

Module 1: SQL Server Security

Module Overview	1-1
Lesson 1: Authenticating Connections to SQL Server	1-2
Demonstration: Authenticating Connections to SQL Server	1-6
Lesson 2: Authorizing Logins to connect to databases	1-8
Demonstration: Authorizing connections to databases	1-12
Lesson 3: Authorization across servers	1-14
Demonstration: Authorization across server instances	1-17
Lesson 4: Contained Databases	1-19
Demonstration: Using Contained Databases	1-22
Review	1-23
Lab 1: SQL Server Security	1-24
Questions	1-28

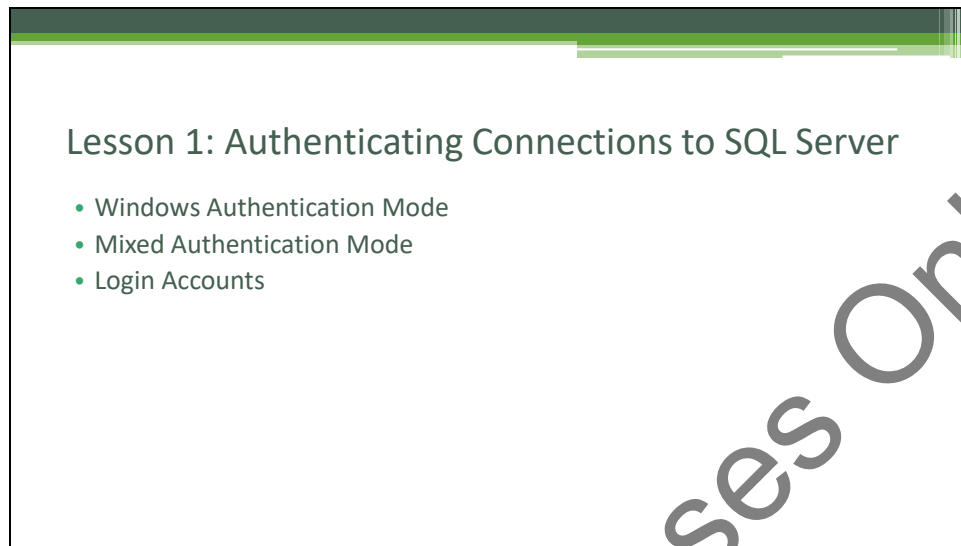
Evaluation Purposes Only

Module Overview

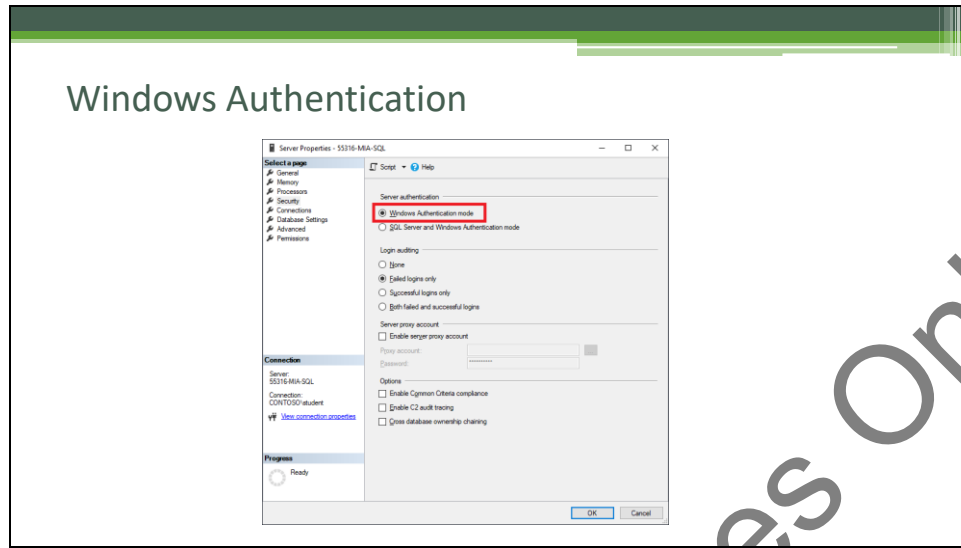
- Lesson 1: Authenticating Connections to SQL Server
- Lesson 2: Authorizing Logins to Connect to databases
- Lesson 3: Authorization Across Servers
- Lesson 4: Contained Databases
- Lab 1: SQL Server Security
- Module Review

SQL Server security can be accomplished in several different ways but primarily authentication at this server and database level are used to protect access to data resources. Another important feature is the ability to use contained databases to isolate in database from the SQL Server instance and other databases. Understanding and using these features will make it easier for database administrators to control access to these resources.

