COHESITY

Version 1.3

July 2024

Use Cohesity DB Migration to Add an MS SQL Database to MS SQL AG

Use the Cohesity SQL Database Migration Solution to Add a Database to a SQL Server Always On Availability Group (AG)

ABSTRACT

Go beyond data management. Use your Cohesity backups to seed a database for SQL AG. In this guide, we use Cohesity's SQL DB Migration feature to prepare a standalone SQL database for its addition into a SQL AG group.

Table of Contents

Figures

Tables

Introduction to Cohesity MS SQL Database Migration

SQL Always On Availability Groups (AG) is fast becoming the SQL replication technology of choice. Leverage your snapshots to introduce a database into an existing SQL AG group. To date, this has often been tricky, but now, Cohesity DB Migration makes it easy.

Partnering Cohesity with Microsoft SQL Server® has three major phases:

- Discovery
- Deployment Considerations
- Solution Procedures

You are at the Solution Procedures phase of pairing Cohesity with Microsoft SQL Server.

This guide focuses on the process of preparing, seeding, and introducing a SQL database into an existing SQL AG group, and is aimed at IT and database administrators who manage data protection for Microsoft SQL Servers.

NOTE: You must contact Cohesity Support to have the Cohesity DB Migration feature enabled before you can use it.

Think About Other Use Cases

The use case in this guide is to introduce a stand-alone database into an existing SQL AG relationship.

There are other implications for this process. Migrating a database to a second SQL server and then making them transactionally identical has several use cases. They are:

- Seeding a Database for SQL Database Mirroring.
- Seeding a Database for SQL Log Shipping.
- Moving a large database from one instance to another for load balancing or upgrading a hardware upgrade.
- Pre-staging databases at a different regional DR site.
- Scaling out the AG relationship by adding a replica.

Know All the Details

There is a lot to know about Microsoft SQL AG technology and how it is applied in the field. Cohesity is uniquely suited to work with SQL AG.

Find out more by reading these articles:

- Cohesity with MS SQL Server AlwaysOn Availability Group Solution Guide
- Prerequisites, Restrictions, and Recommendations for Always On availability groups
- Protect SQL Server with Cohesity — Deployment Configurations Guide

- SQL Server Instance Prerequisites and Restrictions
- What is SQL Server AlwaysOn?

What is Microsoft SQL Server AlwaysOn?

The SQL Always On Availability Groups feature is a high-availability and disaster-recovery (DR) solution that provides an enterprise-level alternative to database mirroring. SQL AG maximizes the availability of a set of user databases for an enterprise. An availability group supports a failover environment for a discrete set of user databases, known as availability databases, that failover together. An availability group supports a set of read-write primary databases and one to four sets of corresponding secondary databases. For more, read Overview of Always On Availability Groups (SQL Server).

Terminology

There are several concepts and terms that are important to understand as you learn about the Microsoft SQL Server Backup features in Cohesity.

Table 1: Migration Concepts

TERM	DEFINITION
Protection Group	A Cohesity Protection Group is a backup job that runs repeatedly, based on an associated Policy, to back up data from a source and store it on the Cohesity cluster. A Protection Group can also store the data on External Targets that can be other clusters, cloud object storage, or tape.
Microsoft DB Migration (Technique)	The process of moving an MS SQL Server database from one SQL Server instance to a different SQL Server instance without data loss. This can be performed manually or in some automated form.
Cohesity DB Migration	The Cohesity implementation of the DB Migration technique. Specifically, moving an initial copy of the database and then sequentially applying incremental backups. * *You must contact Cohesity Support to have the Cohesity DB Migration feature enabled before you can use it.*
AG Replica (Replica)	The term "replica" typically refers to availability replicas. For example, "primary replica" and "secondary replica" always refer to availability replicas.[1] An *availability database* is sometimes called a *database replica* in Transact-SQL, PowerShell, and SQL Server Management Objects (SMO) names.

[1] Overview of Always On Availability Groups (SQL Server), Microsoft.

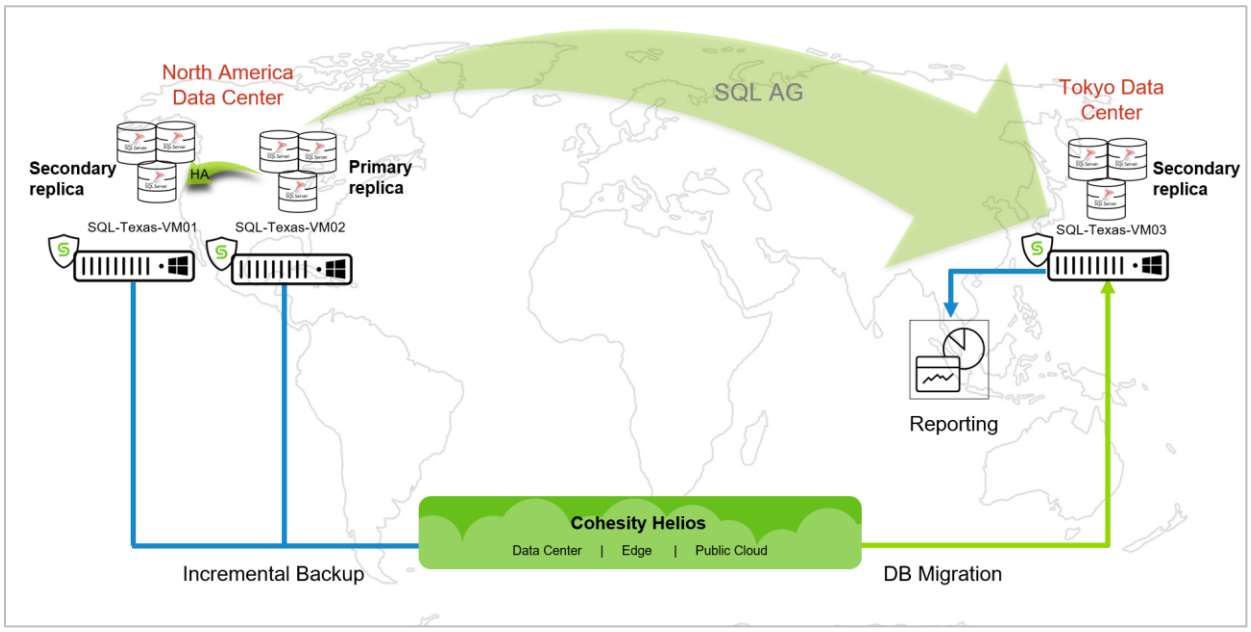
TERM	DEFINITION
Primary Replica	The primary replica is assigned the primary role and hosts read-write databases, which are known as *primary databases*.
Secondary Replica	A secondary replica hosts read-only databases, known as *secondary databases*. Every availability replica is assigned an initial role — the *primary* or *secondary* role, which is inherited by the availability databases of that replica. The role of a given replica determines whether it hosts read-write databases or read-only databases.[2]

Real-World Example

In our example, the customer is headquartered in Dallas and has a branch in Tokyo.

Both locations have a data center (DC); one in Texas (North America) and the other in Tokyo (Japan). The customer wants to set up an AG database between Dallas and Tokyo.[3] Linking the two locations is an existing SQL AG relationship.

Figure 1: Example — Setting Up a SQL AG Database Between Data Centers



In order to establish a SQL AG relationship between two SQL databases located in Dallas and Tokyo, you must have a transactionally identical copy of the database on both the primary (Dallas) and secondary (Tokyo) servers.

[2] Ibid.

[3] There are several reasons to set up an AG database between Dallas and Tokyo: 1) establish a disaster-recovery (DR) site, 2) keep business data in Tokyo up-to-date, and 3) offload business reporting to its Tokyo branch.

The database administrators (DBAs) in Dallas calculate that it will take three days (72 hours) to back up, copy, and restore the database to the Tokyo data center.

In the time that it takes the DBAs to perform this task, the primary database in Dallas continues to receive transactions. Once the copy of the database is restored to Tokyo, how do we make the secondary database in Tokyo identical to the primary database in Dallas, as required for an AG relationship?

In this case, we will establish a copy of the database in Tokyo using Cohesity's DB Migration feature. After that, establishing a SQL AG relationship is easy.

