



Exploring Organizational Security and Auditing



Contents

Organizational security and audit.....	3
Scenario 1: SQL Server audit	3
Scenario 2: Contained Database Authentication	16
Scenario 3: User Defined Server Roles	22
Scenario 4: New Permissions	28
Terms of use	40

Organizational security and audit

Estimated time to complete lab is 30 minutes

Today, businesses require anyone within an organization to be able to access information so that they are able to best perform their role. As businesses become more closely aligned with their suppliers and customers this requirement is extending to these as well. This places requirements to ensure that users are only able to access the appropriate information and that user access is monitored and recorded

You want to be sure that to be sure that users have access only to the data and features that they need to perform their roles. In order to meet this requirement you are going to use the following features on SQL 2014:

- Audit resilience and security
- Contained database authentication
- User defined server roles
- The new server and database permissions

You will be checking that the audit features implemented to meet compliance requirements. As a new application has been recently deployed, you are going to check that the user security has been correctly implemented as per your security policies. As part of the implementation of the security policy database permissions have been defined for the roles within the data administration team. You are going to check that these have been correctly implemented

In this exercise, you will use the following new auditing features in SQL Server 2014:

- **Audit Resilience** so that audit logs are not lost during temporary file and network issues during a failover
- **User Defined Audit** events to allow applications to write custom information to the audit log
- **Audit Filtering** to improve filtering to simplify audit reporting

Audit Resilience

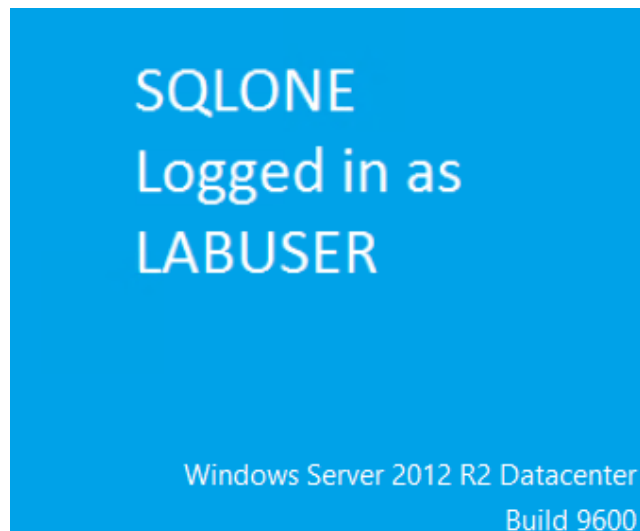
The resilience features implemented in SQL Server Auditing means that the audit logging is now tolerant to loss of connectivity to the target directory and will recover automatically once the network connection is re-established.

Scenario 1: SQL Server audit

You will implement a policy is that processing should continue in the event of an audit log failure. The audit process is also able to record that auditing has been paused or stopped. The policy has been implemented on a new server and Richard has been tasked to confirm that it has been implemented correctly and to check that the audit log correctly records when auditing is disabled and then re-enabled.

Connect to SQLONE computer

1. Click on **SQLONE** button on right side of the screen to connect to the **SQLONE** computer. If you see the following in the lower right corner of the screen, you can jump to step 5 below to set your screen resolution.



2. Click **Send Ctrl-Alt-Del** for **SQLONE** computer and then click **Switch user**.
3. Click **Send Ctrl-Alt-Del** for **SQLONE** computer again and then click **Other user**.
4. Log on to **SQLONE** computer as **labuser** with password **pass@word1**

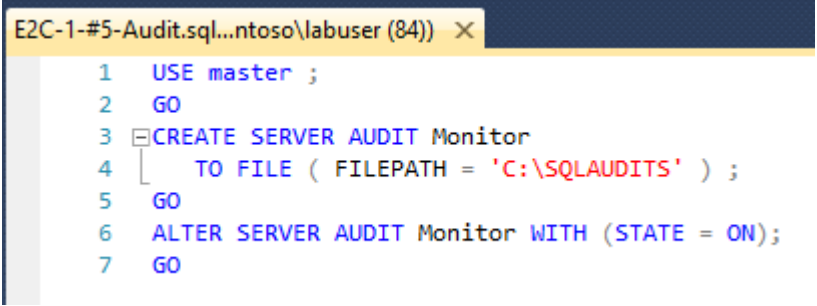
Note, if you have a monitor that supports a larger screen resolution than 1024 x 768, you can change the screen resolution for the lab to go as high as 1920 x 1080. By going to a higher screen resolution, it will be easier to use SQL Server Management Studio.

5. Right click on the desktop and click on **Screen resolution**.
6. Select **1366 x 786** (a good minimum screen size for using SSMS) and click **OK**.
7. Click **Keep Changes**.

8. Resize the client **holLaunchPad Online** window for the lab to fit your screen resolution.

Create an Audit called Monitor

1. Open **SQL Server 2014 Management Studio** with a connection to the **SQLONE** server and ensure **Windows Authentication** is selected before clicking **Connect**.
2. From the **File** menu, select **Open** and then **File...**
3. Browse to **C:\SQLSCRIPTS\E2** and select **E2C-1-#5-Audit**
4. Click **Open**



```
E2C-1-#5-Audit.sql...ntoso\labuser (84) X
1 USE master ;
2 GO
3 CREATE SERVER AUDIT Monitor
4 TO FILE ( FILEPATH = 'C:\SQLAUDITS' ) ;
5 GO
6 ALTER SERVER AUDIT Monitor WITH (STATE = ON);
7 GO
```

5. Click **Execute**

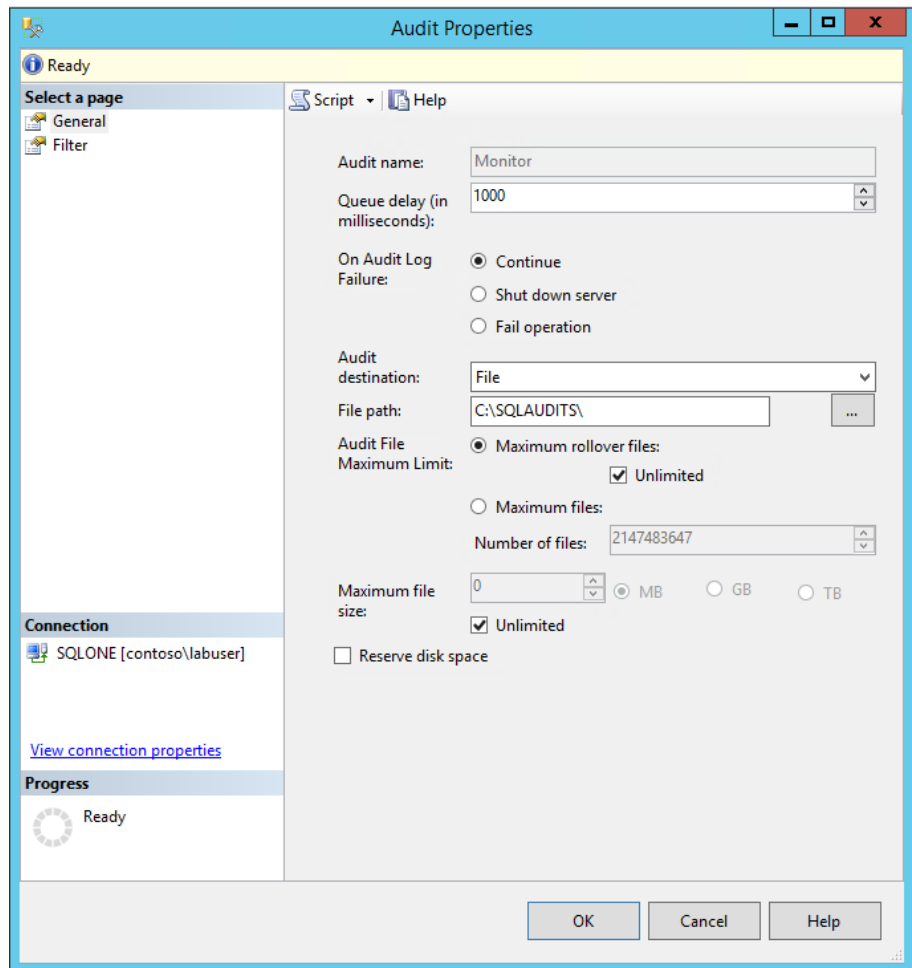
Confirm that the audit was created

1. Expand **Security** in the **Object Explorer** and then **Audits**

NOTE: you may need to click on Refresh  to see the Audit

2. Right-click on **Monitor** and select **Properties**

Here Richard can see the properties of this audit. We see that it is set to continue operation of the server in case of an audit log failure. The audit destination (File) and file path are also defined here

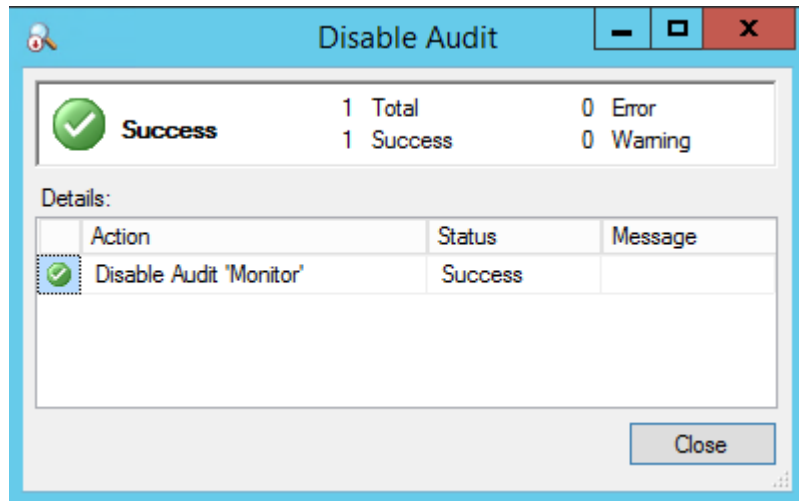


3. Click **OK**

Temporarily disable the audit

This event will be written to the audit log, but the server will remain in operation

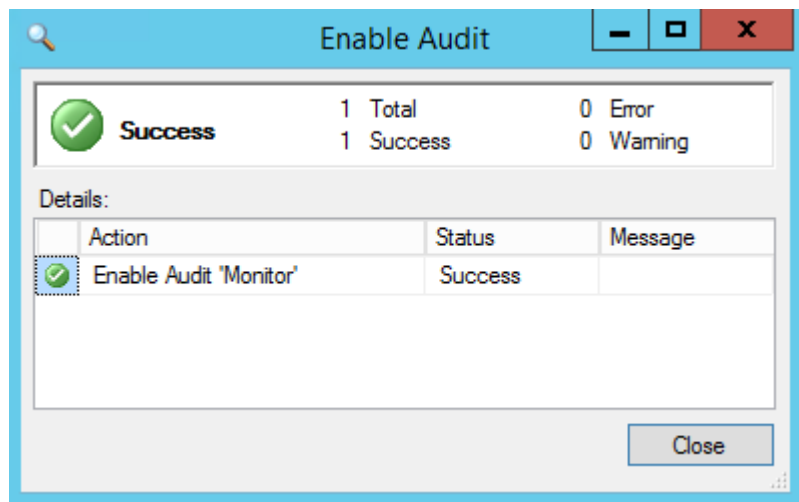
1. Right-click on **Monitor** and select **Disable Audit**



