Microsoft

Implement a secure environment for a database service

Introduction to database authentication, understand encryption solutions, understand data sensitivity and monitor database threats

Objectives

Understand the differences between Windows, SQL Server and Azure Active Directory Authentication

---

Describe and configure both data-at-rest encryption solutions as well as data-in-transit encryption solutions

---

Implement a data sensitivity solution

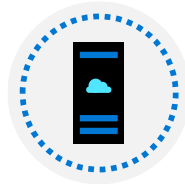# Configure database authentication and authorization

Objectives

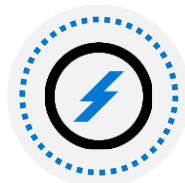Authentication options for Azure SQL Database

Security principals

Roles in Azure SQL Database

Understand permissions within Azure SQL

Understand the concept of least privilege

Azure AD authentication options

SQL Server authentication options

Windows Authentication
User login information is stored in Active Directory

SQL Server Authentication
User login information is stored in the Master or user database

Azure SQL Database and Managed Instance authentication options

Azure Active Directory Authentication
User information is stored in Azure Active Directory

SQL Server Authentication
User login information is stored in the master or user database

What's the difference between Active Directory and Azure Active Directory?

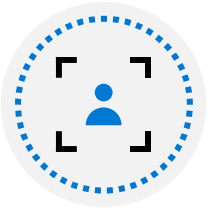|	Active Directory Domain Services	Azure Active Directory
User Management	Yes	Yes
Authentication	NTLM and Kerberos	OpenID Connect, SAML, OAuth
Groups	Yes	Yes
Object Hierarchy	Yes: X.500	Nope
Service Principals	Yes	Yes
Query AD programmatically	LDAP	AD Graph API (REST API)

Authentication and identities
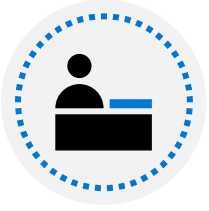
Authentication is the process of proving a user or service is who they say there are
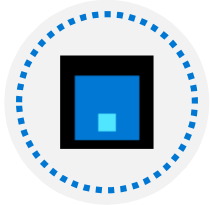
Authorization is a process that occurs after a user is authenticated, and grants them their specified access to resources

Identities can represent users, service principals, or computers

Azure Active Directory authentication

Allows integrated authentication into cloud native dbs - similar to Windows Authentication in traditional deployment of SQL Server

Multi-factor authentication is fully supported

To add Users with Azure Active Directory Authentication, you must first configure the AAD admin

Azure Active Directory admin configuration for Azure SQL Database

Security principals

Security principals are any login, user, group, or role within the server or database

Users within the databases are either mapped to a login, or contained users within the database

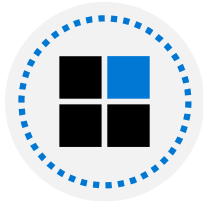Contained users can be based on SQL Authentication or Azure Active Directory

Users can then be mapped to roles in order to give users centrally managed rights, or rights can be granted directly to a user

SQL Server Security overview

Securables
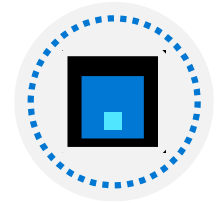Object to which access must be secured

Principals
Security identities (users, service principals, or computers) that access securables to perform actions

Permissions
Actions principals can perform on securables

Security Hierarchies
Securables can contain other securables, and principals can contain other principals (roles)

Schemas and securables
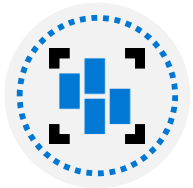
Securables are resources within databases like tables, views, procedures that access is granted to

Securable scopes: <server>.<database>.<schema>.<object>

Securables in Azure SQL Database only have the database and schema scopes

A schema is a collection of objects which allows objects to be grouped into separate namespaces
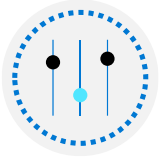
Logins and users

Logins are created in the **master** database and are used for server access

Instance level permissions are applied to logins

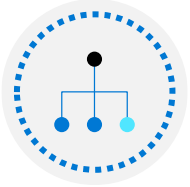Database level permissions are applied to users

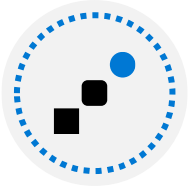Contained users are authenticated at the database

A user has access only to the database in which they are created

Built-in database roles

SQL Server and Azure SQL Database include several fixed roles within each database

Users may be added as members of one or more roles (including custom roles)

The Master database in Azure SQL Database has a couple of unique roles since the *sysadmin* role does not exist

Server roles cannot be granted access to objects within a database directly and are only available in SQL Server and Azure SQL Managed Instance, but not in Azure SQL Database

Fixed server roles
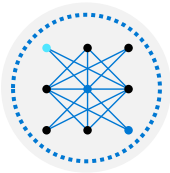
Sysadmin
Can perform any activity on the server

Serveradmin
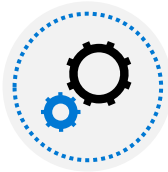Can change server-wide configuration settings and can shutdown the server

Securityadmin
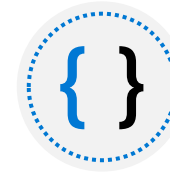Can manage logins and their properties.

Processadmin
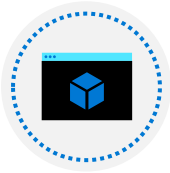Can kill processes running inside of SQL Server

Setupadmin
Can add and remove linked servers using T-SQL
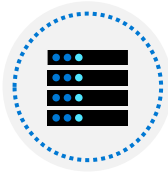
Bulkadmin
Can run the BULK INSERT T-SQL statement

Diskadmin
Can manage backup devices in SQL Server

Dbcreator
Can create, restore, alter, and drop any database

Public
Every SQL Server login belongs to the public user role.

Database roles in SQL Server vs. Azure SQL Database
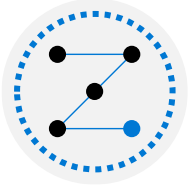
Roles are used to simplify the process of managing privileges in the database
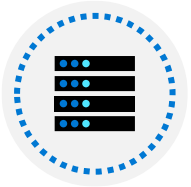
In SQL Server and Managed Instance the scope of the role may be the **database** or the **server**
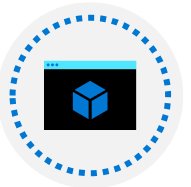
In Azure SQL Database roles are scoped to the individual **database**
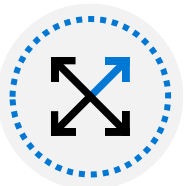
Both SQL Server and Azure SQL Database include built-in **database roles**, and allow for the creation of custom roles

Special Roles for Azure SQL Database

Database level roles available in the virtual master database only

---
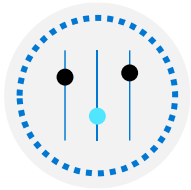
dbmanager – this role can create and delete databases. Equivalent to **dbcreator** fixed server role.