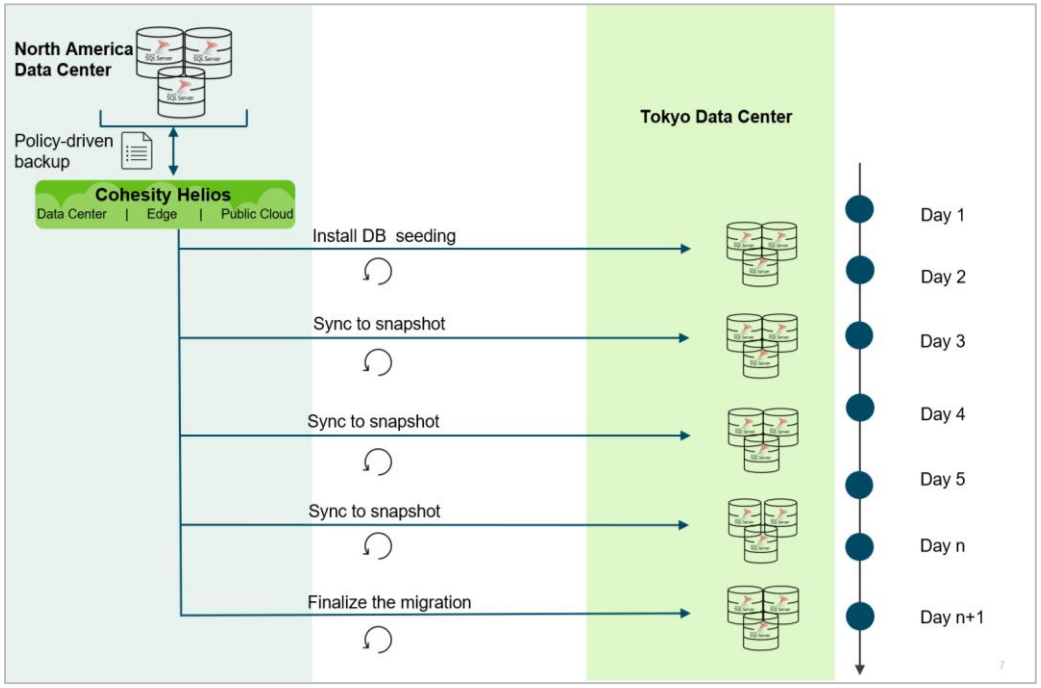
Visualize Database Migration

To continue with our example, the DB Migration feature is used to migrate a copy of the database from Dallas over to Tokyo. DB Migration accomplishes this in one workflow.

The process is:

1. Initial seeding (copy) of the database files from a snapshot on Cohesity over to the SQL instance in Tokyo.
2. A series of snapshot synchronizations keep the copy in Tokyo up-to-date with the most recent incremental snapshot.
3. Finalize the migration
    - Synchronize any outstanding changes with the target.
    - Attach the database to the SQL instance.

Figure 2: Database Migration Workflow



Once the migration is finalized, the Tokyo copy of the database is almost identical to the one in Dallas. The last step is to make the database in Tokyo identical to the database in Dallas.

Let's start with adding a database into a SQL AG relationship, in the next chapter.

Add a Database into a SQL AG Relationship

In the following sections, we will seed a database (which we'll call "NewBraunfels") over to a second SQL instance using Cohesity's DB Migration feature, and then introduce it into a SQL AG relationship.

Map the Workflow to Success

Before jumping into the process, it's important to know what all the stages will be.

To add a database into a SQL AG relationship using Cohesity's DB Migration feature:

1. Jump in with an existing SQL AG
2. Check the prerequisites
3. Migrate the database to a host replica
4. Make the databases identical
5. Tie the knot with SQL AG

> **IMPORTANT**: Cohesity *strongly* recommends you thoroughly test this process in a test environment before applying it in a production environment.

Jump In with an Existing AG Relationship

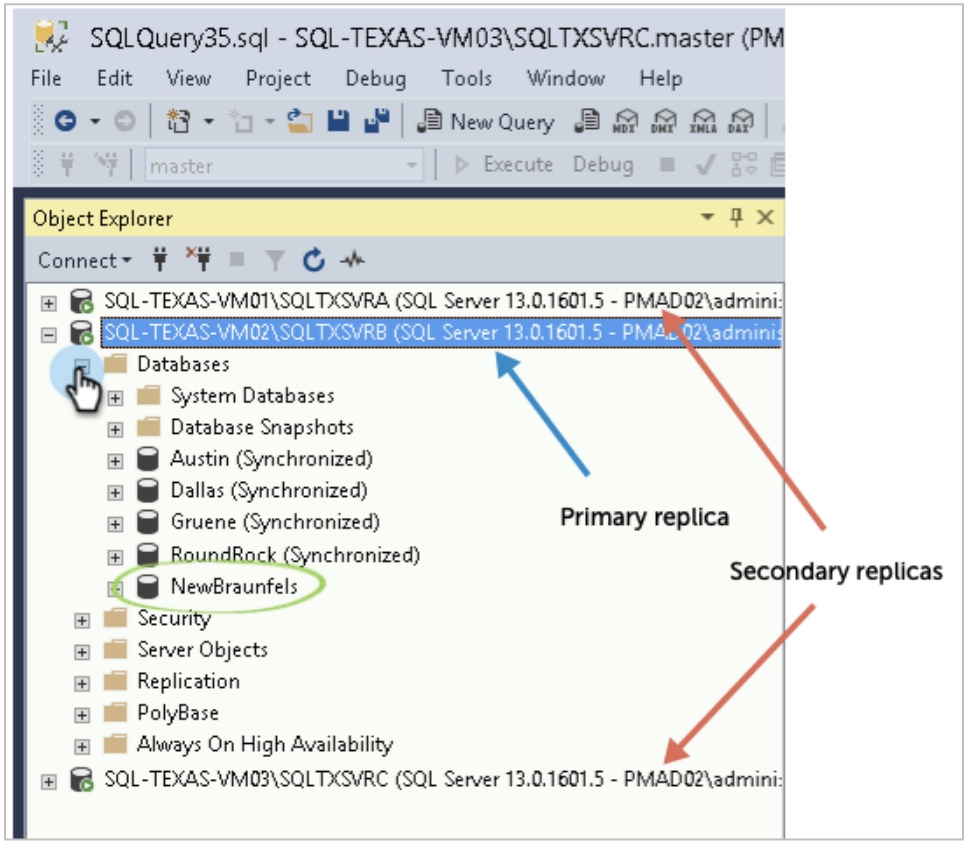
Chances are you are already working with a set of servers that has an existing AG relationship already built. Your first step is knowing the lay of the land so that you can properly place the new database and introduce it into the AG relationship.

We start with an existing SQL AG setup that has three nodes:

- SQL-TEXAS-VM01 (Called Texas 1) — currently a secondary replica.
- SQL-TEXAS-VM02 (Called Texas 2) — currently a primary replica.
- SQL-TEXAS-VM03 (Called Texas 3) — currently a secondary replica.

In Figure 3 below, we can see the existing AG nodes, including the primary replica (SQL-TEXAS-VM02) and the two secondary replicas (SQL-TEXAS-VM01 and SQL-TEXAS-VM03). In the primary replica, we find the NewBraunfels database that we wish to bring into the AG relationship.

Figure 3: AG Nodes — Primary and Secondary Replicas



Check the Prerequisites

Because we are starting with an existing SQL AG setup, most of these prerequisites are already established, but it's always good to check them anyway.

- All the SQL instances and the Cohesity cluster must be joined in an Active Directory Domain.
- All the versions of SQL instances are compatible.
- All the instances are participating in a SQL AG relationship.
- All the SQL instances participating in the SQL AG relationship are registered with your Cohesity cluster.
- The database you are working with is a qualified database.
- The database must be in a normal and open state on the Primary SQL AG instance.
- The database is not already participating in a SQL AG relationship.[4]
- The database has a recent file-based incremental snapshot under a Cohesity Protection Group.