Lesson 1: Authenticating Connections to SQL Server



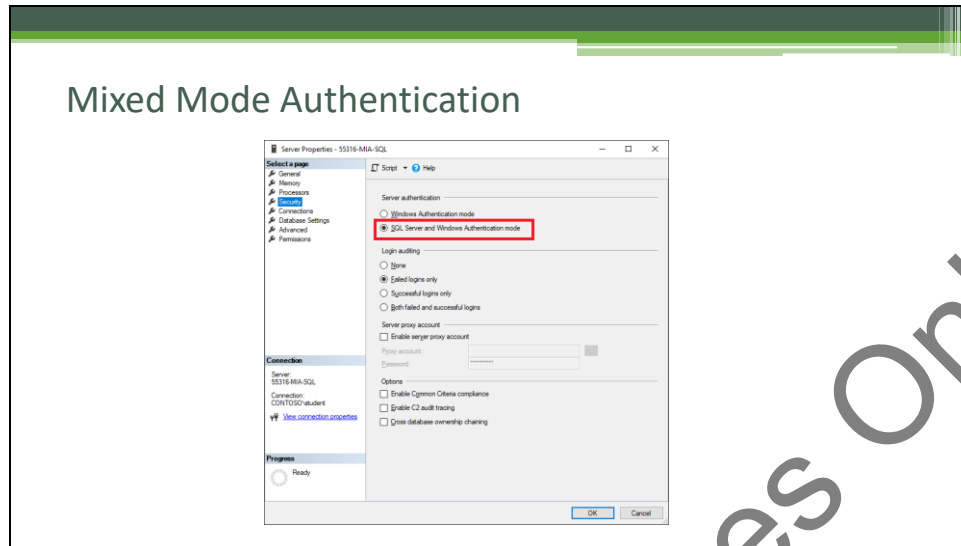
SQL Server authentication can be configured in one of two modes. Windows authentication only, or Windows authentication in combination with SQL Server authentication (Mixed Mode). This option may be configured during the initial setup but may be changed after the server is running. Modifying the authentication mode requires a restart of the instance.



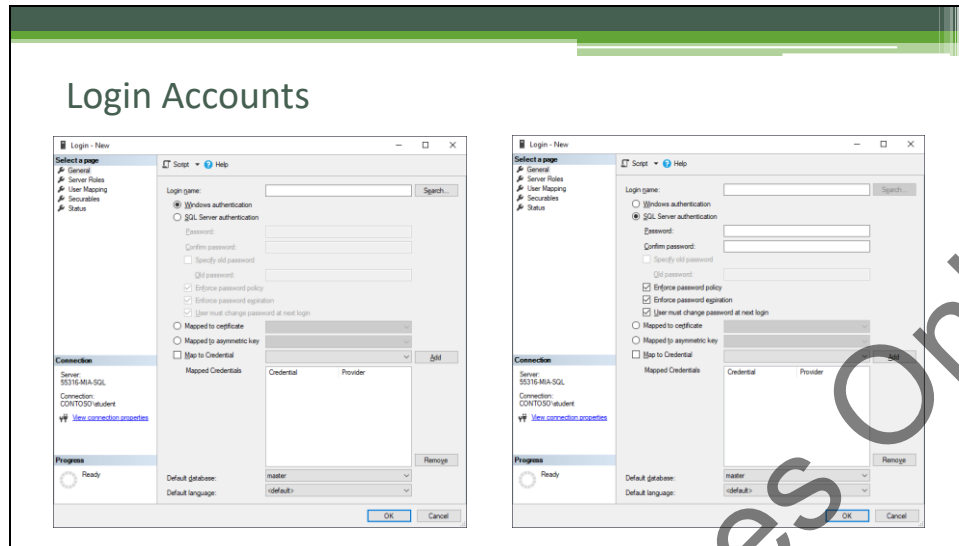
Windows authentication allows the administrator to use local or domain user accounts or groups when accessing SQL Server. This is the default authentication mode for SQL Server instances. When a new login is created, it is linked to an existing Windows or domain account. The account policy settings that secure the account in the operating system, such as password policy, will be enforced on the account in SQL Server as well. Password settings continue to be managed at the operating system level.

All login accounts are created in the master database unless a contained database is being used. The account status and permissions can be modified at any time and will apply the next time the account is used.

Mixed Mode Authentication



The SQL Server instance may be configured to allow SQL Server authentication in addition to windows authentication. These SQL Server based login accounts can be configured to access any instance resource but will not on their own be able to access operating system resources.