---

loginmanager – this role can create and delete logins in the virtual master database. Equivalent to **securityadmin** fixed server role.

Built-in Database Roles

db_owner
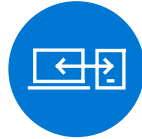
db_backupoperator

db_datareader

db_securityadmin

db_ddladmin

db_denydatawriter

db_accessadmin

db_datawriter

db_denydatareader

Database and object permissions explained
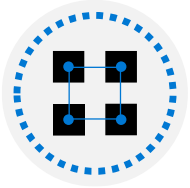
There are four **DML** permissions on tables and views – SELECT, INSERT UPDATE and DELETE

Stored Procedures and Functions have their own permissions – ALTER, CONTROL, EXECUTE, and VIEW DEFINITION

DCL permissions – GRANT, DENY, REVOKE
DDL permissions – CREATE, ALTER, DROP

Permissions which are REVOKED, remove any existing GRANT or DENY permission from the object

Database and object permissions explained
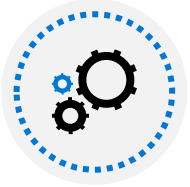
Permissions can be assigned to users or roles within a database
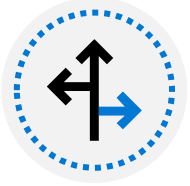
Users may then be assigned to roles

Permissions are additive, with permissions from multiple role memberships applied together

Preventing access through a DENY will override any GRANT to that object

GRANT / DENY example

```sql
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

EXECUTE AS USER / EXECUTE AS LOGIN definition

EXECUTE AS USER and EXECUTE AS LOGIN allows a statement to be executed in the security context of another user or login
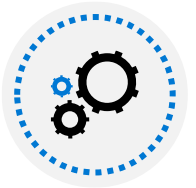
This capability allows for **testing** during the development process to ensure permissions are correctly implemented

Ownership chains explained
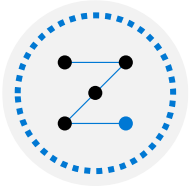
Used when access to an object is needed to complete a task against another object
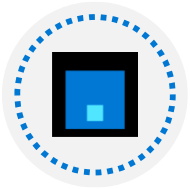
Allows a user to execute a stored procedure without the user needing right to the tables which the stored procedure uses

Permission chaining **only applies to objects with the same owner**
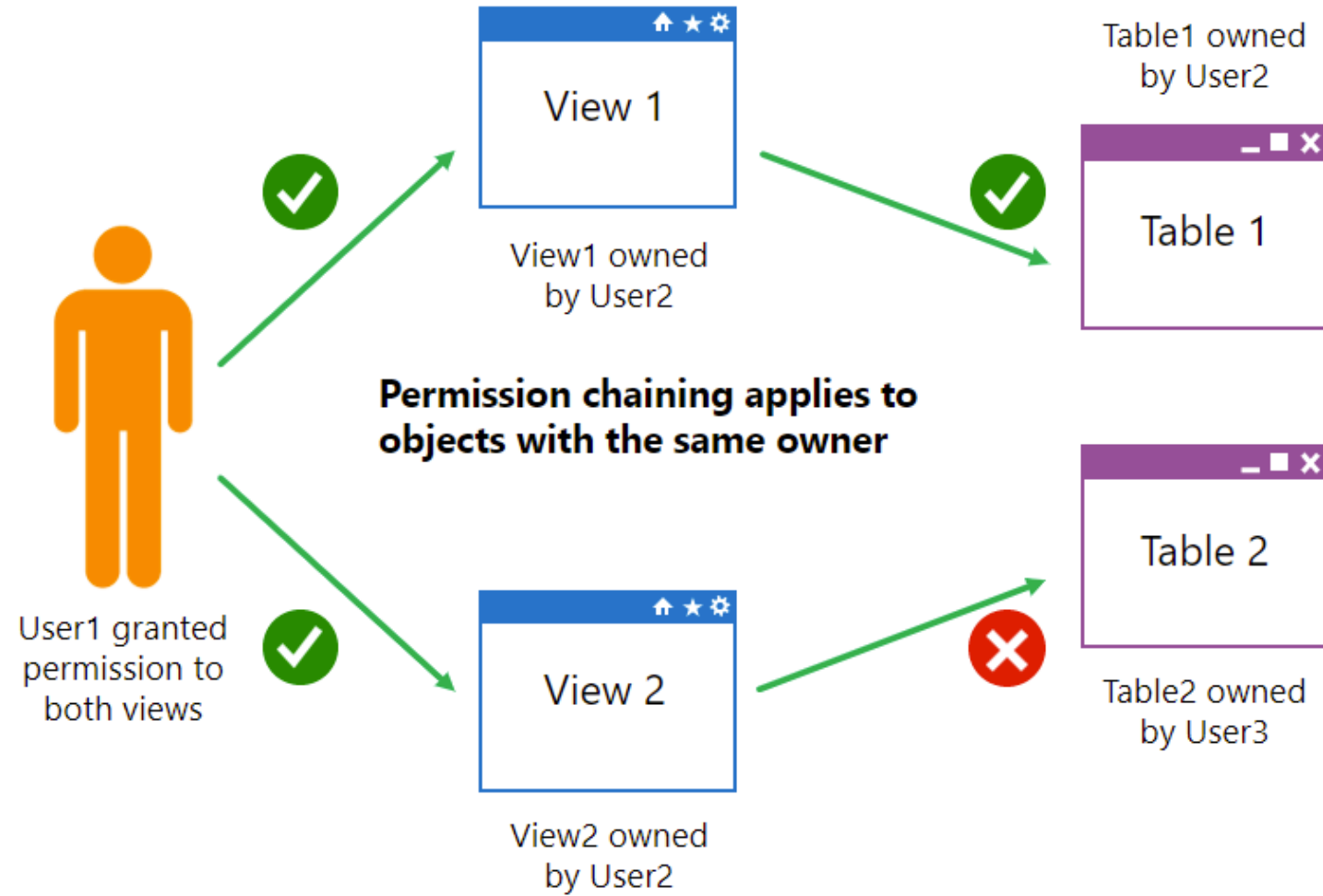
Dynamic SQL

Dynamic SQL is a concept where queries are built programmatically and then executed in-line.

The sample on the right would generate statements to back up each database on a server.