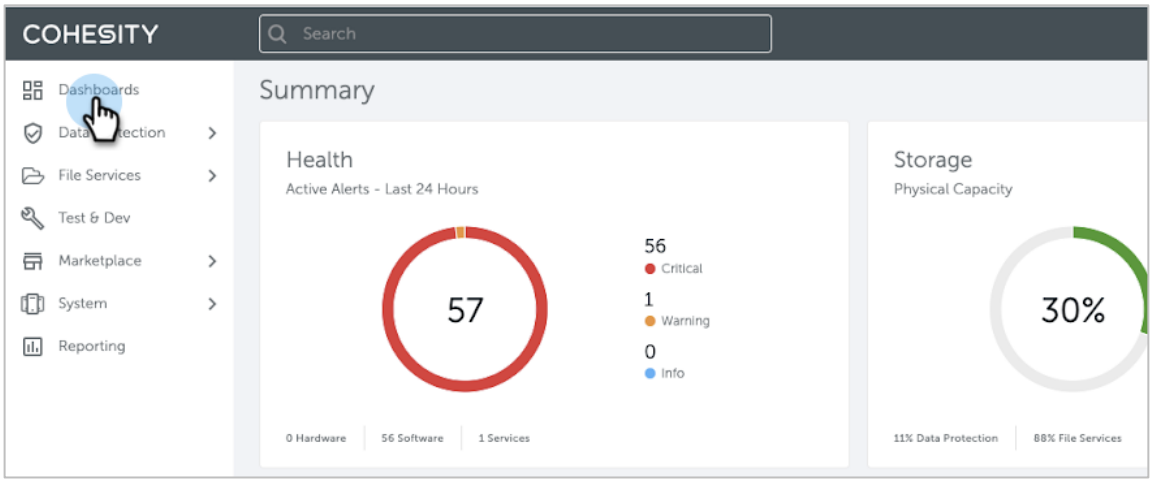
[4] This is an artificial prerequisite used to maintain simplicity in this guide. There are valid scenarios where a user might want to add another database replica to an existing AG group.

Migrate the Database to a Replica Host

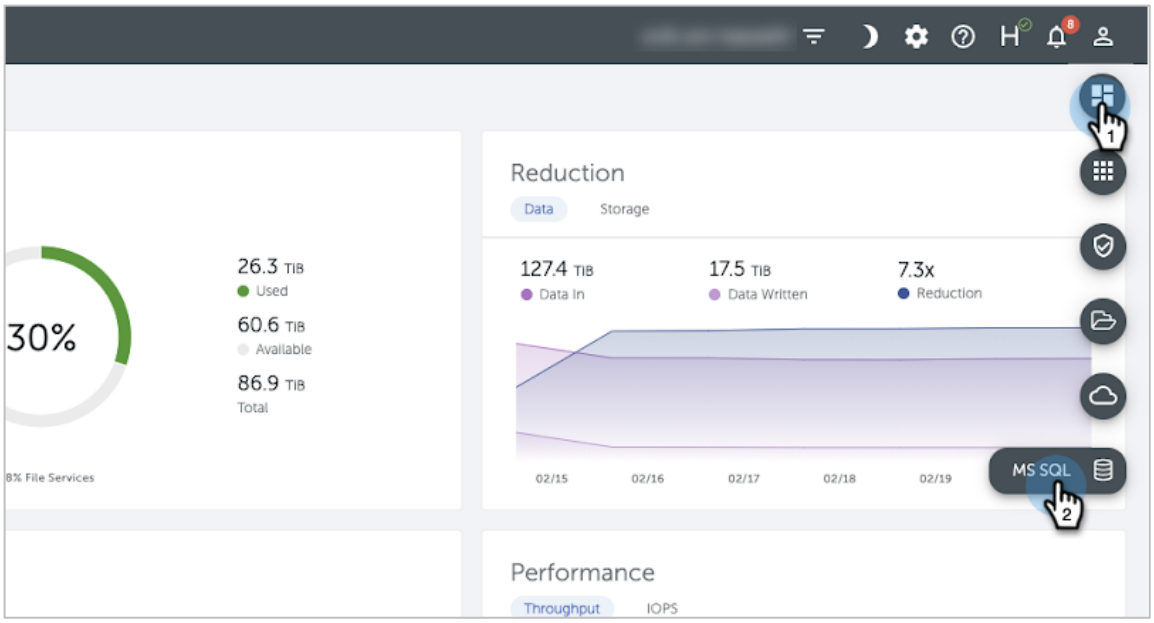
The next step is to migrate the database to get an initial copy onto the secondary host.

To migrate the database:

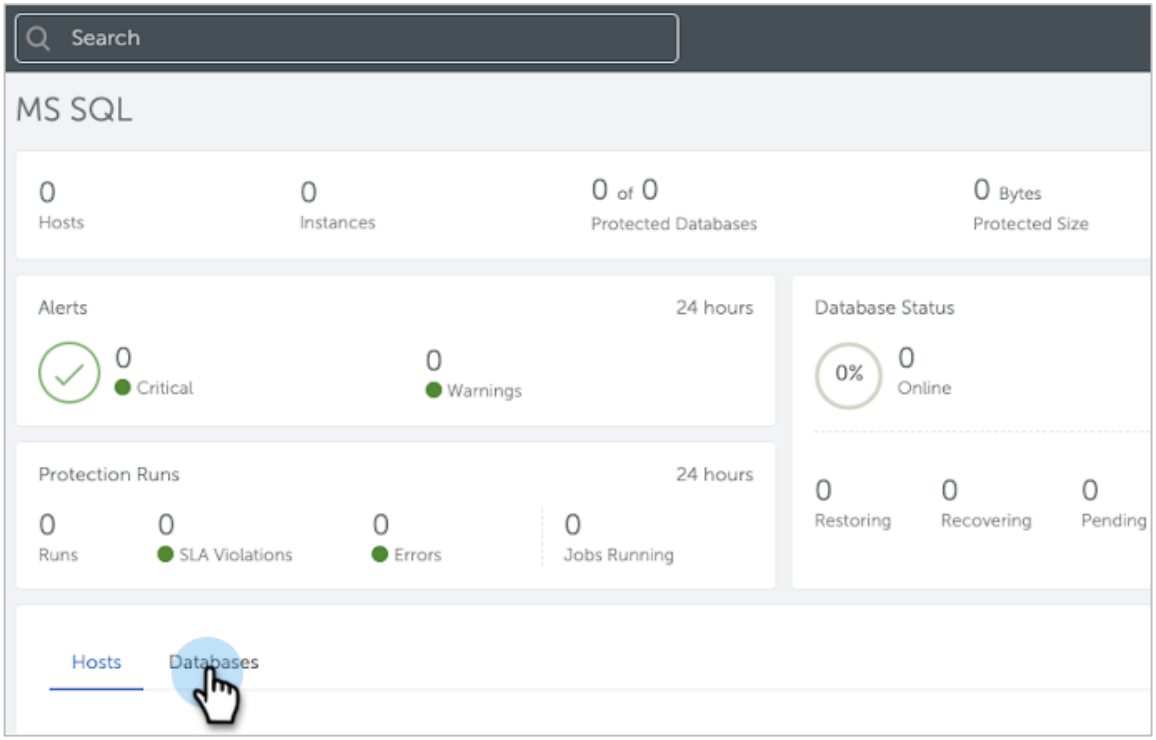
1. Log in to Cohesity.
2. Click **Dashboards**.



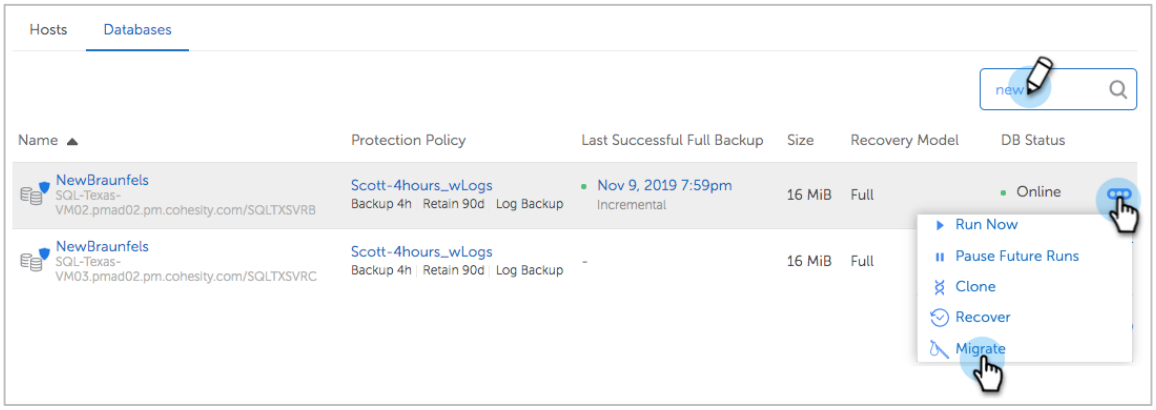
3. Click the **Dashboards** button and select **MS SQL**.



