SQL Server Replication as a tool to migrate very large databases

Prepared by

Data SQL Ninja Engineering Team (datasqlninja@microsoft.com)

release 1.1

Note: The detail provided in this document has been harvested as part of a customer engagement sponsored through the Data SQL Ninja Engineering.

Table of Contents

1 Introduction

Replication is a set of technologies for copying and distributing data and database objects from one database to another and then synchronizing between databases to maintain consistency.

Transactional replication is typically used in server-to-server scenarios that require high throughput, including improving scalability and availability, data warehousing and reporting, integrating data from multiple sites, integrating heterogeneous data, and offloading batch processing.

SQL Replication can be used as a viable tool for some migration scenarios, especially when there is a need to move transactions from the source database to the target continuously. There are heterogeneous sources involved, and the target database must accept updates while data is still migrating.

This paper will discuss how to utilize SQL Server Transactional replication to migrate an extensive (5-10 TB) database to the cloud over a standard internet link from an On-Premises source location to the Azure cloud.

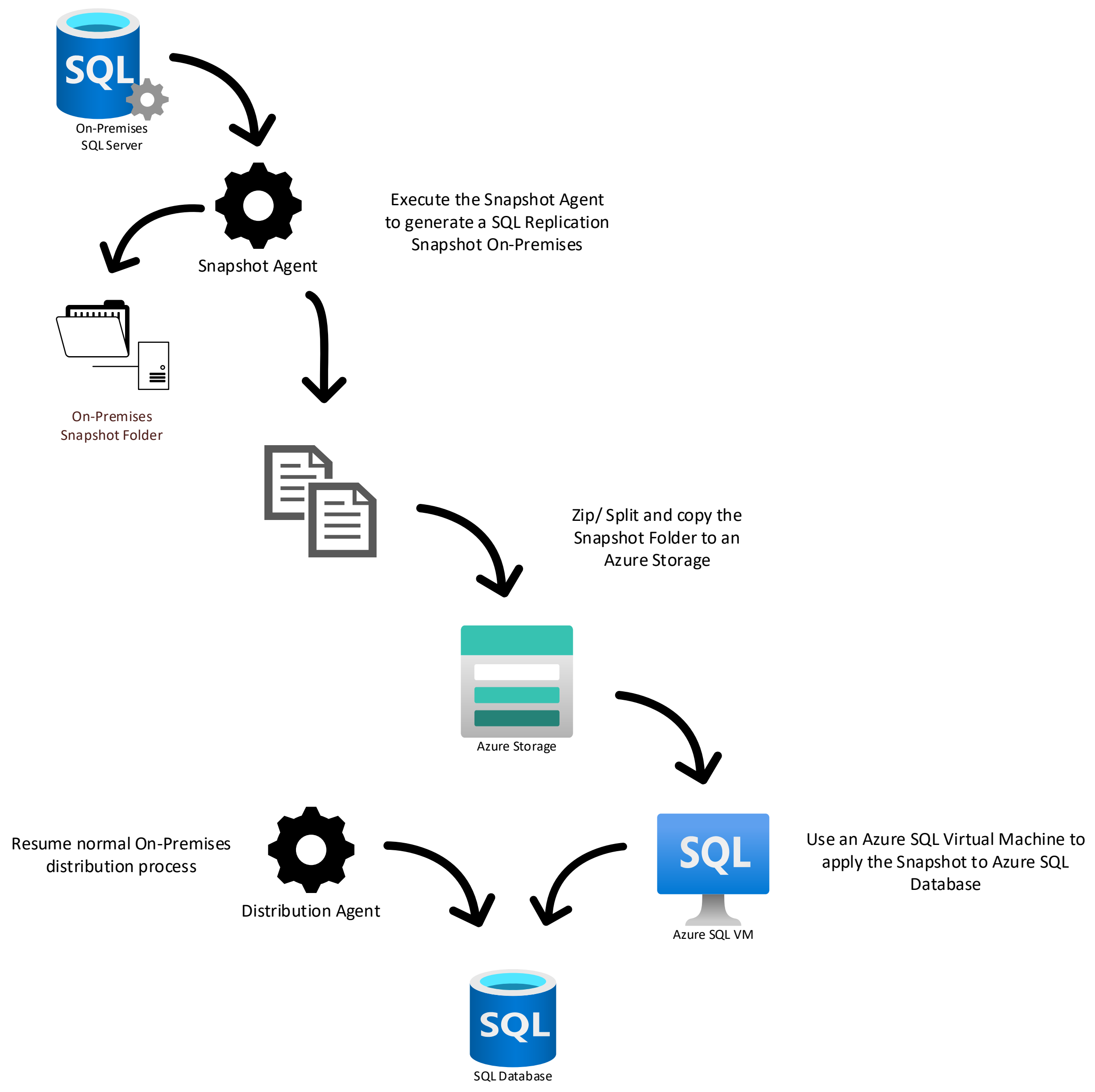
SQL Replication is a perfect use case for specific migration scenarios involving Azure SQL Database and Azure SQL Database Hyperscale as targets when the source database has the correct compatibility requirements for Transactional Replication.