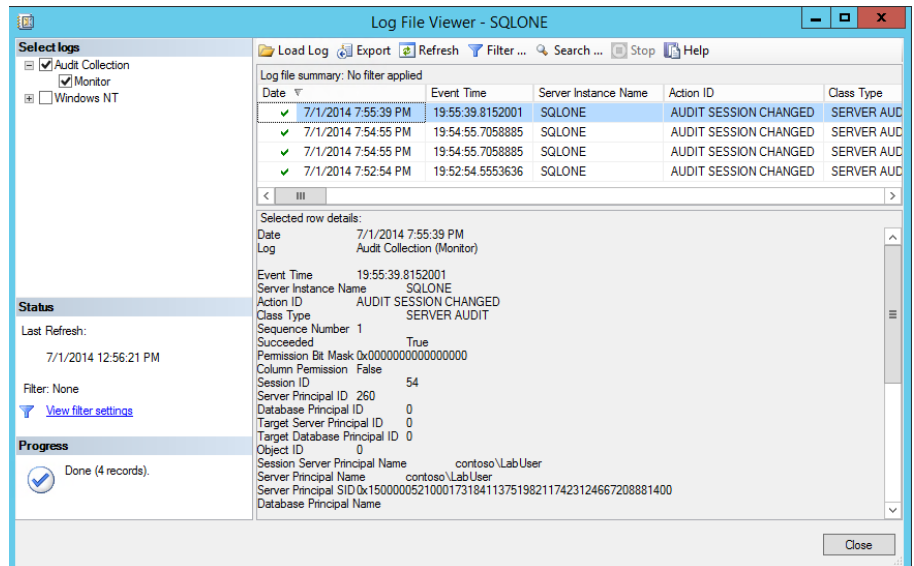
2. Click **Close**

Wait a few seconds

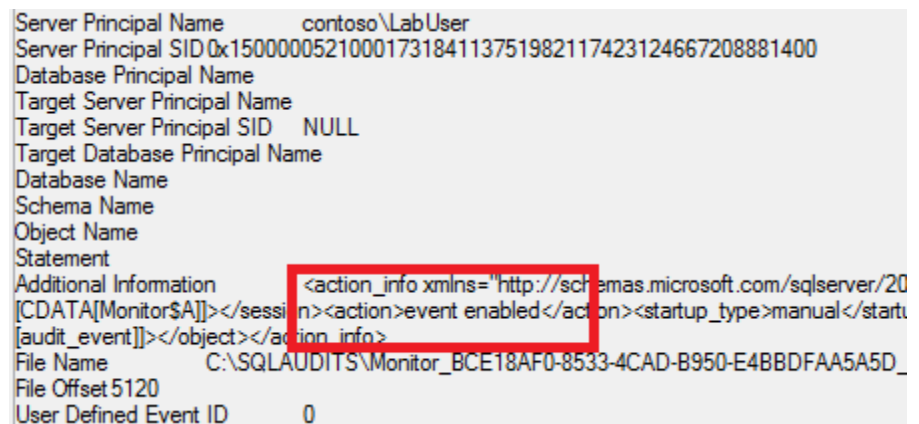
3. Right-click on **Monitor** and select **Enable Audit**



4. Click **Close**
5. Right-click on **Monitor** and select **View Audit Logs** then close after looking at the results



Here you can see that the audit log recorded that auditing was disabled and then re-enabled



6. Click **Close**.

User Defined Audit

The new User-defined audit events allow applications to write custom information to the audit log

As part of an auditing policy you want to be able to log employee's whose salary is increased by more than 20%. You decide to create a User-Defined Audit event. The event will be triggered whenever an employee's salary is increased by more than 20%

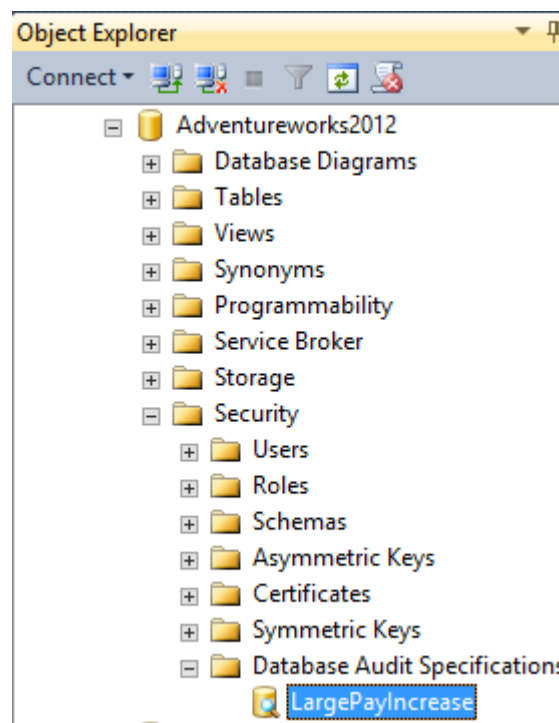
Create an Audit called LargePayIncrease

1. In **SQL Server 2014 Management Studio** open the **File** menu, select **Open** and then **File...**
2. Browse to **C:\SQLSCRIPTS\E2** and select **E2C-1-#6-Audit**
3. Click **Open**


```
E2C-1-#6-Audit.sql...ntoso\labuser (51)) X E2C-1-#5-Audit.sql...ntoso\labuser (84))
1 USE [master]
2 GO
3 CREATE SERVER AUDIT [TestingUserDefinedEvents]
4 TO FILE
5 ( FILEPATH = N'C:\SQLAUDITS'
6 ,MAXSIZE = 5 MB
7 ,MAX_ROLLOVER_FILES = 5
8 ,RESERVE_DISK_SPACE = OFF
9 )
10 WITH
11 ( QUEUE_DELAY = 1000
12 ,ON_FAILURE = CONTINUE
13 )
14 GO
15 ALTER SERVER AUDIT [TestingUserDefinedEvents] WITH (STATE = ON);
16 GO
17 USE [AdventureWorks2012]
18 GO
19 CREATE DATABASE AUDIT SPECIFICATION [LargePayIncrease]
20 FOR SERVER AUDIT [TestingUserDefinedEvents]
21 ADD (USER_DEFINED_AUDIT_GROUP)
22 WITH (STATE = ON)
23 GO
```

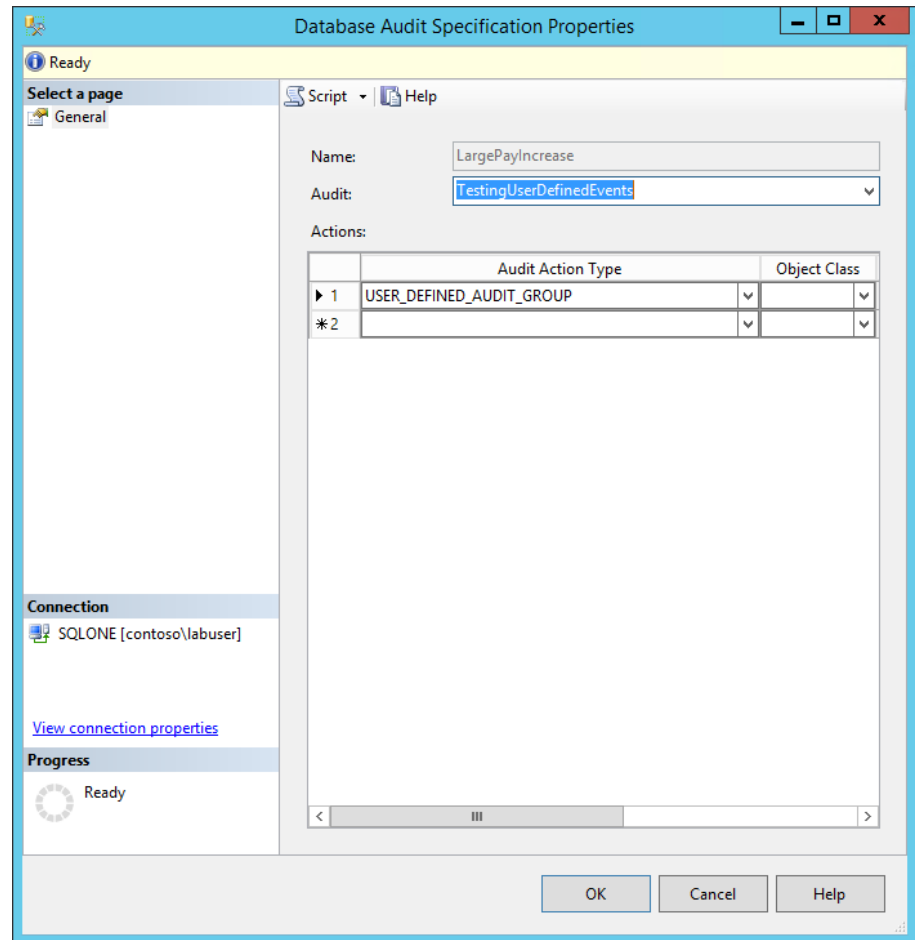
4. Click **Execute**
5. Expand **Databases** and then **AdventureWorks2012, Security, Database Audit Specifications** in the **Object Explorer**

You will see that the Audit Report called LargePayIncrease was created.



6. Right Click on **LargePayIncrease** and select **Properties**

You will notice that the Audit Action Type is set to `USER_DEFINED_AUDIT_GROUP`. This group tracks events raised by using the `sp_audit_write` stored procedure



7. Click on **OK**

Use a trigger to write an audit record using `sp_audit_write`

Create a trigger for when an employee's salary is increased by 20%. This trigger uses the `sp_audit_write` stored procedure, hence an event will be logged when it is used and recorded in the `LargePayIncrease` audit

1. From SQL Server 2014 Management Studio select File and then Open and then File...
2. Browse to **C:\SQLSCRIPTS\E2** and select **E2C-1-#1**
3. Click **Open**

```

E2C-1-#1.sql - SQL...ntoso\labuser (54)  E2C-1-#6-Audit.sql...ntoso\labuser (51)
1  USE AdventureWorks2012
2  GO
3  CREATE TRIGGER [humanresources].[SalaryMonitor] ON [humanresources].[employeepayhistory]
4  AFTER UPDATE
5  AS
6  DECLARE @oldrate money
7  , @newrate money
8  , @empid integer
9  , @msg nvarchar(4000)
10 select @oldrate = d.rate
11 from deleted d
12 select @newrate = i.rate, @empid = i.BusinessEntityID
13 from inserted i
14 IF @oldrate*1.20 < @newrate
15 BEGIN
16 SET @msg = 'Employee '+CAST(@empid as varchar(50))+ ' pay rate increased more than 20%'
17 EXEC sp_audit_write @user_defined_event_id = 27 ,
18 @succeeded = 1
19 , @user_defined_information = @msg;
20 END
21 GO

```

4. Click **Execute**

Test the trigger

Now that you have created a trigger, you can now test the user defined audit event is working by running the following Script, which virtually doubles employee 4's pay rate

1. From **SQL Server 2014 Management Studio** select **File** and then **Open** and then **File...**
2. Browse to **C:\SQLSCRIPTS\E2** and select **E2C-1-#2**
3. Click **Open**

```

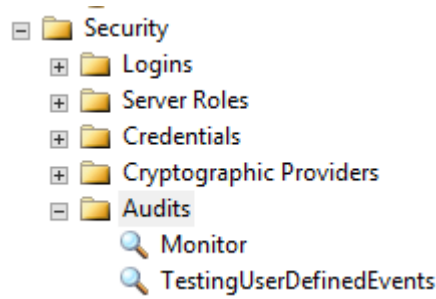
E2C-1-#2.sql - SQL...ntoso\labuser (56)  E2C-1-#1.sql - SQL...ntoso\labuser (54)
1  USE AdventureWorks2012
2  GO
3  select * from HumanResources.EmployeePayHistory where BusinessEntityID= 4
4  GO
5  Update HumanResources.EmployeePayHistory set rate = 69.8462
6  where BusinessEntityID=4 and RateChangeDate = '2006-01-15 00:00:00.000'
7  GO

