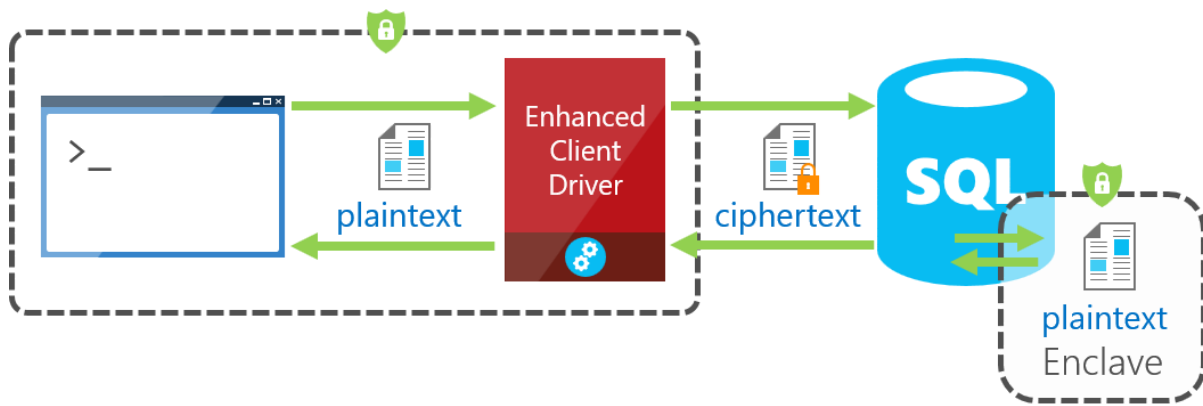
To minimize metadata lookups, the application needs to update the *SqlCommandColumnEncryptionSetting* on the *SqlConnection* objects within the .NET application. These settings must be set for each database query that the application submits.

Secure enclaves

Always Encrypted supports a feature called secure enclaves, which allows more robust querying of encrypted data.

A secure enclave is a secured region of memory within the SQL Server process that acts as a trusted execution environment for processing encrypted data. This enclave appears as a black box to SQL Server, and it isn't possible to view any data or code, even with a debugger.

The image shows the architecture of this process:



Always Encrypted with secure enclaves also addresses some of the limitations of Randomized encryption, which enables pattern matching, comparison operations, and indexing on columns using this encryption type.

5. Enable encrypted connections

<https://learn.microsoft.com/en-us/training/modules/protect-data-transit-rest/5-enable-encrypted-connections>

Enable encrypted connections

Completed

It's possible to encrypt data that is transmitted across a network between an instance of SQL Server and a client application with Transport Layer Security (TLS). The TLS encryption is performed within the protocol layer and is available to all supported SQL Server and Azure SQL database services.

Certificates

You must run SQL Server Configuration Manager with a local administrator account in order to install certificates for use by SQL Server.

Furthermore, the certificate must satisfy the following conditions:

- The certificate must be located in the local computer certificate store or the current user certificate store.
- The SQL Server service account must have permission to access the certificate.
- The certificate must be within a valid period.

Note

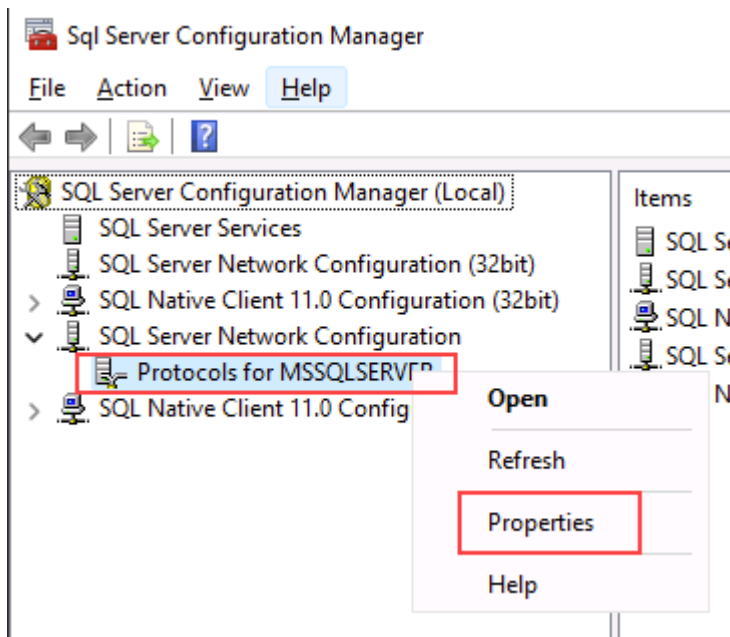
If the correct access isn't provided, restarting SQL Server service fails.

For a complete list of requirements when installing a TLS certificate, see [Enable encrypted connections to the Database Engine](#).

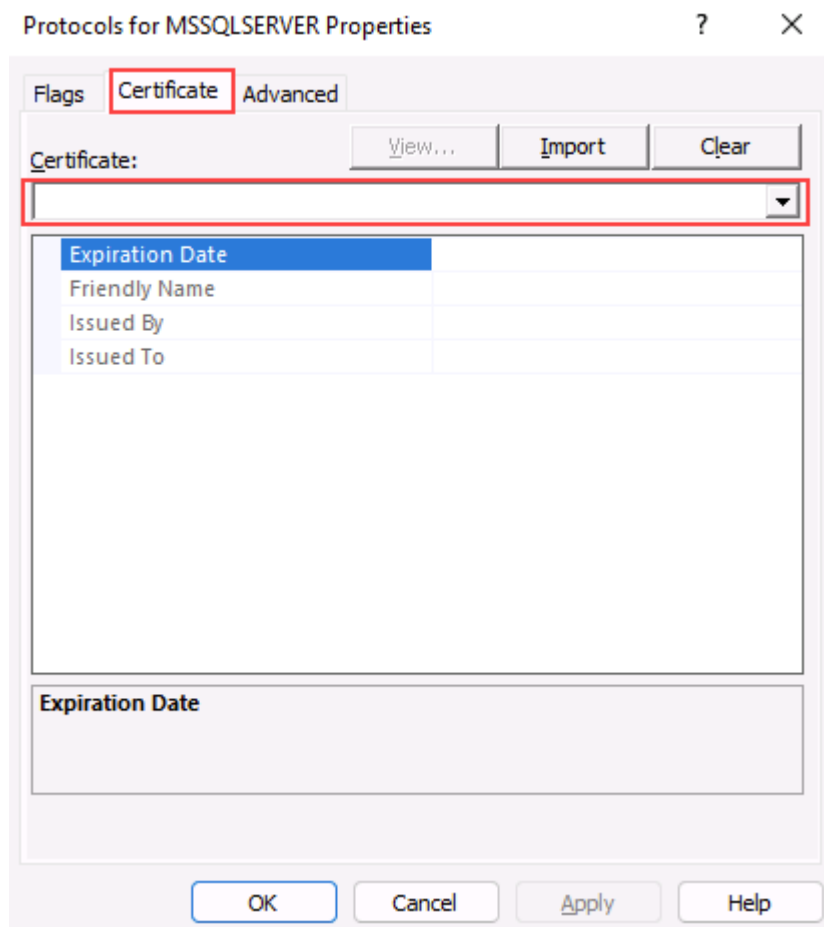
Configure SQL Server instance

You can configure a SQL Server instance to use encrypted connections by following these steps:

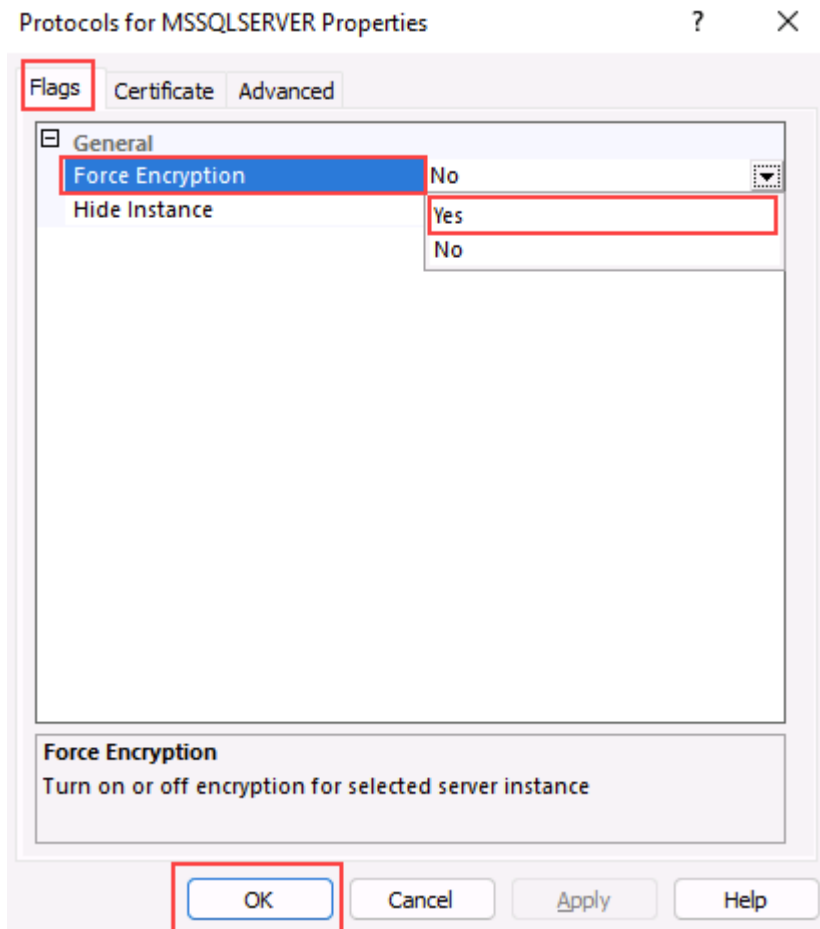
1. On the **SQL Server Configuration Manager**, expand **SQL Server Network Configuration**, right-click **Protocols for <server instance>**, and then select **Properties**.



2. From the **Protocols for <server instance> Properties** dialog box, select the **Certificate** tab, then select the certificate from the **Certificate** drop-down.



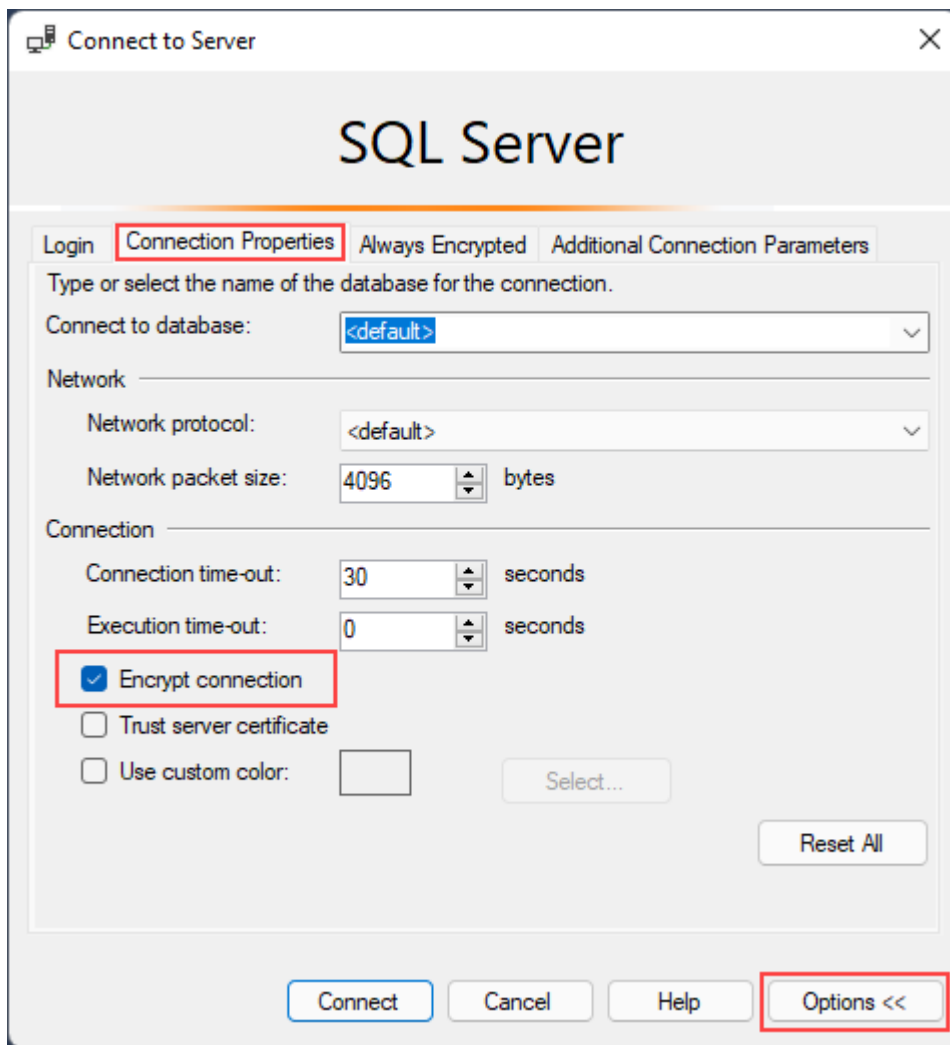
3. On the **Flags** tab, in the **ForceEncryption** property, select **Yes**, and then select **OK**.



4. Restart the SQL Server service.

Once the necessary configuration is in place, you can test the connection through SQL Server Management Studio:

1. In the **Connect to Server** dialog box, complete the connection information, and then select **Options**.
2. On the **Connection Properties** tab, select **Encrypt connection**, and then **Connect**.



