

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

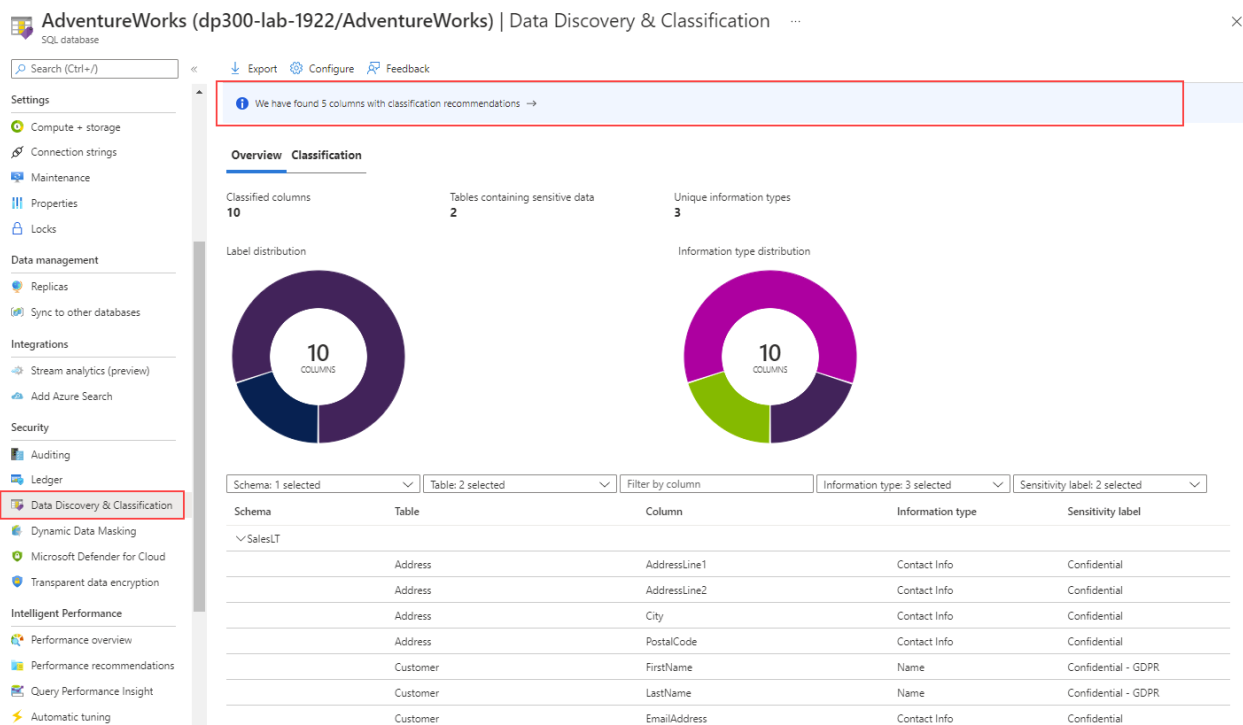
Completed

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')

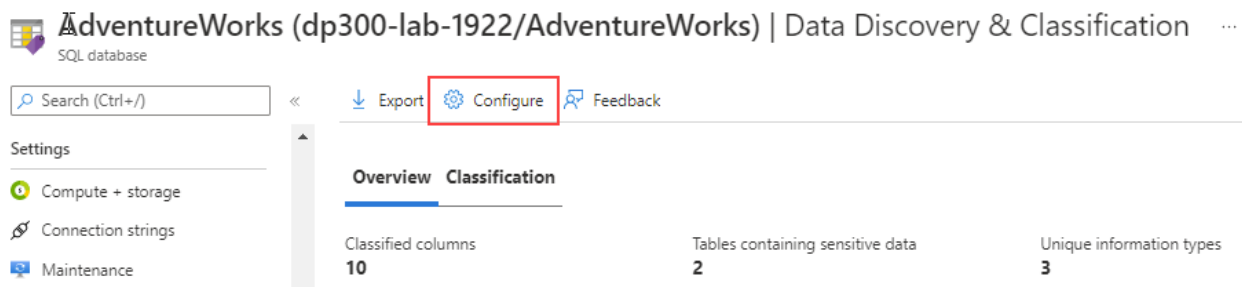
GO
```

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.