```sql
SELECT 'BACKUP DATABASE ' + name +
' TO DISK =''\\backup\sql1\' +
name + '.bak'''

FROM sys.databases
```

Least privilege

The concept of "Least Privilege" means that users never have more permissions than they need to get the task done
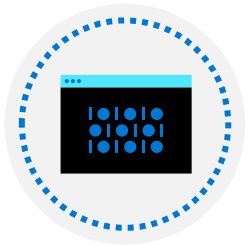
If the user runs an application and the user only needs to use stored procedures, then the user should only have the EXECUTE permission for the stored procedure

Identify authentication and authorization failures

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.

Instructor led labs: Authorize Access to Azure SQL Database with Azure Active Directory

Create users

Manage access to database objects

Validate access

Protect data in-transit and at rest

Objectives

Understand the data encryption options available in the various platforms

Understand how to configure data at rest encryption for Microsoft SQL Server
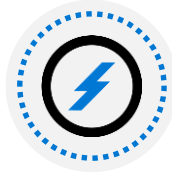
Understand how to configure SQL Server to use Azure Key Vault

Understand the difference between database and instance firewalls in Azure SQL Database

Understand what Dynamic Data Encryption is used for and how to configure it

Encryption at rest

Encryption at rest protects data files, transaction log files, and backup files by requiring a certificate to bring them online

SQL Server and Azure SQL Database implement this through a feature called Transparent Data Encryption
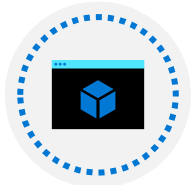
You can also encrypt your Azure VM disks to provide an additional level of protection beyond TDE for your SQL Servers

Encrypt drives before you write sensitive data

Data at rest vs. Data in transit

Encrypt Data

Encrypt Data

Private Network
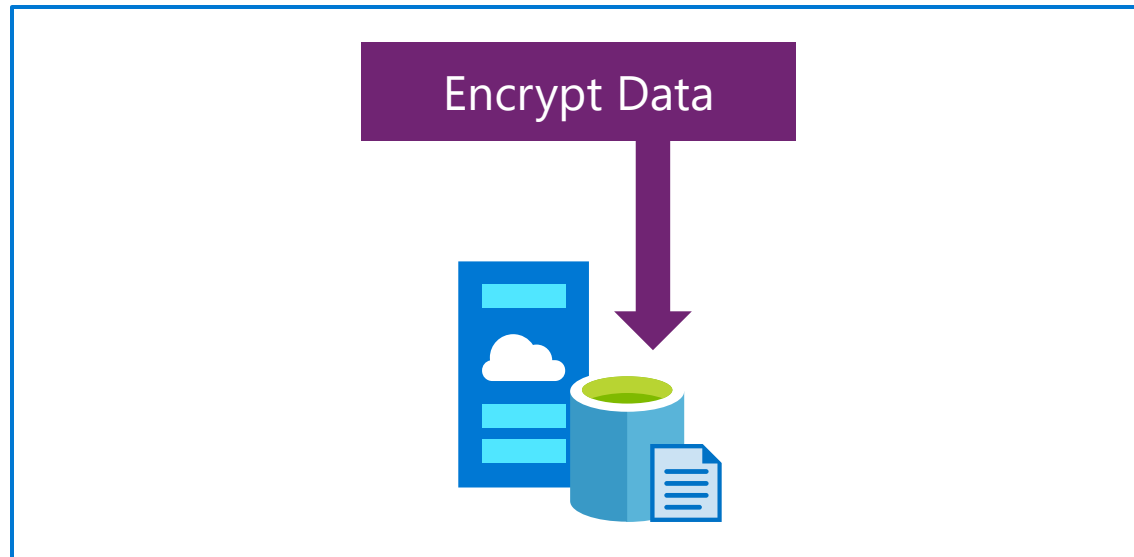
Public Network

Data at rest

Encrypts data while it's on file storage
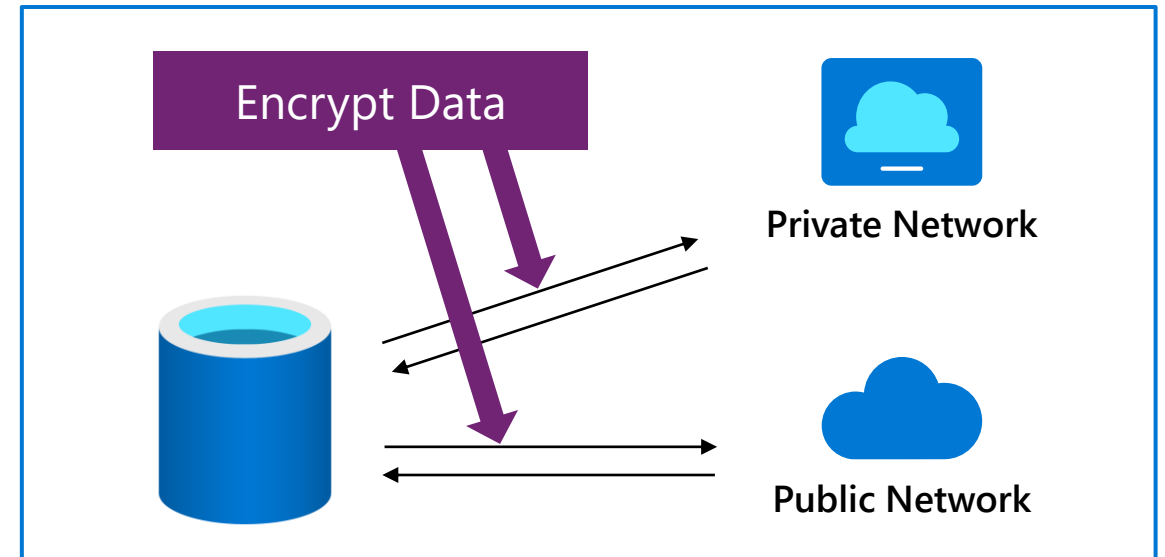
Data in transit

Encrypts data while it travels through private or public network communication channels

Transparent Data Encryption

Transparent Data Encryption (TDE) works by encrypting data in SQL Server and Azure SQL Database on a page level as the page is written to the disk.

- TDE is enabled by **default**

- As data is read from the disk to the buffer pool, it is decrypted before being written to the buffer pool

- Decrypted data is passed to the query processor for joining and returning to the user

- Data is protected as it rests on the disks, and within the backups. Uses symmetric key called the **database encryption key**

Transparent Data Encryption Architecture

Transparent Database Encryption Architecture

Windows Operating System Level
Data Protection API (DPAPI)

DPAPI encrypts the Service Master Key.

SQL Server
Instance Level — Service Master Key — Created at time of SQL Server setup.

Service Master Key encrypts the Database Master Key for the master database.

master
Database Level — Database Master Key — Statement: CREATE MASTER KEY ...

Database Master Key of the master database creates a certificate in the master database.

Statement: CREATE CERTIFICATE ...

The certificate encrypts the Database Encryption Key in the user database.

User Database Level — Database Encryption Key — Statement: CREATE DATABASE ENCRYPTION KEY ...

The entire user database is secured by the Database Encryption Key (DEK) of the user database by using transparent database encryption

Statement: ALTER DATABASE ... SET ENCRYPTION ON

Enabling TDE