Maintaining security for Windows login accounts will be easier for database administrators. The operating system or domain will enforce appropriate rules concerning age, length and configuration of passwords for example. SQL Server login accounts give the administrator a greater level of control over these settings since they can be managed by the instance. Certificate and key mapping options are available for either type of account.

Pre-defined server-level roles are often used to assign instance level or database permissions to login accounts. These permissions cannot be changed (except for the public role). User-defined server-level roles can be created to assign more specific permissions to groups of users. More information about the specific permissions of the pre-defined server-level roles (groups) can be found on the Microsoft documentation website.

Demonstration: Authenticating Connections to SQL Server

Demonstration Tasks:

- Review some of the SQL Server instance properties that affect security.
- Create a SQL Server Login Account.

Demonstration:

Note: Unless otherwise stated, all PS1 and CMD scripts should be executed with Elevated (Administrator) privileges.

1. Login to the **55316-MIA-SQL** virtual machine with your username and password (**Student / Pa\$\$w0rd**).
2. In the **C:\Classfiles** folder, verify that the **AdventureWorks2019.bak** file exists. If not execute the **C:\Classfiles\AdventureWorks_Download.ps1** file.
3. In the **C:\Classfiles** folder, execute the **C:\Classfiles\AdventureWorks_Install.ps1** file. This will restore the original **AdventureWorks2019** database configuration if you have modified it.
4. Launch **SQL Server Management Studio** and connect to the local instance using Windows Authentication.
5. From **Object Explorer**, right-click the SQL Server **Instance Name** then click **Properties**.
6. Click the **Security** page and review the authentication and auditing options.
7. Click the **Connections** page and review the connections and remote server connections options.
8. Click **OK** in the **Server Properties** page when done.
9. From **Object Explorer**, double-click **Security**. Right-click **Logins** then click **New Login**.
10. Choose to create a SQL Login account with the name **SQLLogin3** and a password of **Pa\$\$w0rd**.
11. Review the options in the **General** and **Server Roles** pages.
12. Click the Server Role **securityadmin**.

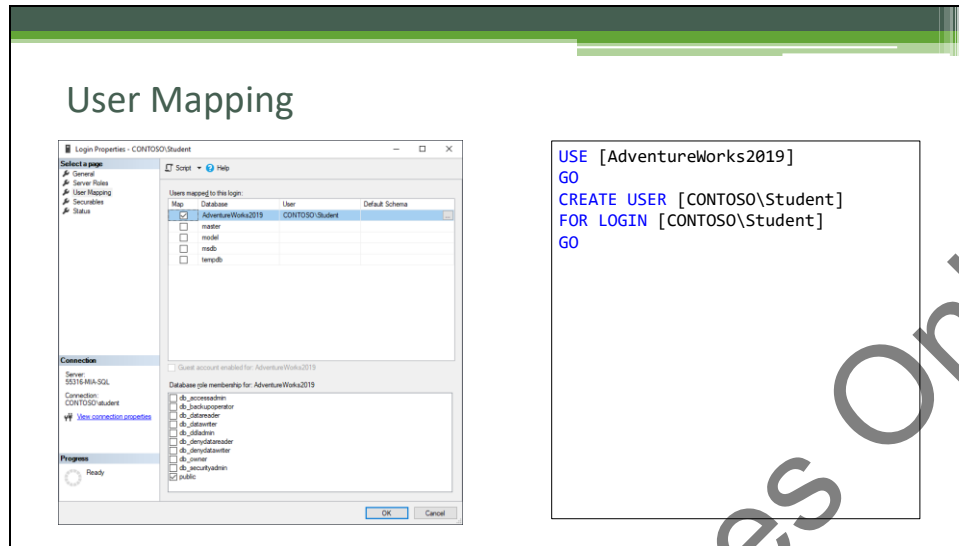
13. Click **OK** in the **Login – New** page to create the login account.
14. Minimize **SQL Server Management Studio**.