All steps must have been executed correctly for you to be able to authenticate through SQL Server Management Studio using TLS.

6. Describe SQL injection

<https://learn.microsoft.com/en-us/training/modules/protect-data-transit-rest/6-describe-sql-injection>

Describe SQL injection

Completed

SQL injection is a common method used for data breaches. The attack involves appending an SQL command to a form field in a web or application front end, usually through a website, with the intent of breaking the original SQL script and executing the injected SQL script. This often occurs when dynamically generated SQL is used within the client application. The primary cause of SQL injection

attacks is poor coding practices in both the client application and database stored procedures. While many developers adopt better practices, SQL injection remains a significant problem due to the prevalence of legacy applications and newer applications built by developers who didn't prioritize SQL injection prevention.

As an example, assume that a front-end web application creates a dynamic SQL statement as follows:

```
SELECT * FROM Orders WHERE OrderId=25
```

This T-SQL is created when the user goes to the sales order history portion of the company's website and enters 25 into the form field for the order ID number. However, suppose the user enters more than just an ID number, for example "25; DELETE FROM Orders;"

In that case, the query sent to your database would be as follows:

```
SELECT * FROM Orders WHERE OrderID=25; DELETE FROM Orders;
```

The way the query in the above example works is that the SQL database is told via the semicolon ";" that the statement has ended and that there's another statement that should be run. The database then processes the next statement as instructed, which would result in the deletion of all rows from the Orders table.

The initial `SELECT` query is run as normal without any errors being generated. However, when you look at the Orders table, you don't see any rows. The second query in the batch, which deletes all the rows, was also executed.

One technique used to prevent SQL injection attacks is to inspect the text of the parameters, or values entered into the form fields, looking for various keywords. However, this solution only provides minimum protection as there are many, many ways to force these attacks to work. Some of these injection techniques include passing in binary data, having the database engine convert the binary data back to a text string, and then executing the string.

You can run the following T-SQL code to see an example of this scenario.

```
DECLARE @v VARCHAR(255)

SELECT @v = cast(0x73705F68656C7064462 AS VARCHAR(255))

EXEC (@v)
```

When accepting data from a user, whether a customer or an employee, it's crucial to validate that the input matches the expected data type to prevent SQL injection attacks. If a number is expected, the client application should verify that the input is indeed a number. If a text string is expected, ensure it is of the correct length and doesn't contain any binary data. The client application should validate all user input. Validation can be done by either informing the user of the issue and allowing them to correct it, or by gracefully handling the error to ensure no commands are sent to the database or file system.

While fixing your application code should always be the priority, there may be cases where it's not possible. In such cases, Advanced Threat Protection can offer an extra layer of security for your sensitive data.

7. Understand Azure Key Vault

<https://learn.microsoft.com/en-us/training/modules/protect-data-transit-rest/7-understand-azure-key-vault>

Understand Azure Key Vault

Completed

Azure Key Vault is a tool used for storing and accessing secrets. Whether they be passwords, certificates or keys, Key Vault acts as a secure area for those secrets to be accessed in a secure fashion, typically programmatically. Key Vault data has its own RBAC policies, separate from the Azure subscription. This means someone who is in the role of subscription admin won't have access to the Key Vault unless explicitly granted.

SQL Server, either within an Azure Virtual Machine or on-premises, supports using Azure Key Vault to store certificates for features such as Transparent Data Encryption, Backup Encryption, or Always Encrypted. While this configuration is complex in an on-premises environment, it's easily managed when using SQL Server on Azure Virtual Machine, as shown in the following image.

Microsoft Azure

Home > New > Marketplace > Get Started > SQL Server 2017 on Windows Server 2016 > Create a virtual machine

Create a virtual machine

Basics Disks Networking Management Advanced **SQL Server settings** Tags Review + create

SECURITY & NETWORKING

* SQL connectivity Private (within Virtual Network) ▼

* Port 1433

SQL AUTHENTICATION

SQL Authentication ⓘ Disable Enable

Azure Key Vault integration ⓘ Disable **Enable**

* Key Vault URL ⓘ https://contosokeyvault.azure.net ✓

* Principal name ⓘ ✓

* Principal secret ⓘ ✓

* Credential name ⓘ ✓

In order to configure the Azure Key Vault integration, you need to set the Key Vault URL, the Principal name, the Principal secret, and the name of the credential. This task can be done at the virtual machine creation or to an existing VM.

Configuring SQL Server to connect to Azure Key Vault first requires creating a normal SQL Server login within the instance. Next a Credential needs to be created and mapped to the login. For the identity of the credential, the name of the key vault should be used. For the secret of the credential, use the application ID from Azure Key Vault.

Once the credential is created, an asymmetric key can be created within the Azure Key Vault. An asymmetric key can then be created within the SQL Server database. The key in database can be mapped to the Azure Key Vault asymmetric key using the `CREATE ASYMMETRIC KEY` command with the `FROM PROVIDER` syntax. Once an asymmetric key is created within the database, that key can be used for TDE, or Backup Encryption or Always Encrypted.

8. Exercise: Configure a server-based firewall rule using the Azure portal

<https://learn.microsoft.com/en-us/training/modules/protect-data-transit-rest/8-exercise-configure-azure-sql-database-firewall>

Exercise: Configure a server-based firewall rule using the Azure portal

Completed

In this exercise, you'll learn how to configure a server-based firewall rule using the Azure portal.

Note

To complete this exercise, you'll need an [Azure subscription](#).

Launch the exercise and follow the instructions.

[Launch Exercise](#)

9. Module assessment

<https://learn.microsoft.com/en-us/training/modules/protect-data-transit-rest/9-knowledge-check>

Module assessment

Completed

10. Summary

<https://learn.microsoft.com/en-us/training/modules/protect-data-transit-rest/10-summary>

Summary

Completed

This module explored the encryption options available within Microsoft SQL Server and Azure SQL. Each of the various platforms supports different database encryption options. You also explored these data encryption options and how to configure them.

Now that you've reviewed this module, you should be able to:

- Understand the difference between server and database firewall rules in Azure SQL Database
- Understand how Always Encrypted is used to protect data in transit
- Understand the role of Azure Key Vault in Transparent Data Encryption
- Explain how to enable encrypted connections

Implement compliance controls for sensitive data

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/implement-compliance-controls-sensitive-data/1-introduction>

Introduction

Completed

This module explores the practices of setting compliance controls on the data that is stored within Azure SQL. We'll also review important security capabilities available to maintain compliance and improve visibility into any security risks.

Learning objectives

After taking this module, you'll understand:

- How data should be classified
- Why server and database audit are important
- How to implement row level security and dynamic data masking
- Understand the usage of Microsoft Defender for SQL
- How Ledger works
- Explore Azure Purview supported capabilities

2. Explore data classification

Explore data classification

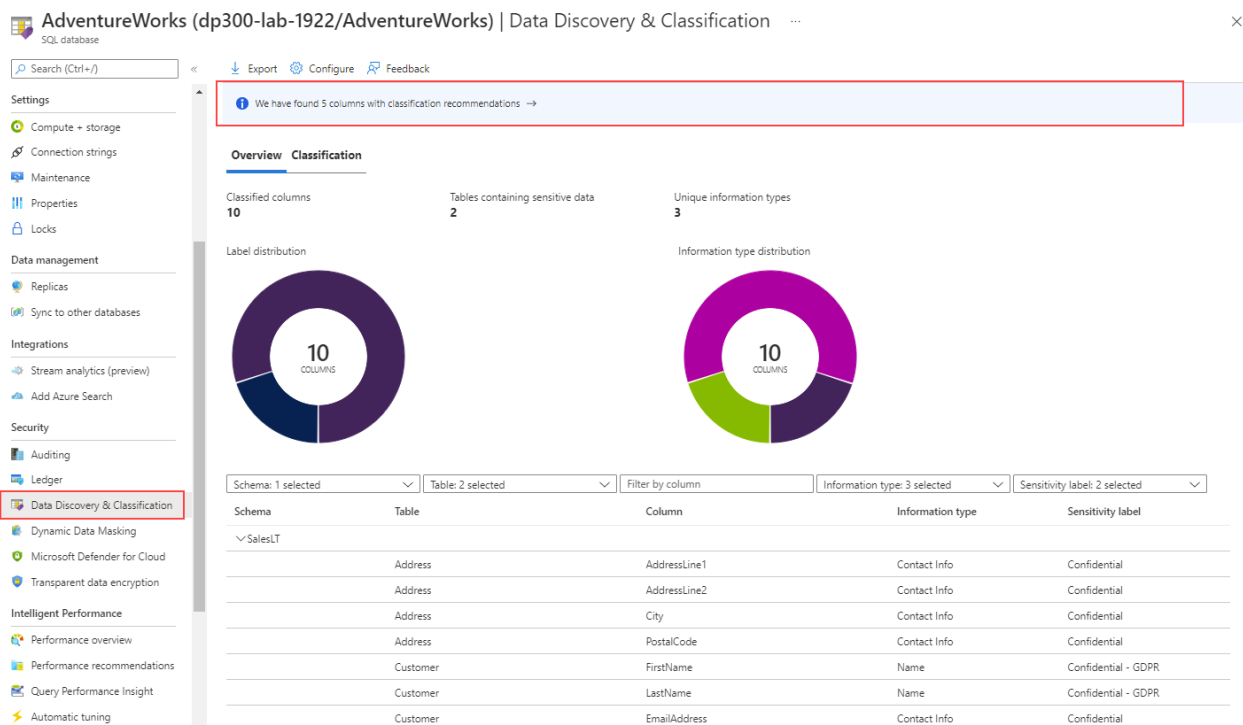
Completed

Confidential data stored within Microsoft SQL Server, Azure SQL Database, or Azure SQL Managed Instance should be classified within the database. This classification helps users and applications understand the sensitivity of the stored data.

Data classification is performed on a column-by-column basis. A single table can have columns classified as public, confidential, or highly confidential.

Initially, data classification was introduced in SQL Server Management Studio, using extended properties of objects to store classification information. Starting with SQL Server 2019, this metadata is stored in a catalog view called `sys.sensitivity_classifications`. This feature is also supported by Azure SQL Database and Azure SQL Managed Instance.

The Azure portal provides a management pane for data classification of your Azure SQL Database. You can access this feature by selecting **Data Discovery & Classification** in the **Security** section of the main blade for your Azure SQL Database.



In both the Azure portal and SQL Server Management Studio, you can configure data classification. The classification engine scans your database to identify columns with names suggesting they might

contain sensitive information. For instance, a column named *email* would be automatically flagged as containing sensitive personal information.

Overview

Classification

5 columns with classification recommendations (Click to minimize)

Accept selected recommendations

Dismiss selected recommendations

☐ Show dismissed recommendations

☐ Select all

Schema: 2 selected

Table: 3 selected

Filter by column

Information type: 4 selected

Sensitivity label: 1 selected

	Schema	Table	Column	Information type	Sensitivity label
☐	dbo	ErrorLog	UserName	Credentials	Confidential
☐	SalesLT	CustomerAddress	AddressType	Contact Info	Confidential
☐	SalesLT	SalesOrderHeader	AccountNumber	Financial	Confidential
☐	SalesLT	SalesOrderHeader	CreditCardApprovalCode	Credit Card	Confidential
☐	SalesLT	SalesOrderHeader	TaxAmt	Financial	Confidential

In the example, there are five columns recommended for classification. The **Information Type** and **Sensitivity label** properties seem consistent with the column name and overall purpose. However, since the recommendations are based on the column name, a column named *column1* that contains email addresses wouldn't be recommended as sensitive personal information.

Columns can also be classified using the sensitivity wizard in SQL Server Management Studio, or by using the `ADD SENSITIVITY CLASSIFICATION` T-SQL command as follows.

```
ADD SENSITIVITY CLASSIFICATION TO
    [Application].[People].[EmailAddress]
WITH (LABEL='PII', INFORMATION_TYPE='Email')

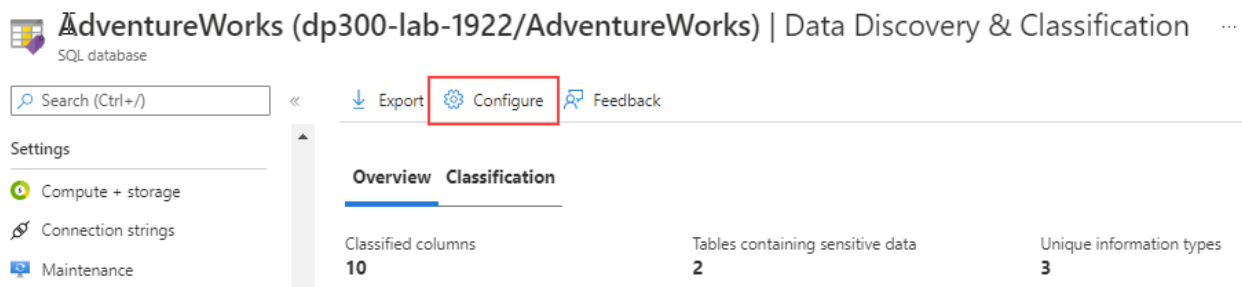
GO
```

Classification of data allows you to easily identify the sensitivity of data within the database. Knowing what columns contain sensitive data allows for easier audits and allows you to more easily identify which columns are good choices for data encryption. Classification allows other employees within the company to make better decisions on how to handle the data which is available within the database.