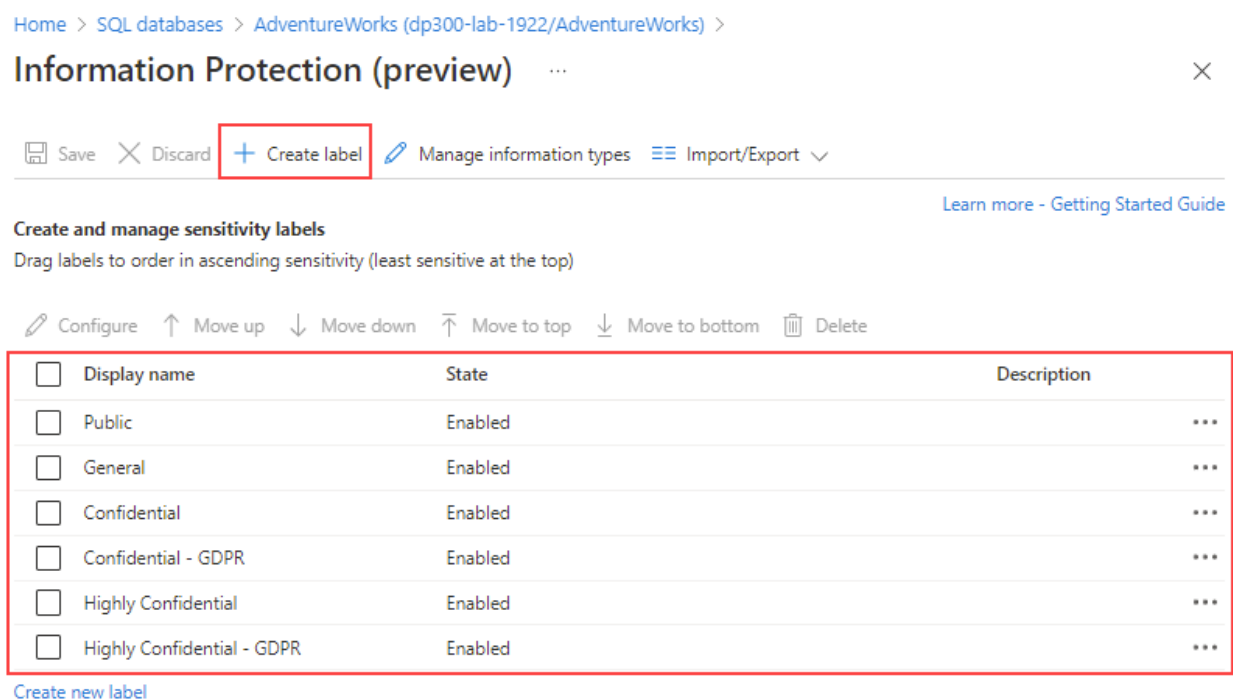
Customize classification taxonomy

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

Select what to mask
Schema

Table

Column

Select how to mask
Masking field format

^

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-",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

Microsoft Defender for Cloud

SQL server