Evaluation Purposes Only

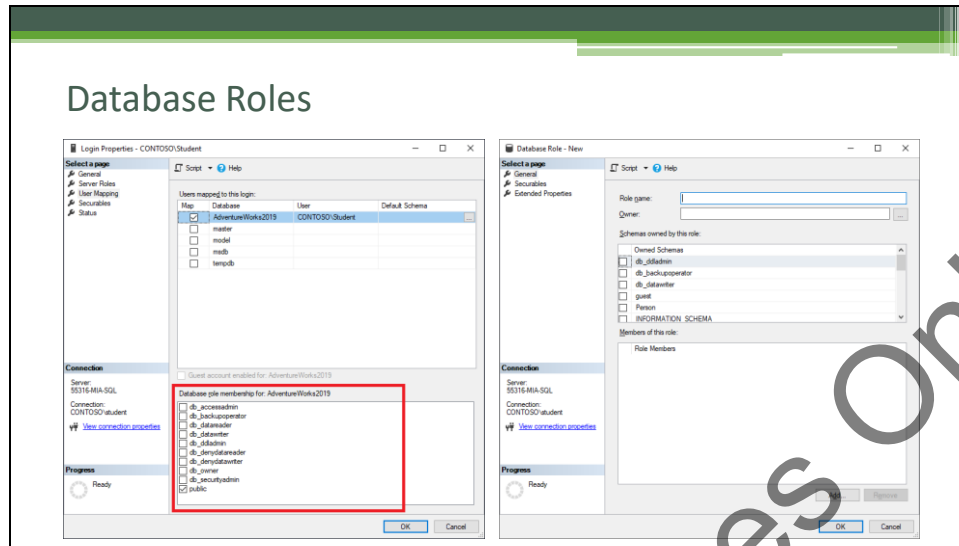
Lesson 2: Authorizing Logins to connect to databases

- User Mapping
- Database Roles
- Securables

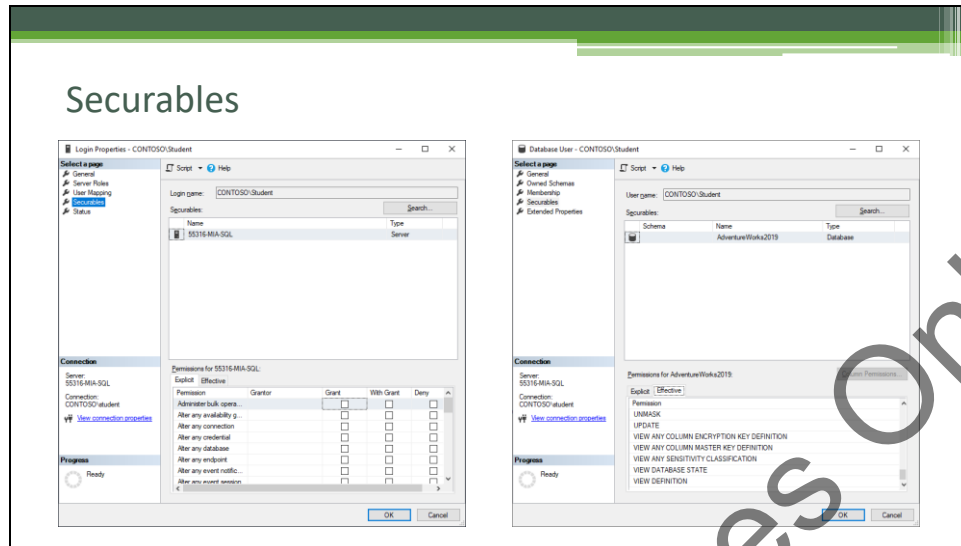
By taking advantage of the different roles, authentication methods and executable options available for an instance, administrators can customize how users access their databases. Understanding how each of these features works will prepare you to use them appropriately depending on why a user is being given access.



Permissions on a database can be given to a login account by mapping it to a user account in the database. A login account can be granted access to multiple databases by creating a mapping to an account in each database. The user account can be created in Management Studio or by using T-SQL with the CREATE USER statement. The default schema (normally dbo) can be changed for each user account.



Database roles are used to assign permissions to a group of users in the database. In addition to the predefined roles, you may create custom roles with customized permissions for better security. Users can be assigned to multiple roles and will inherit the permissions of all the groups they belong to. The public role exists in each database and is automatically used to assign permissions to users that do not already belong to a group. It is often used to assign the lowest level of access desired for any user in the database.



Permissions to any object in a database can be granted or denied to a user. Deny permissions will take precedent if a user inherits both. In addition to granting permissions to a user, an administrator may also decide to give the “With Grant” permissions to them. This is done if they will be delegated the responsibility of assigning that permission to others.

Because a user can have access granted by means of many server and database level roles, it is sometimes difficult to track exactly what level of access they have in a database. The “Effective” level of access can be ascertained in the Database User properties window.

Demonstration: Authorizing connections to databases

Demonstration Tasks:

- Create database user
- Assign database permissions

Demonstration:

Note: Unless otherwise stated, all PS1 and CMD scripts should be executed with Elevated (Administrator) privileges.