The technique presented here optimizes Snapshot delivery, helps minimize cutoff time, increases decoupling of replication processes, and leverages the ability to replicate stored procedure execution during the migration process to reduce data transfer requirements.

2 Empirical Overview

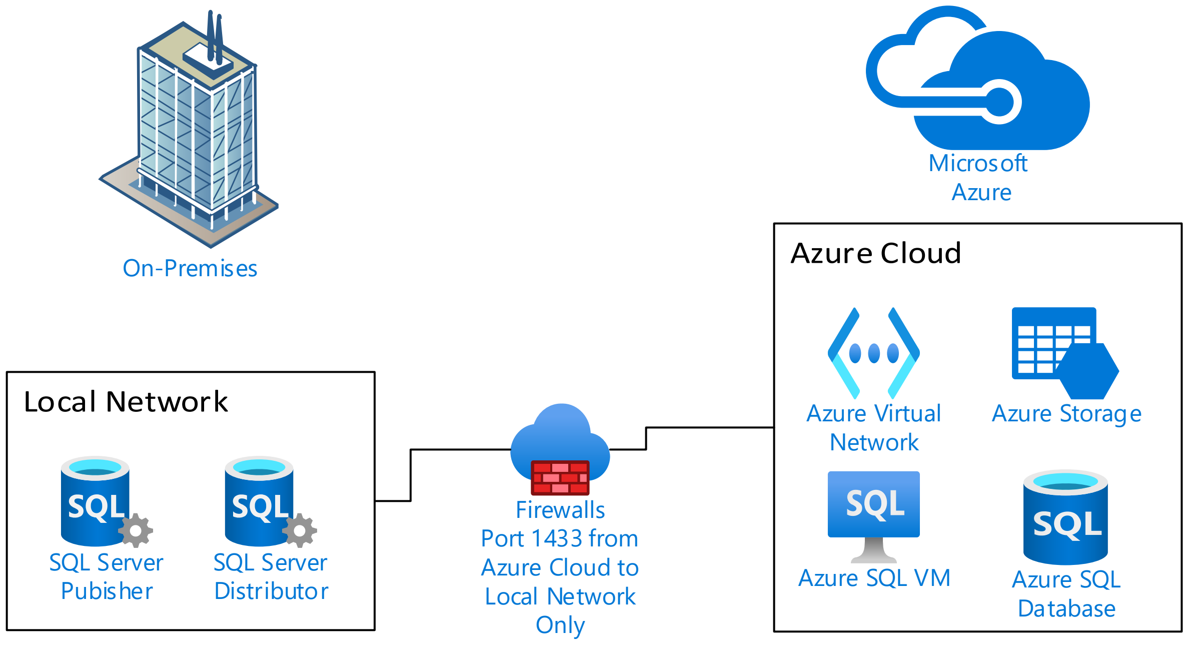
This paper's proposed solution decouples the replication agents' execution, allowing us to generate, compact the snapshot folder, manually upload the snapshot files to the cloud, remotely applying the database snapshot, and then resume the normal replication process in a controlled fashion. This way, we can apply the snapshot from an optimal network location, allowing for the best throughput possible to be achieved, reducing the required time for snapshot delivery. Transactional replication will then maintain the target database in close sync with the source database, allowing reduced cutoff requirements.