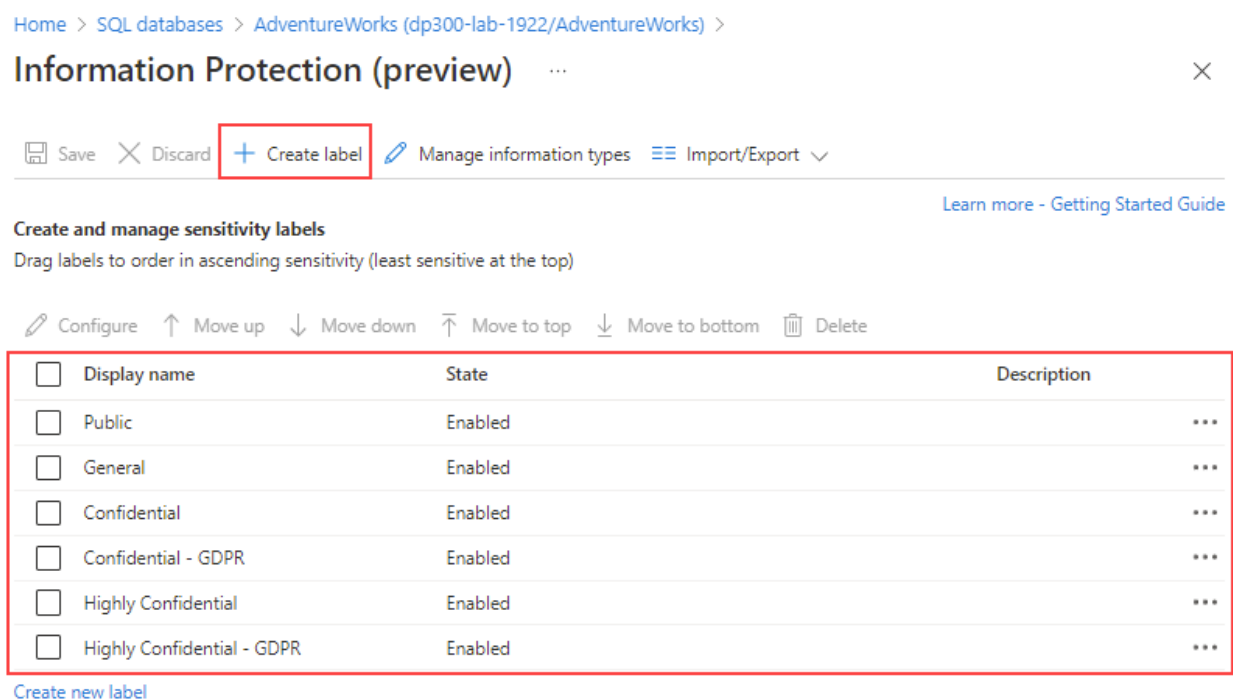
Customize classification taxonomy

Data Discovery & Classification is part of Microsoft Defender for Cloud. You can customize the taxonomy of sensitivity labels and define a set of classification rules specifically for your environment.

You can create and manage sensitivity labels as part of policy management by selecting **Data Discovery and Classification** of the main blade for your Azure SQL Database, and then **Configure**.



On the **Information Protection** page you can define labels, rank them, and link them with a set of information types.



Once you define the patterns, they're added automatically to the discovery logic for identifying this type of data in your databases, and are immediately available.

Note

Only users with administrative rights on the organization's root management group can create and manage sensitivity labels.

3. Explore server and database audit

Explore server and database audit

Completed

[Azure SQL auditing](#) tracks database events and writes them to an audit log in your Azure Storage account, Log Analytics workspace, or Event Hubs. It helps maintain regulatory compliance, analyze activity patterns, and identify deviations that may indicate security violations.

You can define server-level and database-level policies. Server policies automatically cover new and existing databases in Azure.

- If server auditing is enabled, the database is audited, regardless of the database auditing settings.
- In addition to enabling auditing on the server, you can also enable it on the database. This allows both audits to exist simultaneously; the server policy and the database policy.

It's best not to enable both server auditing and database auditing together, unless:

- A different *storage account, retention period, or Log Analytics Workspace* is used for a specific database.
- An audit is needed for a specific database that differs from the rest on the server, such as different event types or categories.

For all other cases, it's recommended to enable only server-level auditing and leave the database-level auditing disabled for all databases.

The default auditing policy for SQL Database includes the following set of action groups:

Action group	Definition
BATCH_COMPLETED_GROUP	Audits all the queries and stored procedures executed against the database.
SUCCESSFUL_DATABASE_AUTHENTICATION_GROUP	This indicates that a principal succeed to log into the database.
FAILED_DATABASE_AUTHENTICATION_GROUP	This indicates that a principal failed to log into the database.

To enable auditing for all databases on an Azure SQL server, select **Auditing** in the **Security** section of the main blade for your server. The **Auditing** page allows you to set the audit log destination.

 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: **Enabled**

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 



Audit log destination (choose at least one):

- ☐ Storage
- ☐ Log Analytics
- ☐ Event Hub

Additional Auditing



Use Ledger for auditing and Compliance

Ledger provides cryptographic proof of data integrity to auditors. This proof can help streamline the auditing process.

[Start](#)

The auditing services for SQL Database and SQL Managed Instance are optimized for availability and performance. SQL Database and SQL Managed Instance may not record some audited events when there's a high rate of activity or high network load.

Note

Auditing on Read-Only replicas is automatically enabled.

Audit sensitive labels

When combined with data classification, you can also monitor access to sensitive data. Azure SQL Auditing was enhanced to include a new field in the audit log called `data_sensitivity_information`.

By logging the sensitivity labels of the data returned by a query, this field provides an easier way to track access to classified columns.

name	action_name	class_type_desc	data_sensitivity_information
audit_event	BATCH COMPLETED	DATABASE	
audit_event	BATCH COMPLETED	DATABASE	
audit_event	BATCH COMPLETED	DATABASE	
audit_event	BATCH COMPLETED	DATABASE	<sensitivity_attributes max_rank="20" max_rank_desc="Medium"><sensitivity_attribute label="Confidential - GDPR" label_id="bf91e08c-f40-478a-b016-25164b2a65ff" information_type="Name"
audit_event	BATCH COMPLETED	DATABASE	<sensitivity_attributes max_rank="20" max_rank_desc="Medium"><sensitivity_attribute label="Confidential - GDPR" label_id="bf91e08c-f40-478a-b016-25164b2a65ff" information_type="Name"
audit_event	BATCH COMPLETED	DATABASE	<sensitivity_attributes max_rank="20" max_rank_desc="Medium"><sensitivity_attribute label="Confidential - GDPR" label_id="bf91e08c-f40-478a-b016-25164b2a65ff" information_type="Name"
audit_event	BATCH COMPLETED	DATABASE	<sensitivity_attributes max_rank="20" max_rank_desc="Medium"><sensitivity_attribute label="Confidential - GDPR" label_id="bf91e08c-f40-478a-b016-25164b2a65ff" information_type="SSN" in

Auditing consists of tracking and recording events that occur in the database engine. Azure SQL auditing simplifies the configuration steps required to enable it, making it easier to track database activities for SQL Database and SQL Managed Instance.

4. Implement Dynamic Data Masking

<https://learn.microsoft.com/en-us/training/modules/implement-compliance-controls-sensitive-data/4-implement-dynamic-data-masking>

Implement Dynamic Data Masking

Completed

Dynamic Data Masking works by obfuscating data in order to limit its exposure. Users who don't need to see sensitive data can view the column that contains the data, but not the actual data itself. Dynamic Data Masking works at the presentation layer, and that unmasked data is always visible by high privileged users.

Dynamic Data Masking has the advantage that it doesn't require many modifications to the application or database. You can configure it through the Azure portal, or using T-SQL as follows.

```

ALTER TABLE [Application].[People] ALTER COLUMN [PhoneNumber] ADD MASKED WITH (FUNCTION = 'partial(0,"XXX-XXX-",4)')
ALTER TABLE [Application].[People] ALTER COLUMN [EmailAddress] ADD MASKED WITH (FUNCTION = 'email()')

EXECUTE AS USER = 'DDMDemo'

SELECT [PersonID]
      ,[FullName]
      ,[PhoneNumber]
      ,[EmailAddress]
FROM [WideWorldImporters].[Application].[People]
WHERE PhoneNumber is NOT NULL

```

	PersonID	FullName	PhoneNumber	EmailAddress
1	2	Kayla Woodcock	XXX-XXX-0102	kXXX@XXXX.com
2	3	Hudson Onslow	XXX-XXX-0102	hXXX@XXXX.com
3	4	Isabella Rupp	XXX-XXX-0102	iXXX@XXXX.com
4	5	Eva Muirden	XXX-XXX-0102	eXXX@XXXX.com
5	6	Sophia Hinton	XXX-XXX-0102	sXXX@XXXX.com
6	7	Amy Trefl	XXX-XXX-0102	aXXX@XXXX.com
7	8	Anthony Grosse	XXX-XXX-0102	aXXX@XXXX.com

In the example, both the *PhoneNumber* and *EmailAddress* columns are hidden from *DDMDemo* user who only has `SELECT` permission on the table. The user is allowed to see the last four digits of the phone number as it's masked using a *partial* function that replaces all but the last four digits in the column. This masking is considered to be a custom function. In addition to T-SQL, if you're using Azure SQL Database, you can create dynamic masking rules in the Azure portal:

Home > AdventureWorks (dp300-lab06-xyz/AdventureWorks) | Dynamic Data Masking > Add masking rule

Add masking rule

Add Discard Delete

Mask name

HumanResources_Employee_NationalID ...

Select what to mask

Schema

HumanResources

Table

Employee

Column

NationalIDNumber (nvarchar)

Select how to mask

Masking field format

Default value (0, xxxx, 01-01-1900)
Default value (0, xxxx, 01-01-1900)
Credit card value (xxxx-xxxx-xxxx-1234)
Email (aXXX@XXXX.com)
Number (random number range)
Custom string (prefix [padding] suffix)

You can reach the screen to add a masking rule by navigating to your database in the Azure portal and selecting **Dynamic Data Masking** in the **Security** section of the main blade for your database.

Dynamic Data Masking supports the following masking patterns that can be used:

Masking function	Definition	T-SQL example
Default	Masks the data in the column without exposing any part of the values to the user. The user would see XXXX for string values, 0 for numbers, and 01.01.1900 for date values.	<pre>ALTER TABLE [Customer] ALTER COLUMN Address ADD MASKED WITH (FUNCTION = 'default()')</pre>
Credit card	Masks all but the final four characters, allowing users to view the final four digits. This masking can be useful for customer service agents who need to view the last four digits of a credit card number but who don't need to see the entire	<pre>ALTER TABLE [Customer] ALTER COLUMN Address ADD MASKED WITH (FUNCTION = 'partial(0,"XXXX-XXXX-XXXX-XXXX-",4)')</pre>

Masking function	Definition	T-SQL example
	number. The data is shown in the usual format of a credit card number XXXX-XXXX-XXXX-1234.	
Email	Only the first letter and the trailing domain suffix aren't masked; for example, "aXXX@XXXXXXX.com"	<pre>ALTER TABLE [Customer] ALTER COLUMN Email ADD MASKED WITH (FUNCTION = 'email()')</pre>
Number	This masking format should be used on numeric columns. It shows a random number as the masked value instead of the actual value. With each query, a different number is displayed.	<pre>ALTER TABLE [Customer] ALTER COLUMN [Month] ADD MASKED WITH (FUNCTION = 'random(1, 12)')</pre>
Custom string	This option allows text to be masked with any value, and to display a custom number of characters at either end of the masked value. If the length of the value being masked is equal to or less than the number of characters which the mask specifies are to be displayed, then only the masked characters are displayed.	<pre>ALTER TABLE [Customer] ALTER COLUMN [PhoneNumber] ADD MASKED WITH (FUNCTION = 'partial(1, "XXXXXXX", 0)')</pre>

To enable users to retrieve unmasked data from the columns for which masking is defined, you need to explicitly grant `UNMASK` permission.

Note

It's possible to identify masked data using inference based on the results. If you're using data masking, you should also limit the ability of the user to run unplanned queries.

For that reason, it's highly recommended to use dynamic data masking with other security features, such as auditing, encryption, row level security in order to better protect sensitive data.

Use case

Data masking is a simple and lightweight feature, and it's ideal for many scenarios, including:

- Mask data from application users who have no direct access to the database.
- Restricting private information for a group of users.
- Provide masked data to external vendors, where you need to protect sensitive information while still preserving the relationships among items in the data.
- Export a copy of your production database to a lower environment for development purposes with a user who doesn't have `UNMASK` permission. The export of the data will be in a masked format.

Import and export data

Copying data from a masked column to another table using `SELECT INTO` or `INSERT INTO` results in masked data in the target table.

When a user without `UNMASK` privilege runs SQL Server Import and Export, the exported data file contains masked data, and the imported database will contain inactively masked data.

To learn more about how Dynamic Data Masking works, see [Dynamic Data Masking](#).

5. Implement Row Level security

<https://learn.microsoft.com/en-us/training/modules/implement-compliance-controls-sensitive-data/5-implement-row-level-security>

Implement Row Level security

Completed

[Row-level security \(RLS\)](#) doesn't use encryption and operates at the database level to restrict access to a table by using a security policy based on group membership or authorization context. This functionally is equivalent to a `WHERE` clause.

The security policy invokes an inline table-valued function to protect access to the rows in a table.

Depending on the attribute of a user, the predicate determines if that user has access to the relevant information. When you run a query against a table, the security policy applies the predicate function. Depending on the business requirements, RLS can be as simple as `WHERE CustomerId = 29` or as complex as required.