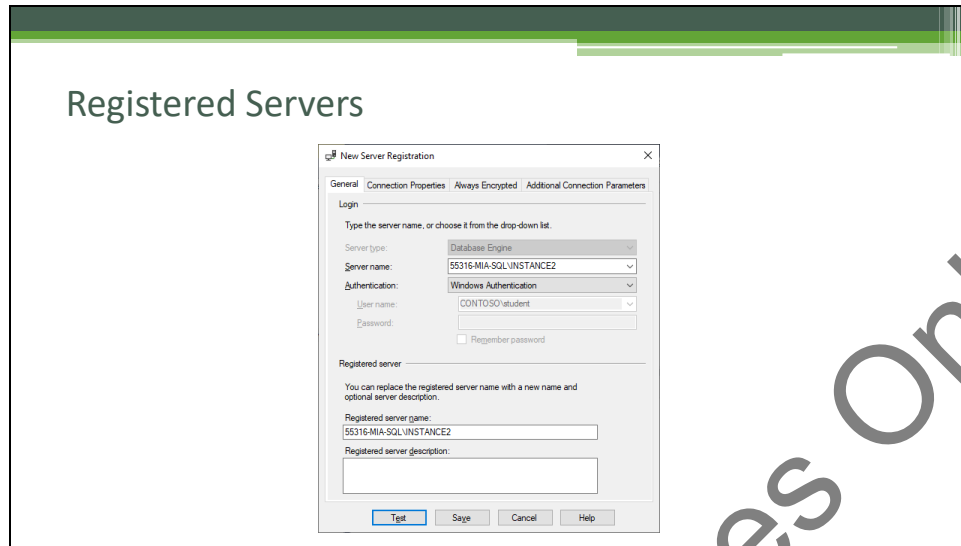
1. Login to the 55316-MIA-SQL virtual machine with your username and password (**Student / Pa\$\$w0rd**).
2. In the **C:\Classfiles** folder, verify that the **AdventureWorks2019.bak** file exists. If not execute the **C:\Classfiles\AdventureWorks_Download.ps1** file.
3. In the **C:\Classfiles** folder, execute the **C:\Classfiles\AdventureWorks_Install.ps1** file. This will restore the original **AdventureWorks2019** database configuration if you have modified it.
4. Repeat the steps above to restore the **AdventureWorksDW2019** database.
5. Launch **SQL Server Management Studio** and connect to the local instance using Windows Authentication.
6. From **Object Explorer**, open the **Logins** folder under **Security**.
7. Double click the new login created earlier, (**SQLLogin3**).
8. In the User Mapping page, select **AdventureWorks2019** and **AdventureWorksDW2019**.
9. Change the default schema for **AdventureWorks2019** to **Person**.
10. In the **Securables** page, view the effective permissions for the login account at the server level.
11. Click **OK** or use the “Script” feature to generate a script to apply the changes and execute it.

12. Open a **New Query** window and connect to it using the credentials of **SQLLogin3**. Try to run a query on the **Person** table in **AdventureWorks2019** (`SELECT * FROM Person.Person`). It will fail because the user account does not have the necessary permissions.
13. In the **Object Explorer**, under **Databases > AdventureWorks2019 > Security > Users**, double click the new user account created earlier (**SQLLogin3**).
14. On the **Membership** page, check the box for **db_datareader** and click **OK**.
15. In the previous query window, try to run the query again. It will succeed. You may also try to run the query using a one-part name for the **Person** table (`SELECT * FROM Person`). This will work because the default schema assigned to **SQLLogin3** is **Person**.
16. Minimize **SQL Server Management Studio**.