Visual description of the overall process:


On-Premises SQL Server
Snapshot Agent
Execute the Snapshot Agent to generate a SQL Replication Snapshot On-Premises
On-Premises Snapshot Folder
Zip/ Split and copy the Snapshot Folder to an Azure Storage
Azure Storage
Resume normal On-Premises distribution process
Distribution Agent
Azure SQL VM
Use an Azure SQL Virtual Machine to apply the Snapshot to Azure SQL Database
SQL Database

Required network connectivity

1. The Azure Virtual machine used remotely to apply the snapshot needs access to the On-Premises SQL Server where the distribution database resides over port 1433 only.
2. The Azure Virtual machine needs access to the Storage folder hosting the snapshot files.



3 Limitations

Tables published for transactional replication must have a primary key. If a table is in a transactional replication publication, you cannot disable any indexes associated with primary key columns. This document will describe a workaround for tables without primary keys.

The source database is accessible to users during the creation of the publication and the initial snapshot. However, it is advisable to create publications during periods of lower activity on the Publisher.

The snapshot files require significant storage space that can, in some instances, be larger than the original database size.

Especially for large data transfers over the internet, it is recommended to have stable and faster internet connectivity. Choose an Azure solution for data transfer | Microsoft Docs

Caution about the source database's transaction log size as the log file grows until we start the Distribution Agent and complete applying the snapshot on the target. Logfile growth depends on factors such as how busy and how extensive the database is.

4 Configurations

We will utilize a customized Transaction Replication strategy that uses a remote snapshot folder and manually controls snapshot creation and delivery.

After we manually initialize the subscription, we will resume the continuous replication process, including replicating the execution of stored procedures.

The process includes generating the Transactional Replication Snapshot during off-peak hours to a local snapshot folder on-premises, then compacting and uploading the snapshot to the cloud.

We will then execute the distribution agent to deliver the snapshot from the virtual machine on the Azure Cloud.

Moving the snapshot data manually to the cloud and executing it from a virtual machine in the cloud allows us to optimize the upload process by compacting and splitting the snapshot folder contents as needed. Besides, applying the snapshot from an optimal network stance will improve throughput and reduce initialization duration.

After the initialization concludes via this customized snapshot delivery process, we resume normal replication from the on-premises Publisher/Distributor. We are free to delete the Azure Virtual Machine and Storage Account.

If you migrate from a source already in the Azure Cloud to a target in the same Azure region, you do not need to move the snapshot manually. Just skip the steps to move the snapshot and run the distributor agent on a virtual machine.

Additional optimization agent parameter settings should optimize subscriber initialization and continuous synchronization performance. Plus, leveraging the replication of stored procedures should minimize network bandwidth usage.

Local on-premises Resources:

Publisher = On-premises SQL Server where source database resides.

Distributor = On-premises remote SQL Server distributor to avoid increasing load on the Publisher.

Azure Resources:

Subscriber = This is target Azure SQL Database.

Azure SQL VM = Azure Virtual Machine with SQL Server that is the same version as the SQL Server **Distributor**. We will only use the File System and execute the SQL Server Distrib.exe replication agent program from this virtual machine. This virtual machine should exist in the same datacenter and as close to the target database as possible.

Azure Storage = Azure storage in the same region as our Azure Virtual Machine.

Steps:

1. Create a Windows virtual machine in Azure and install the same SQL Server version as your local on-premises Distributor server. You can provision from the Azure Gallery if you want. We will only use the Distribution Agent executable program from this Virtual Machine in a command-line prompt. We will **not** configure this Azure Virtual Machine as the Distribution Server for your replication setup.

   Choose a Windows version that supports Azure File Shares and SMB 3.0 and is compatible with your SQL Server Version.
   - Create SQL Server on a Windows virtual machine in the Azure portal - SQL Server on Azure VM | Microsoft Docs
   - Use an Azure file share with Windows | Microsoft Docs

2. Create an Azure Storage account and a File Share.
   - Create a premium Azure file share | Microsoft Docs

3. Attach the Azure Storage to the virtual machine we created on step 1.
   - Create and use an Azure Files share on Windows VMs | Microsoft Docs

4. On your on-premises environment, configure a remote Distributor Server on a separate SQL Server to avoid load on the Source Server.

