Contents

[TYPES OF DATABASES IN SQL SERVER 1](#_Toc165194692)

[CONTAINED DATABASE EXAMPLE 2](#_Toc165194693)

[**Step 1: Setting Up Your Environments** 2](#_Toc165194694)

[**Step 4: Creating Database Users** 3](#_Toc165194695)

[STRETCH DATABASE EXAMPLES 5](#_Toc165194696)

[Manage and Monitor Stretch Database 11](#_Toc165194697)

[Reference URLs 14](#_Toc165194698)

# TYPES OF DATABASES IN SQL SERVER

A DATABASE IS USED IN SQL SERVER TO STORE DATA & OBJECTS. A Database in Sql Server consists of Tables, Stored Procedures, Views, Functions, Synonyms etc

SQL SERVER SUPPORTS DIFFERENT Types of Databases

|  |  |  |  |
| --- | --- | --- | --- |
| **SYSTEM DEFINED DATABASE** | **USER DEFINED DATABASE** | **CONTAINED DATABASE** | **STRETCH DATABASE** |

|  |  |
| --- | --- |
| SYSTEM DEFINED DATABASE | This database is already created by sql server at the time of installation and is used to manage the server configuration. The System Databases which are predefined in Sql Server are master, model,msdb and tempdb. |
| USER DEFINED DATABASE | These databases are created by the user for storing user data. User will design and model these databases and will store data as per the defined model. |
| CONTAINED  DATABASE(2012) | This was introduced in Sql Server 2012. A contained database is one that is isolated from other databases and from the SQL Server instance that hosts the database. The database maintains much of its own metadata and supports database-level authentication, eliminating the need for server-based logins. |
| STRETCH  DATABASE (2016) | This was introduced in Sql Server 2016.Stretch Database is a feature of SQL Server where data can be split between on-premises storage and cloud storage. With Stretch Database, cold, historical data is kept in the cloud and active data is kept on-premises for maximum performance. |

# CONTAINED DATABASE EXAMPLE

Microsoft introduced contained databases in SQL Server 2012.

**A contained database is one that is isolated from other databases and from the SQL Server instance that hosts the database.**

**The database maintains much of its own metadata and supports database-level authentication, eliminating the need for server-based logins.**

As a result, a contained database is more portable than a traditional, non-contained database.

It can also simplify database development and administration, as well as make it easier to support Always On Availability Groups.

Controlling access to a contained database is similar to a non-contained database, except for a few important differences

**Step 1: Setting Up Your Environments**

Before you can implement a contained database, you must enable the SQL Server instance to support this feature, if it’s not already enabled. To use T-SQL to enable contained databases, run the following EXECUTE statement:

|  |  |
| --- | --- |
|  | EXEC sp\_configure 'contained database authentication', 1;  GO  RECONFIGURE;  GO |

**Step 2:** Create the ImportSales1 contained database, using the following T-SQL script:

|  |  |
| --- | --- |
|  | USE master;  GO  DROP DATABASE IF EXISTS ImportSales1;  GO  CREATE DATABASE ImportSales1  CONTAINMENT = PARTIAL;  GO |

When you create a database, you can specify that it should be contained by including the CONTAINMENT clause in the CREATE DATABASE statement and set its value to PARTIAL. The default value is NONE, which disables the contained database feature. The PARTIAL value is used because SQL Server supports only partially contained databases, as opposed to fully contained databases. Currently, SQL Server does not support fully contained databases.

A partially contained database allows you to implement uncontained features that cross the database boundary. For example, you can create a database user that is linked to a SQL Server login in a partially contained database. Fully contained databases do not allow the use of uncontained features.