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.

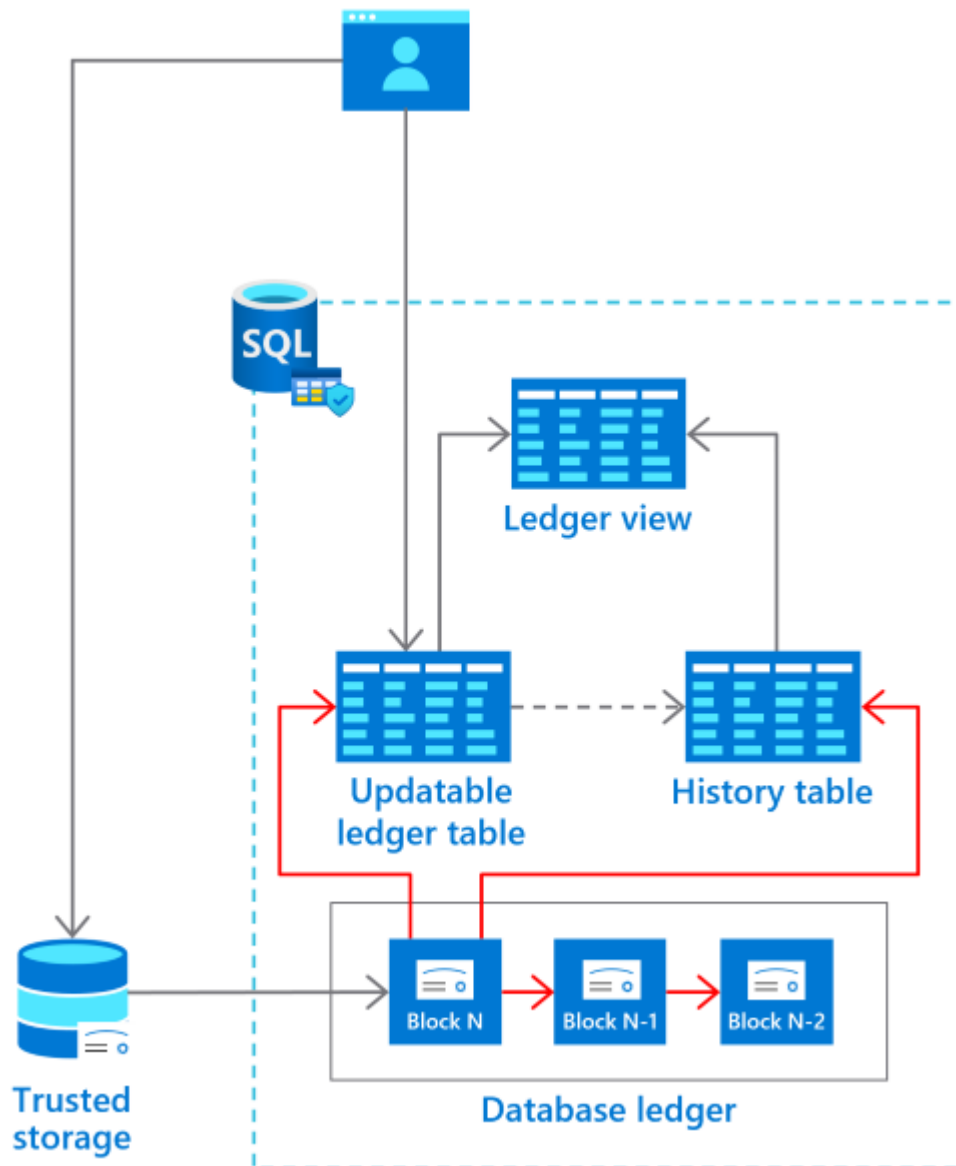
How it works

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

19 / 46

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

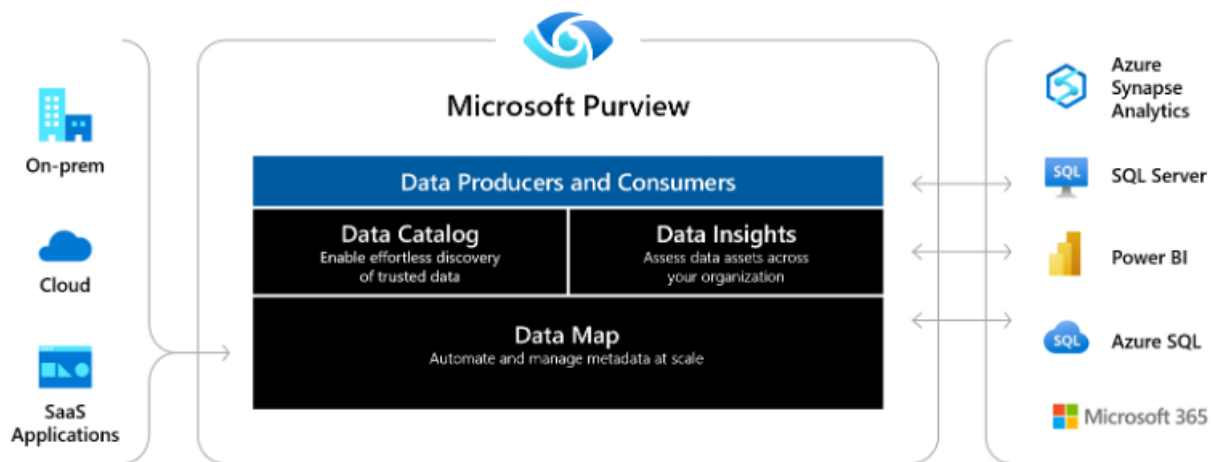
Implement Microsoft Purview

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.



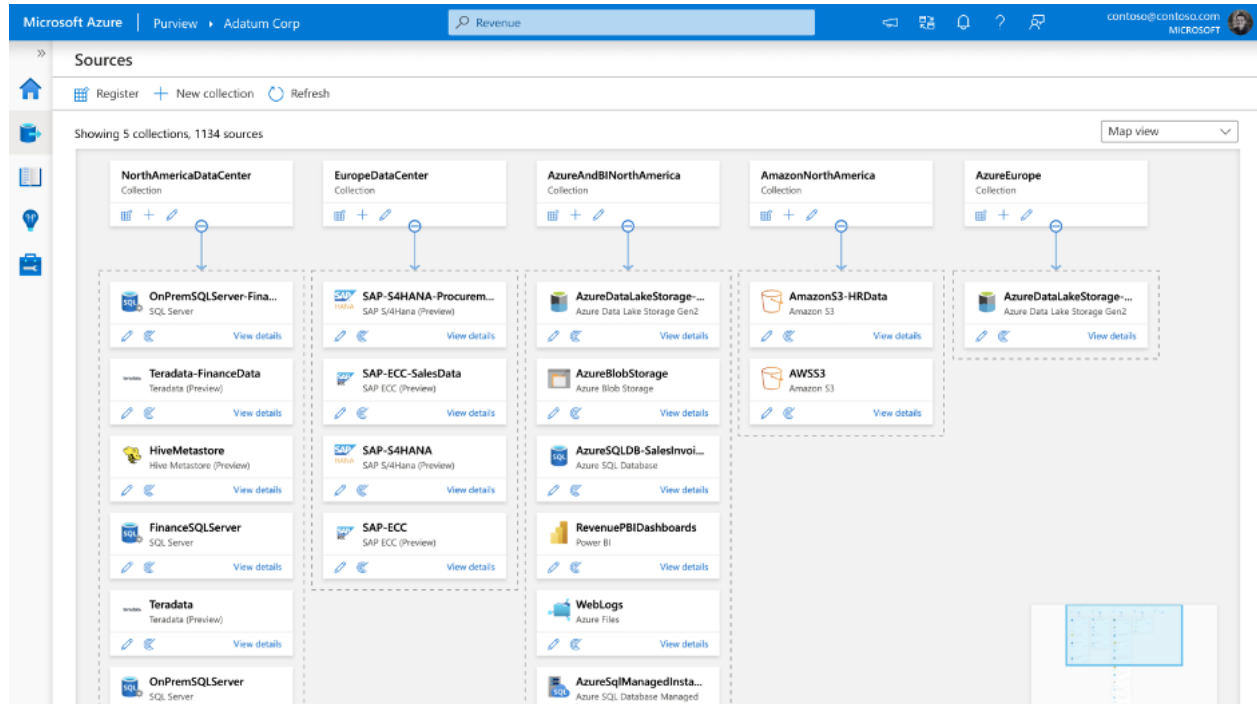