   Check supported versions for your Distributor.
   - Azure SQL Server replication to Azure SQL Database - Azure SQL Database | Microsoft Docs

   Select a local folder on the Distributor Server to serve as the Snapshot folder destination.
   - Configure Distribution - SQL Server | Microsoft Docs

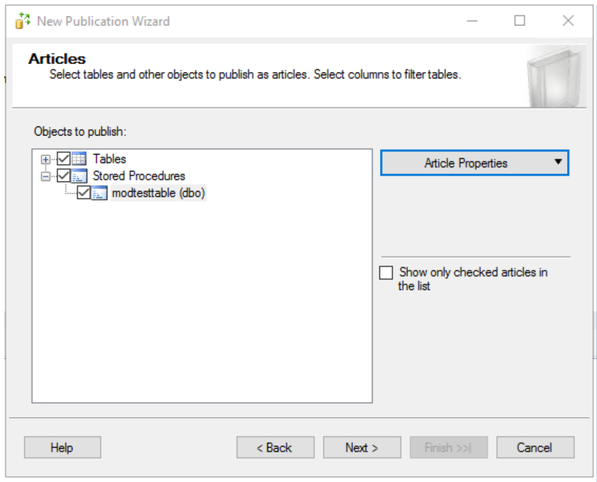
5. Enable your source server for replication as a Publisher.
   - Enable remote Publisher at Distributor (SSMS) - SQL Server | Microsoft Docs

6. Make sure the Log Reader Agent is running.

7. Create a SQL Server single way transactional replication publication but do **NOT** execute the Snapshot Agent.
    - Create a Publication - SQL Server | Microsoft Docs

    Select the option to replicate stored procedure execution

    - Publishing stored procedure execution (Transactional) - SQL Server | Microsoft Docs
    - Publish execution of stored procedure (Transactional) - SQL Server | Microsoft Docs



New Publication Wizard

Articles

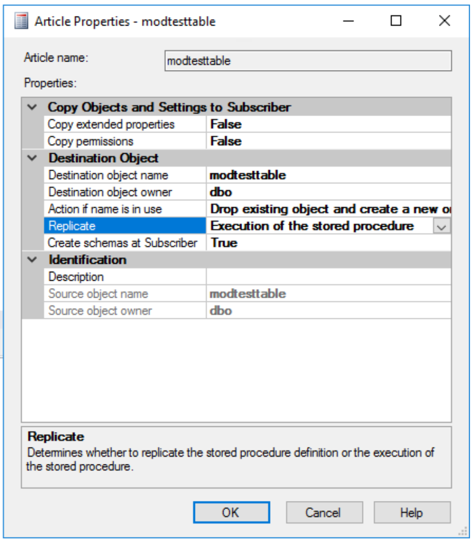
Select tables and other objects to publish as articles. Select columns to filter tables.

Objects to publish:

- ☑ Tables
- ☑ Stored Procedures
    - ☑ modtesttable (dbo)

Article Properties

☐ Show only checked articles in the list



Article Properties - modtesttable

Article name: modtesttable

Properties:

Copy Objects and Settings to Subscriber	
Copy extended properties	False
Copy permissions	False
Destination Object	
Destination object name	modtesttable
Destination object owner	dbo
Action if name is in use	Drop existing object and create a new o
Replicate	Execution of the stored procedure
Create schemas at Subscriber	True
Identification	
Description	
Source object name	modtesttable
Source object owner	dbo

Replicate

Determines whether to replicate the stored procedure definition or the execution of the stored procedure.

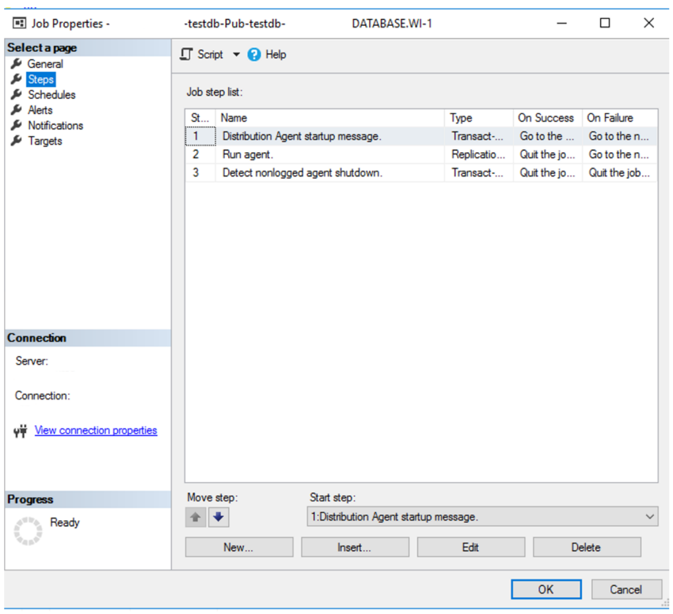
Set the article option to replicate the stored procedure execution.