4. Click the **Databases** tab.

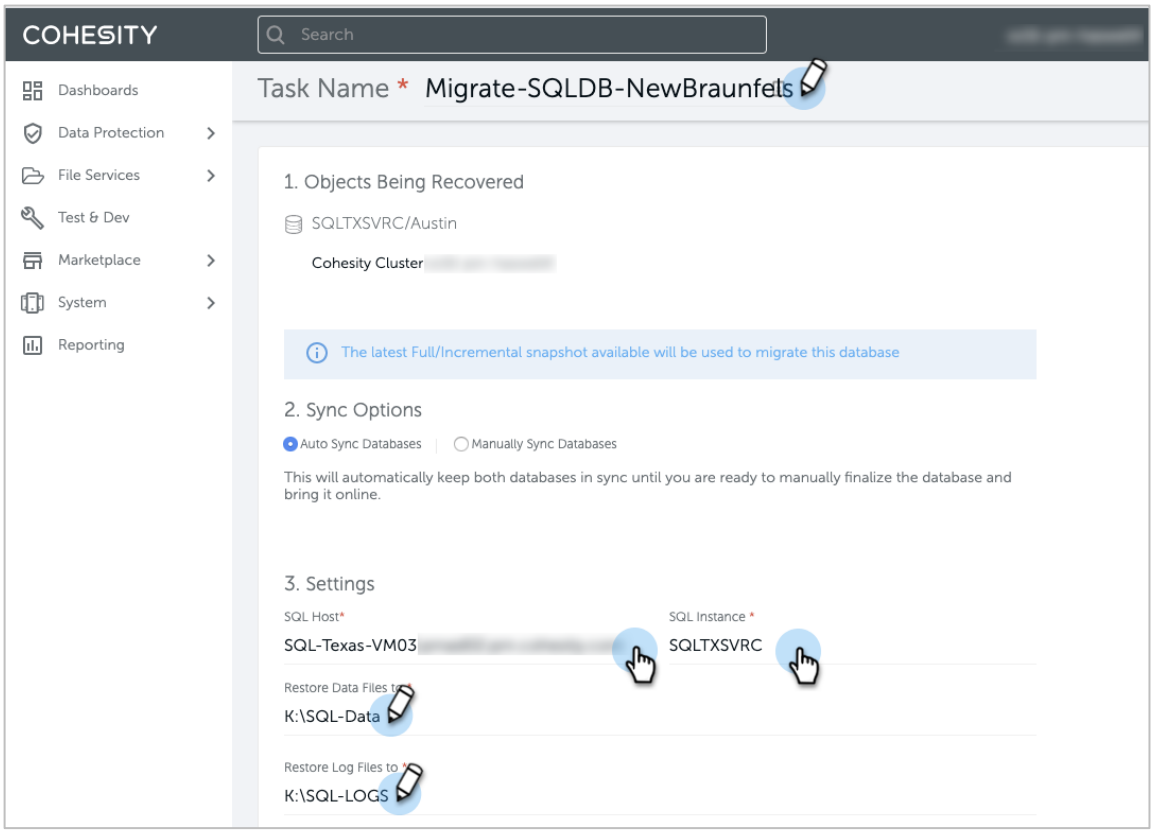


5. In the **Search** box, enter the database name, then click the action menu on the right, and select **Migrate**.

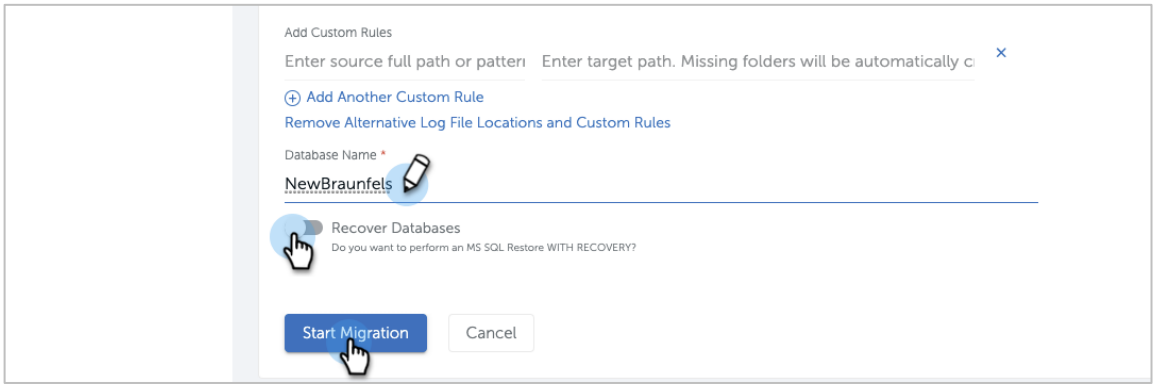


NOTE: Cohesity recommends using the most recent file-based incremental snapshot. Point-in-time (PIT) migration is part of a future enhancement for this feature. Currently, only incremental backups are used to migrate a database.

6. Enter the migration **Task Name**, select a replica node as the **SQL Host** (SQL-TEXAS-VM03 in our example), and enter the locations for **Restore Data Files to** and **Restore Log Files to**.



7. In the same form, enter a **Database Name**, disable **Recover Databases**, and click **Start Migration**.



> **CAUTION**: Be careful to *disable* **Recover Databases** in this form. This is because, for SQL AG, the database must remain in the "restoring" state.

If the rate of change is high in your database, Cohesity recommends that you synchronize the migrated database frequently.

Cohesity's DB Migration feature uses a robust communication mechanism that ensures that interruptions in connectivity during the migration process do not necessitate restarting the migration. If something should happen, the migration will pick up where it left off.

Finalize the Database Migration

The first step to making NewBraunfels and its migrated copy transactionally identical is finalizing the migration. The goal is to have a copy of NewBraunfels on the secondary AG server. The migrated copy of the database will be as close to the original as possible, although they will not yet be identical.

To finalize the database migration:

1. **Take an incremental backup**.

From the Cohesity Protection Group, take an incremental backup of the NewBraunfels database. This is to ensure that the secondary database will now be as similar as a copy to the primary database as possible.

2. **Pause the Protection Group**.

Because you do not want a Protection Group to run against the database during the time you are finalizing the migration and introducing the database into the AG relationship, you must pause the Cohesity Protection Group that protects the SQL AG servers.

IMPORTANT: During this time, nothing should change the state of the database. Therefore, no Protection Group or any other action on the database that would change the state of the database should occur.