There are two types of security policies supported by row-level security:

- **Filter predicates** - restrict data access that violates the predicate.

Access	Definition
SELECT	Can't view rows that are filtered.
UPDATE	Can't update rows that are filtered.
DELETE	Can't delete rows that are filtered.
INSERT	Not applicable.

- **Block predicates** - restrict data changes that violate the predicate.

Access	Definition
AFTER INSERT	Prevents users from inserting rows with values that violate the predicate.
AFTER UPDATE	Prevents users from updating rows to values that violate the predicate.
BEFORE UPDATE	Prevents users from updating rows that currently violate the predicate.
BEFORE DELETE	Blocks delete operations if the row violates the predicate.

Because access control is configured and applied at the database level, application changes are minimal - if any. Also, users can directly have access to the tables and can query their own data.

Row-level security is implemented in three main steps:

1. Create the users or groups you want to isolate access.
2. Create the inline table-valued function that filters the results based on the predicate defined.
3. Create a security policy for the table, assigning the function created previously.

The following T-SQL commands demonstrate how to use RLS in a scenario where user access is segregated by tenant:

```
-- Create supporting objects for this example
CREATE TABLE [Sales] (SalesID INT,
    ProductID INT,
    TenantName NVARCHAR(10),
    OrderQty INT,
    UnitPrice MONEY)
GO

INSERT INTO [Sales] VALUES (1, 3, 'Tenant1', 5, 10.00);
INSERT INTO [Sales] VALUES (2, 4, 'Tenant1', 2, 57.00);
INSERT INTO [Sales] VALUES (3, 7, 'Tenant1', 4, 23.00);
INSERT INTO [Sales] VALUES (4, 2, 'Tenant2', 2, 91.00);
INSERT INTO [Sales] VALUES (5, 9, 'Tenant3', 5, 80.00);
INSERT INTO [Sales] VALUES (6, 1, 'Tenant3', 5, 35.00);
INSERT INTO [Sales] VALUES (7, 3, 'Tenant4', 8, 11.00);

-- View all the rows in the table
SELECT * FROM Sales;
```

Next, create the users and grant them access to the *Sales* table. In this example, each user is responsible for a specific tenant. The *TenantAdmin* user has access to see data from all tenants.

```
CREATE USER [TenantAdmin] WITH PASSWORD = '<strong password>'
GO
CREATE USER [Tenant1] WITH PASSWORD = '<strong password>'
GO
CREATE USER [Tenant2] WITH PASSWORD = '<strong password>'
GO
CREATE USER [Tenant3] WITH PASSWORD = '<strong password>'
GO
```

```

CREATE USER [Tenant4] WITH PASSWORD = '<strong password>'
GO

GRANT SELECT ON [Sales] TO [TenantAdmin]
GO
GRANT SELECT ON [Sales] TO [Tenant1]
GO
GRANT SELECT ON [Sales] TO [Tenant2]
GO
GRANT SELECT ON [Sales] TO [Tenant3]
GO
GRANT SELECT ON [Sales] TO [Tenant4]
GO

```

Next, we create a new schema, an inline table-valued function, and grant user access to the new function. The `WHERE @TenantName = USER_NAME() OR USER_NAME() = 'TenantAdmin'` predicate evaluates if the user name executing the query matches the *TenantName* column values.

```

CREATE SCHEMA sec;
GO

--Create the filter predicate

CREATE FUNCTION sec.tvf_SecurityPredicatebyTenant(@TenantName AS NVARCHAR(10))
    RETURNS TABLE
WITH SCHEMABINDING
AS
    RETURN      SELECT 1 AS result
                WHERE @TenantName = USER_NAME() OR USER_NAME() = 'TenantAdmin'
GO

--Grant users access to inline table-valued function

GRANT SELECT ON sec.tvf_SecurityPredicatebyTenant TO [TenantAdmin]
GO
GRANT SELECT ON sec.tvf_SecurityPredicatebyTenant TO [Tenant1]
GO
GRANT SELECT ON sec.tvf_SecurityPredicatebyTenant TO [Tenant2]
GO
GRANT SELECT ON sec.tvf_SecurityPredicatebyTenant TO [Tenant3]
GO
GRANT SELECT ON sec.tvf_SecurityPredicatebyTenant TO [Tenant4]
GO

--Create security policy and add the filter predicate
CREATE SECURITY POLICY sec.SalesPolicy
ADD FILTER PREDICATE sec.tvf_SecurityPredicatebyTenant(TenantName) ON [dbo].[Sales]
WITH (STATE = ON);
GO

```

At this point, we're ready to test the access:

```
EXECUTE AS USER = 'TenantAdmin';  
SELECT * FROM dbo.Sales;  
REVERT;  
  
EXECUTE AS USER = 'Tenant1';  
SELECT * FROM dbo.Sales;  
REVERT;  
  
EXECUTE AS USER = 'Tenant2';  
SELECT * FROM dbo.Sales;  
REVERT;  
  
EXECUTE AS USER = 'Tenant3';  
SELECT * FROM dbo.Sales;  
REVERT;  
  
EXECUTE AS USER = 'Tenant4';  
SELECT * FROM dbo.Sales;  
REVERT;
```

The *TenantAdmin* user should see all the rows. The *Tenant1*, *Tenant2*, *Tenant3*, and *Tenant4* users should only see their own rows.

If you alter the security policy with `WITH (STATE = OFF);`, you notice that users see all the rows.

SQLQuery2.sql - sql...rks (sql_admin (70))* -p X

```
ALTER SECURITY POLICY sec.SalesPolicy WITH (STATE = OFF);
GO

EXECUTE AS USER = 'Tenant1';
SELECT * FROM dbo.Sales;
REVERT;

EXECUTE AS USER = 'Tenant2';
SELECT * FROM dbo.Sales;
REVERT;

EXECUTE AS USER = 'Tenant3';
SELECT * FROM dbo.Sales;
REVERT;

EXECUTE AS USER = 'Tenant4';
SELECT * FROM dbo.Sales;
REVERT;
```

100 %

Results Messages

	SalesID	ProductID	TenantName	OrderQty	UnitPrice
1	1	3	Tenant1	5	10.00
2	2	4	Tenant1	2	57.00
3	3	7	Tenant1	4	23.00
4	4	2	Tenant2	2	91.00
5	5	9	Tenant3	5	80.00
6	6	1	Tenant3	5	35.00
7	7	3	Tenant4	8	11.00

Note

There's a risk of information leakage if an attacker writes a query with a specially crafted `WHERE` clause and, for example, a divide-by-zero error, to force an exception if the `WHERE` condition is true. This is known as a *side-channel attack*. It's wise to limit the ability of users to run unplanned queries when using row-level security.

Use case

Row-level security is ideal for many scenarios, including:

- When you need to isolate departmental access at the row level.
- When you need to restrict customers' data access to only the data relevant to their company.
- When you need to restrict access for compliance purposes.

Best practices

Here are a few best practices to consider when implementing RLS:

- Create a separate schema for predicate functions, and security policies.
- Avoid type conversions in predicate functions.
- Avoid using excessive table joins and recursion in predicate functions.

6. Understand Microsoft Defender for SQL

<https://learn.microsoft.com/en-us/training/modules/implement-compliance-controls-sensitive-data/6-understand-microsoft-defender-for-sql>

Understand Microsoft Defender for SQL

Completed

[Microsoft Defender for SQL](#) offers a suite of protections for Azure SQL Database and Azure SQL Managed Instance as part of the advanced SQL security features, including SQL vulnerability assessment and Advanced Threat Protection.

SQL vulnerability assessment

SQL vulnerability assessment is a service that uses a knowledge base of security rules to flag items that don't comply when they're scanned. It checks your database for security best practices, and providing visibility into your security state, such as misconfigurations, excessive permissions, and exposure of sensitive data.

To see recommendations for SQL Database and SQL Managed Instance, you must enable Microsoft Defender for SQL at the subscription level (recommended). You also need to provide a storage account. Alternatively, you can choose to receive emails with a summary of the scan results.

The vulnerability assessment feature can detect potential risks in your environment, and help you enhance database security. It also provides insight into your security state and actionable steps to resolve security alerts.

To learn more about SQL vulnerability assessment, see [SQL vulnerability assessment helps you identify database vulnerabilities](#).

Advanced Threat Protection

Advanced Threat Protection monitors the database connections and the queries that are executed in order to detect potentially harmful activities. You can manage and access Advanced Threat Protection via the central Microsoft Defender for SQL portal.

The following threats are supported by Advanced Threat Protection:

Alerts	Definition
Vulnerability to SQL injection	This alert looks for T-SQL code coming into your database that may be vulnerable to SQL injection attacks. An example would be a stored procedure call that didn't sanitize user inputs.
Potential SQL injection	This alert is triggered when an attacker is actively attempting to execute a SQL injection attack.
Access from unusual location	This alert is triggered when a user logs in from an unusual geographic location.
Access from unusual Azure data center	This alert is looking for attacks from an Azure data center that isn't normally accessed.
Access from unfamiliar principal	This alert is raised when a user or applications log on to a database that they haven't previously accessed.
Access from a potentially harmful application	This alert detects common tools that are used to attack databases.
Brute force SQL credentials	This alert is triggered when there a high number of log in failures with different credentials.

To get maximum benefit out of it, you want to enable auditing on your databases. Although it isn't required, enabling auditing allows for deeper investigation into the source of the problem if Advanced Threat Protection detects an anomaly.

The following audit action groups are recommended:

- **BATCH_COMPLETED_GROUP**
- **SUCCESSFUL_DATABASE_AUTHENTICATION_GROUP**
- **FAILED_DATABASE_AUTHENTICATION_GROUP**

How to enable Microsoft Defender for SQL

You must belong to either the SQL security manager role, or one of the database or server admin role to manage Microsoft Defender for SQL settings.

Enable Microsoft Defender for SQL for SQL Database or SQL Managed Instance from the server main blade by selecting **Microsoft Defender for Cloud**, and then the **(Configure)** link.

Home > AdventureWorks (dp300-server/AdventureWorks)

AdventureWorks (dp300-server/AdventureWorks) | Microsoft Defender for Cloud

SQL database

Search (Ctrl+/)

Settings

- Compute + storage
- Connection strings
- Maintenance
- Properties
- Locks
- Data management
 - Replicas
 - Sync to other databases
- Integrations
 - Stream analytics (preview)
 - Add Azure Search
- Security
 - Auditing
 - Ledger
 - Data Discovery & Classification
 - Dynamic Data Masking
 - Microsoft Defender for Cloud**
 - Transparent data encryption

Visit Microsoft Defender for Cloud to manage security across your virtual networks, data, apps, and more

Recommendations: 0

Security alerts: 0

Findings: --

Microsoft Defender for SQL: **Enabled at the subscription-level** [\(Configure\)](#)

[Learn more](#)
[About Microsoft Defender for Cloud](#)
[About Microsoft Defender for SQL](#)

Recommendations

Defender for Cloud continuously monitors the configuration of your SQL Servers to identify potential security vulnerabilities and recommends actions to mitigate them.

No recommendations to display

There are no security recommendations for this resource

[View all recommendations in Defender for Cloud](#)

Security incidents and alerts

Defender for Cloud uses advanced analytics and global threat intelligence to alert you to malicious activity. Alerts displayed below are from the past 21 days.

On the **Server settings** page, make sure the **MICROSOFT DEFENDER FOR SQL** property is set to **ON**.

Server settings

dp300-server

Save Discard Feedback

MICROSOFT DEFENDER FOR SQL

ON OFF

Microsoft Defender for SQL costs 15 USD/server/month. It includes Vulnerability Assessment and Advanced Threat Protection. We invite you to a trial period for the first 30 days, without charge.

Once Microsoft Defender for SQL is enabled, you can view recommendations by selecting **Microsoft Defender for Cloud** on the server blade.

dp300-server

SQL server

Microsoft Defender for Cloud

Search (Ctrl+J)

backups

Deleted databases

Failover groups

Import/Export history

Security

Auditing

Firewalls and virtual networks

Private endpoint connections

Microsoft Defender for Cloud

Transparent data encryption

Identity

Intelligent Performance

Automatic tuning

Recommendations

Monitoring

Logs

Visit Microsoft Defender for Cloud to manage security across your virtual networks, data, apps, and more

Recommendations

Security alerts

Findings

Microsoft Defender for SQL: Enabled at the subscription-level (Configure)

Learn more

About Microsoft Defender for Cloud

About Microsoft Defender for SQL

4

0

--

Recommendations

Defender for Cloud continuously monitors the configuration of your SQL Servers to identify potential security vulnerabilities and recommends actions to mitigate them.

Description	Severity
Auditing on SQL server should be enabled	Low
Private endpoint connections on Azure SQL Database should be enabled	Medium
Public network access on Azure SQL Database should be disabled	Medium
SQL servers should have vulnerability assessment configured	High

View additional recommendations in Defender for Cloud >

Microsoft Defender for SQL provides advanced built-in features to identify and handle threats to the database without the need to be a security expert.