```

4. Click **Execute**

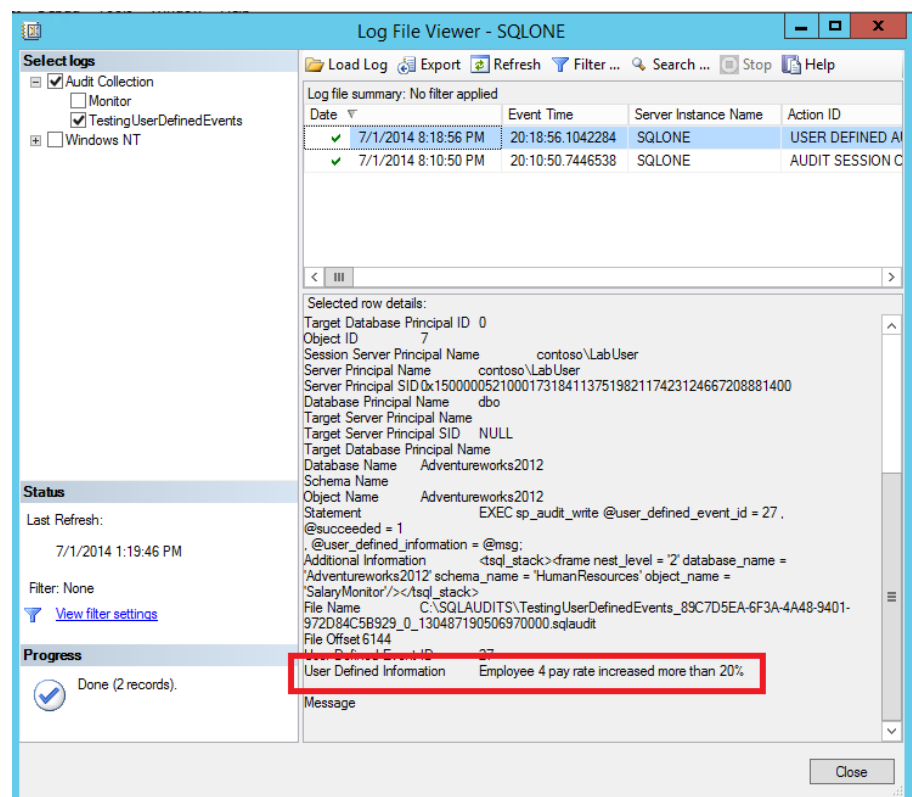
Check that the pay increase was recorded

1. In the **SQL Server 2014 Management Studio** in **Object Explorer**, expand **Security** and then **Audits** (if this node is already open, refresh it by right-clicking on the node and selecting **Refresh**)



- Right Click on **TestingUserDefinedEvents** and select **View Audit Logs**

You can see that the User-defined event was recorded in the logs for employee 4



- Click **Close**.

Audit Filtering

Audit Filtering allows the filtering of unwanted audit events before they are written to the audit log

As part of an auditing policy, you need to be able to log which users are accessing selected tables which contain sensitive data. To do this the you will create an Audit Filter which records the table, user, date and time when the table was accessed

First, you will create an Audit called Payole_Security_Audit

1. In **SQL Server 2014 Management Studio** from the **File** menu, select **Open** and then **File...**
2. Browse to **C:\SQLSCRIPTS\E2** and select **E2C-1-#4-Audit**
3. Click **Open**

```

E2C-1-#4-Audit.sql...ntoso\labuser (55)  X  E2C-1-#2.sql - SQL...ntoso\labuser (56)
1  USE master ;
2  GO
3  -- Create the server audit
4  CREATE SERVER AUDIT Payrole_Security_Audit
5  TO FILE (FILEPATH = 'C:\SQLAUDITS' ) ;
6  GO
7  -- Enable the server audit
8  ALTER SERVER AUDIT Payrole_Security_Audit WITH (STATE = ON);
9  GO
10 -- Move to the target database
11 USE AdventureWorks2012 ;
12 GO
13 -- Create the database audit specification
14 CREATE DATABASE AUDIT SPECIFICATION Audit_Pay_Tables
15 FOR SERVER AUDIT Payrole_Security_Audit
16 ADD (SELECT , INSERT ON HumanResources.EmployeePayHistory BY dbo )
17 WITH (STATE = ON) ;
18 GO

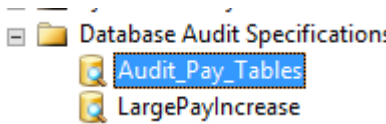
```

4. Click **Execute**

Confirm that the audit was created.

5. Expand **Databases** and then expand **AdventureWorks2012, Security, Database Audit Specifications** in the **Management Studio Object Explorer** (if this node is already open, refresh it by right-clicking on the node and clicking **Refresh**)

There's an Audit Report called Audit_Pay_Tables



6. Right Click on **Audit_Pay_Tables** and select **Properties**

Richard notices that if any user uses SELECT in the EmployeePayHistory table it will be recorded in the Audit logs

Name:

Audit:

Actions:

	Audit Action Type	Object Class	Object	Object Name
1	INSERT	OBJECT	HumanResou...	EmployeePay...
▶ 2	SELECT	OBJECT	HumanResou...	EmployeePay...
*3				

7. Click **OK**

Test the Audit is working by running the following SELECT query

- From **SQL Server 2014 Management Studio** select **File** and then **Open** and then **File...**
- Browse to **C:\SQLSCRIPTS\E2** and select **E2C-1-#3**

This script selects information from the EmployeePayHistory table, so should cause an entry in the audit log.

10. Click **Open**

```

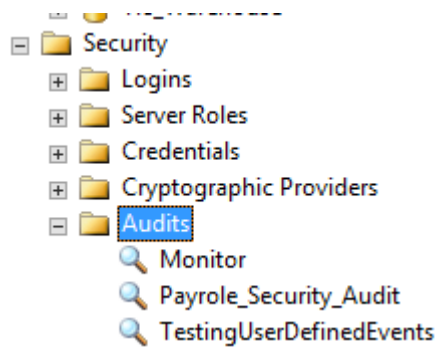
E2C-1-#3.sql - SQL...ntoso\labuser (58) × E2C-1-#4-Audit.sql...ntoso\labuser (55)
1 USE Adventureworks2012
2 GO
3 SELECT TOP 10 [BusinessEntityID]
4     , [RateChangeDate]
5     , [Rate]
6     , [PayFrequency]
7     , [ModifiedDate]
8 FROM [Adventureworks2012].[HumanResources].[EmployeePayHistory]
9 GO

```

11. Click **Execute**

Check that the query was recorded in the Audit logs

- Under the **SQLONE** connection, expand **Security** and then **Audits** (if this node is already expanded, right-click and select **Refresh**)



13. Right-click on **Payrole_Security_Audit** and select **View Audit Logs**

Log file summary: No filter applied

Date	Event Time	Server Instance Name	Action ID
7/1/2014 8:34:46 PM	20:34:46.4350102	SQLONE	SELECT
7/1/2014 8:29:32 PM	20:29:32.0081008	SQLONE	AUDIT SESSION C

Selected row details:

Date 7/1/2014 8:34:46 PM
 Log Audit Collection (Payrole_Security_Audit)

Event Time 20:34:46.4350102

Server Instance Name SQLONE
 Action ID SELECT
 Class Type TABLE
 Sequence Number 1
 Succeeded True
 Permission Bit Mask 0x0000000000000001
 Column Permission True
 Session ID 58
 Server Principal ID 260
 Database Principal ID 1
 Target Server Principal ID 0
 Target Database Principal ID 0
 Object ID 1493580359

Session Server Principal Name contoso\LabUser
 Server Principal Name contoso\LabUser
 Server Principal SID 0x1500000521000173184113751982117423124667208881400
 Database Principal Name dbo
 Target Server Principal Name
 Target Server Principal SID NULL
 Target Database Principal Name

Database Name Adventureworks2012
 Schema Name HumanResources
 Object Name EmployeePayHistory
 Statement SELECT TOP 10 [BusinessEntityID]
 .[RateChangeDate]
 .[Rate]
 .[PayFrequency]
 [ModifiedDate]

You can see that when the SELECT query was run on the Table EmployeePayHistory, the user, date, and time was recorded

14. Close everything without saving

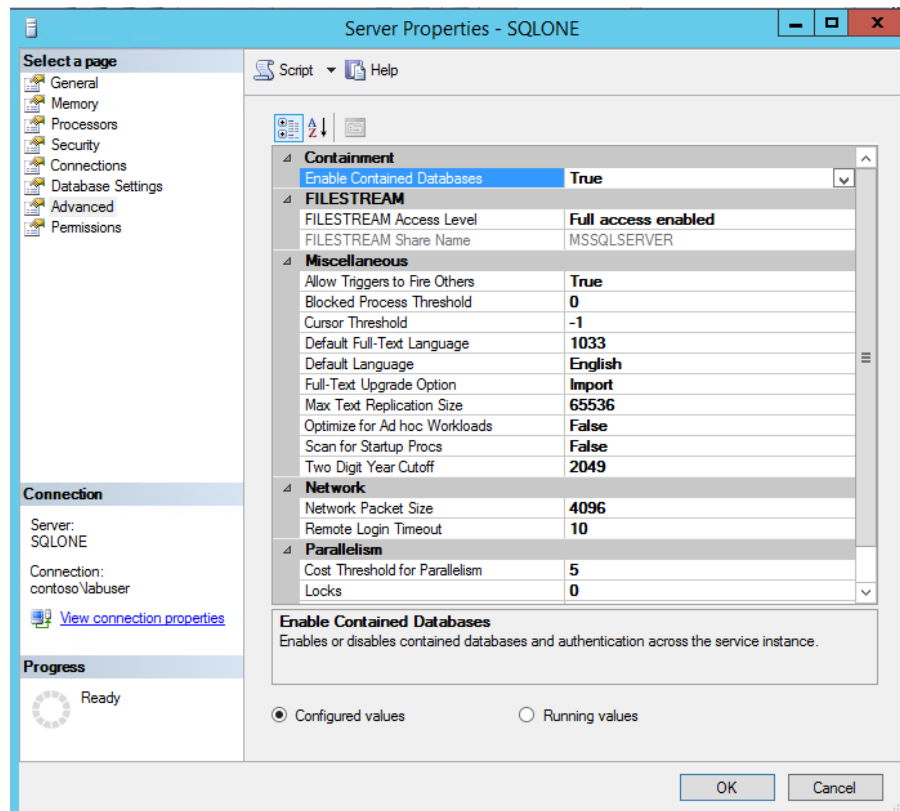
Scenario 2: Contained Database Authentication

A contained database includes all database settings and metadata required to define the database and has no configuration dependencies on the instance of the Database Engine where the database is installed. Users can connect to the database without authenticating a login at the Database Engine level. Isolating the database from the Database Engine makes it possible to easily move the database to another instance of SQL Server

In this exercise, you will create a new Contained Database that user John has read only access.

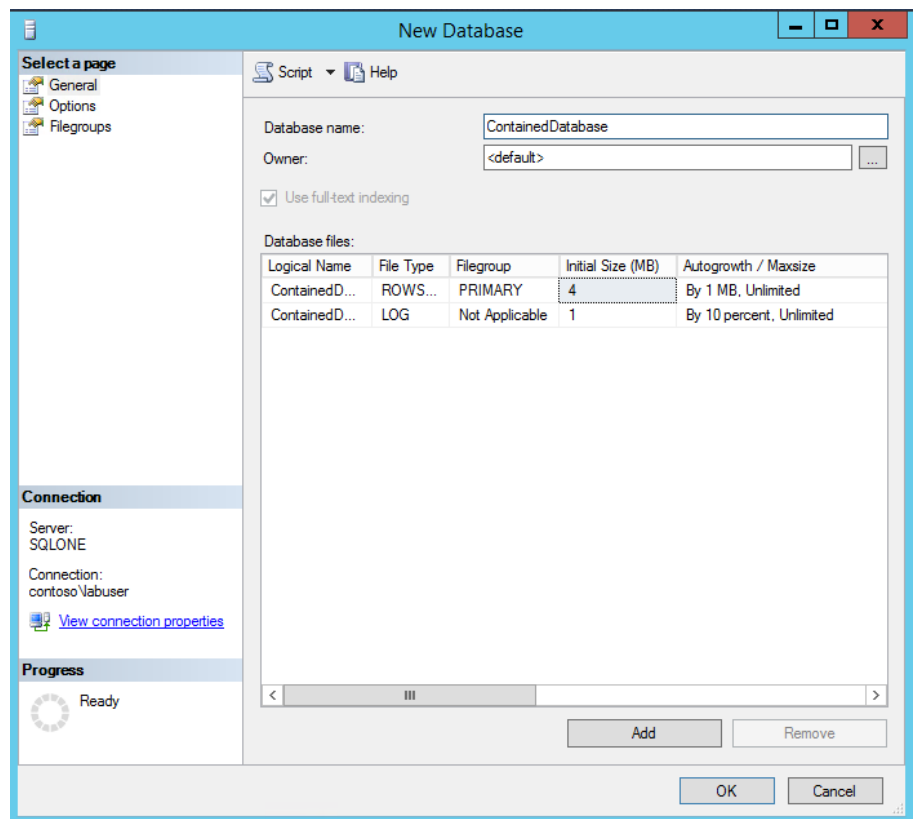
Enable server for contained databases

1. Ensure you have **SQL Server 2014 Management Studio** open with a connection to the **SQLONE** server (if you do not, open **SQL Server 2014 Management Studio** from the Windows Start screen, enter **Database Engine...** as the server type, **SQLONE** as the Server name and ensure **Windows Authentication** is selected before clicking **Connect**.)
2. Right-click on **SQLONE** in the **Object Explorer** and select **Properties**
3. Select **Advanced** and then change **Enable Contained Databases** to **True**