Supported capabilities

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:

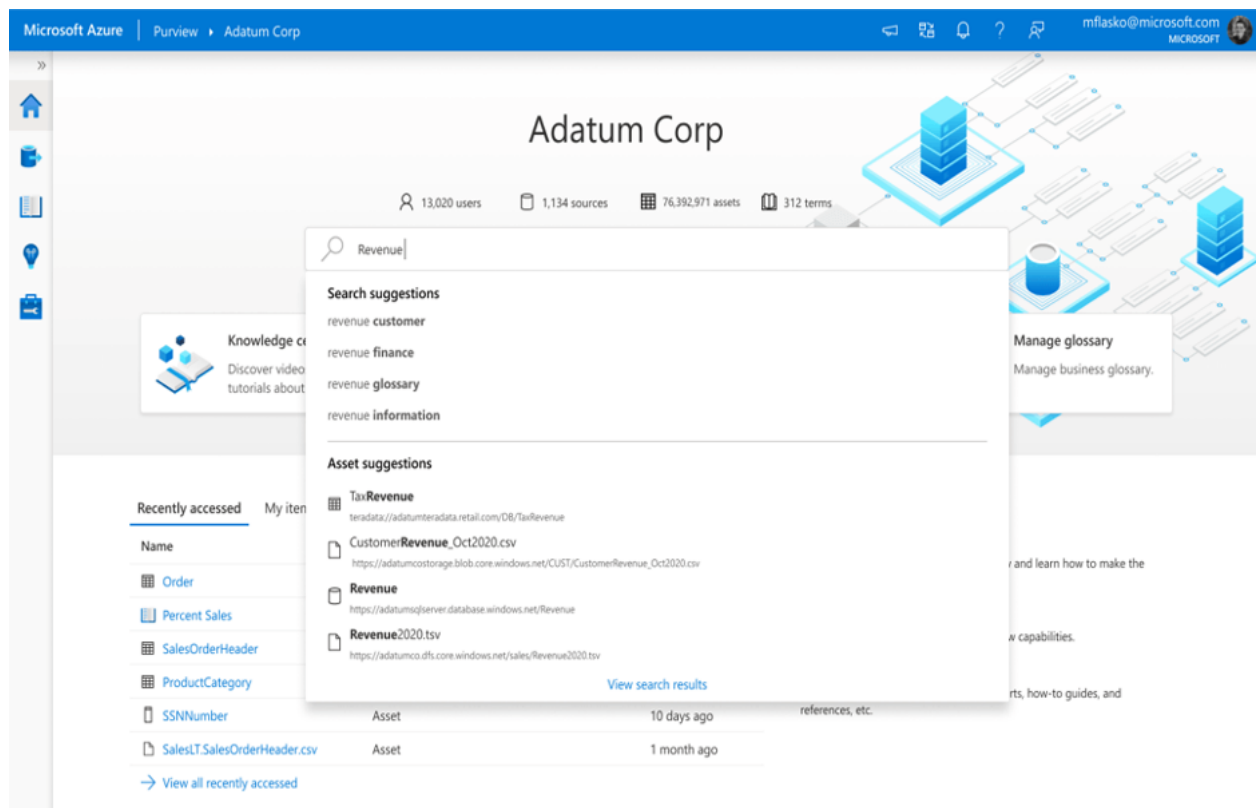
- Automate and manage hybrid resource metadata.
- Classify data using integrated and custom classifications and information protection sensitivity labels.
- Ensure consistent labeling of sensitive data across SQL Server, Azure, Microsoft 365, and Power BI.