Lesson 3: Authorization across servers

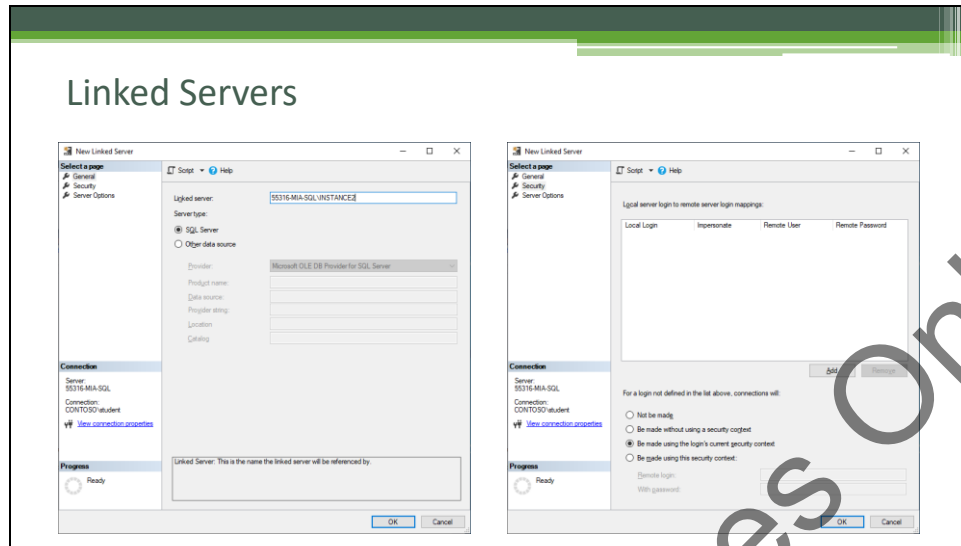
- Registered Servers
- Linked Servers

When database operations must be run across multiple SQL Servers, it is possible to run a single task that affects all the servers. The instances can be registered in a group and managed as such, or individual connections can be made between them. Appropriate server-level and database-level permissions will be required but after these are configured, seamless management across multiple instances will be possible.



Registering multiple servers in SSMS will make it easier to manage many instances that are often worked on together. Once configured, there will be no need to manually add each server again and again when starting SSMS. This feature works with the database engine and other SQL Services like SSIS and SSRS. Windows or SQL Server authentication may be used.

Central Management Servers is another way to register many instances in Management Studio once. This method does not work with SQL Server authentication and only works for the database engine. It stores the connection information in MSDB which is more secure than the XML file method used for registered servers.



Another way to connect to remote database engines is to use linked servers. Unlike registered servers, linked servers can be used to connect to database engines that are not SQL Server. These types of connections are mostly used to access and manage data records on remote servers using OLE DB and ODBC database providers. Connections to Excel spreadsheets and Access databases can be configured using the 32-bit Microsoft Jet.OLEDB.40 OLE DB provider.

Demonstration: Authorization across server instances

Demonstration Tasks:

- Create a Registered Server
- Create a Linked Server

Demonstration:

Note: Unless otherwise stated, all PS1 and CMD scripts should be executed with Elevated (Administrator) privileges.

1. Login to the **55316-MIA-SQL** virtual machine with your username and password (**Student / Pa\$\$w0rd**).
2. In the **C:\Classfiles** folder, verify that the **AdventureWorks2019.bak** file exists. If not execute the **C:\Classfiles\AdventureWorks_Download.ps1** file.
3. In the **C:\Classfiles** folder, execute the **C:\Classfiles\AdventureWorks_Install.ps1** file. This will restore the original **AdventureWorks2019** database configuration if you have modified it.
4. In the **C:\Classfiles** folder, execute the **C:\Classfiles\instance2.cmd** file. This will create a new SQL Server instance (**Instance2**) on the computer.
5. Launch **SQL Server Management Studio** and connect to the local instance using Windows Authentication.
6. From **Object Explorer**, click **Connect > Database Engine**. Browse for the **INSTANCE2** database engine and connect to it using Windows Authentication.
7. From **Object Explorer** on the **INSTANCE2** instance, right-click **Databases** and click **Restore Database**. Under **Source**, click **Device** and browse for the **C:\Classfiles\AdventureWorks2019.bak** file.
8. Click **OK** to restore the database backup file. Verify the **AdventureWorks2019** database is restored successfully on **INSTANCE2**.
9. From the **Management Studio** menu, click **View > Registered Servers**.

10. From **Registered Servers**, right-click **Local Server Group** and click **New Server Registration**.
11. From the **Server name:** list, select **55316-MIA-SQL\INSTANCE2**. Verify the authentication method is Windows Authentication and change the **Registered server name:** to **localinstance2**. (Note: You may also assign an alias for the default instance, 55316-mia-sql).
12. Click **Test** to verify connectivity and then click **Save**.
13. Close both instance connections in **Object Explorer**. Close **Management Studio**.
14. Reopen Management Studio. If prompted for credentials to either instance, **Cancel** the connection.
15. From **Registered Servers**, double-click the aliases under **Local Server Groups** to add both instances to **Object Explorer**.
16. Open a **New Query** window using the default instance. Try to run a query on the **vSalesPerson** view in **AdventureWorks2019** on **55316-MIA-SQL\INSTANCE2** (`SELECT * FROM [55316-MIA-SQL\INSTANCE2].AdventureWorks2019.Sales.vSalesPerson`). It will fail. The default instance does not have a security context for connecting to **INSTANCE2**. Verify this by querying `sys.servers` (`SELECT * FROM sys.servers`).
17. For the default instance, in **Object Explorer**, under **Server Objects**, right-click **Linked Servers** and click **New Linked Server**.
18. On the **General** page, choose **SQL Server** for the server type and **55316-MIA-SQL\INSTANCE2** for the server. On the **Security** page, select **Be made using the login's current security context**. Click **OK**.
19. Refresh the page and verify the new **Linked Server** appears in the list.
20. Run a query from the default instance on the **vSalesPerson** view in **AdventureWorks2019** on **55316-MIA-SQL\INSTANCE2** (`SELECT * FROM [55316-MIA-SQL\INSTANCE2].AdventureWorks2019.Sales.vSalesPerson`). The distributed query should work.
21. Minimize **SQL Server Management Studio**.