After you create the ImportSales1 database, you can add tables and then populate them, just like you can with a non-contained database

**Step 3 : CREATE TABLES IN THE CONTAINED DATABASE**

|  |  |
| --- | --- |
|  | USE ImportSales1;  GO  CREATE SCHEMA Sales;  GO  CREATE TABLE Sales.Customers(    CustID INT NOT NULL PRIMARY KEY,    Customer NVARCHAR(100) NOT NULL,    Contact NVARCHAR(50) NOT NULL,    Category NVARCHAR(50) NOT NULL);  GO  INSERT INTO Sales.Customers(CustID, Customer, Contact, Category)  SELECT CustomerID, CustomerName,    PrimaryContact, CustomerCategoryName  FROM WideWorldImporters.Website.Customers  WHERE BuyingGroupName IS NOT NULL;  GO |

**Step 4: Creating Database Users**

In SQL Server, you can create users that are specific to a contained database and not linked to server-level logins. Contained users make it possible to maintain the separation between the contained database and the SQL Server instance, so it’s easier to move the database between instances.

SQL Server supports two types of contained users: *SQL user with password* and *Windows user*. The password-based user is a database user that is assigned a password for authenticating directly to the database. The user is not associated with any Windows accounts.

To create a password-based user, you must include a WITH PASSWORD clause in your CREATE USER statement. For example, the following CREATE USER statement defines a user named sqluser02 and assigns the password tempPW@56789 to the user:

|  |  |
| --- | --- |
|  | USE ImportSales1;  GO  CREATE USER sqluser02  WITH PASSWORD = 'tempPW@56789';  GO |

When a password-based user tries to access a contained database, the user account is authenticated at the database level, rather than the server level. In addition, all authorization granted through assigned permissions is limited to the database.

After you’ve created your contained users, you can grant, deny, or revoke permissions just like you can with any database users. For example, the following GRANT statement grants the SELECT permission on the Sales schema to both users:

|  |  |
| --- | --- |
|  | GRANT SELECT ON SCHEMA::Sales TO sqluser02, [win10b\winuser02];  GO |

You can also add contained users to fixed and user-defined database roles, and assign permissions to the user-defined roles.

|  |
| --- |
| STEP 1:  EXEC sp\_configure 'contained database authentication', 1;  GO  RECONFIGURE;  GO  STEP 2 :  USE master;  GO  DROP DATABASE IF EXISTS ImportSales1;  GO  CREATE DATABASE ImportSales1  CONTAINMENT = PARTIAL;  GO  STEP 3:  USE ImportSales1;  GO  CREATE SCHEMA Sales;  GO  CREATE TABLE Sales.Customers(  CustID INT NOT NULL PRIMARY KEY,  Customer NVARCHAR(100) NOT NULL,  Contact NVARCHAR(50) NOT NULL,  Category NVARCHAR(50) NOT NULL);  GO  insert into Sales.Customers values(1,'sunil','8877991122','IT')  insert into Sales.Customers values(2,'KAPIL','7766554433','HR')  STEP 4:  USE ImportSales1;  GO  CREATE USER sqluser02  WITH PASSWORD = 'tempPW@56789';  GO  enosis@123  TESTING THE CONTAINED DATABASE USING THE USER  EXECUTE AS USER = 'sqluser02';  SELECT \* FROM ImportSales1.Sales.Customers;  REVERT;  GO |

# STRETCH DATABASE EXAMPLES

Microsoft has introduced an amazing benefit to organizations of all sizes with its Stretch Database offering in SQL Server 2016. Stretch Database allows companies to use on-premises instances of SQL Server to "stretch" their data into the Microsoft Azure SQL Database. Stretch DB is relatively simple to configure and implement on your environment.

In this example we will show you how to migrate the historical data transparently and securely to the Microsoft Azure cloud, which provides for cost effective availability of cold data while keeping it safe and unchanged. You will see how to stretch your on-premises data to the cloud but still have the ability to query the fully accessible and online table and to monitor and manage the stretched tables.

To implement a stretch database,you must enable the feature on both the instance and database.

STEP 1:

Let's enable the Stretch Database feature on the Instance by running the below command.