4. Click **OK**

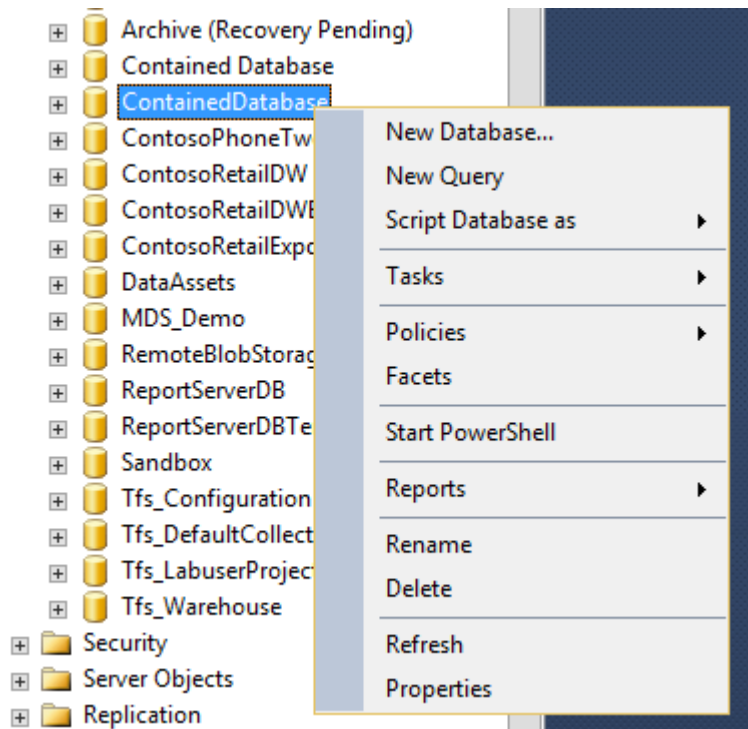
5. Right-click on **Databases** in the **Object Explorer** and select **New Database**
6. In Database name enter **ContainedDatabase**



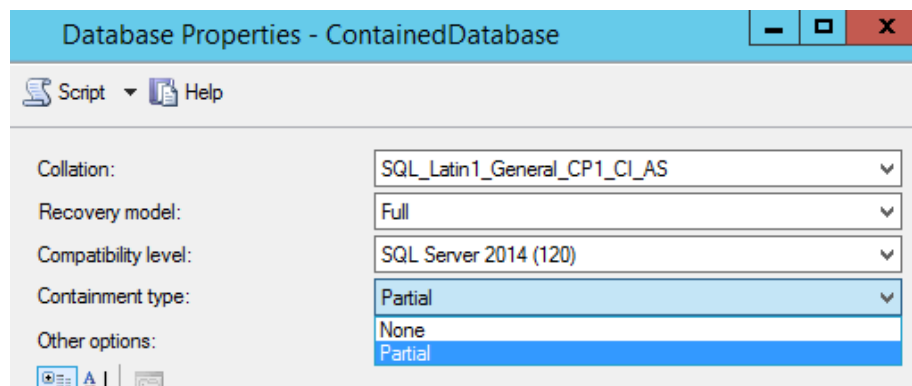
7. Click **OK**
8. Click on **Refresh**  in **Object Explorer**.

Set the database as being contained

1. Expand **Databases**
2. Right-click on **ContainedDatabase** in the **Object Explorer** and then select **Properties**



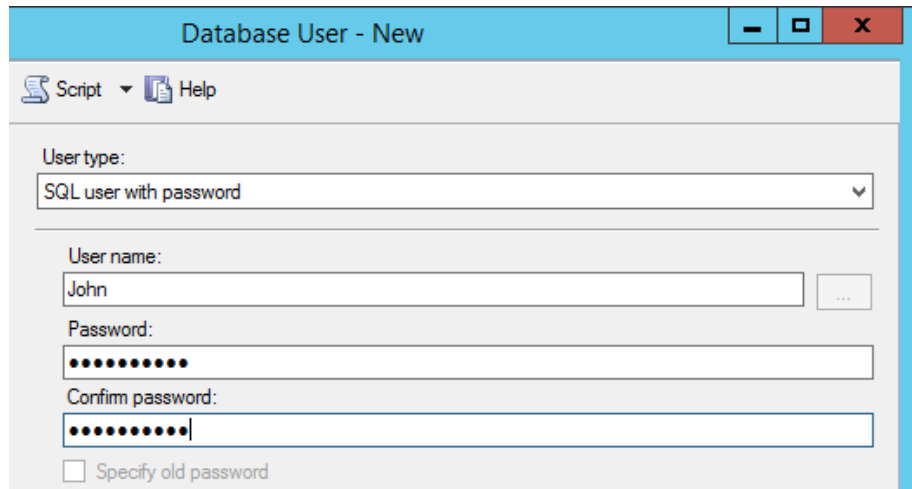
3. Click on the **Options** page
4. Change **Containment type** to **Partial**



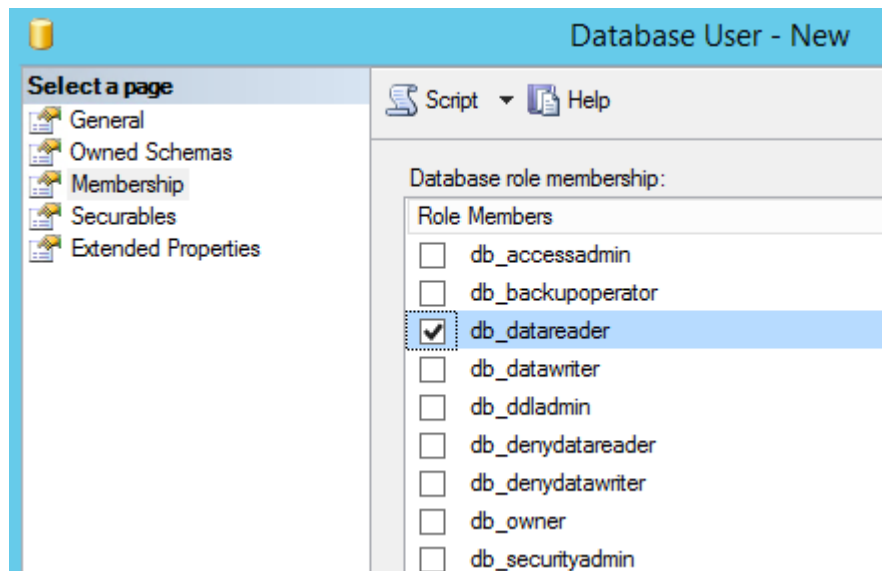
5. Click **OK**

Add the user John to the database

1. Expand the **ContainedDatabase** node in the **Object Explorer** and then **Security** and then **Users**
2. Right-click on **Users** and select **New User**
3. In User name enter **John** and in Password enter **pass@word1**. Ensure the User type is **SQL user with password** and enter the same password in **Confirm Password** dialog



4. Click on **Membership** and tick **db_datareader**

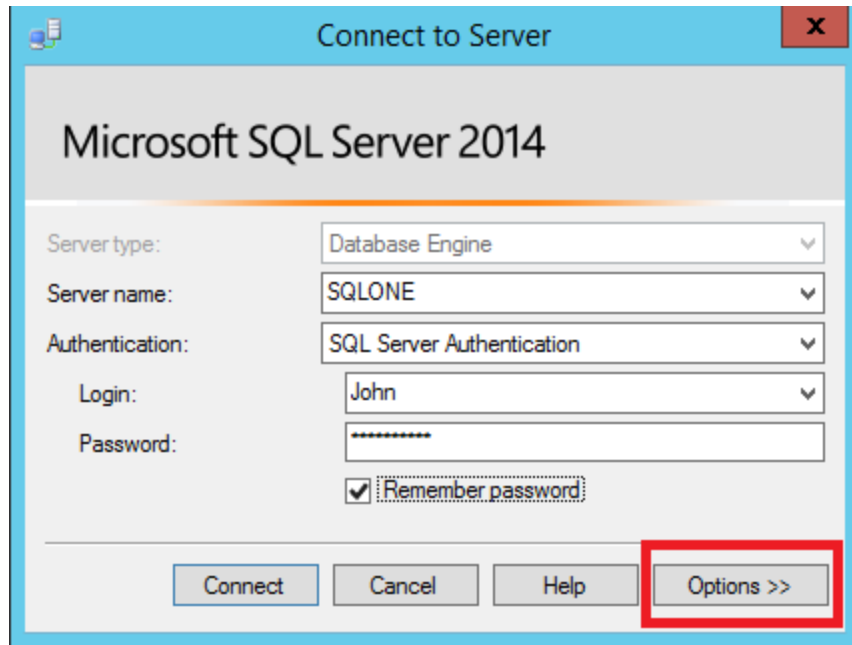


5. Click **OK**

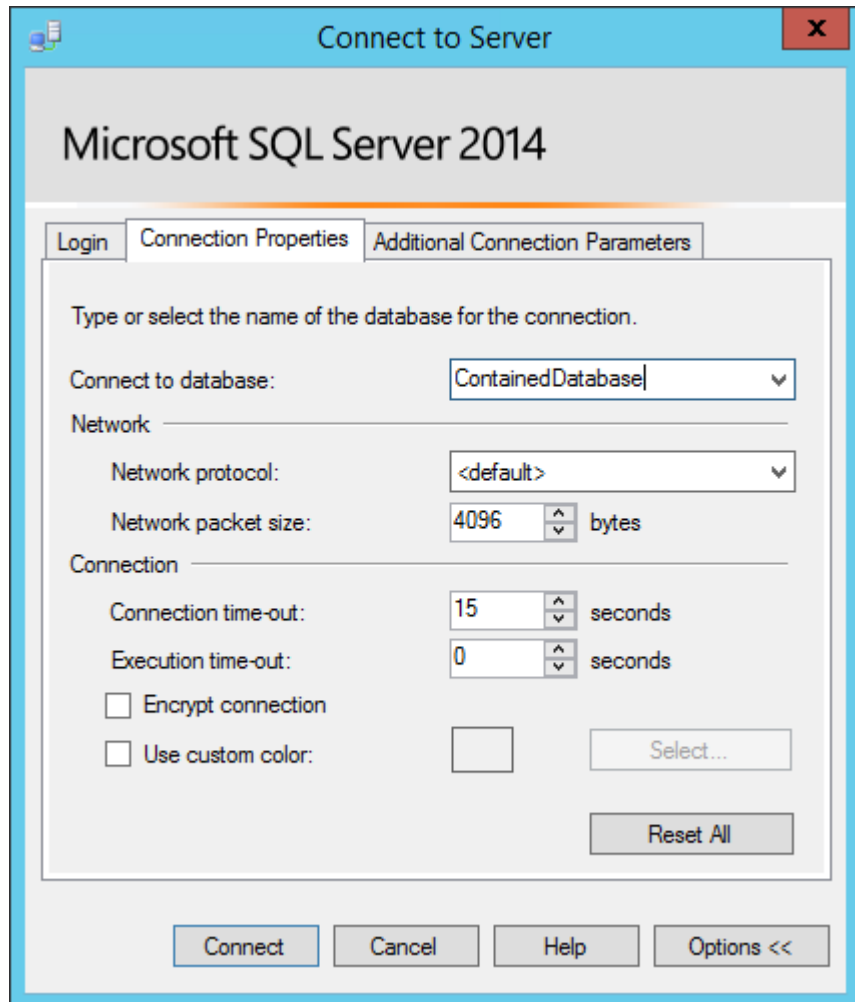
You now created a contained Database that John has read access to
 NOTE: John does not have a SQL login account, he only has a logon account to the database called 'ContainedDatabase'