Do not check the box to execute the Snapshot Agent on the next screen.

8. Add your Azure SQL Database Hyperscale as a subscriber to your replication.

    Select the option to run the distribution agent at the distributor

9. **Disable** the Distribution Agent Job.

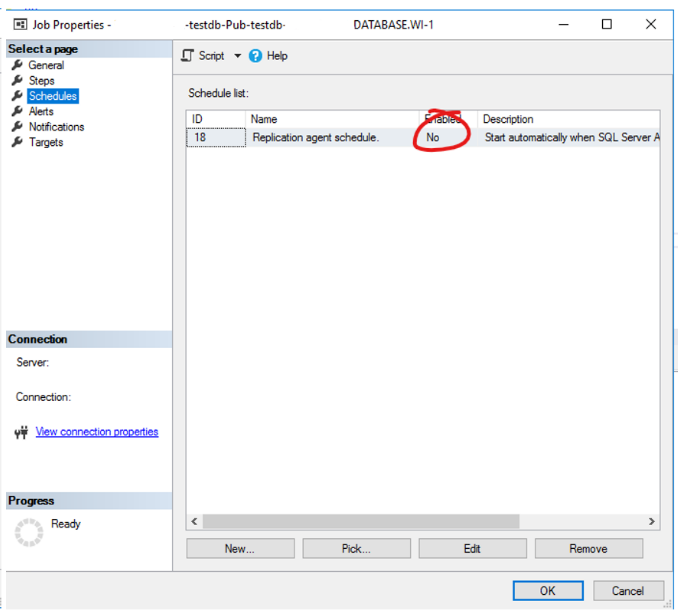

Job Properties - -testdb-Pub-testdb- DATABASE.WI-1

Select a page: General, **Steps**, Schedules, Alerts, Notifications, Targets

Job step list:

St...	Name	Type	On Success	On Failure
1	Distribution Agent startup message.	Transact-...	Go to the ...	Go to the n...
2	Run agent.	Replicatio...	Quit the jo...	Go to the n...
3	Detect nonlogged agent shutdown.	Transact-...	Quit the jo...	Quit the job...

Start step: 1:Distribution Agent startup message.

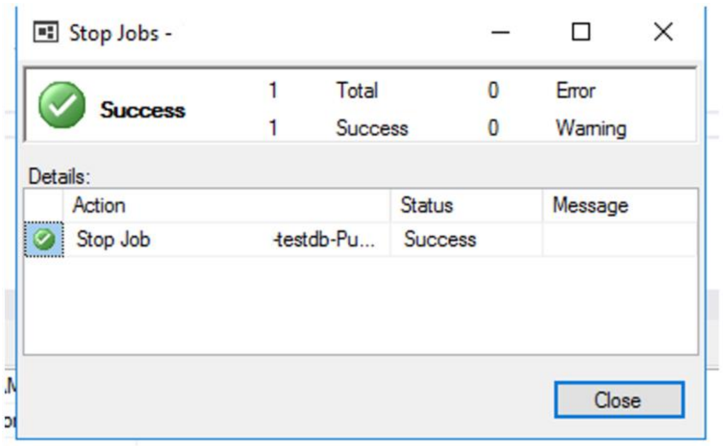

Job Properties - -testdb-Pub-testdb- DATABASE.WI-1

Select a page: General, Steps, **Schedules**, Alerts, Notifications, Targets

Schedule list:

ID	Name	Enabled	Description
18	Replication agent schedule.	No	Start automatically when SQL Server A

10. Stop the Distribution Job on the Distributor


Stop Jobs -

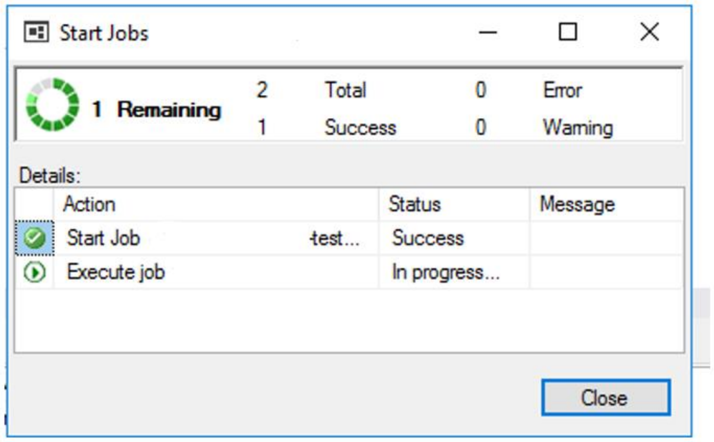
Success

1	Total	0	Error
1	Success	0	Warning

Details:

Action	Status	Message
Stop Job -testdb-Pu...	Success	

11. Select an off-business-peak hour and manually execute the Snapshot Agent.


Start Jobs

1 Remaining

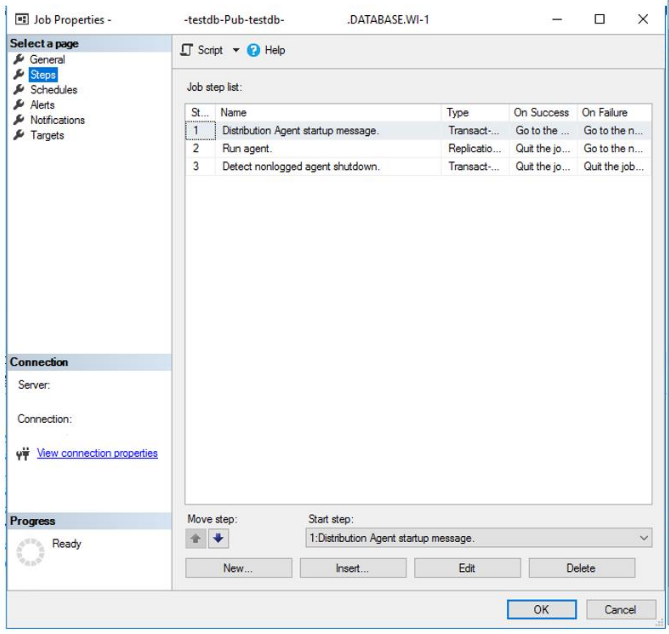
2	Total	0	Error
1	Success	0	Warning

Details:

Action		Status	Message
Start Job	test...	Success	
Execute job		In progress...	

12. After the Snapshot Agent finishes, Zip, split, and upload the complete snapshot folder to the Storage from step 2.

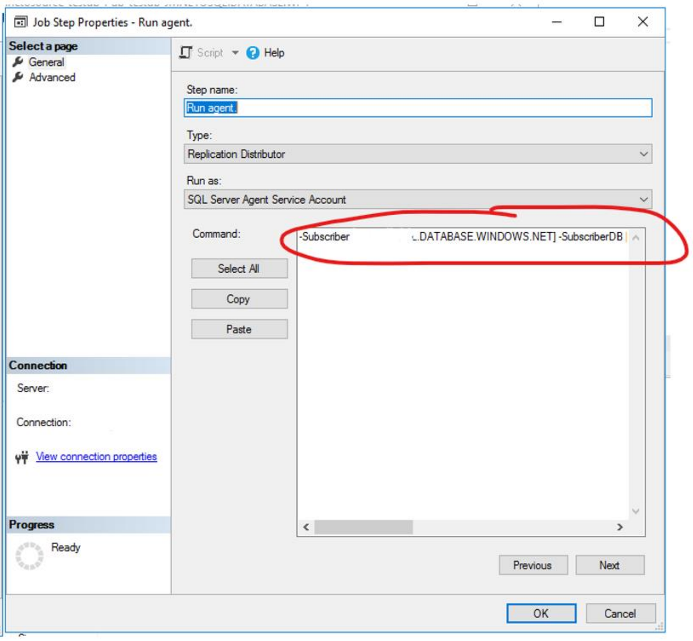
13. Copy the distributor command line from the Distribution Server Job on the distributor server.


Job Properties - -testdb-Pub-testdb- .DATABASE.WI-1

Job step list:

St...	Name	Type	On Success	On Failure
1	Distribution Agent startup message.	Transact-...	Go to the ...	Go to the n...
2	Run agent.	Replicatio...	Quit the jo...	Go to the n...
3	Detect nonlogged agent shutdown.	Transact-...	Quit the jo...	Quit the job...

Start step: 1:Distribution Agent startup message.


Job Step Properties - Run agent.

Step name: Run agent.