```sql
SQL

USE master;
GO

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Pa55.w.rd';
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
```
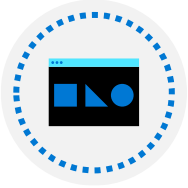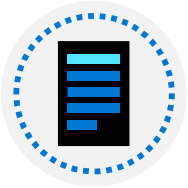
Managing TDE

Always backup the certificate that is created in the master database

You will not be able to restore this database to this, or another server without first restoring the certificate

Without a certificate, you will not be able to restore or read the database from backup or attach the data or log files

When setting up a database within an Availability Group, restore this certificate to each server within the Availability Group

Configure server and database firewall rules

Each Azure SQL Database maps to a public IP address which is hosted by Microsoft Firewalls are designed to prevent people from accessing resources that they should not be accessing

By default all access should be blocked with access opened as needed

In Azure SQL Database there are firewalls at the server level as well as at the database level
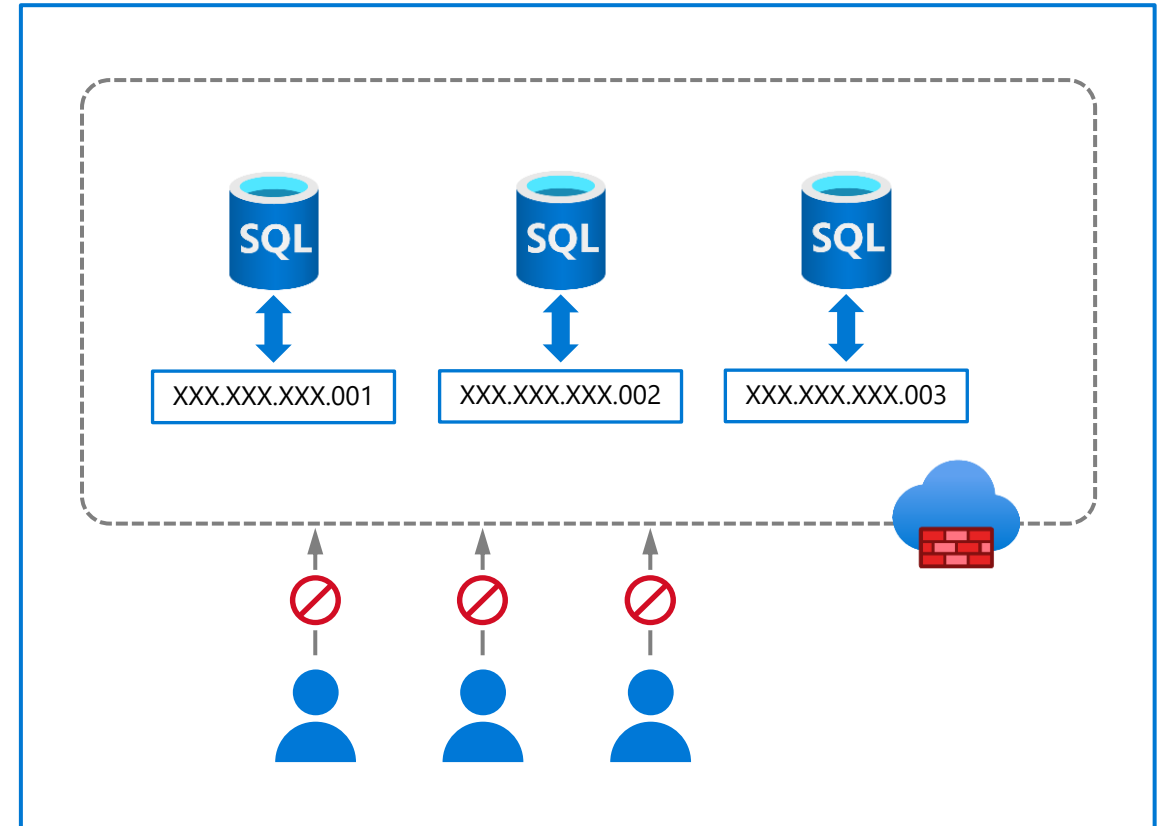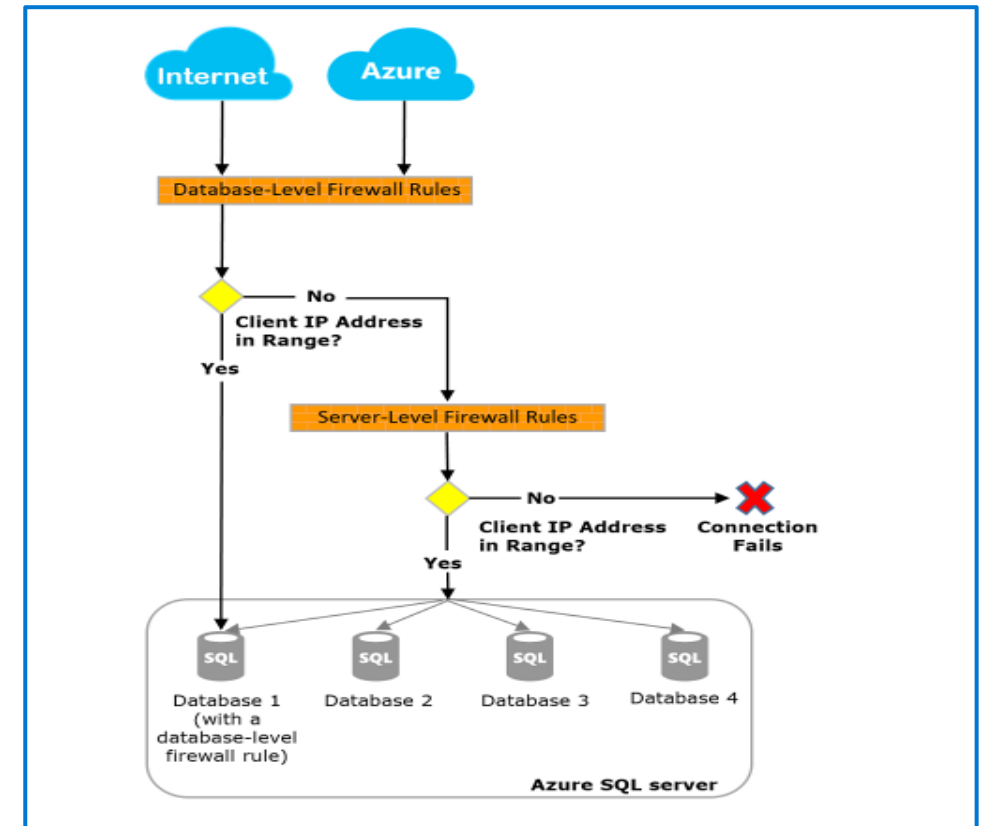
Server and database firewall rules

Connection attempts from the internet and Azure must pass through the firewall before they reach your server or database.

Server Level - sp_set_firewall_rule, sp_delete_firewall_rule

DB Level - sp_set_database_firewall_rule, sp_delete_database_firewall_rule

Virtual network endpoints

Network Peering allows connectivity from within a Virtual Network to Azure PaaS services including Azure SQL Database

Network Endpoint peering is limited to one Azure region

Network Endpoint peering allows only server level connections, not a database one

Requires outbound access to Azure SQL Database Public IP addresses

Private link

Private link allows you to connect various Azure services (including Azure SQL DB) to a private endpoint

A private endpoint is a private IP address within a specific virtual network subnet
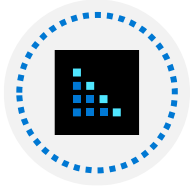
This allows for you to use network rules to prevent data exfiltration

Can route directly to Azure SQL DB over ExpressRoute or Point to Point VPN without traversing the public internet

Object level encryption

While TDE encrypts the data files and the backup files, it does not encrypt the data within tables

SQL Server uses **Always Encrypted** to encrypt table data, so that only the application with the correct column key can decrypt the data

This protects data from being exfiltrated (unauthorized data transfer) by a malicious administrator