Test John's user with the contained database

1. Click on **Connect** at the top of the **Object Explorer** and then **Database Engine...**
2. Ensure Authentication is set to **SQL Server Authentication** and the Server Name is **SQLONE**
3. In Login enter **John** and in Password enter **pass@word1**

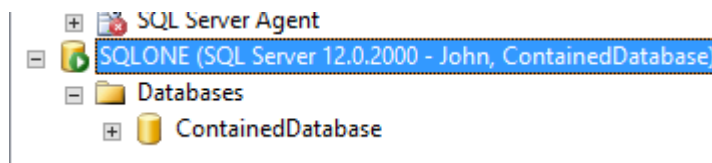


4. Click on **Options**
5. Select the Tab **Connection Properties** and in **Connect to database** enter **ContainedDatabase**



6. Click **Connect**
7. Expand **Databases**

Notice how John was able to access the Contained Database without having a SQL login account



8. Close all windows

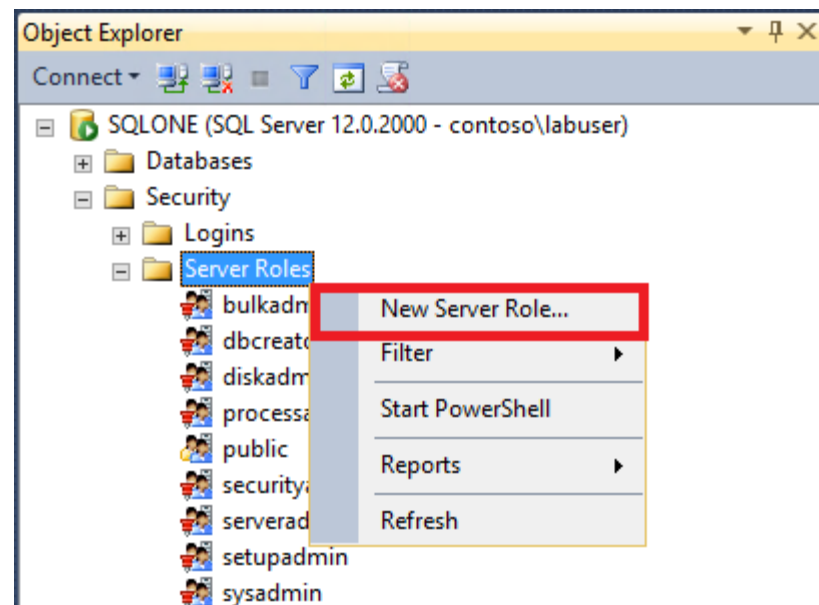
Scenario 3: User Defined Server Roles

User-Defined Server Roles increases flexibility and manageability, and it facilitates compliance through clearer separation of duties. It allows creation of server roles to suit different organizations that separate administrative duties according to roles. Roles also can be nested to allow more flexibility in mapping to hierarchical structures in organizations. User-Defined Server Roles also helps prevent organizations from using sysadmin (sa) for database administration. For example, a special database administration role can be created for common database administration without the ability to access user data

You will create a server role called PerfCheck for user Peter who needs to monitor SQL server performance. This role is allowed to see all the databases but not the data in them.

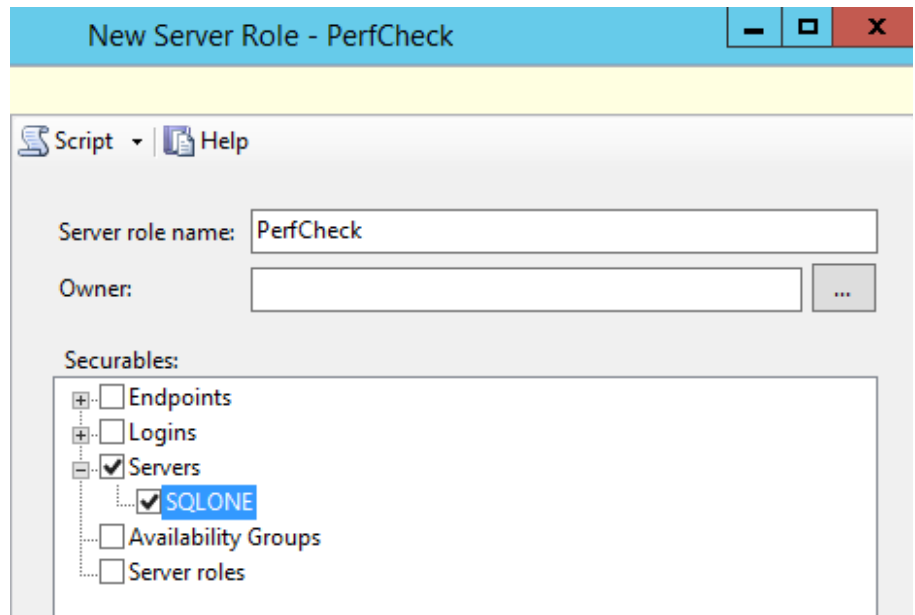
Create the new role PerfCheck

1. Ensure you have **SQL Server 2014 Management Studio** open with a connection to the **SQLONE** server (if you do not, open **SQL Server 2014 Management Studio** from the Windows Start screen, enter **Database Engine...** as the server type, **SQLONE** as the Server name and ensure **Windows Authentication** is selected before clicking **Connect**.)
2. Expand **Security** in the **Object Explorer**
3. Right click on **Server Roles** and select **New Server Roles...**



4. In Server Role Name enter **PerfCheck**

- Place a tick in **Servers** and then expand **Servers** and make certain **SQLONE** is selected



- Grant the following Permissions by placing a tick in the boxes under the **Grant** title next to:

- Alter Trace**

Permissions for SQLONE:

Explicit

Permission	Grantor	Grant	With Grant
Alter server state		<input type="checkbox"/>	<input type="checkbox"/>
Alter settings		<input type="checkbox"/>	<input type="checkbox"/>
Alter trace		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Authenticate server		<input type="checkbox"/>	<input type="checkbox"/>
Connect Any Database		<input type="checkbox"/>	<input type="checkbox"/>
Connect SQL		<input type="checkbox"/>	<input type="checkbox"/>

- Create DDL event notification**

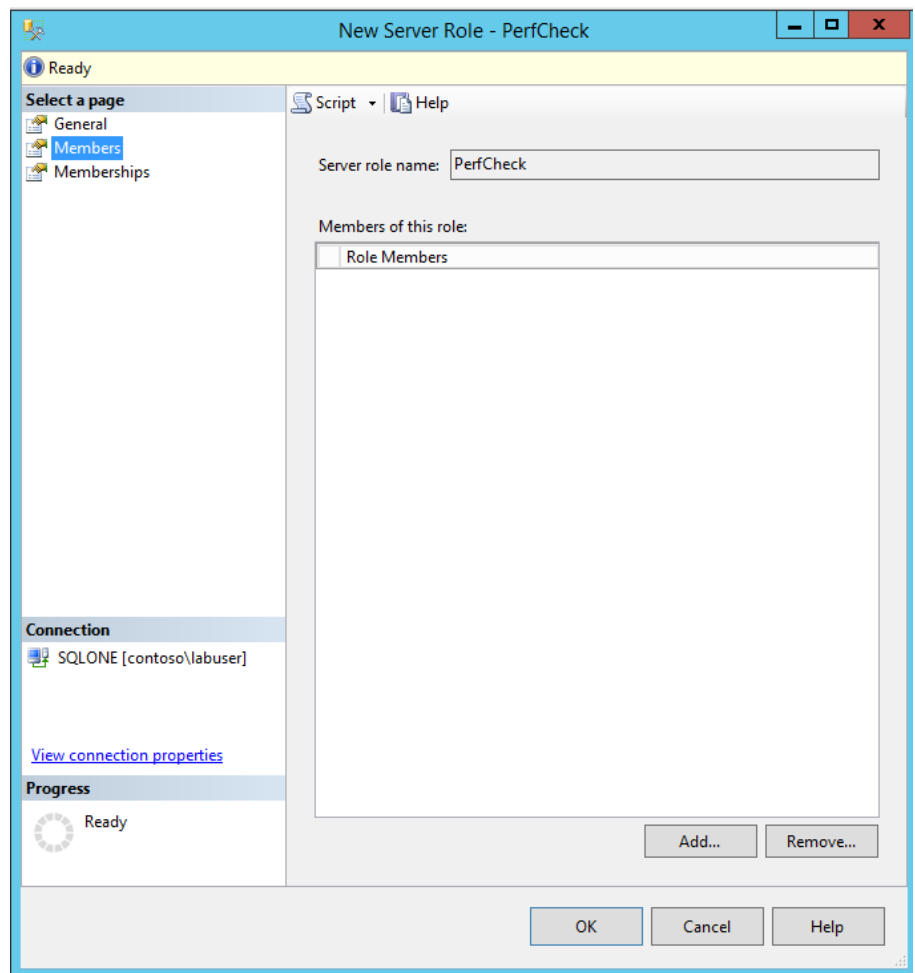
Explicit

Permission	Grantor	Grant	With Grant
Create availability group		<input type="checkbox"/>	<input type="checkbox"/>
Create DDL event notification		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Create endpoint		<input type="checkbox"/>	<input type="checkbox"/>
Create server role		<input type="checkbox"/>	<input type="checkbox"/>

- View Server State**

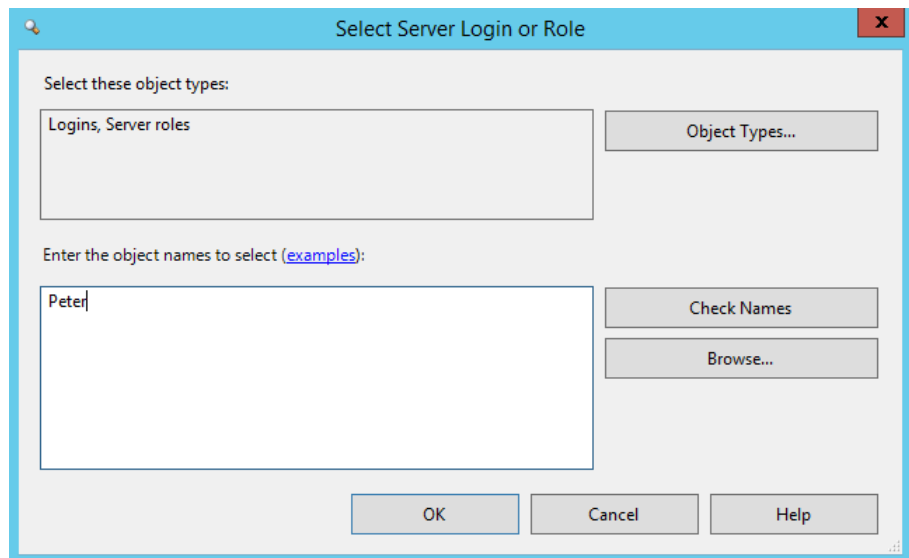
Permission	Grantor	Grant	With Grant
Shutdown		<input type="checkbox"/>	<input type="checkbox"/>
Unsafe assembly		<input type="checkbox"/>	<input type="checkbox"/>
View any database		<input type="checkbox"/>	<input type="checkbox"/>
View any definition		<input type="checkbox"/>	<input type="checkbox"/>
View server state		<input checked="" type="checkbox"/>	<input type="checkbox"/>