Examples of actions that would change the state of the database are:

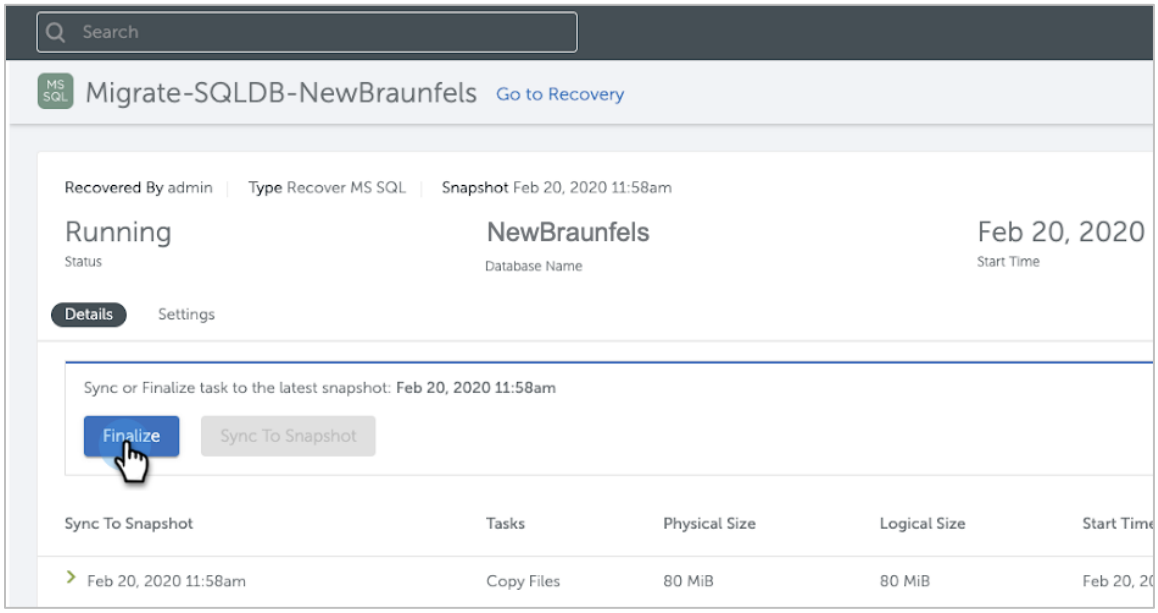
- A database log backup
- A database Full backup
- Taking the database offline
- Changing the RECOVERY Model
- Changing the database to READ ONLY

3. **Synchronize the snapshot**.

Synchronizing the snapshot will copy over any changes captured by the primary database's snapshot and apply them to the secondary database.

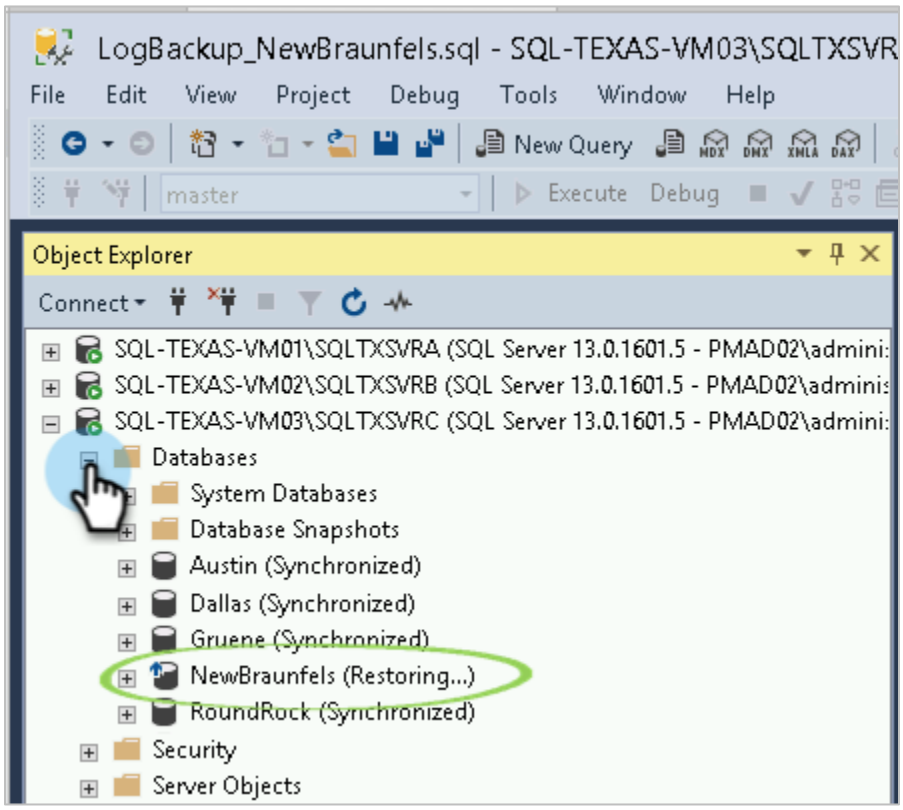
NOTE: Application transactions might still be occurring on the primary database (NewBraunfels, in our example). These will be captured and applied to the secondary copy in the next task. If the application generates a large volume of transactions during this short time, consider briefly pausing the application.

4. **Finish the migration**. Navigate to the migration task and click **Finalize**.



This copies the latest changes, applies them to the migrated database, and attaches the database to the secondary SQL AG Instance.

In your SQL Server Management Studio (SSMS), the migrated database should now appear on the TEXAS 03 secondary replica, as a database in the "**Restoring…**" state.



Make the Databases Identical

Finally, because application transactions can still be occurring on the primary database during DB Migration, we need to capture and apply these newest transactions to the secondary copy to ensure that the databases are fully identical.

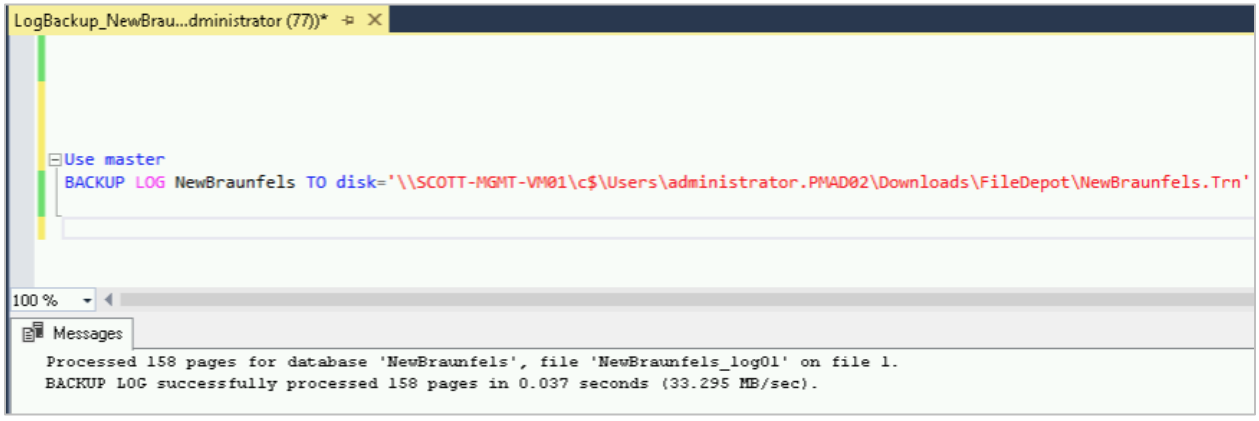
To capture the outstanding transactions:

1. Take a SQL native log backup from the primary database
2. Restore the log backup to the secondary database.

Take Log Backup from Primary Database

Go back to the primary database and use SSMS to take a log backup of that database. This captures the changes and internal differences between the primary and the migrated database.

For example:



The T-SQL command used in this example is:

```
BACKUP LOG NewBraunfels TO disk= '\\SCOTT-MGMT-
VM01\FileDepot\NewBraunfels.Trn' WITH COPY_ONLY
```

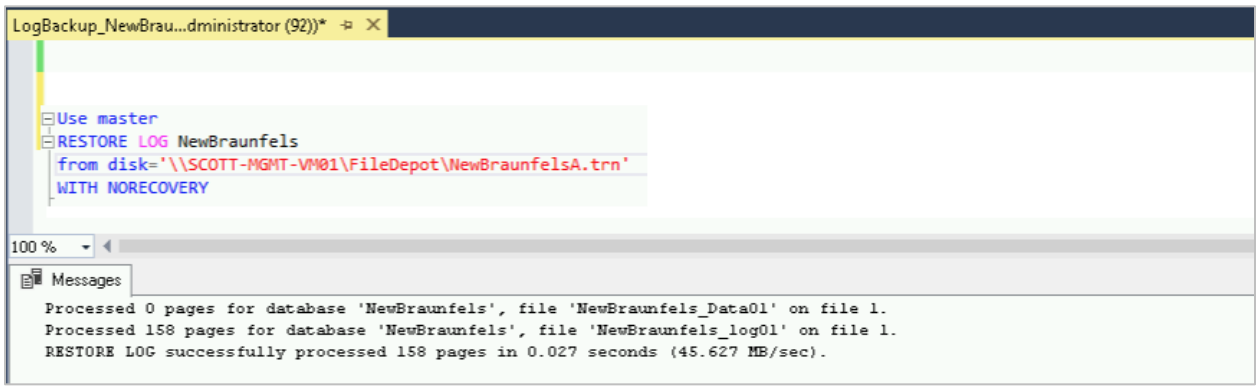
IMPORTANT: You must specify “COPY_ONLY” on the log backup. If you do not, you will break the log chain.[5]

[5] For a deeper discussion on the COPY_ONLY option, see Microsoft’s Copy-Only Backups, Microsoft.