Always Encrypted benefits

Encrypts data within tables, protects the data:

- As it **rests** on the disk
- As it is **in flight** between the SQL Server and the client

Allows for data within the database to be encrypted without the database engine ever seeing plain text data

Data is encrypted with certificates created by the database, but stored in the application

Even administrators cannot decrypt the encrypted data

Always Encrypted encryption types

Deterministic

Should be used with data that has many distinct values

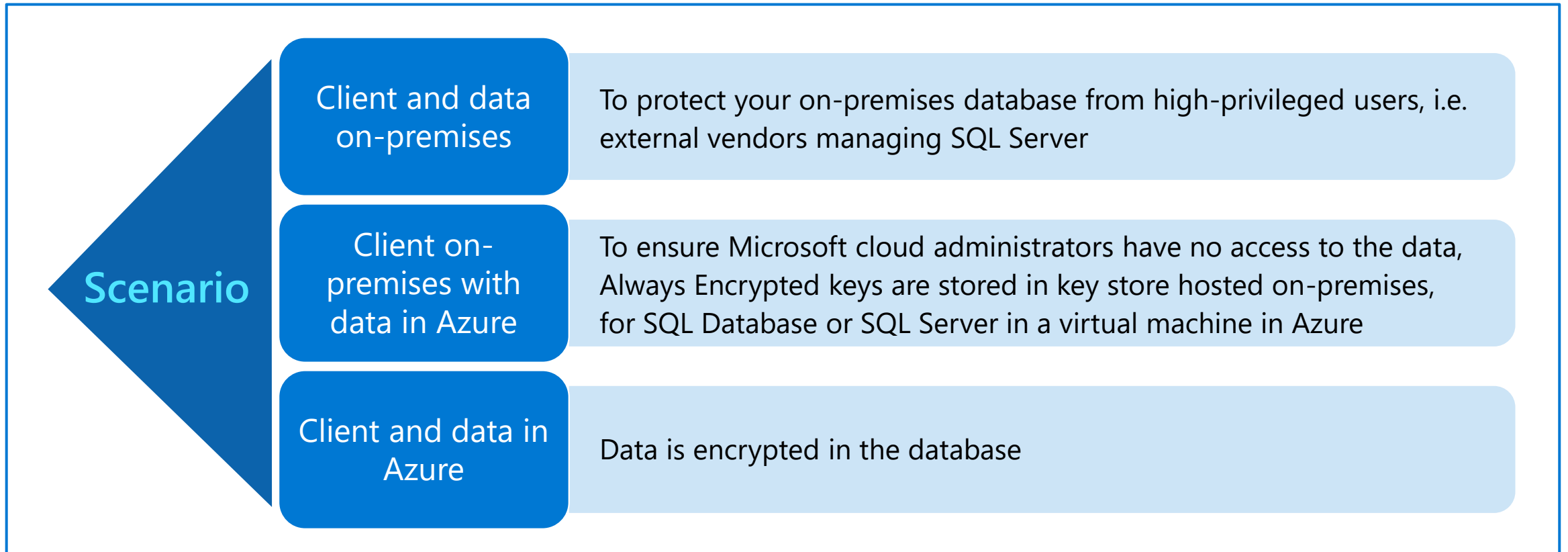**Allows equality joins, grouping and indexing on encrypted columns**

Randomized

Most secure

Good for columns with few distinct values

Prevents searching, grouping, indexing, joining on encrypted columns and equality operations
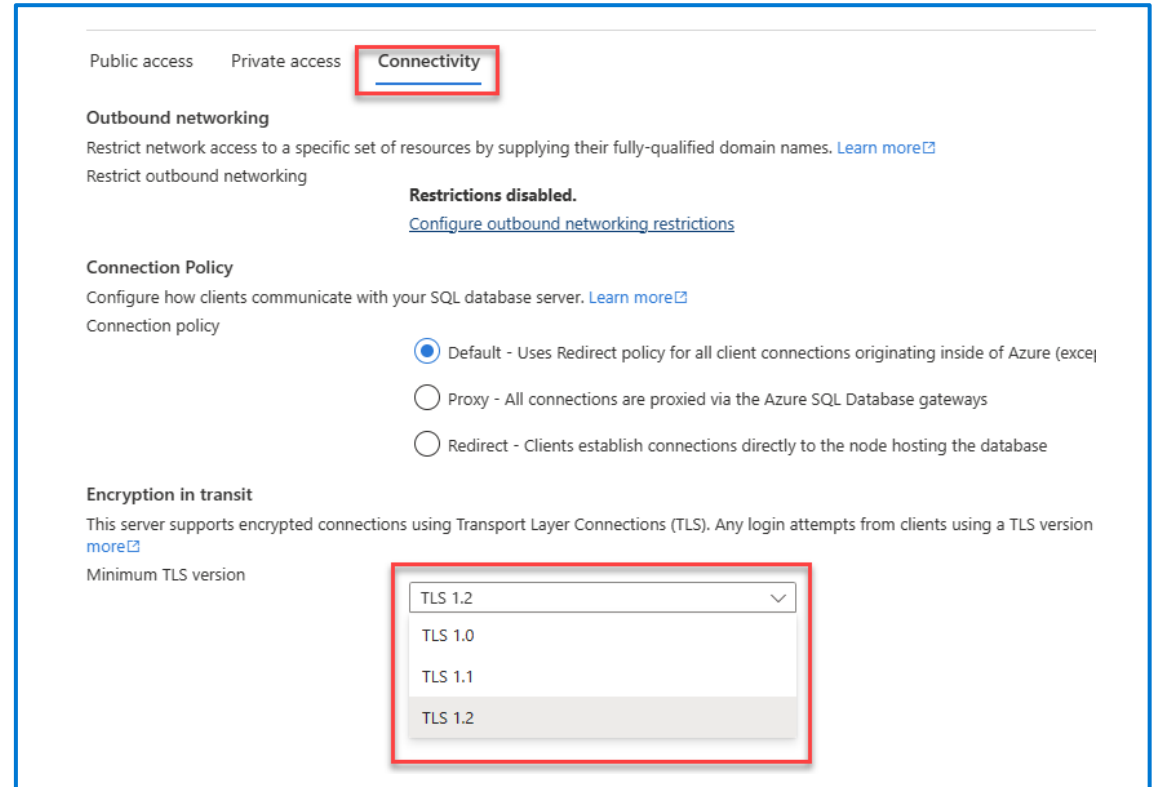
Always Encrypted use cases

Scenario		
Client and data on-premises	To protect your on-premises database from high-privileged users, i.e. external vendors managing SQL Server
Client on-premises with data in Azure	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 in a virtual machine in Azure
Client and data in Azure	Data is encrypted in the database

What are secure enclaves?

Added to SQL Server 2019, secure enclaves provide a secure protected area of server memory to decrypt data and perform calculations

Allows for even randomized encrypted data to be compared in queries that do pattern matching or range comparison
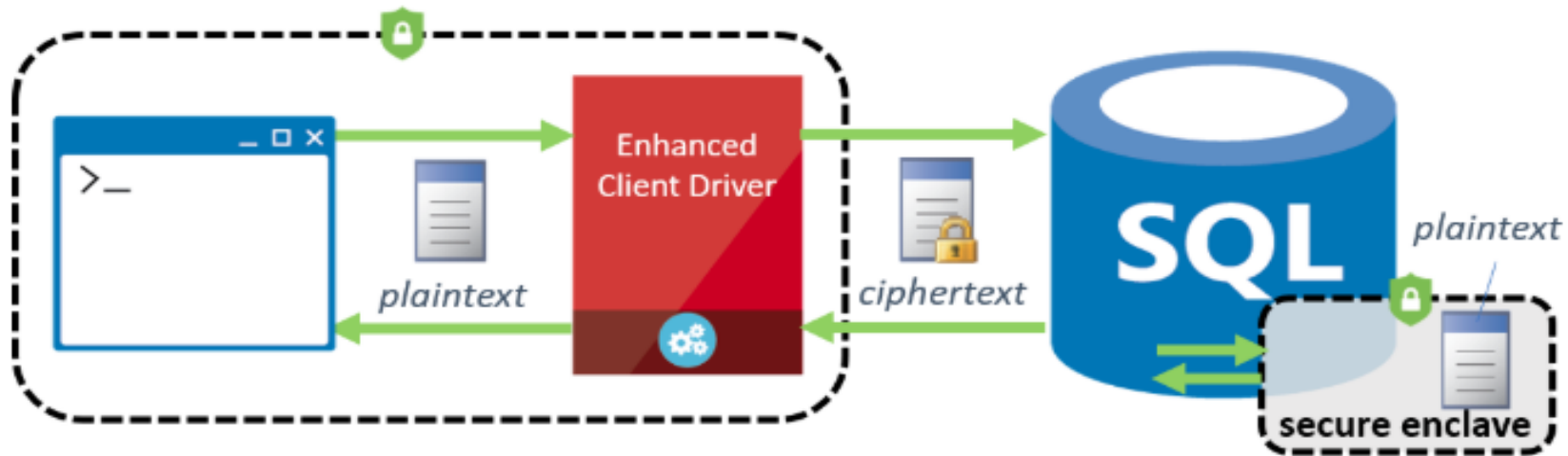
Enable encrypted connections on Azure SQL Database

Transport Layer Security (TLS) encryption is performed at the protocol layer and is available to all supported SQL Server and Azure SQL database services

Public access   Private access   **Connectivity**

Outbound networking
Restrict network access to a specific set of resources by supplying their fully-qualified domain names. Learn more⬈
Restrict outbound networking

Restrictions disabled.
Configure outbound networking restrictions

Connection Policy
Configure how clients communicate with your SQL database server. Learn more⬈
Connection policy

◉ Default - Uses Redirect policy for all client connections originating inside of Azure (exce

○ Proxy - All connections are proxied via the Azure SQL Database gateways

○ Redirect - Clients establish connections directly to the node hosting the database

Encryption in transit
This server supports encrypted connections using Transport Layer Connections (TLS). Any login attempts from clients using a TLS version
more⬈
Minimum TLS version

TLS 1.2   ⌄