7. Explore Ledger

<https://learn.microsoft.com/en-us/training/modules/implement-compliance-controls-sensitive-data/7-explore-azure-sql-database-ledger>

Explore Ledger

Completed

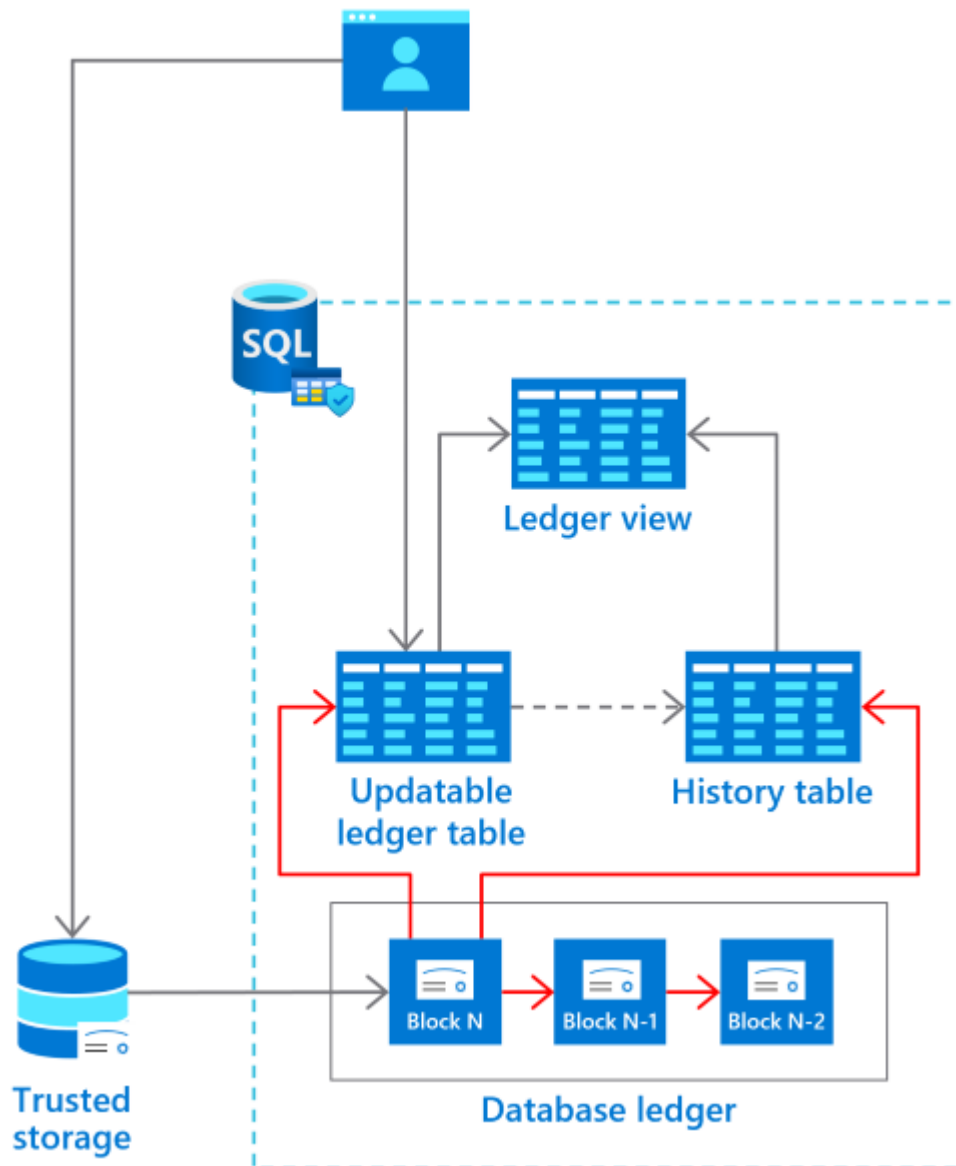
Ledger provides tamper-evidence capabilities in your database. You can cryptographically attest to other parties, such as auditors or other business parties, that your data hasn't been tampered with.

How it works

Cryptography and blockchain have begun to appear in far reaching areas of technology with varying degrees of success. One place where it has proved useful and beneficial is in being used as the technology behind the Ledger. Ledger provides tamper-evidence capabilities in your database. Using this feature, you can provide concrete proof to auditors, business partners or any interested parties what data has been changed or tampered with.

A traditional ledger is defined as a collection of accounts of a particular type and that's exactly what the Azure SQL Database Ledger feature provides in your environment. It provides transparent protection of your data from bad actors including but not limited to attackers or even database or

cloud administrators. It provides guarantees of cryptographic data integrity while maintaining the power, flexibility, and performance of Azure SQL Database.



Each transaction that the database receives is cryptographically hashed (SHA-256). The function cryptographically links all transactions together, like a blockchain.

Components

Ledger function currently exists for tables in two forms: The Updatable Ledger Tables and the Append-only Ledger Tables.

Updatable ledger tables

Updatable ledger table can be used for applications that issue updates and deletes and inserts. It works well for system of record applications and transactional systems where matter of fact record keeping and auditing is required and happen. The updatable ledger tables track history of changes

to any rows and uses the built-in system versioning to create a history table that stores the previous version of the row for full history is kept for any updates or deletes.

Append-only ledger tables

Append-only ledger tables work well with insert only applications such as an accounting system, which still needs auditing or security information and event management (SIEM) applications. The append-only ledger table blocks all updates and deletes at the API level so not only does it provide certainty, it aides in management.

Benefits

Ledger provides multiple benefits:

Ease Audits – Audits are frequently enacted to ensure that proper security controls are in place to reduce potential attacks, backup and restore practices are as required, and thorough disaster recovery procedures are in place. Ledger provides documented proof that your data hasn't been altered in an auditing process.

Increased trust – Ledger also can help establish trust between multiple-party business processes without the complexity and performance implications that network consensus can introduce.

Data integrity – Querying the data on a blockchain network without sacrificing performance can be a serious challenge. Ledger provides data integrity for off-chain storage of blockchain networks, which helps ensure complete data trust through the entire system.

Enable ledger on a SQL database

You can enable the ledger capability only during the database creation process. Once the database is created, you cannot modify it.

1. During the dababase creation, select **Ledger** under the **Security** tab of the **Create SQL Database** page in Azure portal.
2. Select the **Configure ledger** link. On the **Configure ledger** page, select the **Enable for all future tables in this database** checkbox. Provide the information to create a new storage for your database digests. Select **Apply**.

Configure ledger ...

Create SQL Database

Ledger database

Enabling the ledger functionality at the database level will make all tables in this database updatable ledger tables. This option cannot be changed after the database is created. If you do not select this option now, you can still create ledger tables (updatable or append-only) using T-SQL. After enabling the ledger functionality for a table, you cannot disable it. [Learn more](#)

Enable for all future tables in this database



Digest storage

If you want ledger to generate digests automatically and store them for your verification later, you need to configure an Azure Storage account or Azure Confidential Ledger. Alternatively, you can manually generate digests and store them in your own secure location. [Learn more](#)

Enable automatic digest storage ⓘ



Storage type



Azure Storage



Azure Confidential Ledger

Storage account *

[Create new](#)

Storage container ⓘ



To prevent tampering of your digest files, configure and lock a retention policy for your container. [Learn more](#)

Note

This setting ensures that all future tables in the database will be ledger tables. For this reason, all data in the database will show any evidence of tampering. By default, new tables will be created as updatable ledger tables, even if you don't specify `LEDGER = ON` in the `CREATE TABLE` statement.

You can also leave this option unselected. You're then required to enable ledger functionality on a per-table basis when you create new tables by using Transact-SQL.

8. Implement Microsoft Purview

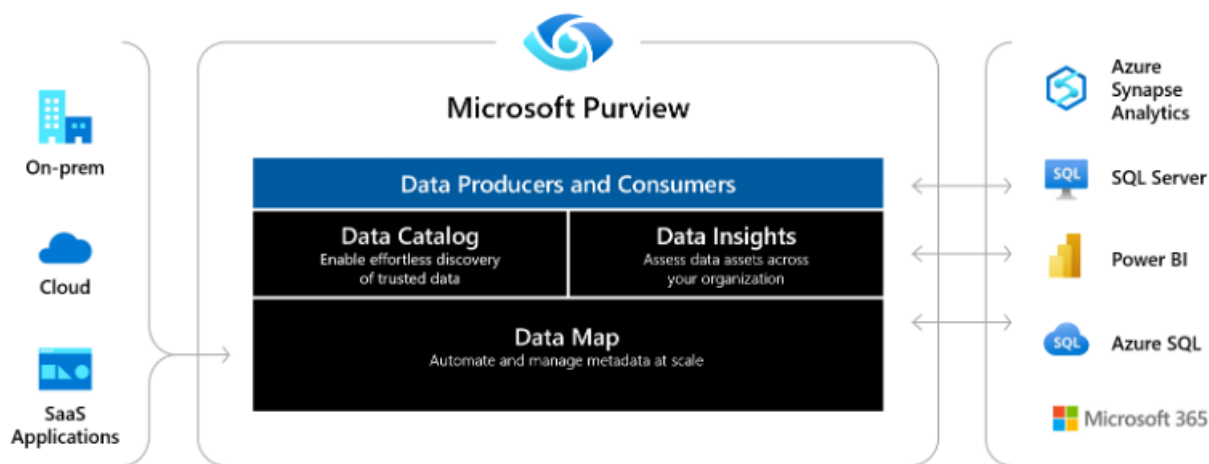
Implement Microsoft Purview

Completed

Microsoft Purview is a unified data governance service that helps you manage and govern your on-premises, multicloud, and software-as-a-service (SaaS) data. Create a holistic, up-to-date map of your data landscape with automated data discovery, sensitive data classification, and end-to-end data lineage. Enable data curators to manage and secure your data estate. Empower data consumers to find valuable, trustworthy data.

How it works

Microsoft Purview automates data discovery by providing data scanning and classification as a service for assets across your data estate. Metadata and descriptions of discovered data assets are integrated into a holistic map of your data estate. Atop this map, there are purpose-built apps that create environments for data discovery, access management, and insights about your data landscape.



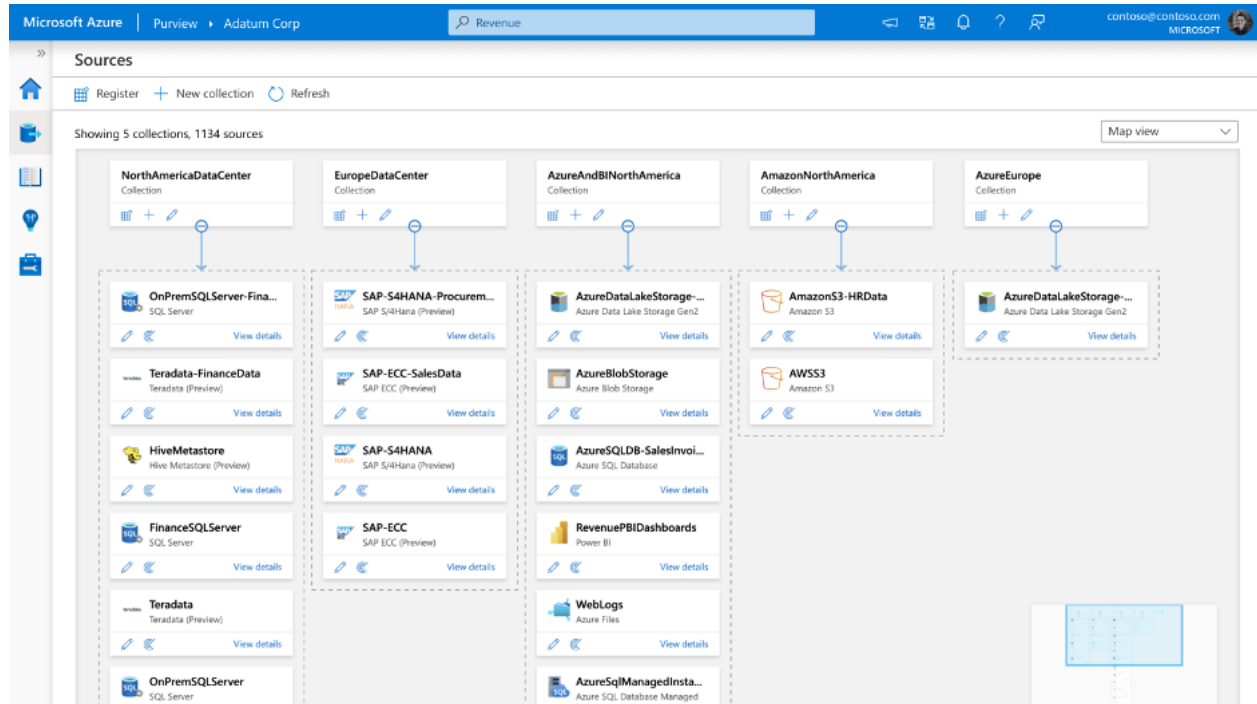
Supported capabilities

Understanding the location and movement of sensitive data across the entire data domain is one of the main features of Microsoft Purview for Azure SQL Database.