Lesson 4: Contained Databases

- Database Level Authentication
- Database Level Metadata

Contained databases is not the traditional way of providing access to databases, but it does provide useful way for developers and others to control access to their resources with limited control from the instance level. In the same way SQL Server logins provide some autonomy for database administrators from domain administrators, contained databases can provide a level of control for application developers and others who need more control over their databases.

Database Level Authentication

Traditional Model (created in master / user database)	Contained Database Model (created in user database)
<code>CREATE LOGIN <login> WITH PASSWORD=<password>;</code>	<code>CREATE USER <user> WITH PASSWORD = <password>;</code>
<code>CREATE USER <user> FOR LOGIN <login>;</code>	

Contained databases are isolated from the SQL Server instance, and its master database where login accounts are stored. Access to the database is authorized using user accounts in the database. This method of authentication will work with accounts using Windows or SQL Server authentication. The user will be limited to resources in the database and will not be able to access the instance or objects in other databases. If a duplicate account exists in another database on the instance, this can be used to access those resources.

Database Level Metadata

- Master Database
- Dependencies
- Moving the database

In addition to the normal metadata stored by any database, contained databases also store data related to authentication and login normally kept in the master database. This is in line with the strategy of removing dependencies outside of the user database.

The use of resources outside of the contained database are permitted. These uncontained resources may be referenced by scripts or code. They may present problems if the database needs to be moved. Contained databases which have eliminated the use of any resources outside of itself can be moved easily between instances while maintaining consistent administration settings.

Demonstration: Using Contained Databases

Demonstration Tasks:

- Creating a contained database

Demonstration:

Note: Unless otherwise stated, all PS1 and CMD scripts should be executed with Elevated (Administrator) privileges.

1. Login to the **55316-MIA-SQL** virtual machine with your username and password (**Student / Pa\$\$w0rd**).
2. In the **C:\Classfiles** folder, verify that the **AdventureWorks2019.bak** file exists. If not execute the **C:\Classfiles\AdventureWorks_Download.ps1** file.
3. In the **C:\Classfiles** folder, execute the **C:\Classfiles\AdventureWorks_Install.ps1** file. This will restore the original **AdventureWorks2019** database configuration if you have modified it.
4. Use the **C:\Classfiles\Demofiles\MOD01\containeddb.sql** script to walk through the process of creating a contained database and user accounts. Alternatively, you may walk through the steps using the **Management Studio** GUI interface.

Review

1. What account type allows instance administrators to modify password age independent of operating system settings?
2. How can you automatically assign default level permissions for any new user in a database?
3. What kind of database allows authentication at the database level?

1. What account type allows instance administrators to modify password age independent of operating system settings?
2. How can you automatically assign default level permissions for any new user in a database?
3. What kind of database allows authentication at the database level?

Lab 1: SQL Server Security

Exercise 1: Authenticating Connections to SQL Server

Exercise 2: Authorizing Connections to databases

Exercise 3: Authorization across server instances

Exercise 4: Authorizing Connections to databases

Login Information:

- VM Name: 55316-MIA-SQL
- Username: Student
- Password: Pa\$\$w0rd

Time: 60 - 75 minutes

Lab 1: SQL Server Security

Note: Unless otherwise stated, all PS1 and CMD scripts should be executed with Elevated (Administrator) privileges.

Exercise 1: Authenticating Connections to SQL Server

1. Login to the **55316-MIA-SQL** virtual machine with your username and password (**Student / Pa\$\$w0rd**).
2. In the **C:\Classfiles** folder, verify that the **AdventureWorks2019.bak** and **AdventureWorksDW2019.bak** files exists. If not execute the **AdventureWorks_Download.ps1** and **AdventureWorksDW_Download.ps1** files.
3. In the **C:\Classfiles** folder, execute the **AdventureWorks_Install.ps1** and **AdventureWorksDW_Install.ps1** files. This will restore the original database configurations for **AdventureWorks2019** and **AdventureWorksDW2019**.
4. Launch **SQL Server Management Studio** and connect to the local instance using Windows Authentication.
5. From **Object Explorer**, right-click the SQL Server **Instance Name** then click **Properties**.
6. Click the **Security** page and review the authentication and auditing options.

