Initialize Subscription from a Backup (avoid using Snapshot)

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:31 AM PST

Contents

Issue

A subscription to a transactional publication is by default initialized with a snapshot. The snapshot is generated by the Snapshot Agent and applied by the Distribution Agent, which is reasonably fast in most on-premise environments.

But snapshots can run into significant delays if the published dataset is large or if the Distributor and Subscriber are connected over slow networks.

Investigation / Analysis

Slow Subscriber connectivity can occur if the on-premise SQL Server needs to publish its data to a Managed Instance Subscriber database, where the internet upload link only has limited throughput. In the worst case, it might take hours to create the large snapshot, and then several days for applying the snapshot over the internet into the cloud database; it might even exceed the default retention period of 72 hours = 3 days.

This issue can be avoided by initializing the Subscriber database with a backup of the Publisher database. Initializing with a backup is the fastest way to deliver data to the Subscriber and is convenient, because any recent backup can be used if it was taken after the publication was enabled for initialization with a backup.

Content of this article:

- This article describes the steps for initializing a Managed Instance Subscriber from a backup.
- It only applies to replicating from on-premise to Managed Instance.
- It does NOT apply to the MI-to-on-premise.
- It does NOT apply to the MI-to-MI scenario.

The article is following the steps from the public article: Initialize a Transactional Subscription from a Backup.
For a detailed sample script with comments, see Initialize Subscription from a backup (on-premise to MI)

Mitigation

These are the high-level steps for initializing a Subscriber database with a backup from the Publisher:

Step 1a - Existing Publication

For an existing publication, ensure that the publication has enabled the `allow initialize from backup` flag by executing sp_helppublication (Transact-SQL) at the Publisher on the publication database. Note the value of `allow_initialize_from_backup` and `immediate_sync` in the result set:

- If `allow_initialize_from_backup` = 1 then the publication already supports this functionality
- If `allow_initialize_from_backup` = 0 then you need to enable it. Execute sp_changepublication (Transact-SQL) at the Publisher on the publication database. Specify a value of "allow_initialize_from_backup" for `@property` and a value of "true" for `@value`.
- If `immediate_sync` = 0 then it is recommended to also change it to "true". This option will guarantee that transactions are cached within the retention period, therefore not being cleaned up while the backup is copied and restored to the Subscriber.

Step 1b - New Publication

For a new publication, execute sp_addpublication with parameters `@allow_initialize_from_backup = N'true'` and `@immediate_sync = N'true'`.

Step 2 - Backup Published database on the on-premise SQL Server

OPTION 1 - Works with all Publisher SQL Server versions
Backup the database to a local storage path, for example "M:\Backups". Note the `COMPRESSION` flag to minimize the file size for quicker upload to Azure, and the `COPY_ONLY` flag to avoid breaking the backup chain.
Older SQL versions must use a local storage path for writing the backup file, and you then have to upload the file to a Blob storage container yourself to make it available to the MI restore command. This option is also still valid on newer SQL versions, e.g. if the SQL Server machine doesn't have direct access to Azure storage.

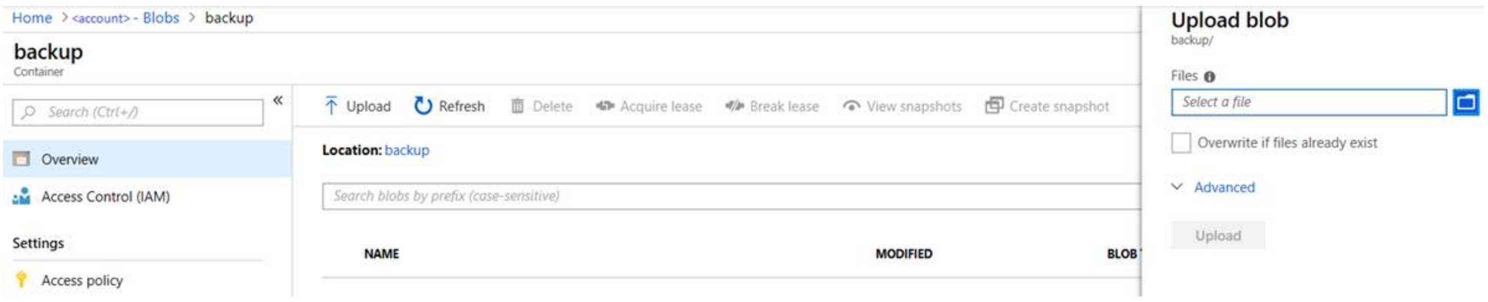
OPTION 2 - Applies only to Publisher SQL Server versions 2017 CU21 / 2019 CU7 / 2022 and newer
Backup the database directly to an Azure Blob storage container. Skip the following step 3 and continue with step 4.
The Cumulative Update described in KB4569425 - FIX: Transactional replication publications can support URL type device allows using `backupdevicetype = 'URL'` for the sp_addsubscription (Transact-SQL) command in Step 5 below. This avoids the manual upload of the backup file in Step 3.

```
/*** OPTION 1 ***/
BACKUP DATABASE [publisherdatabase]
TO DISK = 'M:\Backups\publisherdatabase.bak' WITH COPY_ONLY, INIT, COMPRESSION


/*** OPTION 2 ***/
-- Create a credential for the Blob storage container if you haven't already
-- use SAS secret with first "?" character removed
USE master
CREATE CREDENTIAL [https://youraccount.blob.core.windows.net/backup]
WITH
    IDENTITY='SHARED ACCESS SIGNATURE',
    SECRET = 'sv=2021-06-08&ss=bfqt&srt=sco&sp=rwdlacupiytfx&se=2024-11-30T...xxxx...Fw0%3D'
GO

BACKUP DATABASE [publisherdatabase]
TO URL = 'https://youraccount.blob.core.windows.net/backup/publisherdatabase.bak' WITH COPY_ONLY, INIT, COMPRE
```