--enable instance for stretch

EXEC sp\_configure ‘remote data archive’, ‘1’

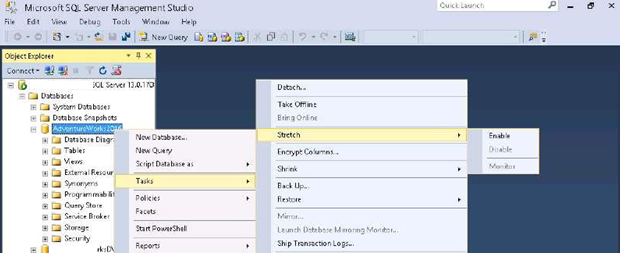
GO

RECONFIGURE

GO

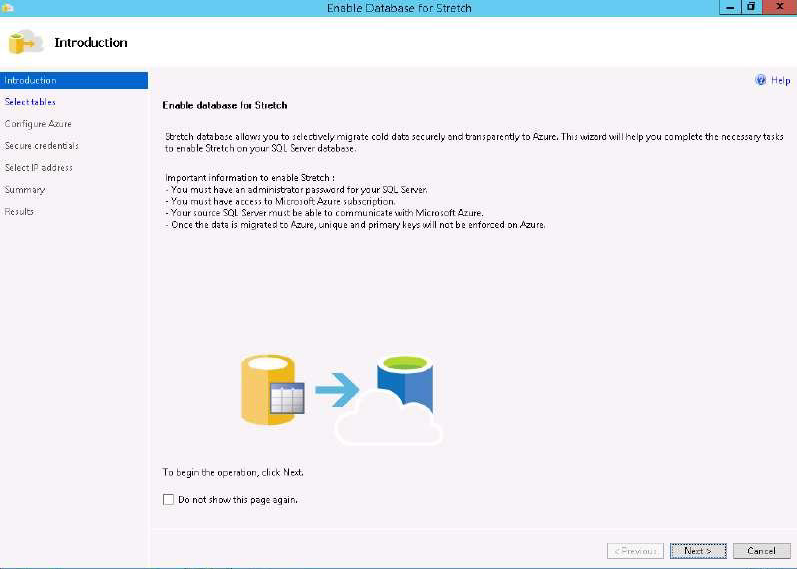
STEP 2:

After enabling the Stretch Database feature on the Instance, you need to enable the feature on database, as shown in Fig. 1. From SQL Server Management Studio (SSMS), right click the database that holds the table(s) you want to stretch to Azure and select Tasks > Stretch > Enable.



**Fig. 1 - Start the stretch wizard**

**Enabling this will open the Enable Database for Stretch Wizard, as shown in Fig. 2. You can select the entire table contents or you can select specific rows to stretch.**



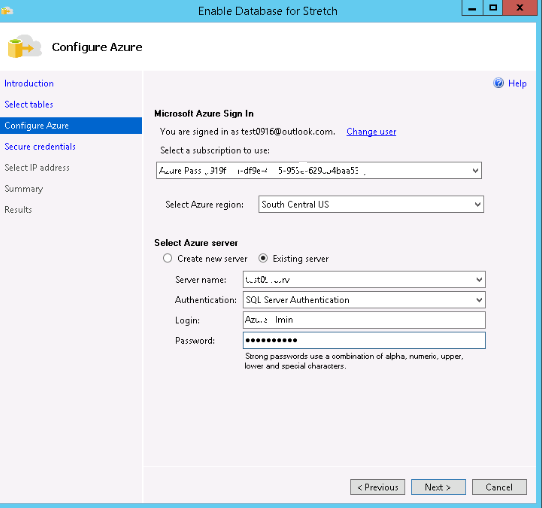
**Fig. 2 - The Stretch Database wizard**

For this example, we have selected the entire OrderTracking table to be stretched, shown in Fig. 3.