7. Click the **Connections** page and review the connections and remote server connections options.
8. Click **OK** in the **Server Properties** page when done.
9. From **Object Explorer**, double-click **Security**. Right-click **Logins** then click **New Login**.
10. Choose to create a SQL Login account with the name **SQLLogin3** and a password of **Pa\$\$w0rd**.
11. Review the options in the **General** and **Server Roles** pages.
12. Click the Server Role **securityadmin**.
13. Click **OK** in the **Login – New** page to create the login account.

Exercise 2: Authorizing Connections to databases

1. Launch **SQL Server Management Studio** and connect to the local instance using Windows Authentication.
2. From **Object Explorer**, open the **Logins** folder under **Security**.
3. Double click the new login created earlier, (**SQLLogin3**).
4. In the User Mapping page, select **AdventureWorks2019** and **AdventureWorksDW2019**.
5. Change the default schema for **AdventureWorks2019** to **Person**.
6. In the **Securables** page, view the effective permissions for the login account at the server level.
7. Click **OK** or use the “Script” feature to generate a script to apply the changes and execute it.
8. Open a **New Query** window and connect to it using the credentials of **SQLLogin3**. Try to run a query on the Person table in **AdventureWorks2019** (SELECT * FROM Person.Person). It will fail because the user account does not have the necessary permissions.
9. In the **Object Explorer**, under **Databases > AdventureWorks2019 > Security > Users**, double click the new user account created earlier (**SQLLogin3**).
10. On the Membership page, check the box for **db_datareader** and click **OK**.

11. In the previous query window, try to run the query again. It will succeed. You may also try to run the query using a one-part name for the **Person** table (SELECT * FROM Person). This will work because the default schema assigned to **SQLLogin3** is **Person**.

Exercise 3: Authorization across server instances

1. In the **C:\Classfiles** folder, execute the **C:\Classfiles\instance2.cmd** file. This will create a new SQL Server instance (**Instance2**) on the computer.
2. Launch **SQL Server Management Studio** and connect to the local instance using Windows Authentication.
3. From **Object Explorer**, click **Connect > Database Engine**. Browse for the **INSTANCE2** database engine and connect to it using Windows Authentication.
4. From **Object Explorer** on the **INSTANCE2** instance, right-click **Databases** and click **Restore Database**. Under **Source**, click **Device** and browse for the **C:\Classfiles\AdventureWorks2019.bak** file.
5. Click **OK** to restore the database backup file. Verify the **AdventureWorks2019** database is restored successfully on **INSTANCE2**.
6. From the **Management Studio** menu, click **View > Registered Servers**.
7. From **Registered Servers**, right-click **Local Server Group** and click **New Server Registration**.
8. From the **Server name:** list, select **55316-MIA-SQL\INSTANCE2**. Verify the authentication method is Windows Authentication and change the **Registered server name:** to **localinstance2**. (Note: You may also assign an alias for the default instance, 55316-mia-sql).
9. Click **Test** to verify connectivity and then click **Save**.
10. Close both instance connections in **Object Explorer**. Close **Management Studio**.
11. Reopen Management Studio. If prompted for credentials to either instance, **Cancel** the connection.
12. From **Registered Servers**, double-click the aliases under **Local Server Groups** to add both instances to **Object Explorer**.
13. Open a **New Query** window using the default instance. Try to run a query on the **vSalesPerson** view in **AdventureWorks2019** on **55316-MIA-SQL\INSTANCE2** (SELECT * FROM [55316-MIA-SQL\INSTANCE2].AdventureWorks2019.Sales.vSalesPerson). It will fail.

The default instance does not have a security context for connecting to **INSTANCE2**. Verify this by querying sys.servers (SELECT * FROM sys.servers).

14. For the default instance, in **Object Explorer**, under **Server Objects**, right-click **Linked Servers** and click **New Linked Server**.
15. On the **General** page, choose **SQL Server** for the server type and **55316-MIA-SQL\INSTANCE2** for the server. On the **Security** page, select **Be made using the login's current security context**. Click **OK**.
16. Refresh the page and verify the new **Linked Server** appears in the list.
17. Run a query from the default instance on the **vSalesPerson** view in **AdventureWorks2019** on **55316-MIA-SQL\INSTANCE2** (SELECT * FROM [55316-MIA-SQL\INSTANCE2].AdventureWorks2019.Sales.vSalesPerson). The distributed query should work.
18. Minimize **SQL Server Management Studio**.

Exercise 4: Authorizing Connections to databases

1. Use the **C:\Classfiles\Labfiles\MOD01\containeddb.sql** script to walk through the process of creating a contained database and user accounts. Alternatively, you may walk through the steps using the **Management Studio** GUI interface.



Questions?

Review & Questions

Before continuing, ask the students if they have any questions on the material, demonstrations, or labs.