Restore Log Backup to Secondary Database

Now use SSMS to restore the SQL native log to the secondary database. Because the database is in the "Restoring" state, it will accept the changes captured in the transaction log backup. Apply the log file and leave the database in a restoring state.

For example:



The T-SQL command used in this example is:

```
RESTORE LOG NewBraunfels from disk=
'\\SCOTT-MGMT-VM01\FileDepot\NewBraunfels.trn'
WITH NORECOVERY
```

Both the primary database and the migrated database are now transactionally identical, and the database is now ready to be introduced into the SQL AG relationship.

Tie the Knot with SQL AG

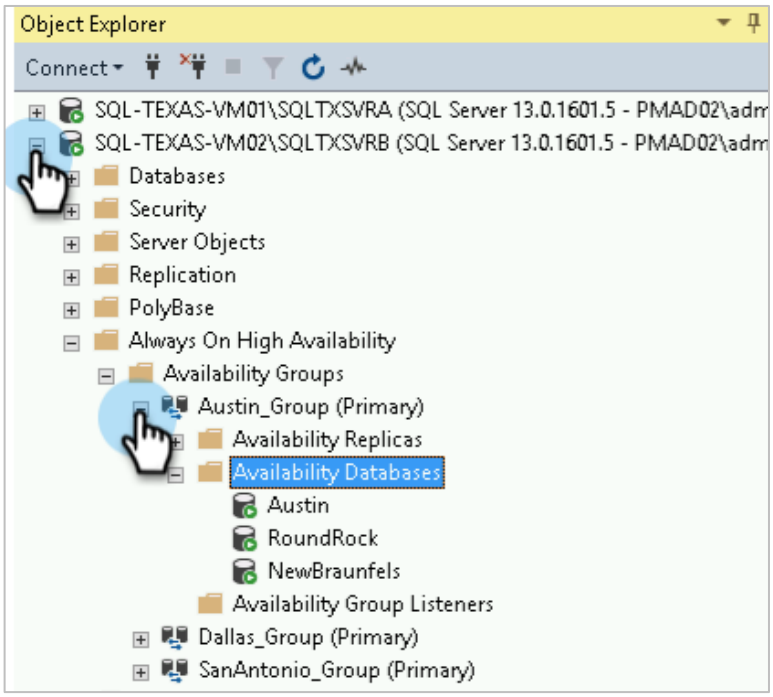
Now that our two databases are identical, we can introduce them into the SQL AG group.

Add Database to Existing AG Group on Primary AG Host

Before we can join the database to AG on the secondary host, we must add it to an AG group on the primary host.

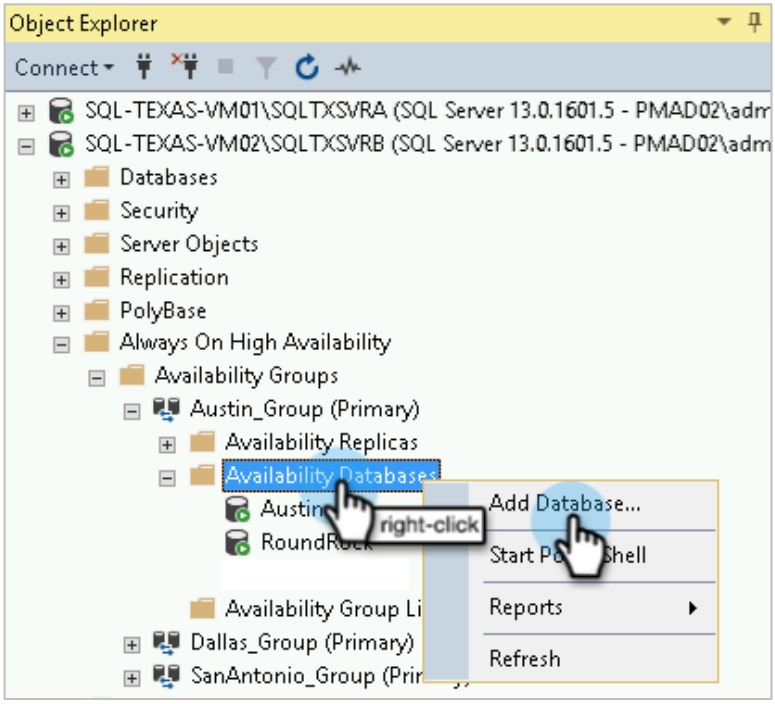
To add a database to an existing AG group on the primary AG host:

1. Open SSMS and, under the primary AG host (TEXAS 02 in our example), navigate to **Always On High Availability > Availability Groups > <*AG_Name*> (Primary) > Availability Databases**. In our example, we'll use the "Austin_Group" AG.

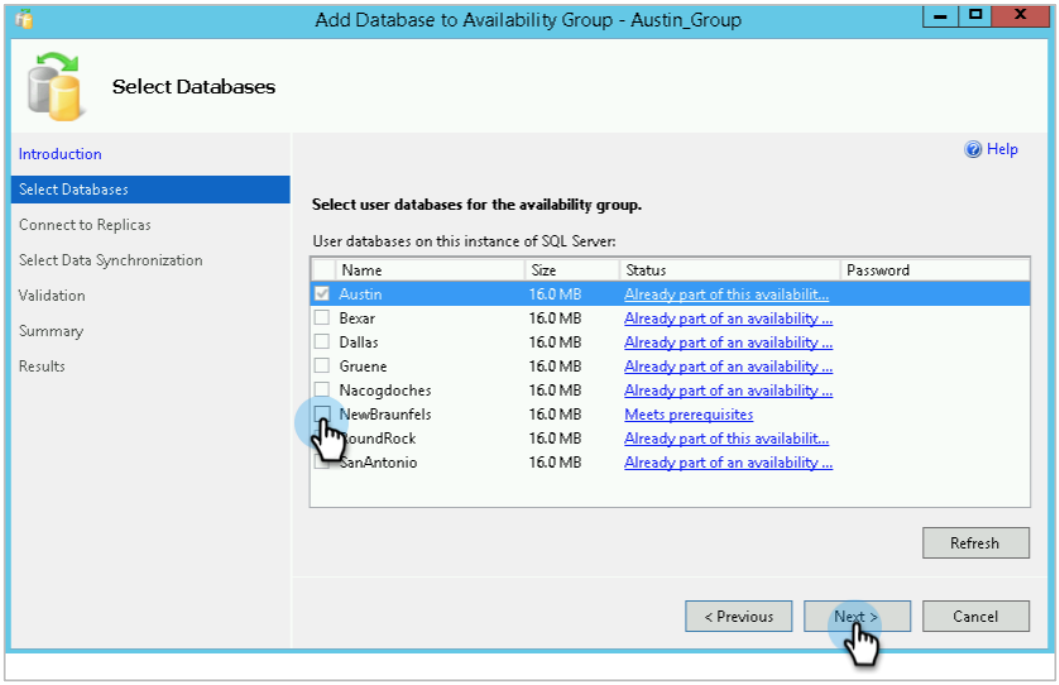


IMPORTANT: You must start the introduction with the AG group labeled **Primary**. In the example above, the primary group resides on SQL instance TEXAS 02. Later, we will introduce the migrated copy into the AG group on the other end, where it sits on TEXAS 03.