- Easily integrate all your data systems using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



Make data easy to find

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

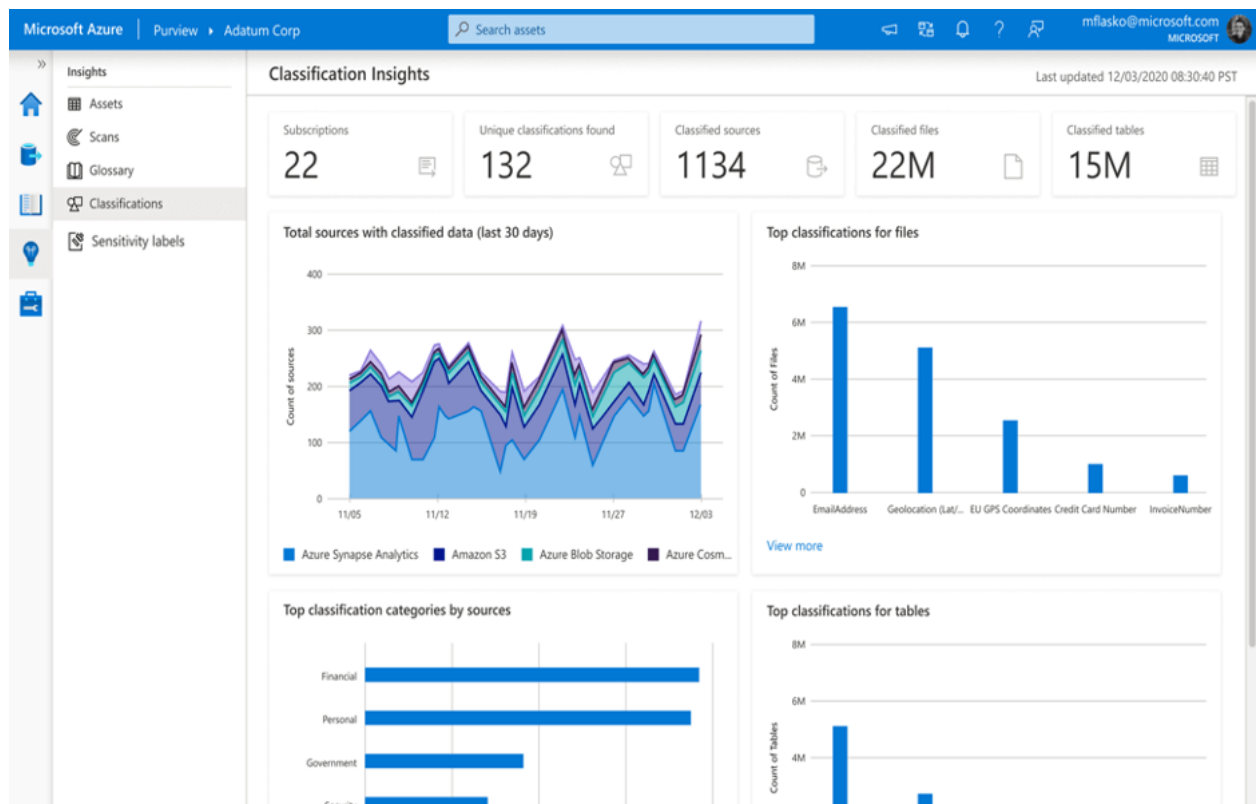
- Ensure optimal business value for your data users' data with Microsoft Purview Data Catalog.
- Eliminate the need for data dictionaries in Excel with a business-level business dictionary.
- Gain insight into the origin of your data with interactive visualization of data origin.
- Provide data scientists, 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.

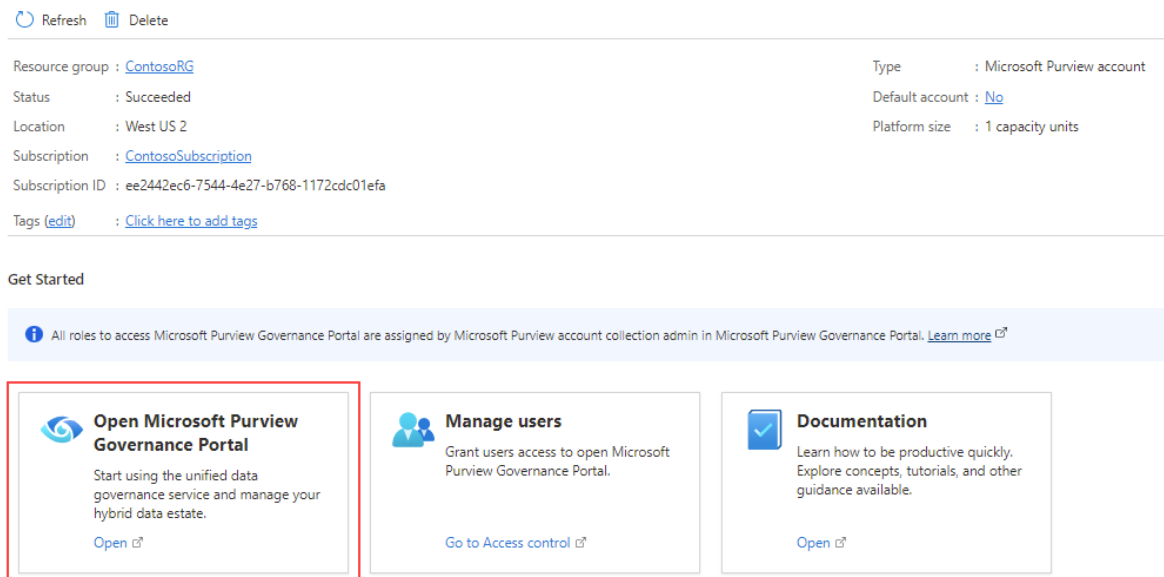
Register and scan SQL Database using Microsoft Purview

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

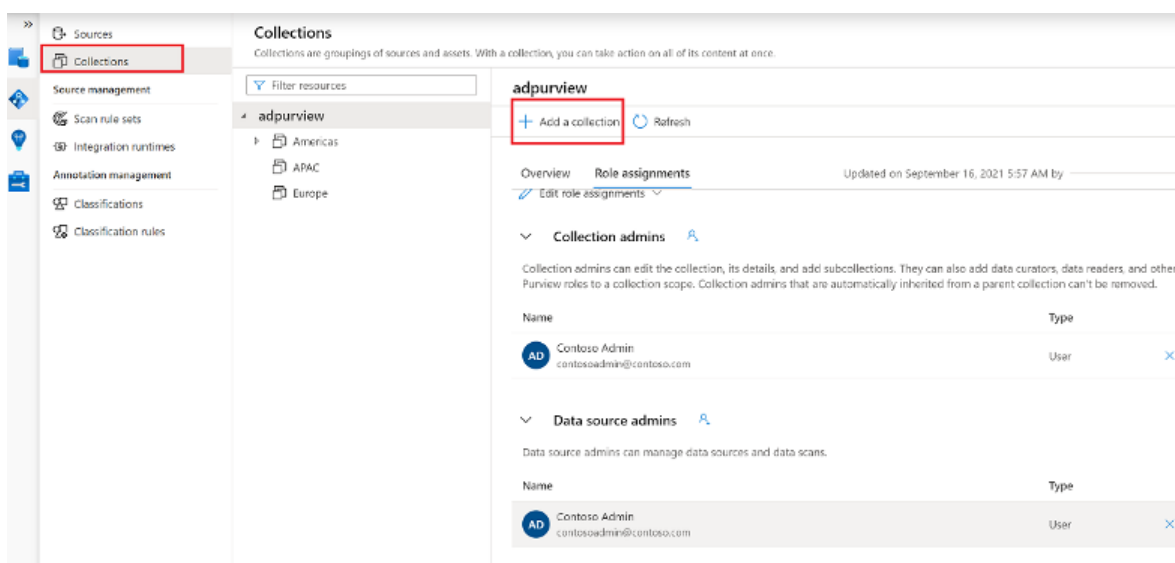
Register the data source