7. Click on **Members**



8. Click **Add...**

9. In **Enter the object names to select** enter **Peter**



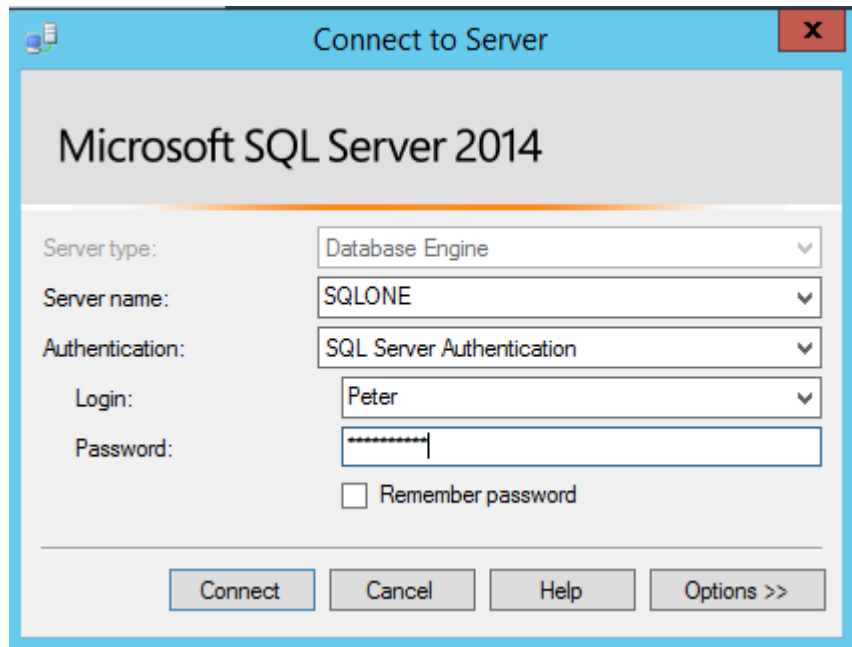
10. Click **OK**

11. Click **OK**

Check out the use of the new role for user Peter

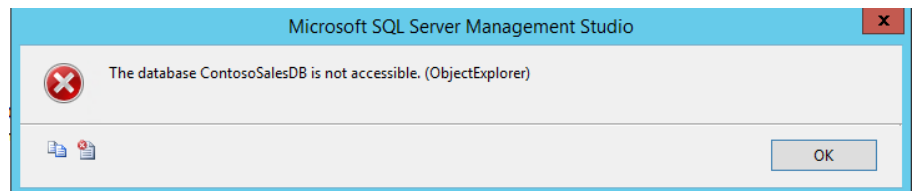
Now that you have created the new role, you are going to check that it is working

1. Click on **Connect** at the top of the **Object Explorer** then select **Database Engine**
2. Ensure Authentication is set to **SQL Server Authentication** and the Server Name is **SQLONE**
3. In Login enter **Peter** and in Password enter **pass@word1**



4. Click **Connect**
5. Expand Databases and then try to expand the database **ContosoSalesDB**

Notice the error message that pops up

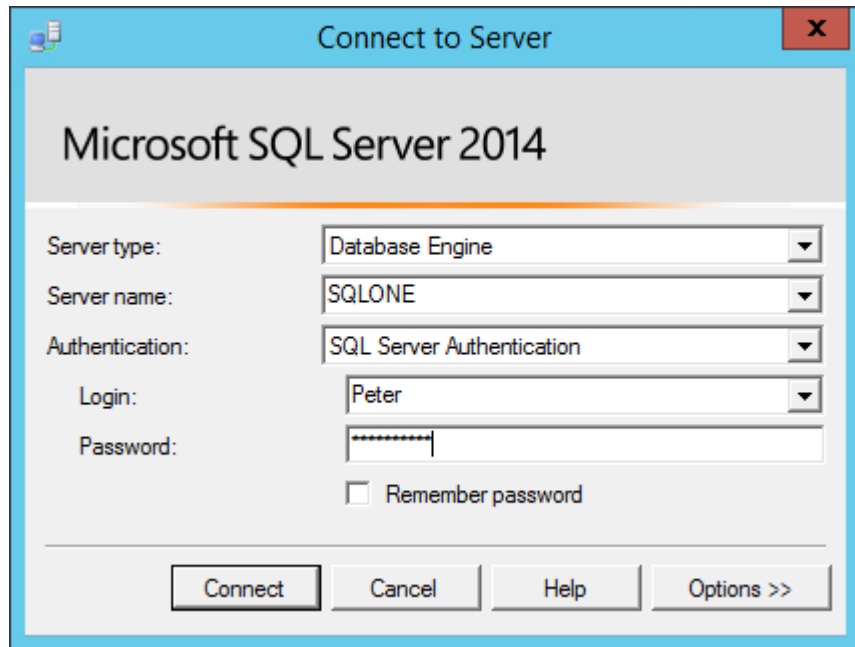


The role PerfCheck is allowed to see all the databases, but is not allowed access to the data in them. However the role does allow appropriate operations to monitor SQL server performance without the user requiring sysadmin (sa) rights

6. Click **OK**

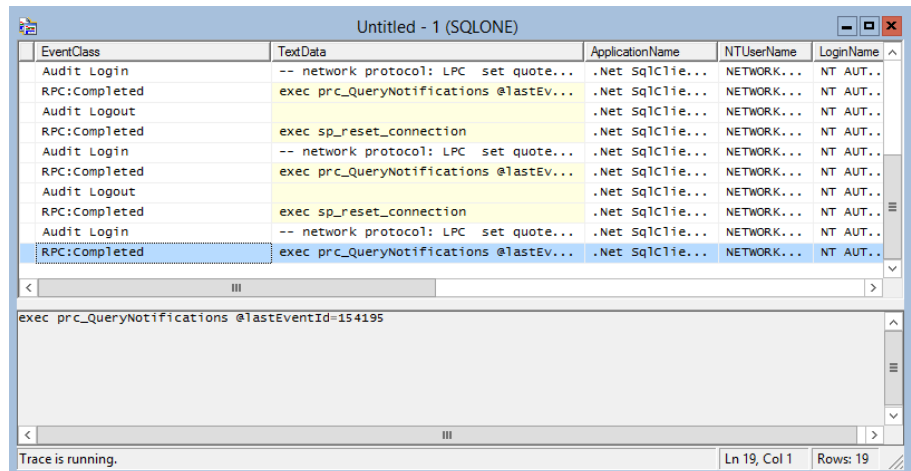
Test user Peter for ability to perform a SQL Trace


1. From the menu click on **Tools** and then select **SQL Server Profiler**
2. Ensure Authentication is set to **SQL Server Authentication**
3. In Login enter **Peter** and in **Password** enter **pass@word1**



4. Click **Connect**
5. Click **Run** to start a Trace

Notice how the traces are captured from all databases in the server, even though the role PerfCheck does not have sysadmin (sa) rights and access to user data



6. Stop the Trace by clicking on 
7. Close the **SQL Server Profiler**
8. Close all windows

Scenario 4: New Permissions

You are going to use the following new permissions available in SQL Server 2014 that can be used to limit the access of a person assigned the sysadmin role.:

- CONNECT ANY DATABASE
- SELECT ALL USER SECURABLES
- IMPERSONATE ANY LOGON

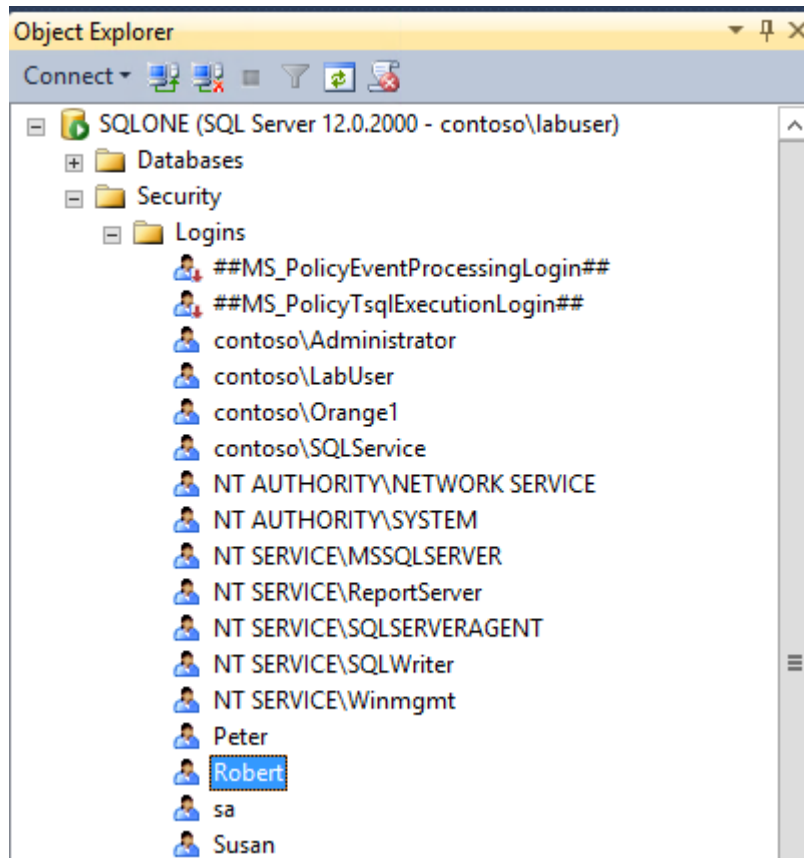
CONNECT ANY DATABASE

A new server level permission. Grant CONNECT ANY DATABASE to a user that must connect to all databases that currently exist and to any new databases that might be created in future. Does not grant any permission in any database beyond connect.

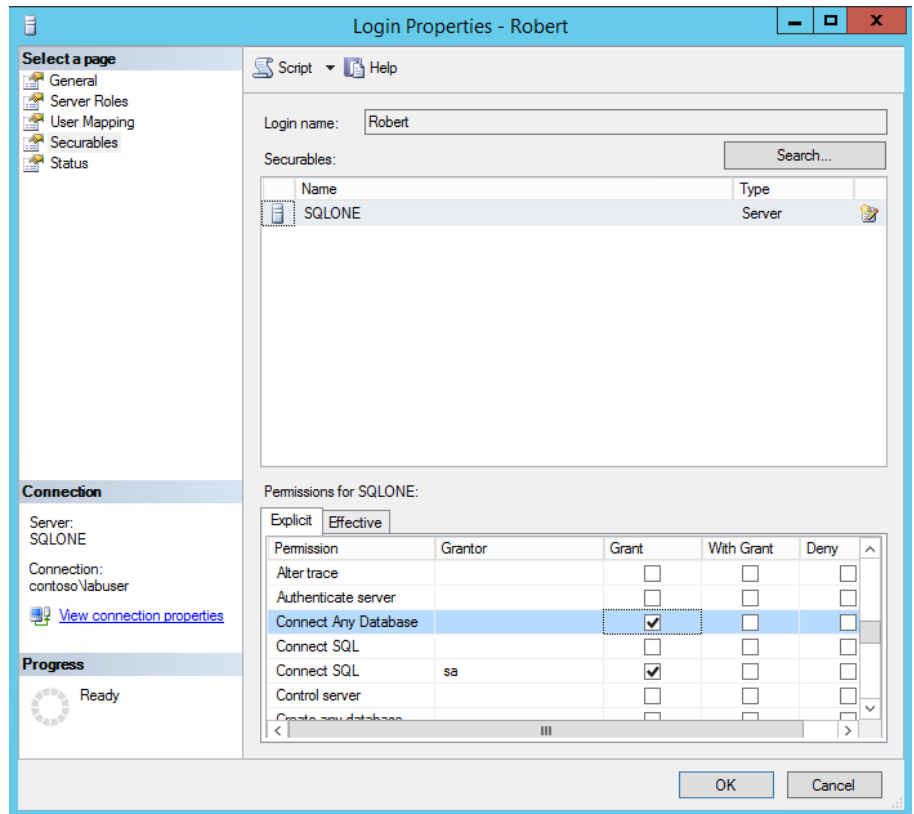
Previously, any increase in the number of databases that the data administration team are required to support has required either granting server level role or creating a login for the administrator on each database. However now with the CONNECT ANY DATABASE this is no longer required as this grants the user the ability to connect and view any database on the server. With a new data administrator (Robert) soon to start you decide to use CONNECT ANY DATABASE to give him access to the databases on the server.