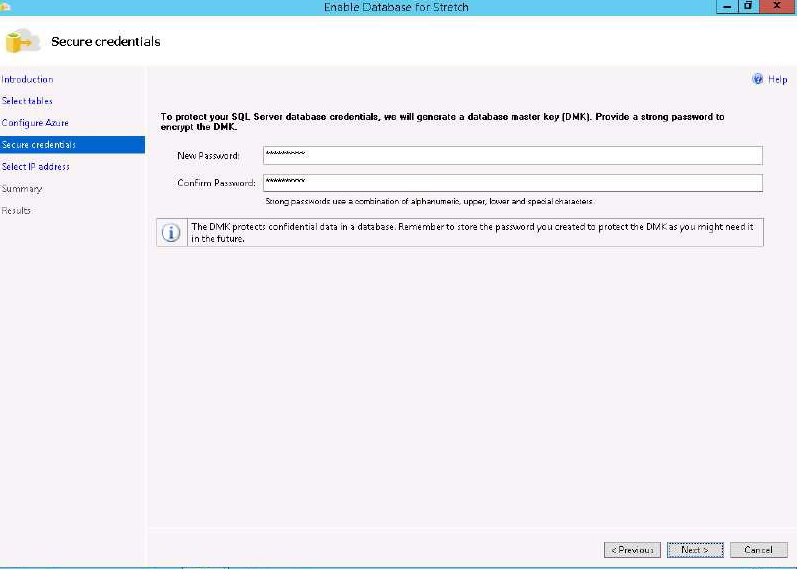
**Fig. 3 - Select the contents to stretch**

The next step is to Configure Azure. Sign in to Microsoft Azure with a Microsoft account and select the existing Azure subscription and Azure region to use for Stretch Database. Specify whether you want to use an existing server or create a new Azure server. We have selected an existing server in this example.



**Fig. 4 - Enter credentials for Azure and choose a server**

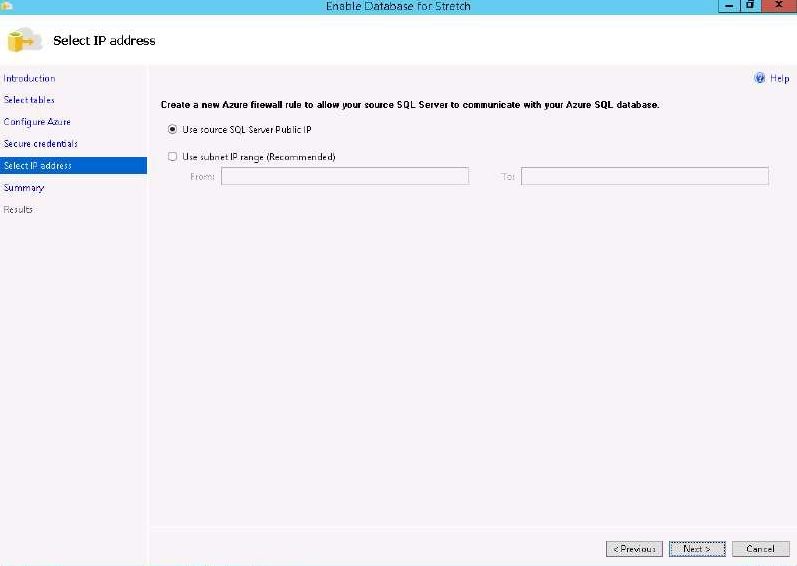
In order to stretch a database table to Azure, **the database must have a database master key (DMK). Specify (and save) the password for the DMK by creating the credential in the Wizard as follows on the Secure Credentials page as shown in Fig. 5.**



**Fig. 5 - Set a password for the DMK**

On the Select IP Address page, you can select the subnet IP address range, or the public IP address of your SQL Server, to create a firewall rule on Azure that lets SQL Server communicate with the remote Azure server.