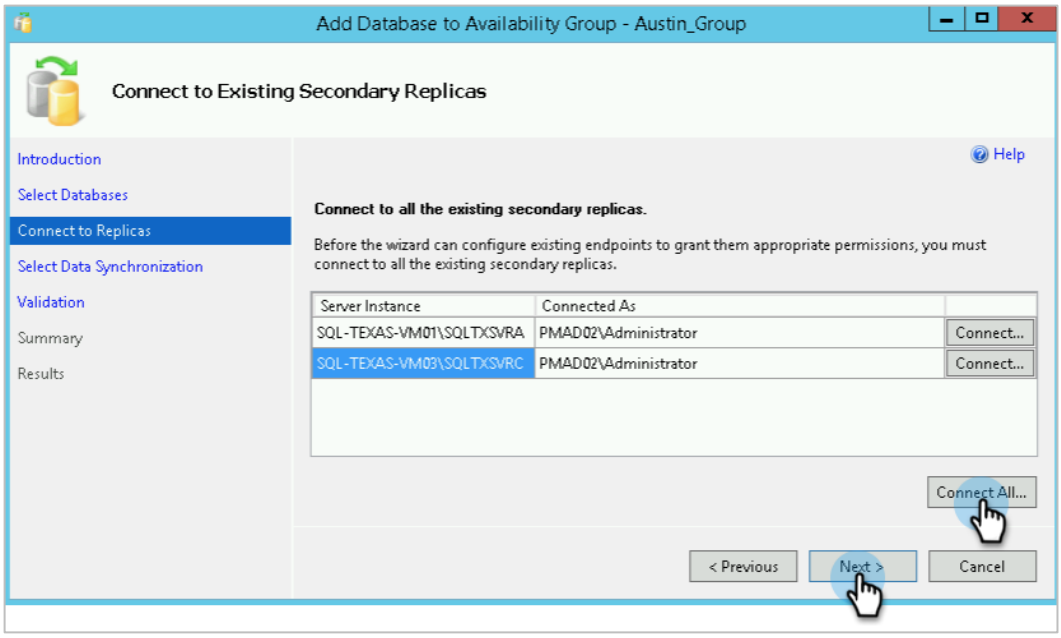
2. Right-click **Availability Databases** and select **Add Database…**.



3. Select the database and click **Next**. (Note that in our example, the NewBraunfels database is listed as **Meets prerequisites**.)



4. Connect to the replica hosts. In this existing SQL AG relationship, we have two replica hosts. Click **Connect All** to connect to all secondary SQL AG instances, then click **Next**.



5. See Table 2 below to choose the right data synchronization option for you. Make your selection and click **Next**.

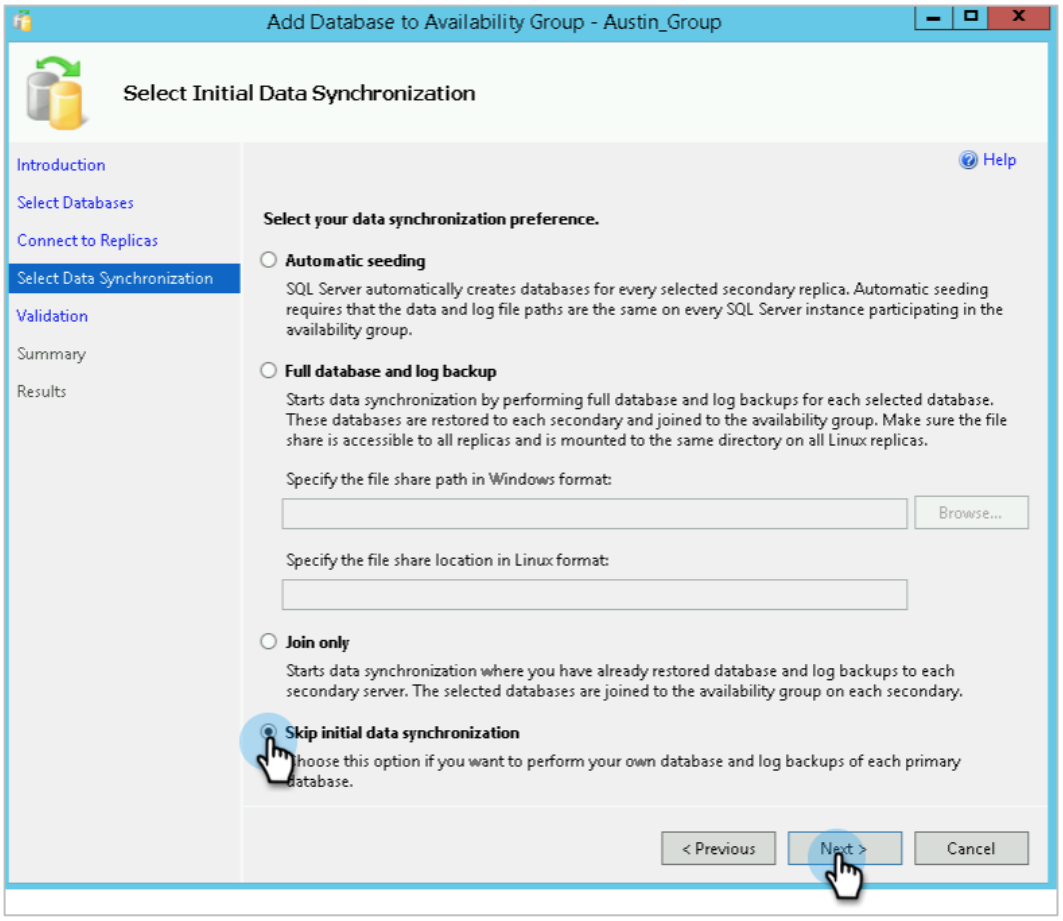


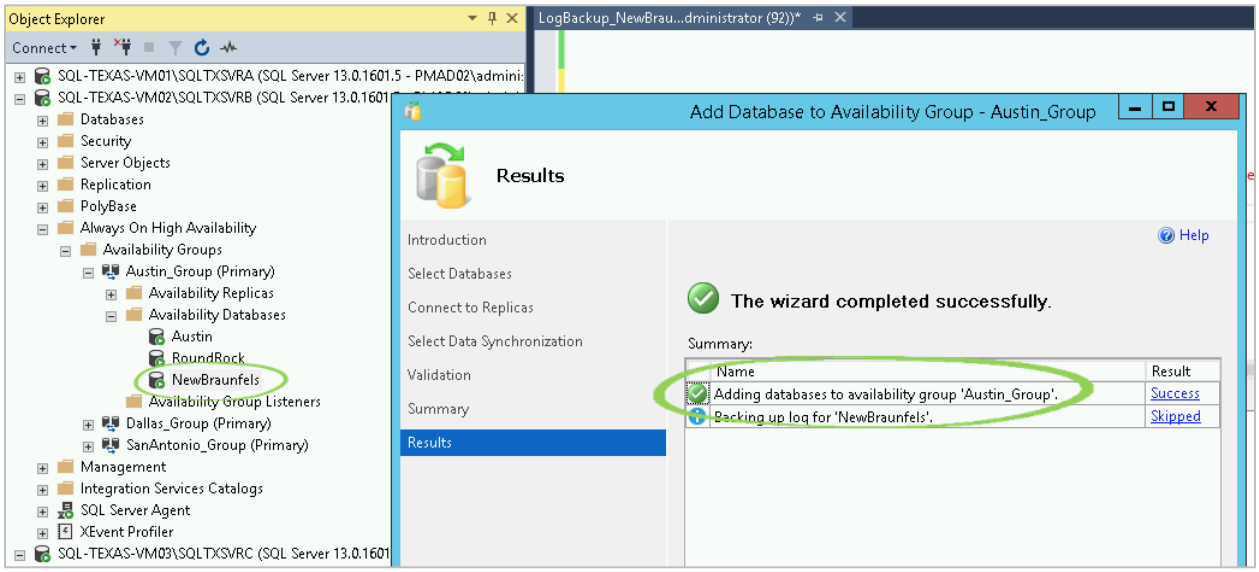
Table 2: Data Synchronization Options

OPTION	DEFINITION
Automatic Seeding	SQL Server automatically creates the secondary replicas for every database in the group.
Full database and log backup	For each primary database, this option performs several operations in one workflow: create a full and log backup of the primary database, create the corresponding secondary databases by restoring these backups on every server instance that is hosting a secondary replica, and join each secondary database to availability group.
Join Only	Select this option only if the new secondary databases already exist on each server instance that hosts a secondary replica for the availability group. The wizard will attempt to join each existing secondary database to the availability group.