You will grant Robert the CONNECT TO ANY DATABASE permission

1. Ensure you have **SQL Server 2014 Management Studio** open with a connection to the **SQLONE** server (if you do not, open **SQL Server 2014 Management Studio** from the Windows Start screen, enter **Database Engine...** as the server type, **SQLONE** as the Server name and ensure **Windows Authentication** is selected before clicking **Connect**.)
2. Expand **Security** and then **Logins** in the **Object Explorer**



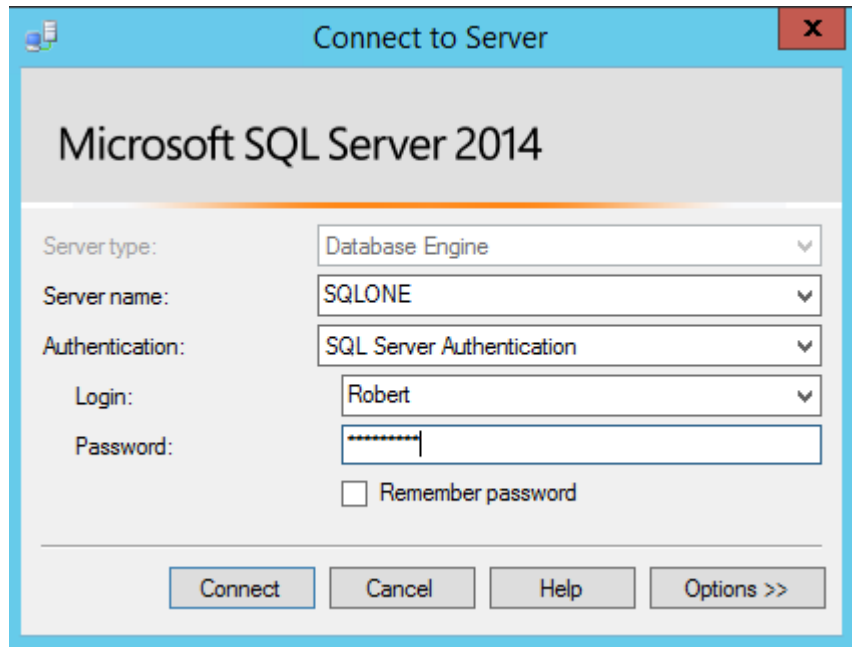
3. Right-click on the login **Robert** and select **Properties**
4. Click on **Securables** and then grant the permission **Connect Any Database**



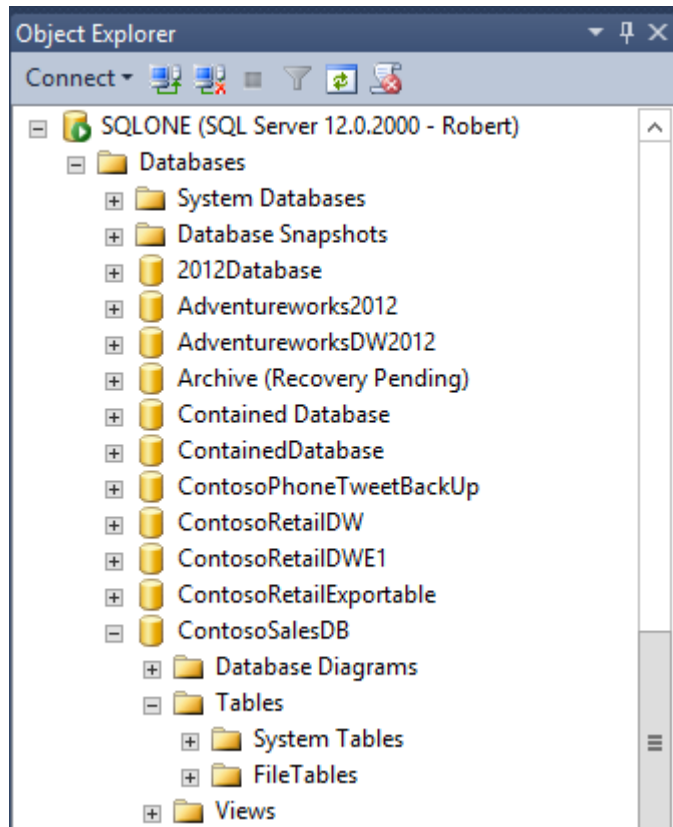
5. Click **OK**

Test the **CONNECT TO ANY DATABASE** permission is working for **Robert**

1. From **SQL Server 2014 Management Studio** in **Object Explorer**, click on **Connect to SQLONE** and then **Database Engine...**
2. Ensure Authentication is set to **SQL Server Authentication** and in Login enter **Robert**
3. In Password enter **Password1**

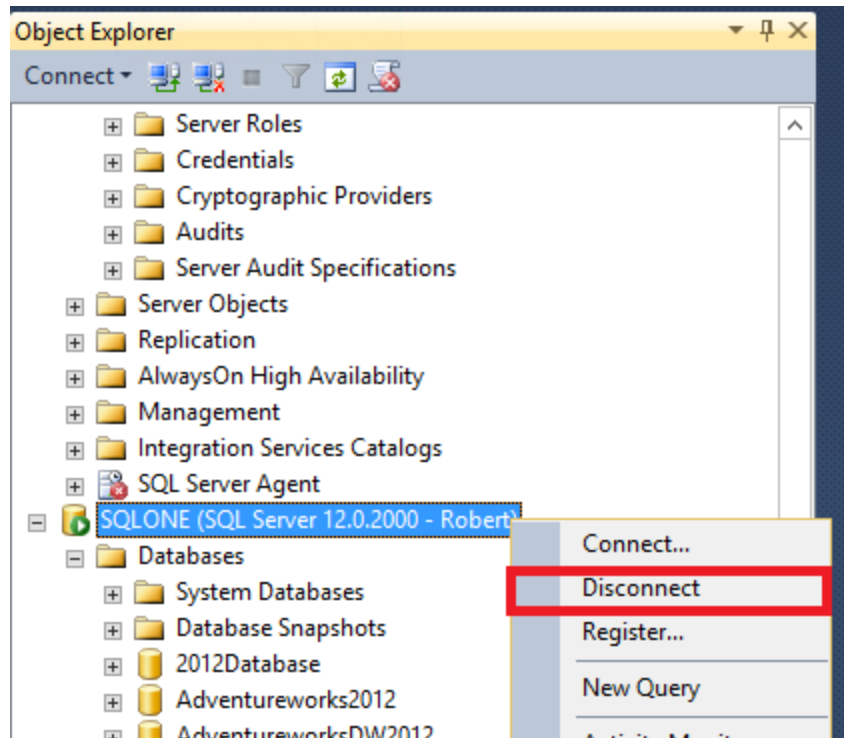


4. Click **Connect**
5. Expand **Databases**
6. Expand **ContosoSalesDB**
7. Expand **Tables**



Although Robert is not a sysadmin (sa) he is able to see all the Databases with the CONNECT ANY DATABASE permission, but he cannot see the data in the databases

8. Right click on **SQLONE** for Robert in the **Object Explorer** and click the **Disconnect** command.

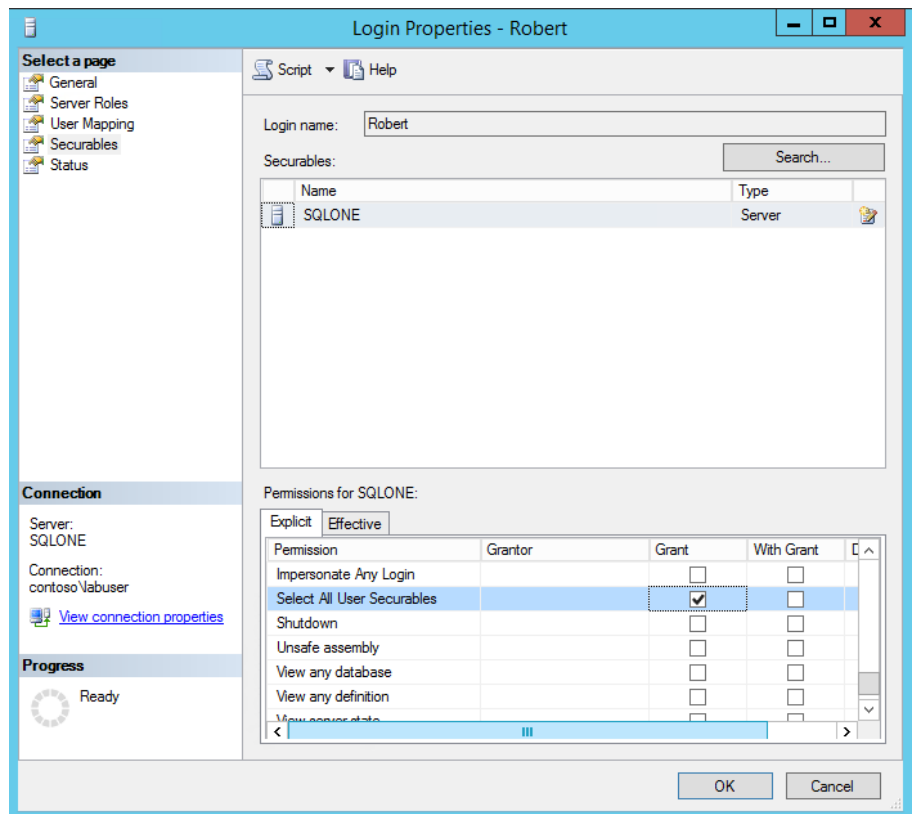


SELECT ALL USER SECURABLES

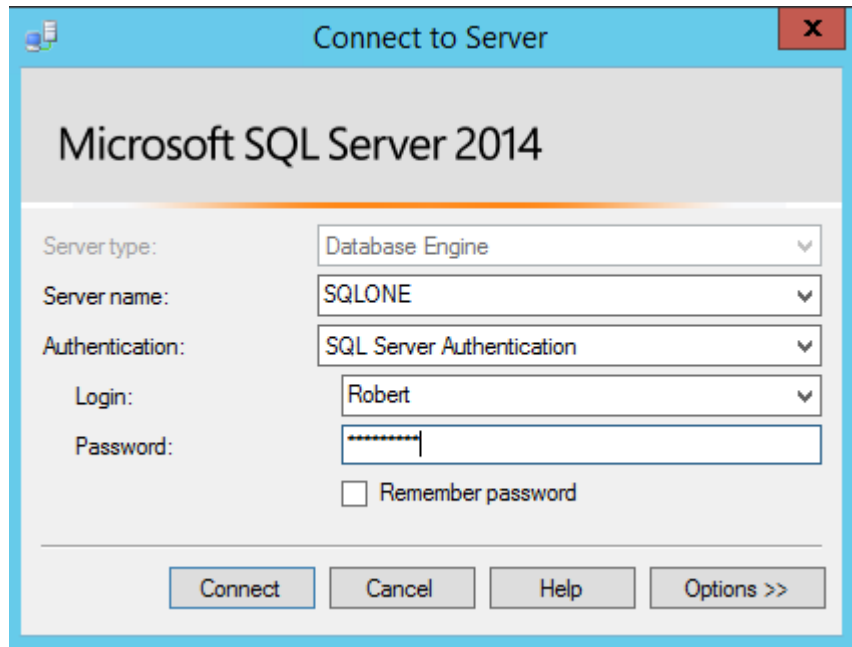
This is an extension of the CONNECT ANY DATABASE permission so that the user is also able to see the data in the databases.

You will use Select All User Securables permission to allow user Robert to view data.