Create a unified map of data across the entire data domain

Microsoft Purview helps you lay the foundation for effective data management, including the following capabilities:

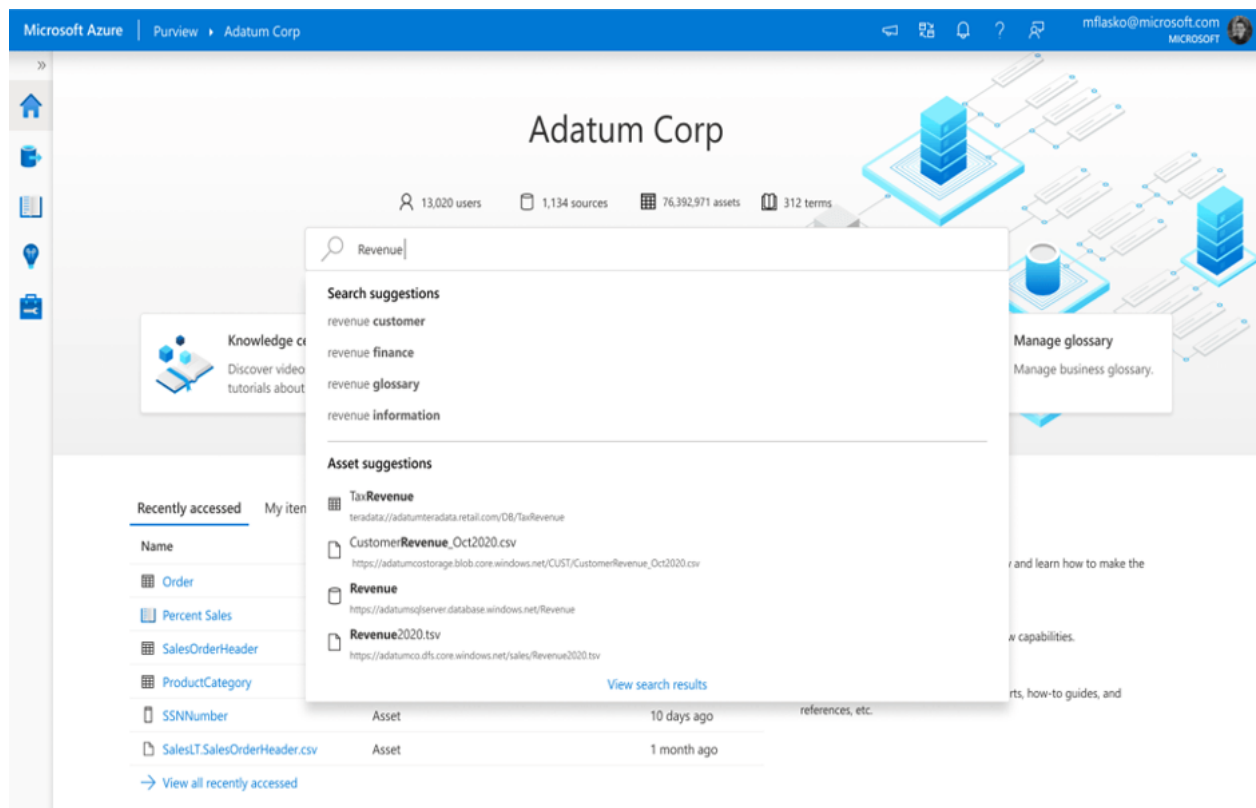
- Automate and manage hybrid resource metadata.
- Classify data using integrated and custom classifications and information protection sensitivity labels.
- Ensure consistent labeling of sensitive data across SQL Server, Azure, Microsoft 365, and Power BI.
- Easily integrate all your data systems using Apache Atlas APIs.



Make data easy to find

Make data easy to find using familiar business and technical search terms, including the following capabilities:

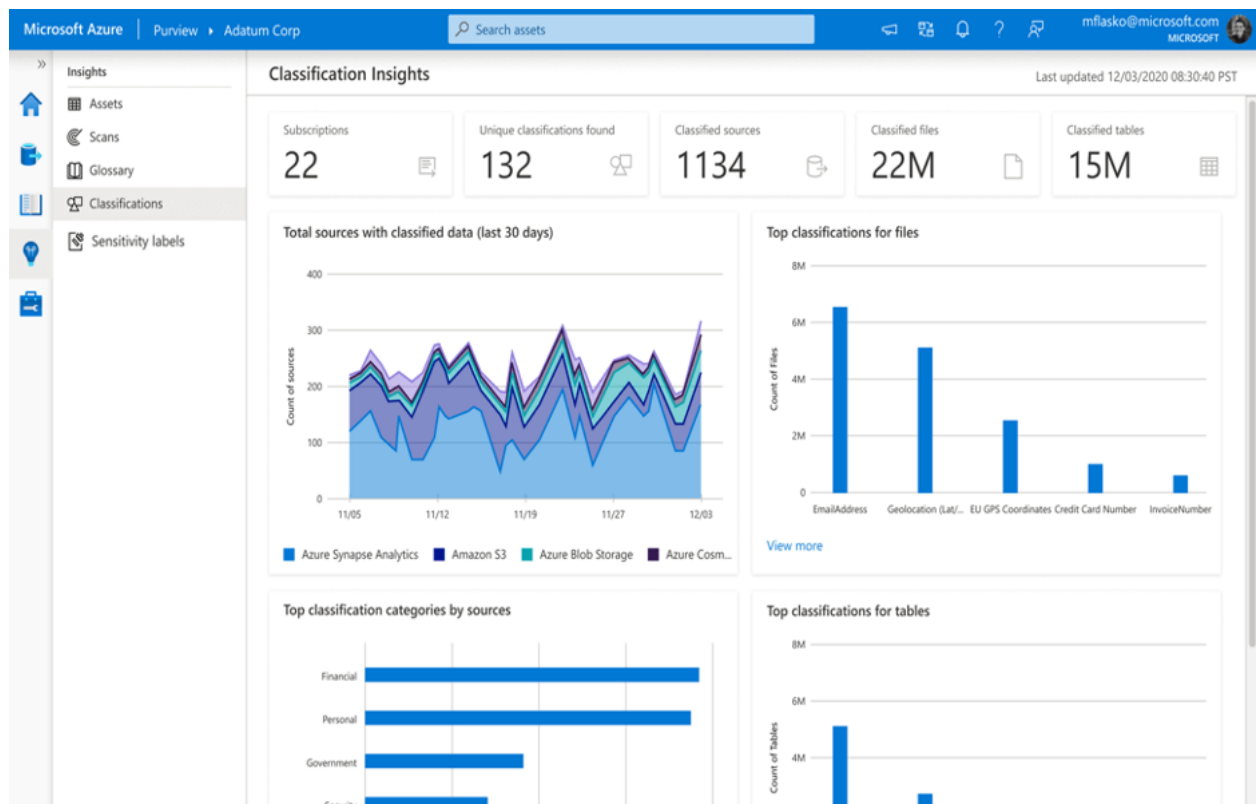
- Ensure optimal business value for your data users' data with Microsoft Purview Data Catalog.
- Eliminate the need for data dictionaries in Excel with a business-level business dictionary.
- Gain insight into the origin of your data with interactive visualization of data origin.
- Provide data scientists, engineers, and analysts with the data they need for BI, analytics, AI, and machine learning.



Get an overview of sensitive data

Microsoft Purview provides a comprehensive view of your data management operations with Data Insights (in preview), including the following capabilities:

- View your entire data domain and its distribution by asset dimension, such as source type, classification, and file size.
- Receive status updates on the number of scans that passed, failed, or canceled.
- Get key insights to add or redistribute glossary terms for better search results.



Requirements

Before you get started with Microsoft Purview, ensure the following requirements are met:

- Access to Microsoft Azure with a development or production subscription.
- Ability to create Azure resources including Microsoft Purview.
- Access to data sources such as Azure Data Lake Storage or Azure SQL in test, development, or production environments.
 - For Data Lake Storage, the required role to scan is **Reader**.
 - For Azure SQL, the identity must be able to query tables for sampling of classifications.
- Access to Microsoft Defender for Cloud or ability to collaborate with Defender for Cloud Admin for data labeling.
- An active Microsoft Purview account.
- You need to be a **Data Source Administrator** and **Data Reader** to register a source and manage it in the Microsoft Purview governance portal.

Security considerations

Let's review some important security capabilities when scanning a SQL Database using Microsoft Purview.

Firewall settings

If your database server has a firewall enabled, you need to update the firewall to allow access in one of two ways:

- **Allow Azure connections through the firewall** – A straightforward option to route traffic through Azure networking, without needing to manage virtual machines.
- **Install a self-hosted integration runtime** – Install a self-hosted integration runtime on a machine in your network and give it access through the firewall. If you have a private virtual network set up within Azure, or have any other closed network set up, using a self-hosted integration runtime on a machine within that network will allow you to fully manage traffic flow and utilize your existing network.
- **Use a managed virtual network** – You can use the Azure integration runtime in a closed network by setting up a managed virtual network using your Microsoft Purview account to connect to Azure SQL.

Authentication

To scan your data source, you need to configure an authentication method in the Azure SQL Database. The following authentication options are supported when preparing for a scan:

- **System-assigned managed identity (recommended)** – This is an identity associated directly with your Microsoft Purview account that allows you to authenticate directly with other Azure resources without needing to manage a go-between user or credential set. The system-assigned managed identity is created when your Microsoft Purview resource is created, is managed by Azure, and uses your Microsoft Purview account's name. The system-assigned managed identity can't currently be used with a self-hosted integration runtime for Azure SQL.
- **User-assigned managed identity (preview)** – Similar to system-assigned managed identity, a user-assigned managed identity is a credential resource that allows Microsoft Purview to authenticate against Microsoft Entra ID. The user-assigned managed by users in Azure, rather than by Azure itself, which gives you more control over security. The user-assigned managed identity can't currently be used with a self-hosted integration runtime for Azure SQL. For more information, see our guide for user-assigned managed identities.
- **Service Principal** – A service principal is an application that can be assigned permissions like any other group or user, without being associated directly with a person. Their authentication has an expiration date, and so can be useful for temporary projects.
- **SQL Authentication** – Connect to the SQL database with a username and password.

Note

If you're using a self-hosted integration runtime to connect to your resource, system-assigned and user-assigned managed identities won't work. You need to use service principal authentication or SQL authentication.

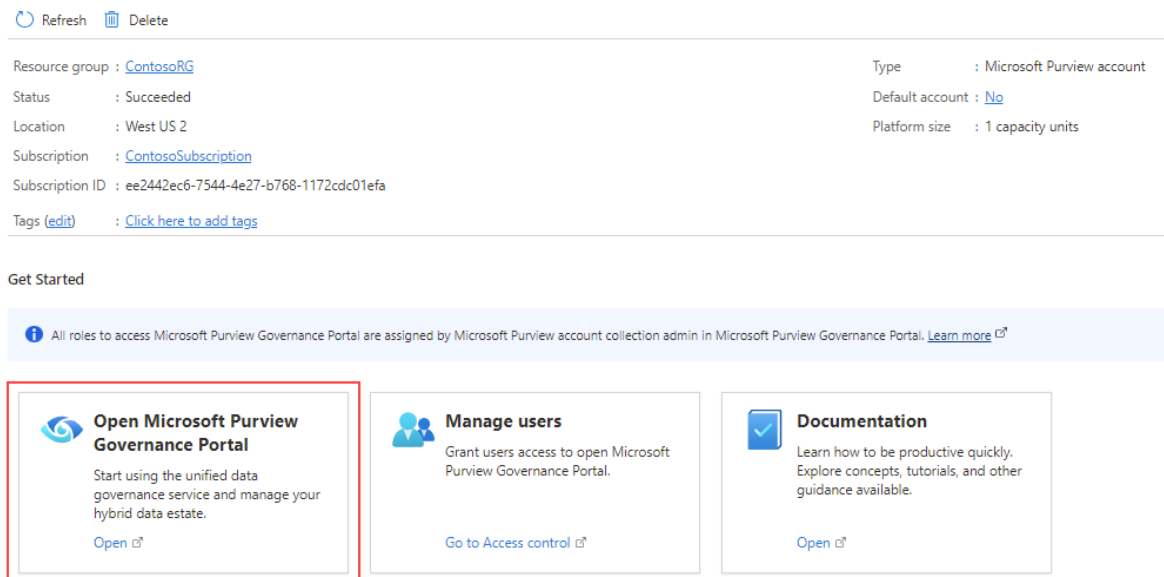
Register and scan SQL Database using Microsoft Purview

This section enables you to register the Azure SQL Database data source and set up a scan.

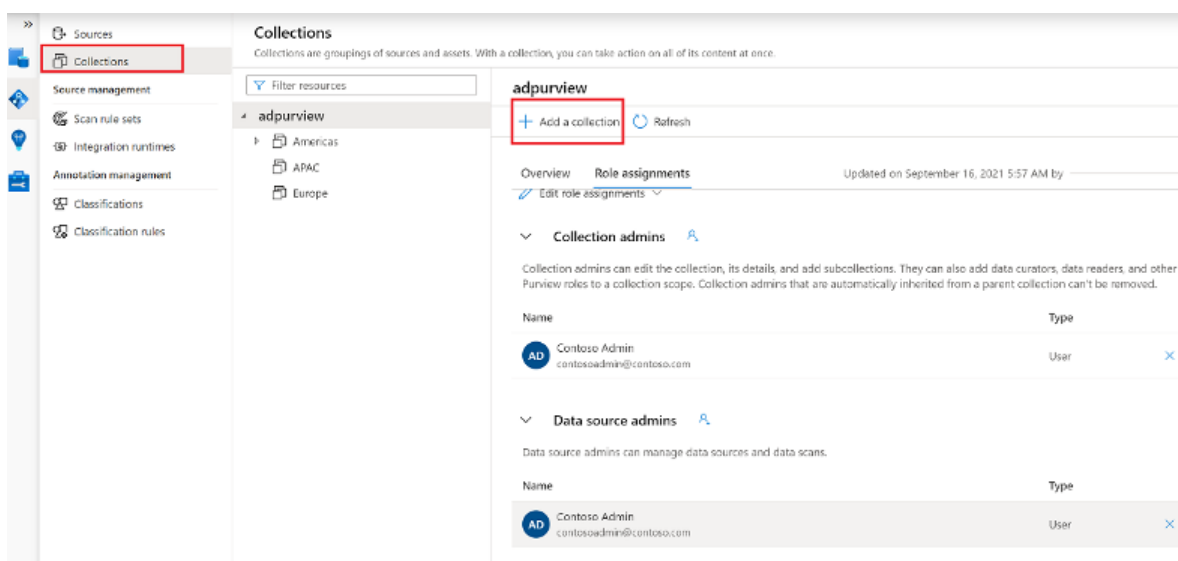
Register the data source

It's required to register the data source in Microsoft Purview before setting up a scan.