TLS 1.0

TLS 1.1

TLS 1.2

Enable encrypted connections on SQL Server
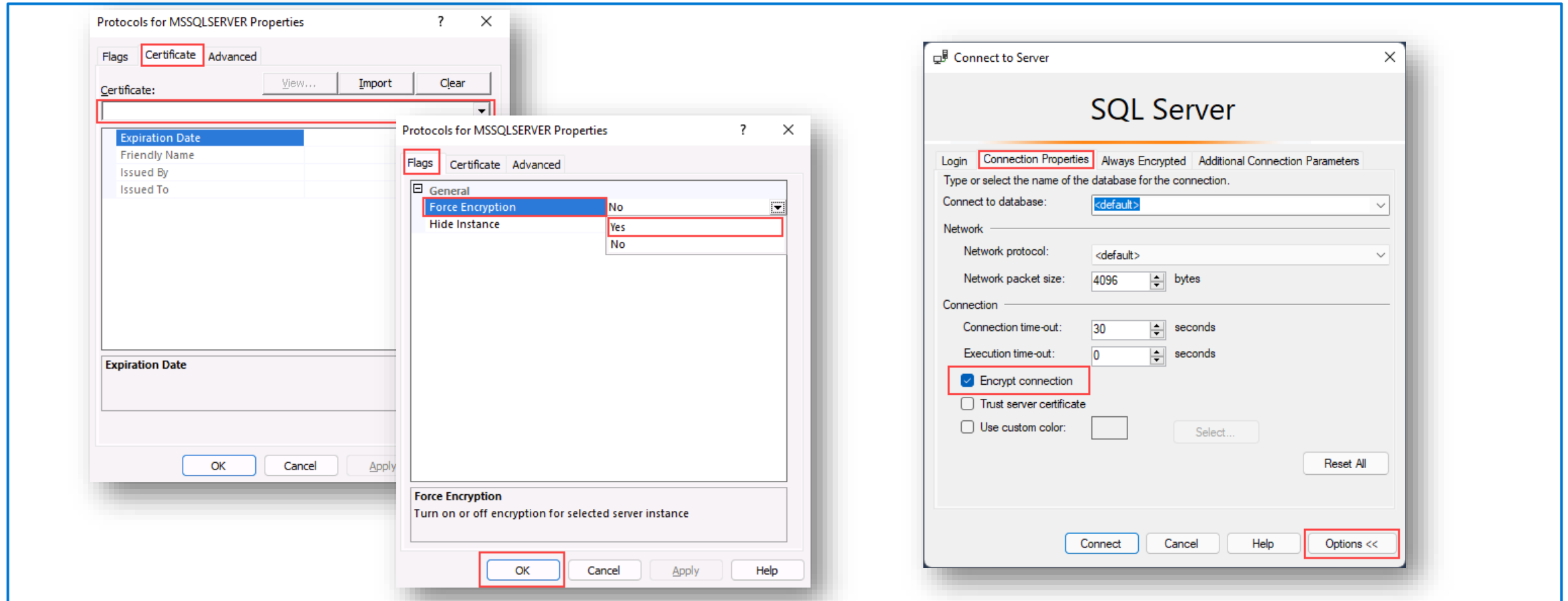
What is SQL Injection?

- SQL Injection is an attack in which malicious code is inserted into strings that are later passed to a database engine

- SQL Injection is a possibility whenever dynamic SQL is used with user input

- An example is shown here >

C#:
```
var shipCity;
shipCity = Request.form ("shipCity");
var sql = "select * from OrdersTable where shipCity
= '" + ShipCity + '";
```
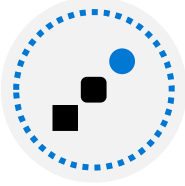
User Input:
```
Redmond'; drop table OrdersTable--
```

Executed SQL:
```
SELECT * FROM OrdersTable WHERE ShipCity =
'Redmond'; drop table OrdersTable--'
```

What is Azure Key Vault?

Key Vault is a solution to safeguard cryptographic keys and other secrets

It can be used in conjunction with Azure VMs running SQL Server to store Transparent Data Encryption and Always Encrypted certificates

This can also be used with on-premises servers
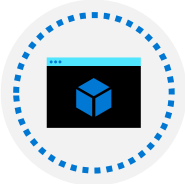
Azure SQL VM resource provider integrates with Key Vaults and can store your TDE certificates in Azure Key Vault

Instructor led labs: Configure a server-based firewall rule using the Azure portal

Configure Azure SQL Database firewall rules

Validate access

Implement compliance controls for sensitive data

Objectives

How data should be classified
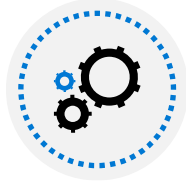
Why data classification should be done

How to implement row level security and dynamic data masking

Understand the usage of Microsoft Defender for SQL
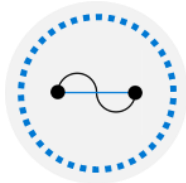
How Azure SQL Database Ledger works

Data classification

Describes how the data must be treated

Companies should build policies around data governance that take the classification of data into account. For example:

Data that is marked as Highly Confidential should not be included in reports that are distributed to the entire company
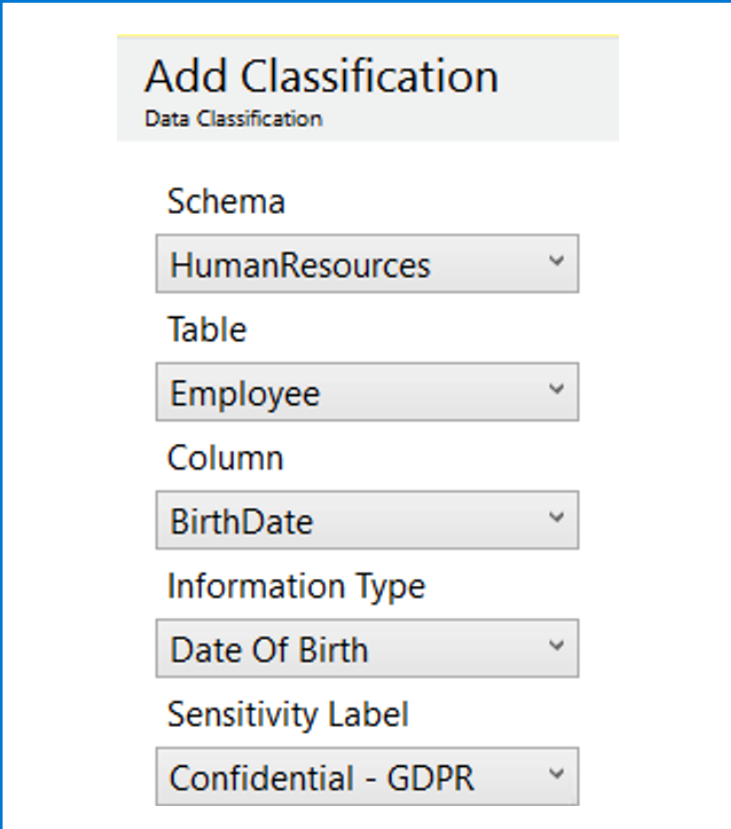
Data that is marked as GDPR should be included in any requests to be forgotten

By classifying data, and setting proper permissions to view data, you can ensure that users should only be seeing the data that they are allowed to see with the database

How your data should be classified

- Data within tables is classified on a column-by-column basis

- A single table can have data that is Public, General, Confidential and Highly Confidential

- Azure SQL Database or SQL Server Management Studio can automatically classify your database, based on column names