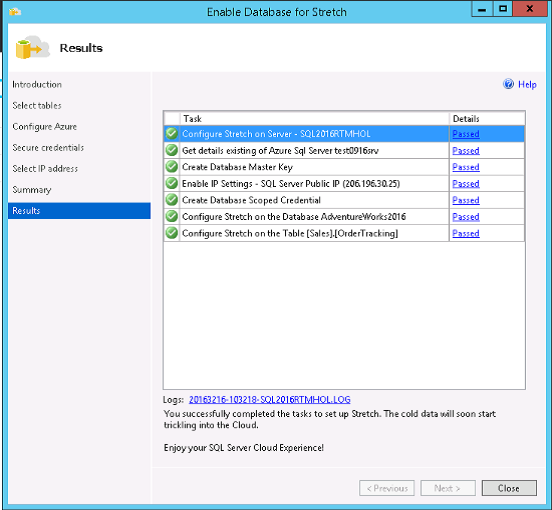
The IP address or addresses that you provide on this page tell the Azure server to allow incoming data, queries, and management operations initiated by SQL Server to pass through the Azure firewall. The wizard doesn't change anything in the firewall settings on the SQL Server.



**Fig. 6 - Choose an IP address**

After specifying the IP addresses, click next for the Summary and Results pages as shown in Fig. 7.

In the Summary page, review the details, and then click Finish. This will start provisioning the database to the Azure SQL Stretch database server. When complete, the summary page will show a list of tasks and their status. Make sure that all tasks have Passed status, and then click Close to complete the setup and close the wizard. You have successfully completed the tasks to set up Stretch.