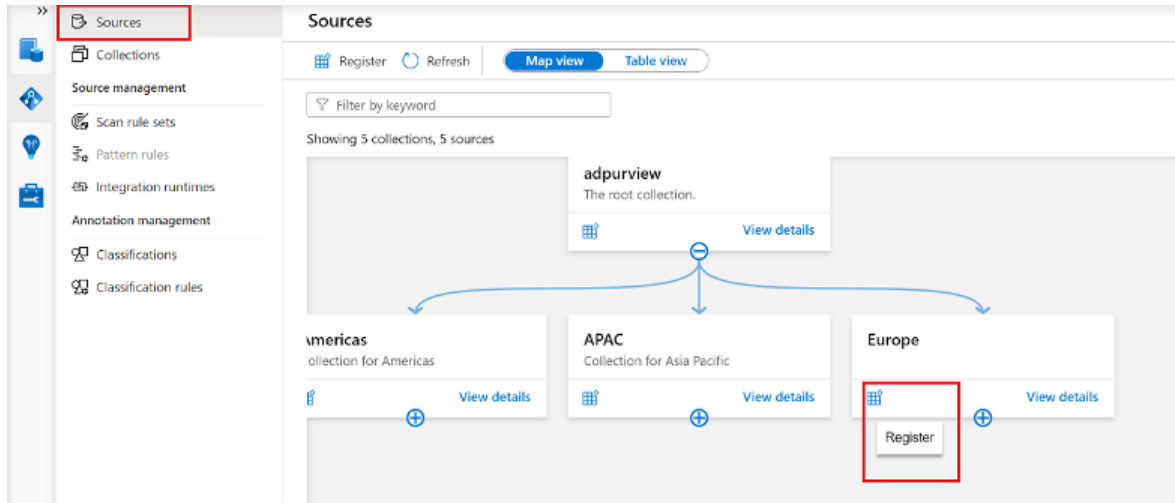
1. Open your Microsoft Purview account, and select **Open Microsoft Purview Governance Portal**.



2. Select **Data Map > Collections** from the left pane to open collection management page. Create the collection hierarchy using the **Collections** menu, and assign permissions to individual subcollections, as required.

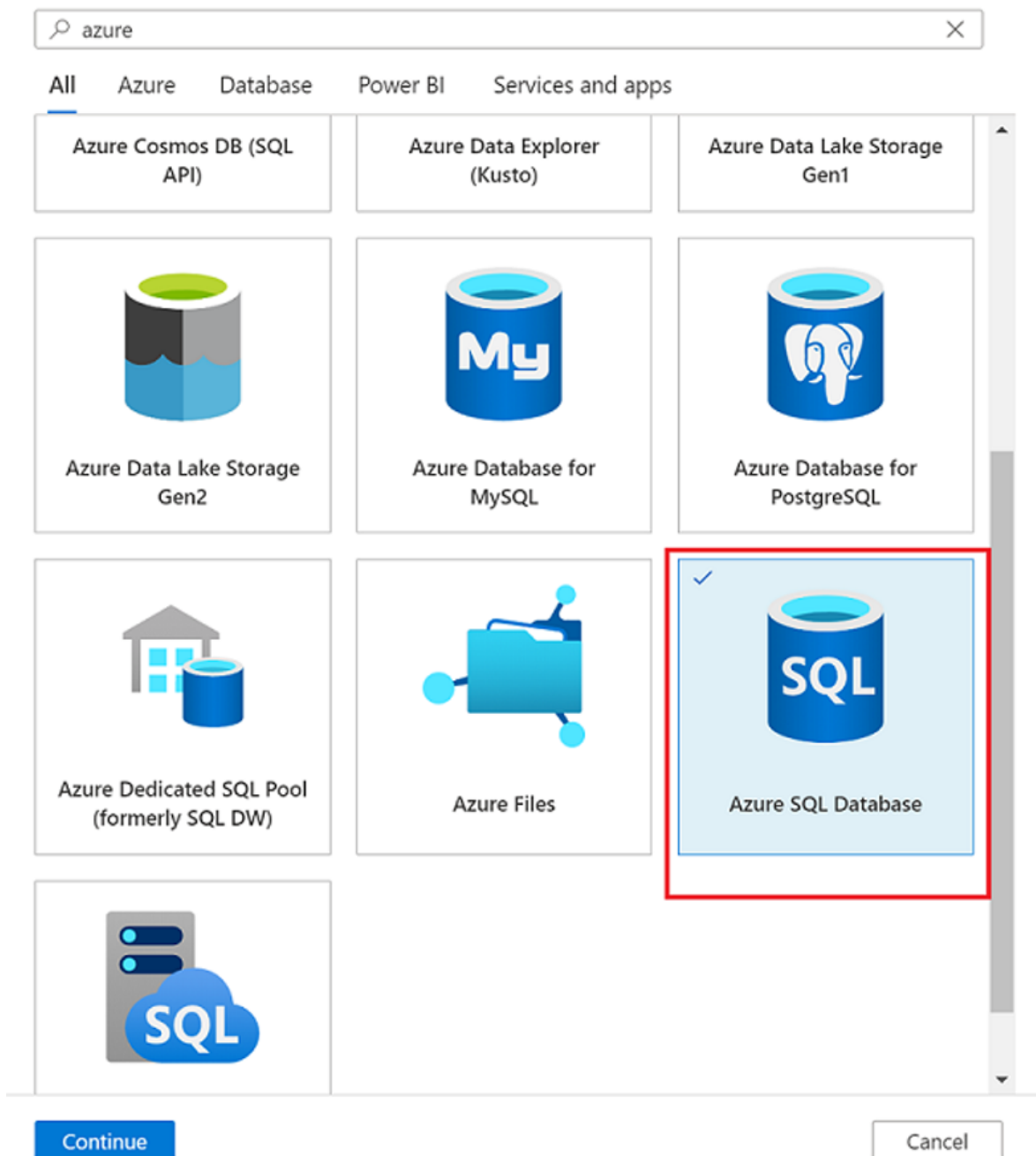


3. Navigate to the appropriate collection under the **Sources** menu, and then select **Register** to register a new SQL Database.



4. Select the Azure SQL Database data source, and then select **Continue**.

Register sources



5. Provide a name for the data source, select an Azure subscription, select the SQL Database server name, and then select **Apply**.

Edit source (Azure SQL Database)

Name *
AzureSqlDatabase-N6s

Azure subscription
Purview_E2ETest (abcdef01-2345-6789-0abc-def012345678) ▼

Server name *
sqlserverdatascans1rg1ause ▼

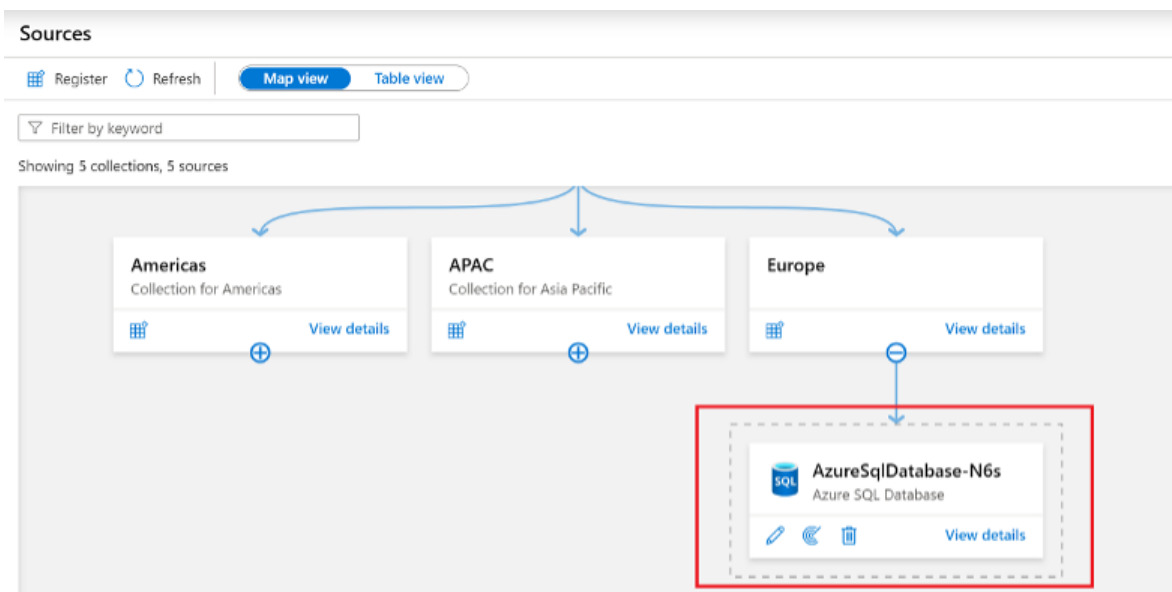
Endpoint
sqlserverdatascans1rg1ause.database.windows.net 

Select a collection * ⓘ
adpurview > Europe ▼

 All assets under this source will belong to the collection you select.

Apply **Cancel**

6. The Azure SQL Database appears under the selected collection.



Create a scan

To create and set up a scan, follow these steps:

1. Open your Microsoft Purview account and select the **Open Microsoft Purview** governance portal.

Refresh Delete

Resource group : [ContosoRG](#) Type : Microsoft Purview account
 Status : Succeeded Default account : [No](#)
 Location : West US 2 Platform size : 1 capacity units
 Subscription : [ContosoSubscription](#)
 Subscription ID : ee2442ec6-7544-4e27-b768-1172cdc01efa
 Tags ([edit](#)) : [Click here to add tags](#)

Get Started

All roles to access Microsoft Purview Governance Portal are assigned by Microsoft Purview account collection admin in Microsoft Purview Governance Portal. [Learn more](#)



Open Microsoft Purview Governance Portal

Start using the unified data governance service and manage your hybrid data estate.

[Open](#)



Manage users

Grant users access to open Microsoft Purview Governance Portal.

[Go to Access control](#)

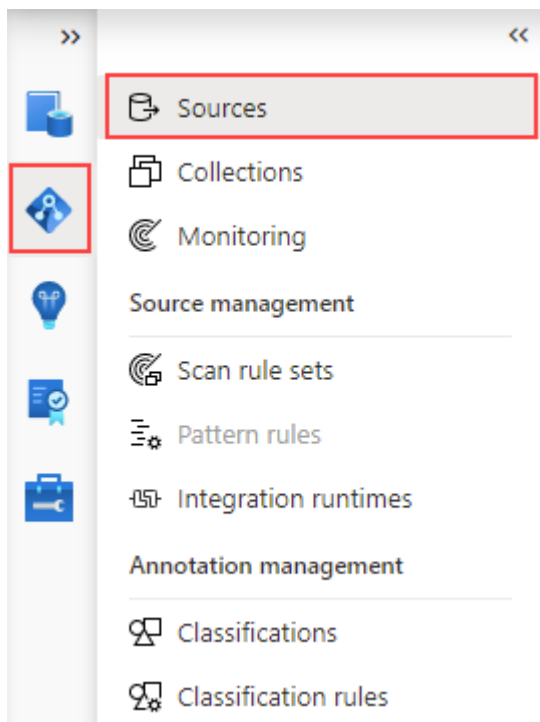


Documentation

Learn how to be productive quickly. Explore concepts, tutorials, and other guidance available.

[Open](#)

2. Select the **Data map** icon, then **Sources** to view the collection hierarchy.



3. Select the **New Scan** icon under the Azure SQL Database registered earlier.

Scan "AzureSqlDatabase-eP4"

Name *

Scan-nYB

Connect via integration runtime * ⓘ

Azure AutoResolveIntegrationRuntime

Server endpoint *

sqlserver-east-jp.database.windows.net

Database selection method

☒ From Azure subscription ☐ Enter manually

Database name *

Select...

Credential *

Microsoft Purview MSI (system)



Before you set up your scan you must give the managed identity of the Microsoft Purview account permissions to connect to your Azure SQL Database. [See more](#) ✓

Lineage extraction (preview) ⓘ

☒ On

ⓘ To successfully turn on Lineage extraction, you must do the following:

- Provide the db_owner role in Azure SQL Database for Microsoft Purview MSI
- Run "create Master Key" in Azure SQL Database (only if not already exists)

Scan charges apply. [Learn more](#) ↗

Select a collection

DP300test > LATAM

ⓘ All assets scanned will be included in the collection you select.

Continue

 Test connection

Cancel

4. Provide a name for the scan, select **Enter manually** for **Database selection method** property, enter the **Database name**, and select the **Credential**. Choose the appropriate collection for the scan, and select **Test connection** to validate the connection. If the connection is successful, select **Continue**.

Scan "AzureSqlDatabase-N6s"

Name *

Scan-yY6

Connect via integration runtime * ⓘ

Azure AutoResolveIntegrationRuntime

Server endpoint *

sqlserverdatascans1rg1ause.database.windows.net

Database selection method

☐ From Azure subscription ☒ Enter manually

Database name *

TestDB2

Credential *

cred-azsql

Select a collection

adpurview > Europe

ⓘ All assets scanned will be included in the collection you select.