OPTION	DEFINITION
Skip initial data synchronization	Select this option if you want to perform your own database and log backups of every primary database, restore them to every server instance that hosts a secondary replica. After you exit the wizard, you will then need to join every secondary database on every secondary replica.

For more, see Microsoft's Select Initial Data Synchronization Page (Always On Availability Group Wizards).

In our example, when the wizard completes successfully, it will show that the NewBraunfels database has been added to the primary side of the the "Austin_Group" AG.

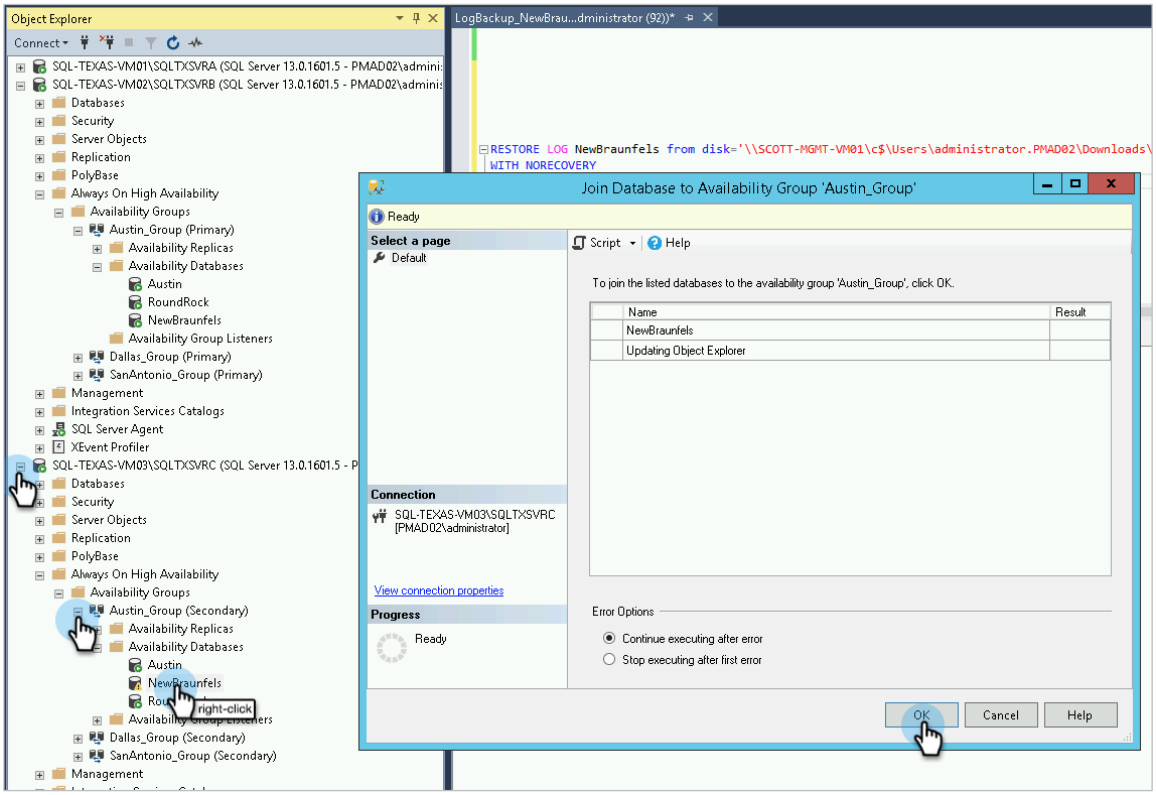


Join Database to Existing AG Group on Secondary AG Host

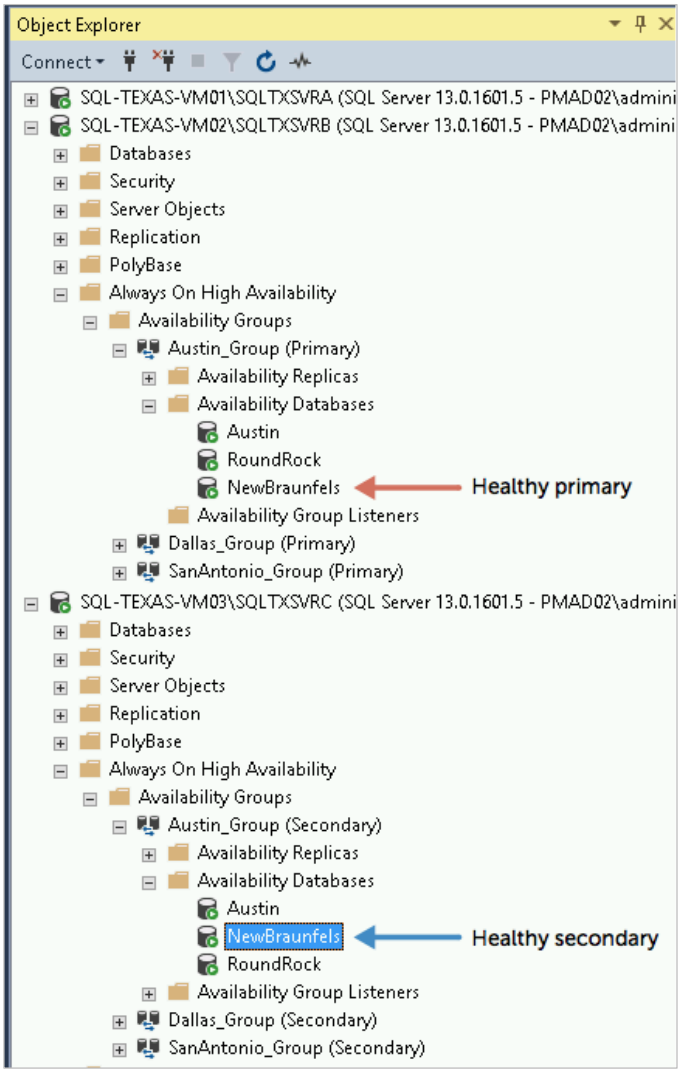
After adding the database to an AG group on the primary host, we are now ready to join it to an AG group on the secondary host.

To join a database to an existing AG group on the secondary AG host:

1. Open SSMS and, under the secondary AG host (TEXAS 03 in our example), navigate to **Always On High Availability > Availability Groups > <*AG_Name*> (Primary) > Availability Databases**. In our example, we're using the "Austin_Group" AG. Right-click the database name (**NewBraunfels**) and when prompted, click **OK** to join the database.



2. Confirm that you have successfully created the AG database in SSMS.



IMPORTANT: You should go back and enable the Protection Group that protects these servers. Check the Protection Group and confirm that the NewBraunfels database is protected.

Your Feedback

Was this document helpful? Send us your feedback!

About the Author

Scott Lorenz is a SQL Solutions Engineer at Cohesity. In his role, Scott focuses on business-critical applications, MS SQL Server databases, and data protection with Enterprise and Cloud Storage.

Other essential contributors included:

- Bart Abicht, Senior Technology Writer and Editor at Cohesity
- Freddy Grahn, Senior Technical Field Enablement Engineer

Document Version History

VERSION	DATE	DOCUMENT HISTORY
1.3	July 2024	Republishing
1.2	Sep 2021	Rebranding updates
1.1	Feb 2020	First release
1.0	Nov 2019	Original document (Internal)

ABOUT COHESITY

Cohesity is a leader in AI-powered data security and management. Aided by an extensive ecosystem of partners, Cohesity makes it easier to protect, manage, and get value from data – across the data center, edge, and cloud. Cohesity helps organizations defend against cybersecurity threats with comprehensive data security and management capabilities, including immutable backup snapshots, AI-based threat detection, monitoring for malicious behavior, and rapid recovery at scale. Cohesity solutions are delivered as a service, self-managed, or provided by a Cohesity-powered partner. Cohesity is headquartered in San Jose, CA, and is trusted by the world's largest enterprises, including six of the Fortune 10 and 44 of the Fortune 100.

Visit our website and blog, follow us on Twitter and LinkedIn and like us on Facebook.