1. Right click on **Robert** login under **Security** -> **Logins** in **Object Explorer** and click on **Properties**.
2. Click on **Securables** and then grant the permission **Select All User Securables**

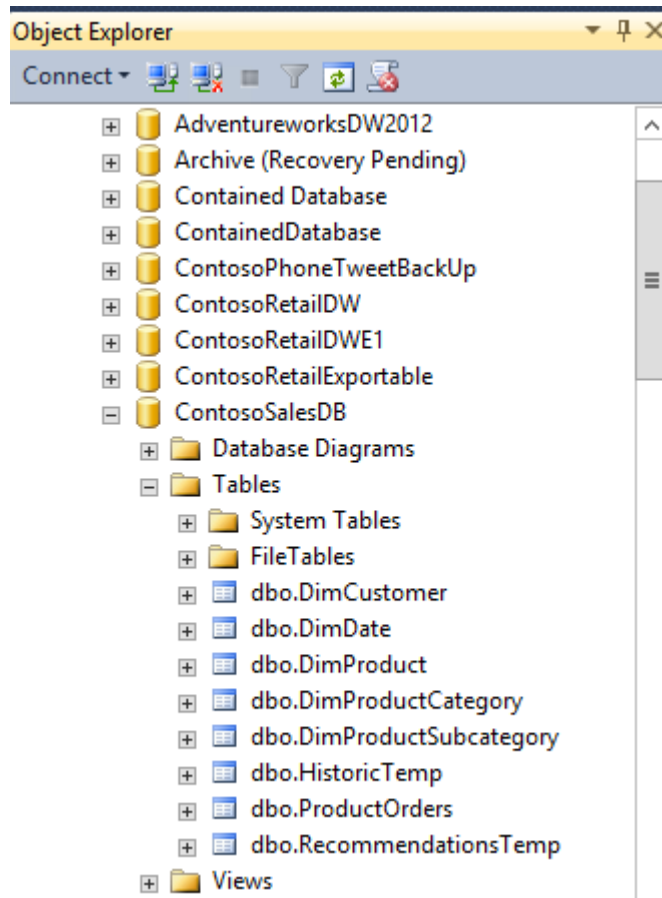


3. Click **OK**
4. From **SQL Server 2014 Management Studio** in **Object Explorer**, click on **Connect to SQLONE** and then **Database Engine...**
5. Ensure Authentication is set to **SQL Server Authentication** and in Login enter **Robert**
6. In Password enter **Password1**



7. Click **Connect**
8. Expand **Databases**
9. Expand **ContosoSalesDB**
10. Expand **Tables**

As Robert now has the permissions `CONNECT ANY DATABASE & SELECT ALL USER SECURABLES` he can now see all the data in the databases



IMPERSONATE ANY LOGIN

The EXECUTE AS statement allows the execution context of a session to be switched to the specified login or user name. However for this to work the user that is executing the EXECUTE AS statement must have the sysdamin (sa) server role. The new IMPERSONATE ANY LOGON permission removes this restriction

To reduce security concerns the data administration team are keen to reduce the number of users that are members of the sysadmin (sa) role. You decide to give Robert the new IMPERSONATE ANY LOGIN permission and is also going to test that it is working

1. In **Object Explorer** and select the **Robert** connection.
2. Expand **Databases** in the **Object Explorer** under the connection for Robert and then right-click on **ContosoSalesDB** and select **New Query**
3. In the query window enter the following script

```
Execute as user = 'Susan'
```

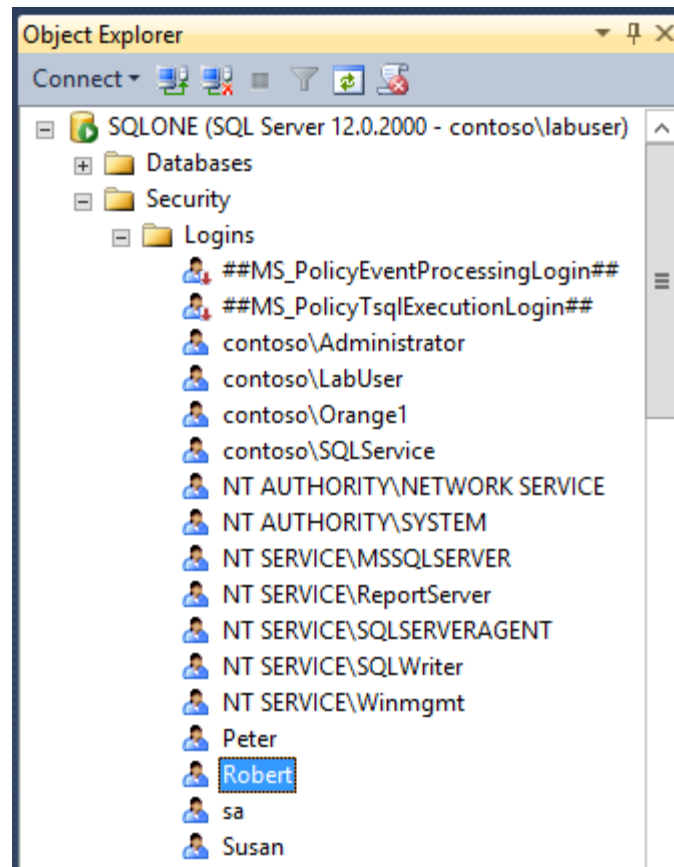
4. Click **Execute**

Notice the error message stating that the principal cannot be impersonated. As Robert is not a sysadmin (sa) and does not have the new IMPERSONATE ANY LOGIN permission he is not able to impersonate Susan

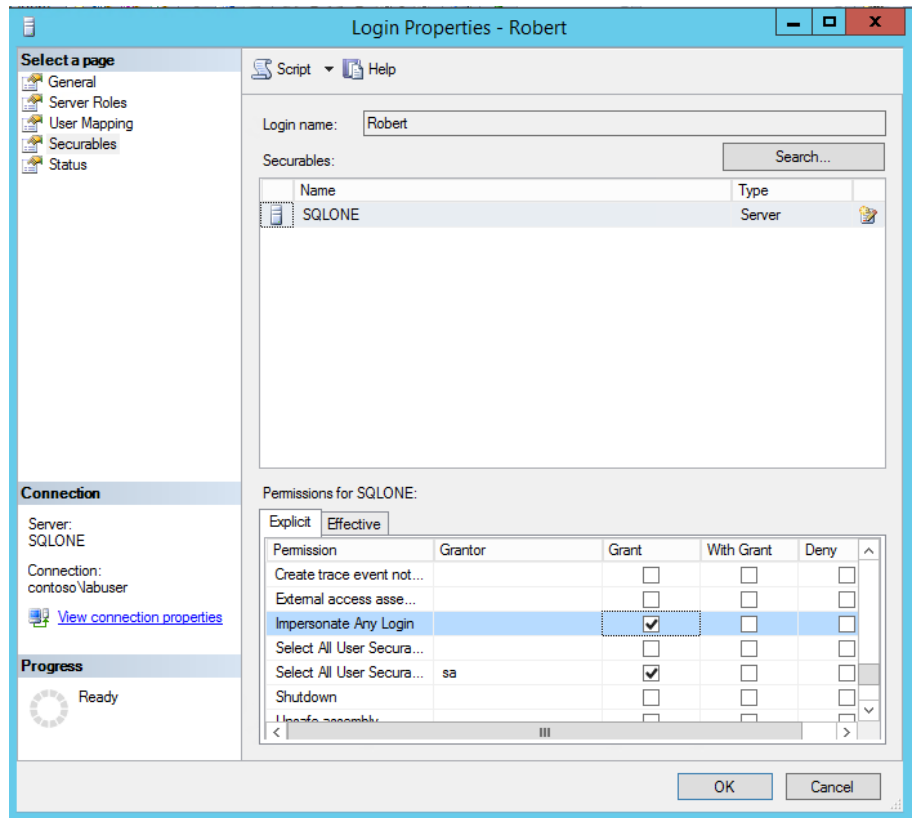
this type of principal cannot be impersonated,

You will now give Robert the IMPERSONATE ANY LOGIN permission

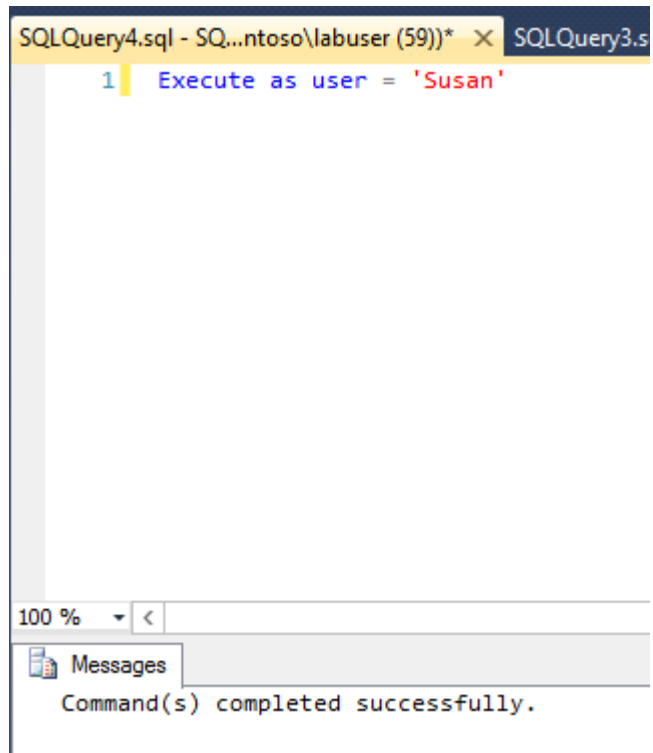
5. In the **Object Explorer**, under the main **SQLONE** connection expand **Security** and then **Logins** (if this connection does not exist, open a new connection dialog box and connect to the **SQLONE** database engine using **Windows Authentication**.)



6. Right-click on the user **Robert** and select **Properties**
7. Click on **Securables** and then Grant Robert the permission **IMPERSONATE ANY LOGIN**



8. Click **OK**
9. Go back to your query window (in step 1) and Click **Execute** again



The screenshot shows a SQL Server Enterprise Manager window with a query editor. The title bar indicates the file is 'SQLQuery4.sql' and the user is 'labuser (59)'. The query editor contains a single line of T-SQL code: 'Execute as user = 'Susan''. Below the query editor, a 'Messages' pane displays the output: 'Command(s) completed successfully.' The zoom level is set to 100%.

```
1 Execute as user = 'Susan'
```

100 % <

Messages
Command(s) completed successfully.

As Robert now has IMPERSONATE ANY LOGIN permission he was able to impersonate Susan without having sysadmin (sa) rights.

Terms of use

© 2014 Microsoft Corporation. All rights reserved.

By using this Hands-on Lab, you agree to the following terms:

The technology/functionality described in this Hands-on Lab is provided by Microsoft Corporation in a “sandbox” testing environment for purposes of obtaining your feedback and to provide you with a learning experience. You may only use the Hands-on Lab to evaluate such technology features and functionality and provide feedback to Microsoft. You may not use it for any other purpose. You may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, or sell this Hands-on Lab or any portion thereof.

COPYING OR REPRODUCTION OF THE HANDS-ON LAB (OR ANY PORTION OF IT) TO ANY OTHER SERVER OR LOCATION FOR FURTHER REPRODUCTION OR REDISTRIBUTION IS EXPRESSLY PROHIBITED.

THIS HANDS-ONLAB PROVIDES CERTAIN SOFTWARE TECHNOLOGY/PRODUCT FEATURES AND FUNCTIONALITY, INCLUDING POTENTIAL NEW FEATURES AND CONCEPTS, IN A SIMULATED ENVIRONMENT WITHOUT COMPLEX SET-UP OR INSTALLATION FOR THE PURPOSE DESCRIBED ABOVE. THE TECHNOLOGY/CONCEPTS REPRESENTED IN THIS HANDS-ON LAB MAY NOT REPRESENT FULL FEATURE FUNCTIONALITY AND MAY NOT WORK THE WAY A FINAL VERSION MAY WORK. WE ALSO MAY NOT RELEASE A FINAL VERSION OF SUCH FEATURES OR CONCEPTS. YOUR EXPERIENCE WITH USING SUCH FEATURES AND FUNCTIONALITY IN A PHYSICAL ENVIRONMENT MAY ALSO BE DIFFERENT.

FEEDBACK. If you give feedback about the technology features, functionality and/or concepts described in this Hands-on Lab to Microsoft, you give to Microsoft, without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft software or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement.

MICROSOFT CORPORATION HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE HANDS-ON LAB , INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. MICROSOFT DOES NOT MAKE ANY ASSURANCES OR

REPRESENTATIONS WITH REGARD TO THE ACCURACY OF THE RESULTS, OUTPUT THAT DERIVES FROM USE OF THE VIRTUAL LAB, OR SUITABILITY OF THE INFORMATION CONTAINED IN THE VIRTUAL LAB FOR ANY PURPOSE.

DISCLAIMER

This lab contains only a portion of new features and enhancements in Microsoft SQL Server 2014. Some of the features might change in future releases of the product. In this lab, you will learn about some, but not all, new features.