Continue

🔗 Test connection

Cancel

Scope and run the scan

To scope and run the scan, follow these steps:

1. You can scope your scan to specific database objects by choosing the appropriate items in the list.

Scope your scan

 Refresh

All future assets under a certain parent will be automatically selected if the parent is fully or partially checked.

- ☒   AzureSqlDatabase-N6s
- ☒  dbo.customclassifications
- ☒  dbo.supportedclassifications

Continue

Back

Cancel

2. Select a scan rule set. You can choose between the system default, existing custom rule sets, or create a new rule set inline.

Select a scan rule set

 New scan rule set  Refresh

Select one scan rule set to be used by your scan.



AzureSqlDatabase SYSTEM DEFAULT

Microsoft default scan rule set that includes all supported system classification rules
[View detail](#)

Continue

Back

Cancel

3. Select **New scan rule set**, and provide a new scan rule set name.

New scan rule set

Source Type *

Azure SQL Database



Scan rule set name *

azsql-scanrule

Scan rule description

Continue

Cancel

4. You can then select the classification rules to be included in the scan rule, and then select **Create**.

Select classification rules

Choose classification rules that will run on the dataset.

System rules

☐	> Government
☒	> Financial
☐	▼ Personal
☒	Age of an individual
☒	Date of Birth
☒	Email Address
☒	Ethnic Group
☒	EU GPS Coordinates
☒	EU Mobile Phone Number
☒	EU Phone Number
☒	Geolocation (Lat/Lon)
☒	Japanese My Number – Corporate
☐	Japanese My Number – Personal
☐	Person's Name
☐	Personal IP Address
☐	U.S. Phone Number
☒	> Security
☒	> Miscellaneous

Create

Back

195 classification rules selected

Cancel

5. The **Select a scan rule set** page will the scan rule set you've created.

Select a scan rule set

[+ New scan rule set](#) [Refresh](#)

Select one scan rule set to be used by your scan.



AzureSqlDatabase SYSTEM DEFAULT

Microsoft default scan rule set that includes all supported system classification rules
[View detail](#)



azsql-scanrule

Scan any Azure SQL Database data. [View detail](#)

6. On the **Set a scan trigger** page, configure your scan trigger. Select **Continue**.

Set a scan trigger

Set a scan trigger to run the scan at specific dates and times. If once, the scan will start after set up is completed. If recurring, the scan will start at a date and time you choose. The initial scan is a full scan and every subsequent scan is incremental.

☒ Recurring ☐ Once

Time zone * ⓘ

(UTC) Coordinated Universal Time

Recurrence *

Every 1 Month(s)

☒ Month days ☐ Week days

Select day of the month to scan

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	Last			

Schedule scan time (UTC)

h:mm:ss AM

Start recurrence at (UTC) *

2021-09-08



9:42:00 AM

☐ Specify recurrence end date (UTC)

Continue

Back

Cancel

7. Review your scan, and then select **Save and run**.

Review your scan

Review your scan before running it.

Basics

Name	Scan-slf
Collection	Adatum Corp

Credential

Type	SQL authentication
Name	SQL-Canary

Scan scope

Scope	Full
-------	------

Lineage extraction

On

Scan rule set

Name	AzureSqlDatabase
Type	System

Scan trigger

Start at	Immediately
Recurrence	Once

Save and run | ▾

Back

Cancel

Data lineage

Generally, data lineage represents the journey the data takes from its origin to where it moves across the data estate over time. Among its many uses are troubleshooting, tracing the root cause in data pipelines, and debugging.

Microsoft Purview Data Catalog connects with other data storage, processing, and analytics platforms to collect lineage information. As a result, the Catalog contains a generic, scenario-specific lineage experience.

Microsoft Purview supports data lineage from Azure SQL Database. At the time of setting up a scan, you can enable lineage extraction toggle button to extract lineage information.

Prerequisites for setting up scan with Lineage extraction

1. Follow steps under authentication for a scan using Managed Identity section to authorize Microsoft Purview scan your Azure SQL Database.
2. Sign in to Azure SQL Database with Microsoft Entra account and assign proper permission (for example: **db_owner**) to Purview Managed identity. Use below example SQL syntax to create user and grant permission by replacing *purview-account* with your account name.

```
CREATE user <purview-account> FROM EXTERNAL PROVIDER
GO
EXEC sp_addrolemember 'db_owner', <purview-account>
GO
```

3. Run below command on your Azure SQL Database to create a master key.

```
CREATE MASTER KEY
GO
```

Create scan with lineage extraction toggle turned on

1. Enable lineage extraction toggle in the scan screen.

Scan "AzureSqlDatabase-eP4"

Name *

Scan-nYB

Connect via integration runtime * ⓘ

Azure AutoResolveIntegrationRuntime

Server endpoint *

sqlserver-east-jp.database.windows.net

Database selection method

☒ From Azure subscription ☐ Enter manually

Database name *

Select...

Credential *

Microsoft Purview MSI (system)



Before you set up your scan you must give the managed identity of the Microsoft Purview account permissions to connect to your Azure SQL Database. [See more](#) ✓

Lineage extraction (preview) ⓘ

☒ On

ⓘ To successfully turn on Lineage extraction, you must do the following:

- Provide the db_owner role in Azure SQL Database for Microsoft Purview MSI
- Run "create Master Key" in Azure SQL Database (only if not already exists)

Scan charges apply. [Learn more](#) ↗

Select a collection

DP300test > LATAM

ⓘ All assets scanned will be included in the collection you select.

Continue

 Test connection

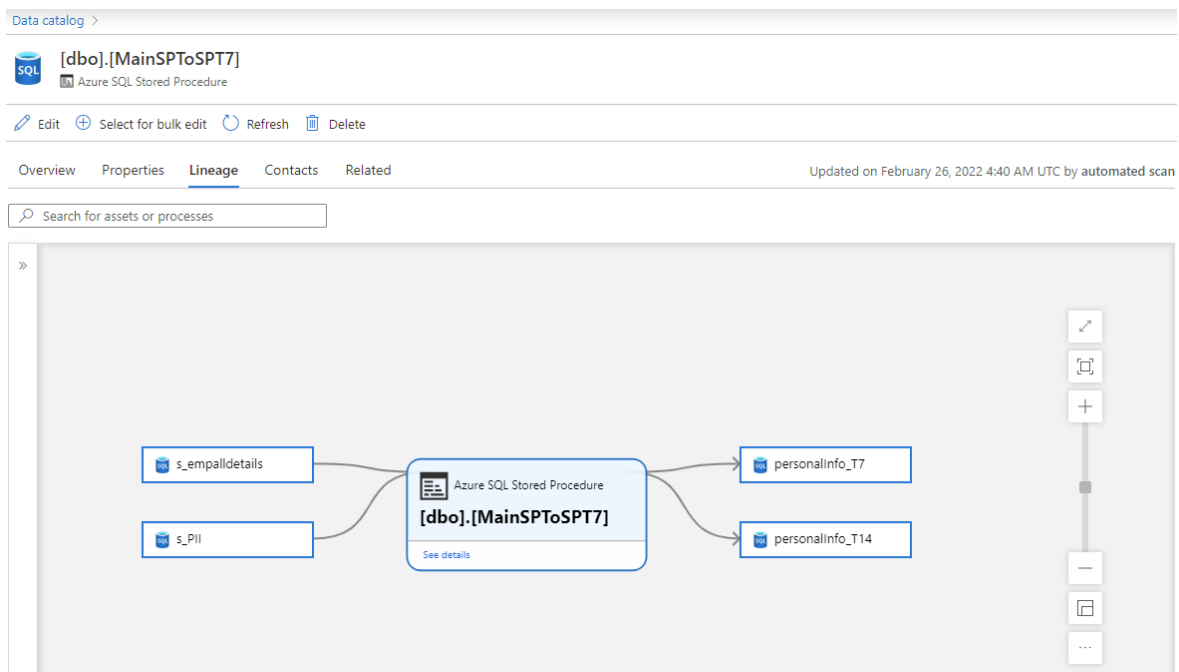
Cancel

2. Select your method of authentication by following steps in the scan section.
3. Once the scan is successfully set up from previous step, a new scan type called Lineage extraction runs incremental scans every 6 hours to extract lineage from Azure SQL Database. Lineage is extracted based on the actual stored procedure runs in the Azure SQL Database.

Search Azure SQL Database assets and view runtime lineage

You can browse the data catalog or search the data catalog to view asset details for Azure SQL Database following the steps below:

1. Go to asset -> lineage tab, you can see the asset lineage when applicable. Refer to the supported capabilities section on the supported Azure SQL Database lineage scenarios. For more information about lineage in general, see data lineage and lineage user guide



2. Go to stored procedure asset -> Properties -> Related assets to see the latest run details of stored procedures

Data catalog >

[dbo].[MainSPToSPT7]

Azure SQL Stored Procedure

Edit
Select for bulk edit
Refresh
Delete

Overview

Properties

Lineage

Contacts

Related

Filter by property key

Show properties without a value

Showing 4 of 9 properties

Properties

inputs

s_PII
s_empalldetails

outputs

personalInfo_T7
personalInfo_T14

qualifiedName

Related assets

Runs

[dbo].[MainSPToSPT7]

- Select the stored procedure hyperlink next to Runs to see Azure SQL Stored Procedure Run Overview. Go to properties tab to see enhanced run time information from stored procedure. For example: executedTime, rowcount, Client Connection, and so on

Data catalog >

[dbo].[MainSPToSPT7]

Azure SQL Stored Procedure

Edit
Select for bulk edit
Refresh
Delete

Overview

Properties

Lineage

Contacts

Related

Filter by property key

Show properties without a value

Showing 4 of 9 properties

Properties

inputs

s_PII
s_empalldetails

outputs

personalInfo_T7
personalInfo_T14

qualifiedName

mssql://sqlserv

Related assets

Runs

[dbo].[MainSPToSPT7]

[dbo].[MainSPToSPT7]

Azure SQL Stored Procedure Run

Edit
Select for bulk edit
Refresh
Delete

Overview

Properties

Contacts

Updated on February 28, 2022 4:40 AM UTC by automated scan

["Source":"mssql://sqlserverdatacanus2eup.database.windows.net/sqlineagedb2/dbo/s_empalldetails";"Sink":"mssql://sqlserverdatacanus2eup.database.windows.net/sqlineagedb2/dbo/personalInfo_T7";"ColumnMapping":{"Source":"credit_card_number";"Sink":"credit_card_number"};{"Source":"firstName";"Sink":"firstName"};{"Source":"us_city";"Sink":"us_city"}];["DatasetMapping":{"Source":"mssql://sqlserverdatacanus2eup.database.windows.net/sqlineagedb2/dbo/s_PII";"Sink":"mssql://sqlserverdatacanus2eup.database.windows.net/sqlineagedb2/dbo/personalInfo_T14";"ColumnMapping":{"Source":"Passport_Number";"Sink":"credit_card_number"};{"Source":"firstName";"Sink":"firstName"};{"DatasetMapping":{"Source":"mssql://sqlserverdatacanus2eup.database.windows.net/sqlineagedb2/dbo/s_empalldetails";"Sink":"mssql://sqlserverdatacanus2eup.database.windows.net/sqlineagedb2/dbo/personalInfo_T14";"ColumnMapping":{"Source":"australian_automobile_association";"Sink":"LastName"};{"Source":"us_city";"Sink":"us_city"}];

cpuTime

0

duration

12224

executedTime

Mon, 28 Feb 2022 01:07:47 UTC

inputs

s_PII
s_empalldetails

lastRunId

###5f5f-aa6a-bb7b-cc8c-dddd9d9d9d

nestLevels

2

numberOfResponseRows

0

outputs

personalInfo_T7
personalInfo_T14

qualifiedName

mssql://sqlserverdatacanus2eup.database.windows.net/sqlineagedb2/dbo/[MainSPToSPT7]#runs/00112233-4455-6677-8899-AA8BCCDDEEFF

queriesInStoredProcedure

2

queryHash

AA11B822CC33D044EE55FF66AA77888CC99DD00

queryText

Exec MainSPToSPT7

rowCount

20

username

dd5d3d3d-ee4e-ff5f-aa6a-bbbbbb7b7b7b@###5f5f-aa6a-bb7b-cc8c-dddd9d9d9d

Related assets

Client Connection

Core .Net SqlClient Data Provider:PRASADGMAGREE:eeeeeeee-###-aaaa-5555-666666666666

Stored Procedure

[dbo].[MainSPToSPT7]

Close

86 / 88

9. Exercise: Enable Microsoft Defender for SQL and Data Classification

<https://learn.microsoft.com/en-us/training/modules/implement-compliance-controls-sensitive-data/9-exercise-enable-advanced-data-security-classification>

Exercise: Enable Microsoft Defender for SQL and Data Classification

Completed

In this exercise, you'll learn how to enable Microsoft Defender for SQL and data classification.

Note

To complete this exercise, you'll need an [Azure subscription](#).

Launch the exercise and follow the instructions.

[Launch Exercise](#)

10. Module assessment

<https://learn.microsoft.com/en-us/training/modules/implement-compliance-controls-sensitive-data/10-knowledge-check>

Module assessment

Completed

11. Summary

Summary

Completed

This module explored the practices of setting compliance controls on the data that is stored within the SQL Server database. You also reviewed several security features available for Azure SQL, including the Microsoft Defender for SQL.

Now that you've reviewed this module, you should be able to:

- How data should be classified
- Why server and database audit are important
- How to implement row level security and dynamic data masking
- Understand the usage of Microsoft Defender for SQL
- How Ledger works
- Explore Azure Purview supported capabilities