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

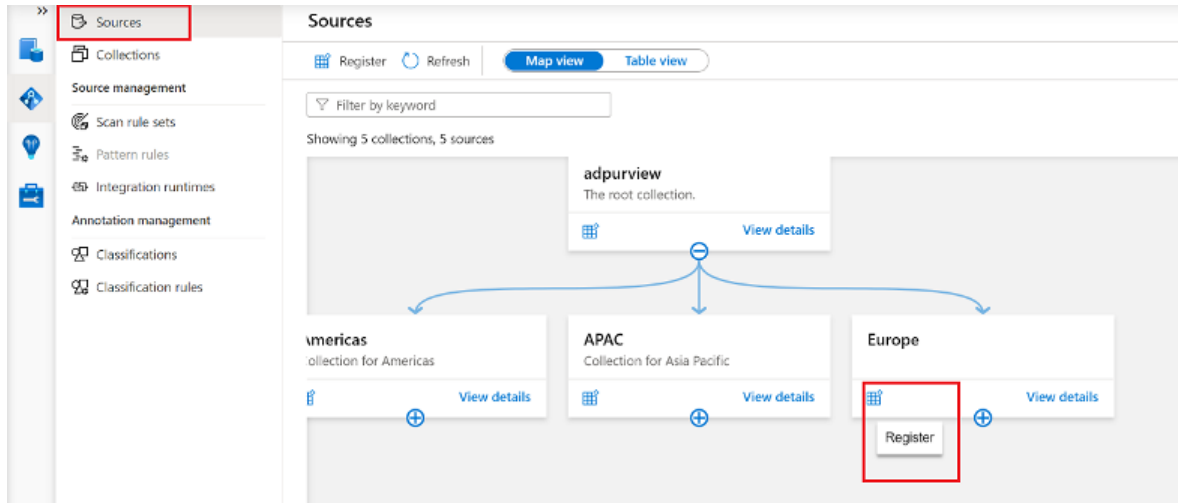
1. Open your Microsoft Purview account, and select **Open Microsoft Purview Governance Portal**.



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.

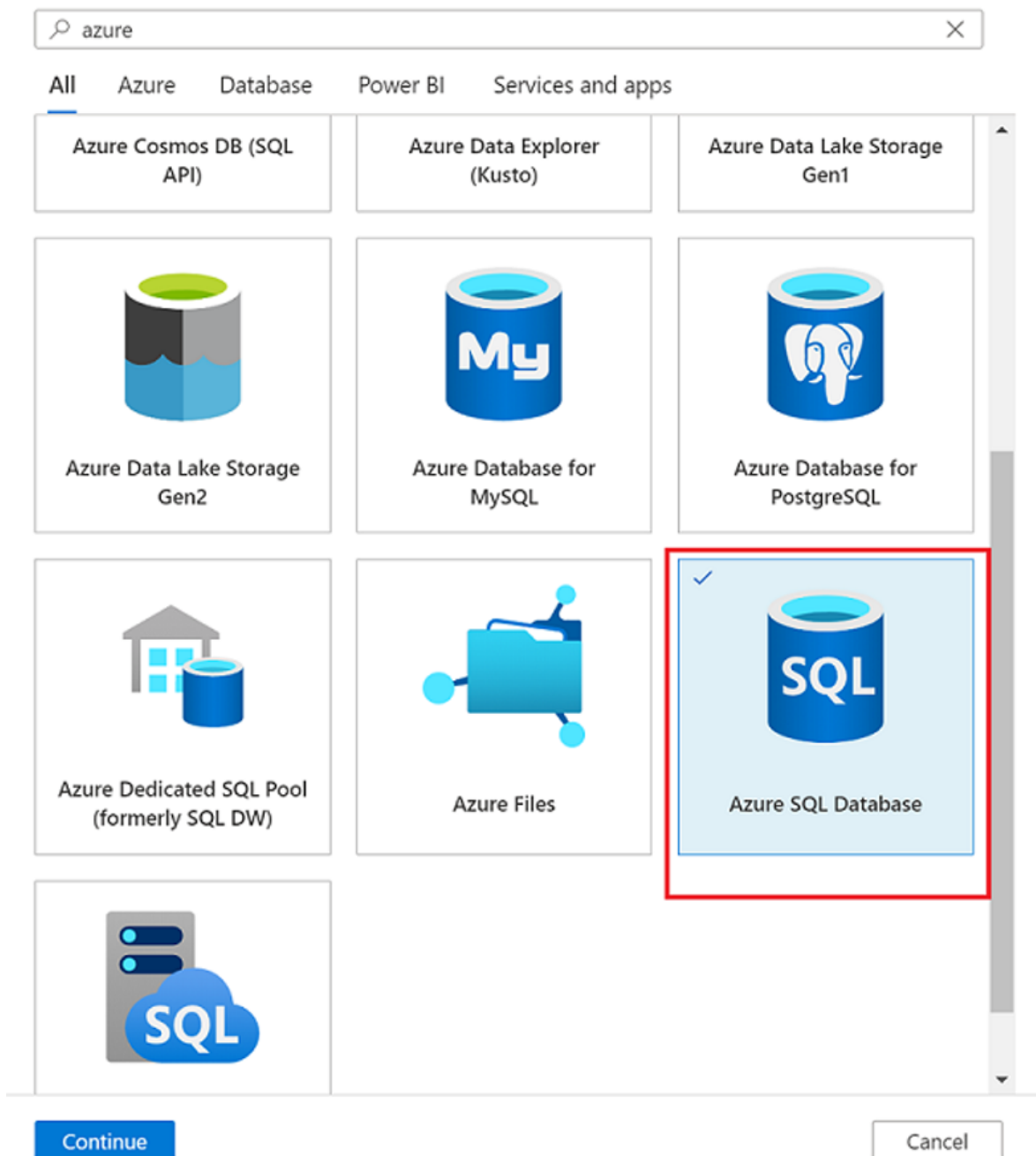


3. Navigate to the appropriate collection under the **Sources** menu, and then select **Register** to register a new SQL Database.



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 ▼

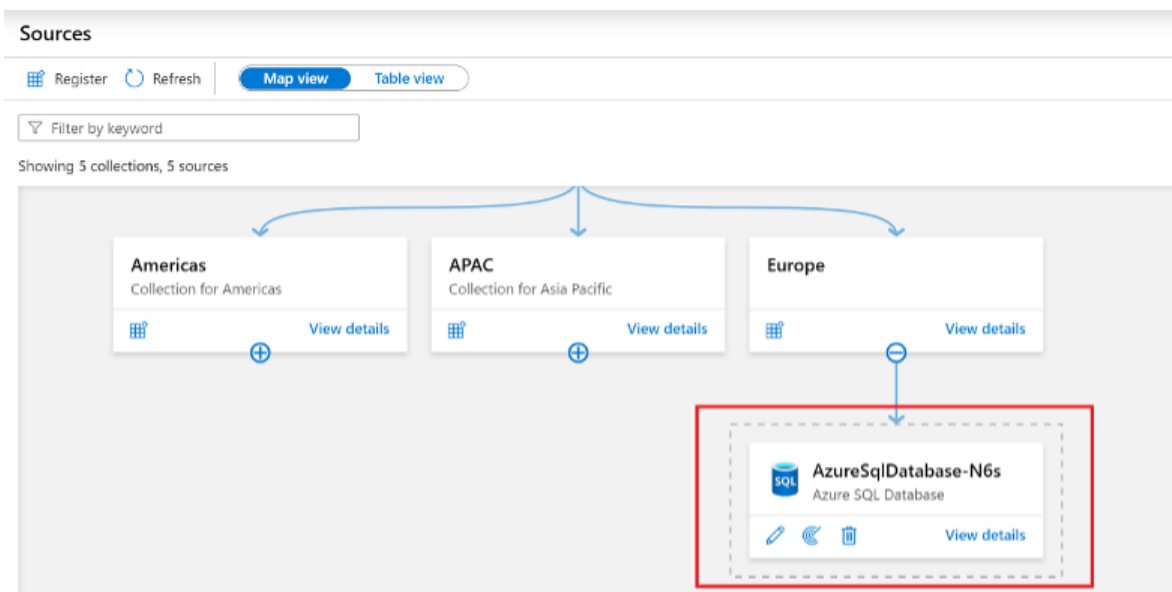
Endpoint
sqlserverdatascans1rg1ause.database.windows.net 

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

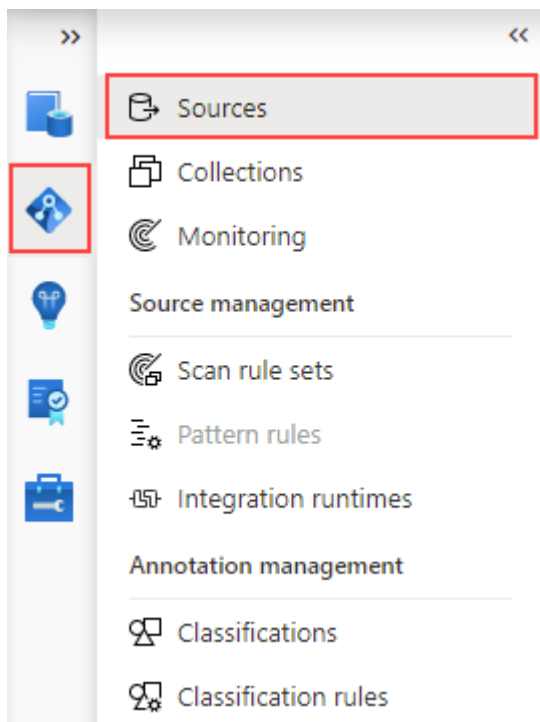
 All roles to access Microsoft Purview Governance Portal are assigned by Microsoft Purview account collection admin in Microsoft Purview Governance Portal. [Learn more](#) 