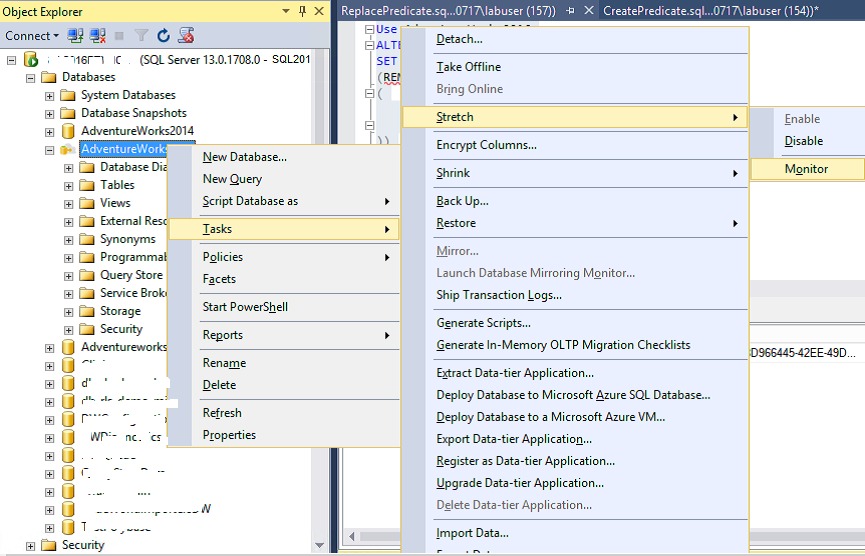


**Fig. 7 - Confirm the settings**

## Manage and Monitor Stretch Database

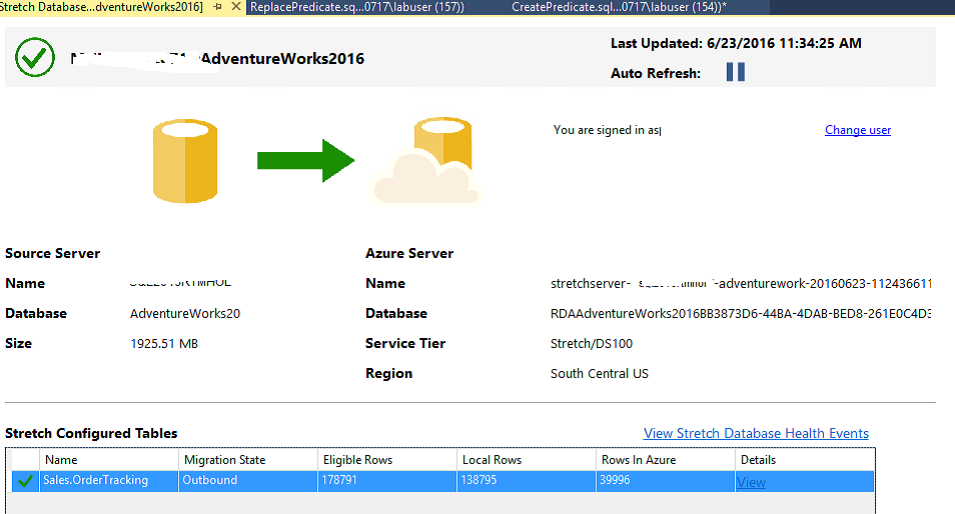
You have just enabled a SQL Stretch Database for the AdventureWorks2016 database. You can now manage and monitor its activities using the monitoring option in Stretch Database. This provides a dashboard view of information on the remote Microsoft Azure SQL Stretch database and tables where your cold data is stored. You can also check how many batches and rows of data have migrated. Finally, you can access details about the Stretch configured tables, such as how many rows are locally stored and how many have stretched to Microsoft Azure.

Open Stretch Monitor for the Stretch Database Monitor option, right-click Database AdventureWorks2016. Select Task and Stretch, and then choose Monitor as shown in Fig. 8.



**Fig. 8 - Monitoring status**

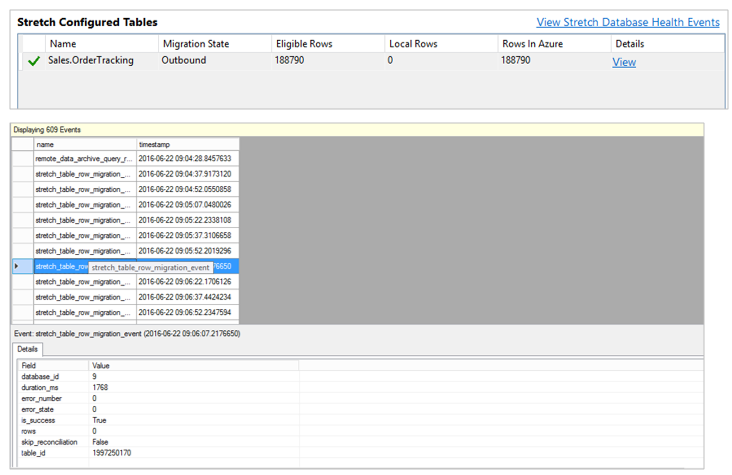
Selecting the monitor option from the stretch task will open the dashboard view of the database that has been stretched to Microsoft Azure. It provides details such as Microsoft Azure SQL server name, database name, and database size as shown in Fig. 9.



**Fig. 9 - Viewing a report on your data**

You can also check the status of the Stretch-enabled table (such as stretch eligible row, number of rows on-premises, and Azure SQL Stretch database) and view the event health by clicking View Stretch Database Health Event, or check event details for troubleshooting as shown in Fig. 10. Event details includes the error details and state. You can use this information to diagnose the error.

For more information and documentation on how to monitor or troubleshoot Stretch Database, go to the MSDN site: [**https://msdn.microsoft.com/en-us/library/mt484151.aspx**](https://msdn.microsoft.com/en-us/library/mt484151.aspx).



**Fig. 10 - Viewing health of the feature**

Summary

This article covered very basic information on how to set up a Stretch Database.Its an easy way to migrate archive data to Microsoft Azure and lets you dynamically stretch warm and cold transactional data from Microsoft SQL Server 2016 to Microsoft Azure. Straightforward implementation, convenient monitoring are characteristics that makes it a preferable solution for institutions that are required to keep old data for a long and even indefinite period of time.

# Reference URLs

https://www.sqlservercentral.com/articles/implementing-stretch-database