Step 3 - Upload backup file to an Azure Blob Storage container

Skip this step if you have used Step 2 Option 2

The Managed Instance can only access Azure Blob storage containers but not file shares or other external resources.



Step 4 - Restore the backup to Managed Instance

Restore the backup from the Azure storage URL to the Subscriber server. Note that you must have a credential based on an SAS token in place to get the permissions to the Blob container. See [Tutorial: Use Azure Blob Storage with SQL Server 2016]() for more information.

```
-- Create a credential for the Blob storage container if you haven't already
-- use SAS secret with first "?" character removed
USE master
CREATE CREDENTIAL [https://youraccount.blob.core.windows.net/backup]
WITH
    IDENTITY='SHARED ACCESS SIGNATURE',
    SECRET = 'sv=2021-06-08&ss=bfqt&srt=sco&sp=rwdlacupiytfx&se=2024-11-30T...xxxx...Fw0%3D'
GO

-- Restore the backup on the MI Subscriber
-- run this from a SSMS connection to the target MI
-- credential to the URL needs to be in place before running this
RESTORE DATABASE [subscriberdatabase] FROM URL = 'https://youraccount.blob.core.windows.net/backup/publisherda
GO

/* Variation if you want to use the same backup for an on-premise Subscriber:
-- same approach for the on-premise restore - requires to include the MOVE option:
RESTORE DATABASE [subscriberdatabase] FROM URL = 'https://youraccount.blob.core.windows.net/backup/publisherda
    WITH MOVE 'publisherdatabase' to 'M:\data\subscriberdatabase.mdf',
    MOVE 'publisherdatabase_Log' to 'M:\data\subscriberdatabase.ldf', REPLACE
GO
*/
```

Step 5 - Create the Push subscription at the Distributor

After restoring the backup from the Azure storage URL, you can now create the Push subscription at the on-premise Distributor.

If you have used Step 2 Option 1 earlier: Note that the `backupdevicename` uses the same local storage path to which the backup had been written initially - you cannot specify the URL on Azure storage or any other, random path.

If you have used Step 2 Option 2 earlier: You can use the same Azure storage URL that you have used for the backup and restore commands.

```
/*** OPTION 1 ***/
-- Adding the Push subscription from on-premise to MI
exec [Repl_PUB]..sp_addsubscription @publication = N'publicationname',
     @subscriber = 'yourmi.8f08e6d34d3b.database.windows.net', -- FQDN
     @destination_db = N'subscriberdatabase',
     @subscription_type = N'Push',
     @sync_type = N'initialize with backup',
     @backupdevicetype = N'disk',
     @backupdevicename = N'M:\Backups\publisherdatabase.bak',
     @article = N'all',
     @update_mode = N'read only',
     @subscriber_type = 0
GO

/*** OPTION 2 ***/
-- Adding the Push subscription from on-premise to MI
exec [Repl_PUB]..sp_addsubscription @publication = N'publicationname',
     @subscriber = 'yourmi.8f08e6d34d3b.database.windows.net', -- FQDN
     @destination_db = N'subscriberdatabase',
     @subscription_type = N'Push',
     @sync_type = N'initialize with backup',
     @backupdevicetype = N'URL',
     @backupdevicename = N'https://youraccount.blob.core.windows.net/backup/publisherdatabase.bak',
     @article = N'all',
     @update_mode = N'read only',
     @subscriber_type = 0
GO
```

How good have you found this content?