**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)

[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

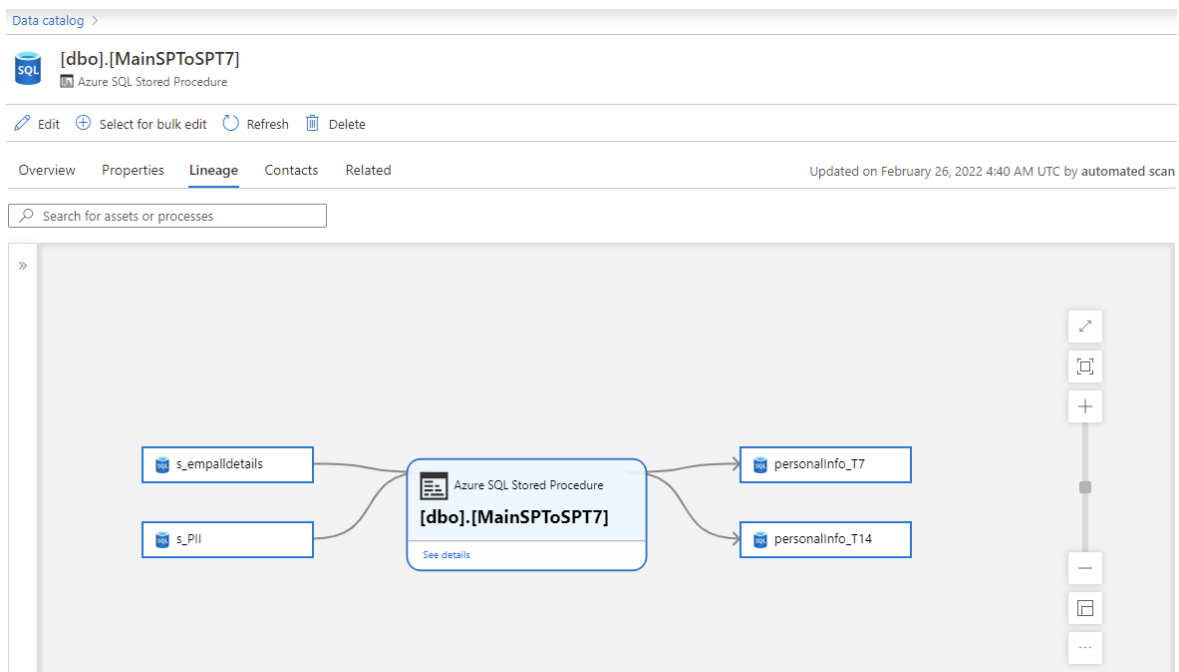
Cancel

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://sqlserverdatacanus2euap.database.windows.net/sqlineagedb2/dbo/s_empalldetails", "Sink": "mssql://sqlserverdatacanus2euap.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://sqlserverdatacanus2euap.database.windows.net/sqlineagedb2/dbo/s_PII", "Sink": "mssql://sqlserverdatacanus2euap.database.windows.net/sqlineagedb2/dbo/personalInfo_T14", "ColumnMapping": [{"Source": "Passport_Number", "Sink": "credit_card_number"}, {"Source": "firstName", "Sink": "firstName"}], "DataSetMapping": [{"Source": "mssql://sqlserverdatacanus2euap.database.windows.net/sqlineagedb2/dbo/s_empalldetails", "Sink": "mssql://sqlserverdatacanus2euap.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-ddddd9d9d9d

nestLevels

2

numberOfResponseRows

0

outputs

personalInfo_T7
personalInfo_T14

qualifiedName

mssql://sqlserverdatacanus2euap.database.windows.net/sqlineagedb2/dbo/[MainSPToSPT7]#runs/00112233-4455-6677-8899-AA8BCCDDEEFF

queriesInStoredProcedure

2

queryHash

AA11B822CC33D044EE55FF66AA77B888CC99DD00

queryText

Exec MainSPToSPT7

rowCount

20

username

dd5d3d3d-ee4e-ff5f-aa6a-bb7b-cc8c-ddddd9d9d9d

Related assets

Client Connection

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

Stored Procedure

[dbo].[MainSPToSPT7]

Close

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