- You can also manually classify data using SSMS or T-SQL (`ADD SENSITIVITY CLASSIFICATION`)

Explore server and database audit

- Tracks database events and writes them to an audit log in your Azure Storage account, Log Analytics workspace or Event Hubs

- You can define **server-level and database-level policies**

- It is recommended that you enable only server-level auditing and leave the database-level auditing **disabled** for all databases

Explore server and database audit cont'd

Default auditing policy for SQL Database includes:

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.

Explore server and database audit cont'd

Audit sensitive labels: when combined with data classification, you can also monitor access to sensitive data.

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-f4f0-478a-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-f4f0-478a-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-f4f0-478a-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-f4f0-478a-b016-25164b2a65ff" information_type="SSN" i

Dynamic Data Masking

Dynamic Data Masking (DDM) is used to prevent users from seeing sensitive data.

- Dynamic Data Masking hides data from view by using a user defined value instead of the actual data which is stored

- The data is masked server side, meaning unmasked data is never transmitted over the network

- Data can be unmasked by simply granting a right to a user

- Server admins will always have access to unmasked data

Dynamic Data Masking implementation

Data can be masked in different ways – the engine provides several built-in masks:

Default	Credit card	Social Security number	Random number	Custom text
Full Masking based on data types of the field (no data exposed)	Shows the last four digits of a credit card number	Shows the last 4 digits XXX-XX-1234	Generates random numbers according to the boundaries	Exposes first and last characters and adds custom string in the middle

Dynamic Data Masking use cases

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.

Row level security

Operates at the database level to restrict access to a table by using a security policy – equivalent to a WHERE clause

Depending on the attribute of a user, the predicate determines if the user has access
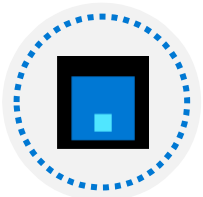
- **Filter predicate -** restrict data access that violate predicate (SELECT, UPDATE, DELETE, INSERT)

- **Block predicate -** restrict data changes that violate predicate (AFTER INSERT, AFTER UPDATE, BEFORE UPDATE, BEFORE DELETE)

Row level security use cases and best practices

Use cases:

- To isolate departmental access at the row level.
- To restrict customers' data access to only the data relevant to their company.
- To restrict access for compliance purposes.
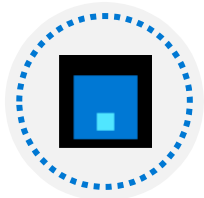
Best practices:

- 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.

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

Advanced Threat Protection watches for:
- Suspicious database activities
- Potential database vulnerabilities
- SQL Injection Attacks
- Anomalous database access and query patterns
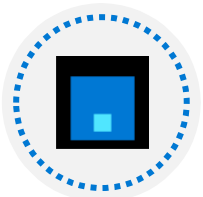
SQL vulnerability assessment is a service that uses a knowledge base of security rules to flag items that do not comply when they are scanned.

Azure SQL Database Ledger

Cryptographically attests to other parties, such as auditors or other business parties, that your data hasn't been tampered with

Collection of accounts of a particular type
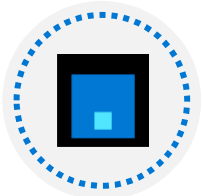
Provides transparent protection of your data from bad actors including but not limited to attackers or even database or cloud administrators

Azure SQL Database Ledger benefits

Ease Audits

Increased trust
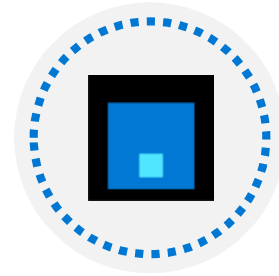
Data integrity

Azure SQL Database Ledger benefits

Microsoft Purview

Microsoft Purview
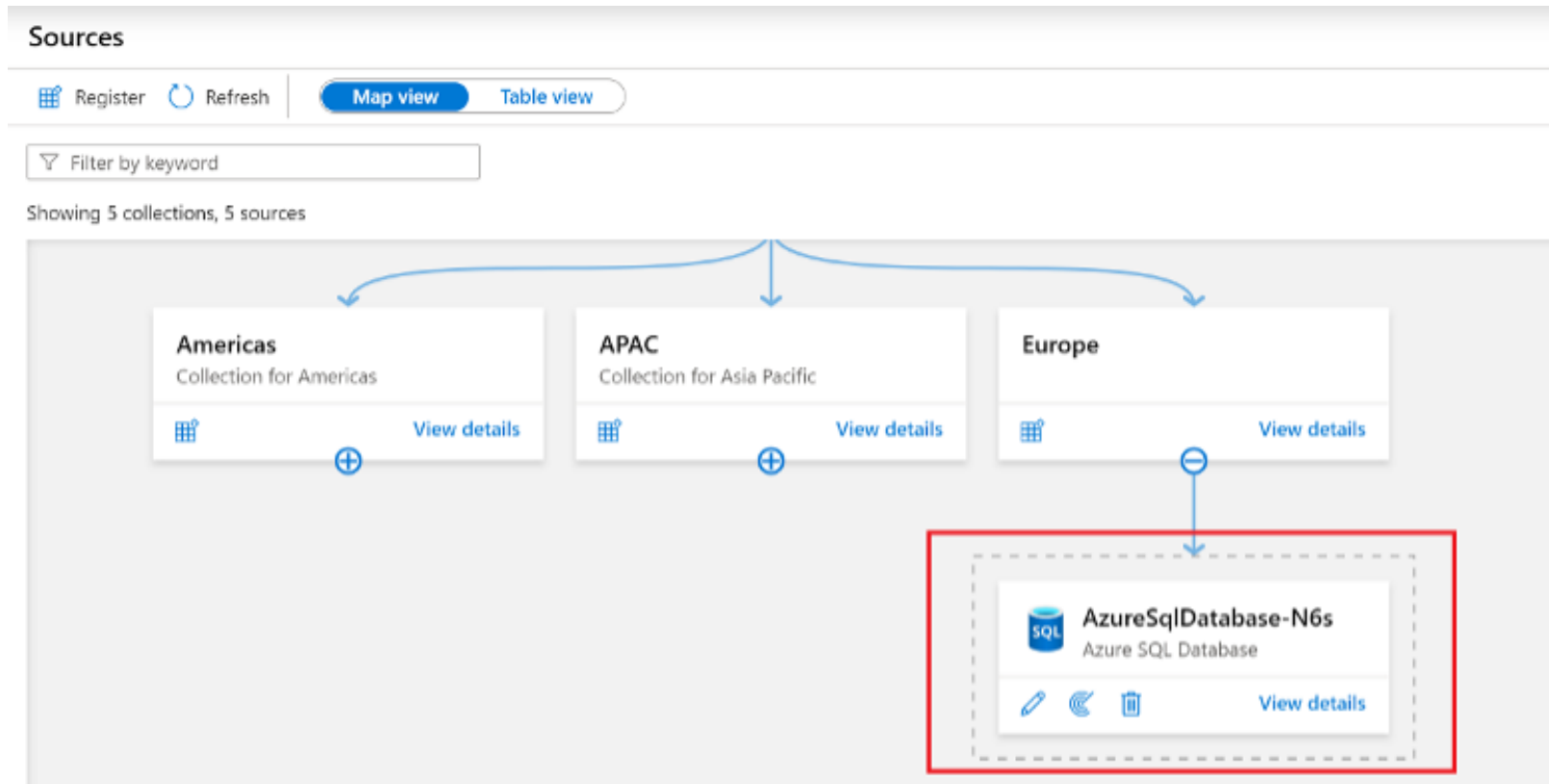
How to check for vulnerabilities in Azure SQL Database

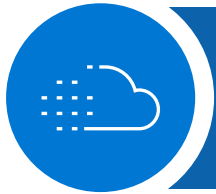Issues that Advanced Threat Protection identifies can be found in the "Microsoft Defender for Cloud" under "Security" section

Click on the *View additional findings in Vulnerability Assessment* in order to see the current vulnerability assessment for the database

Vulnerability assessment is available when scanning results are stored in a storage account

Instructor led labs: Enable Microsoft Defender for SQL and Data Classification

Enable Microsoft Defender for Azure SQL Database

Configure Data Classification for Azure SQL Database

Summary

Configure database authentication and authorization:

- Understand Authentication Options for Azure SQL Database

- Learn what Security Principals are

- Understand Roles in Azure SQL Database and SQL Server

Protect data in-transit and at-rest:

- Implement Transparent Data Encryption and Always Encrypted

- Use Dynamic Data Masking to protect data

- How to manage firewalls in Azure SQL Database

Implement compliance controls for sensitive data:

- Understand data classification and Dynamic Data Masking

- Learn the benefits of Advanced Threat Detection

- Explore Azure SQL Database Ledger and Azure Purview