Type: Replication Distributor

Run as: SQL Server Agent Service Account

Command: -Subscriber .DATABASE.WINDOWS.NET] -SubscriberDB

Example:

```
-Subscriber [<Your-Server>.DATABASE.WINDOWS.NET] -SubscriberDB [<Your-Target-
Database>] -Publisher [<Your-Publisher-Server>] -Distributor [<Your-
Distributor-Server] -DistributorSecurityMode 1 -Publication [<Your-
Publication>] -PublisherDB [<Your-Source-Database>] ~~-Continuous~~
```

14. On the Virtual Machine from step 1, open a command line and execute the SQL Replication Agent Program Distrib.exe to deliver the Snapshot Agent to Azure SQL Database. For this step we are going to use the parameters from we collected in step 13 plus some adjustments:
    - [Replication Distribution Agent - SQL Server | Microsoft Docs]

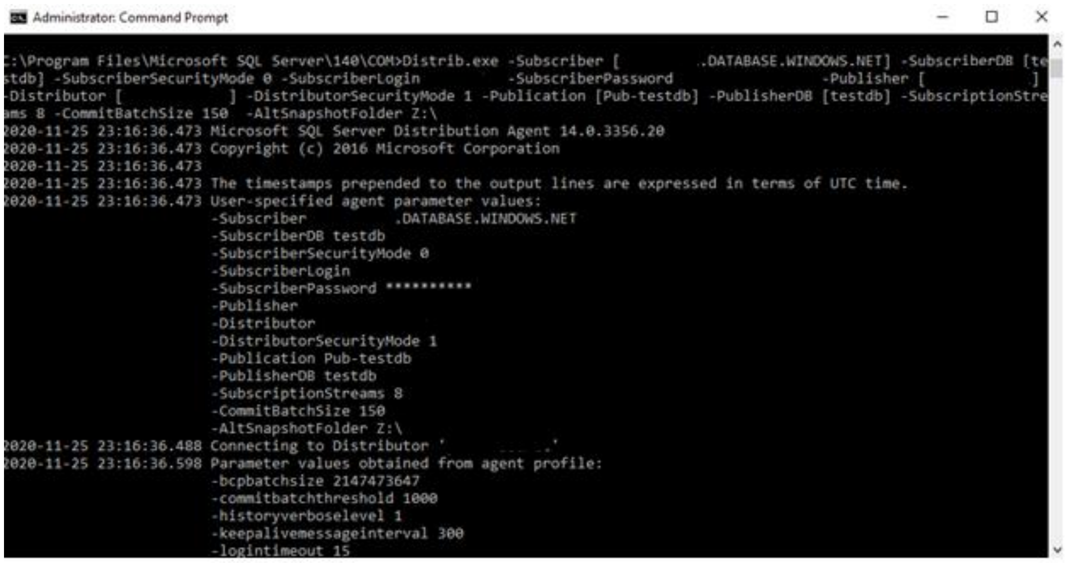
    a. Add distribution agent Distrib.exe parameter -AltSnapshotFolder pointing to the location of the snapshot folder on the Virtual Machine from step 10
    b. Add distribution agent Distrib.exe parameter –SubscriptionStreams with a value equal to 8
    c. Change distribution agent Distrib.exe parameter parameter -CommitBatchSize to 150

    Alternatively, you can use SQL Server Standard Authentication to connect to the distributor, requiring only port 1433 connectivity between Azure and On-Premises

```
CD C:\Program Files\Microsoft SQL Server\140\COM
```

```
Distrib.exe -Subscriber [<Your-Server>.DATABASE.WINDOWS.NET] -
SubscriberDB [<Your-Target-Database>] -SubscriberSecurityMode 0 -
SubscriberLogin <Your-Target-Database-user> -SubscriberPassword <Your-
Target-Database-password> -Publisher [<Your-Publisher-Server>] -
Distributor [<Your-Distributor-Server] -DistributorSecurityMode 1 -
Publication [<Your-Publication>] -PublisherDB [<Your-Source-Database>]
-SubscriptionStreams 8 -CommitBatchSize 150  -AltSnapshotFolder <Your-
Alternate-Snapshot-Folder-Location>
```

    Remove -Continuous so the agent stops after the snapshot is delivered



```
:\Program Files\Microsoft SQL Server\140\COM>Distrib.exe -Subscriber [        .DATABASE.WINDOWS.NET] -SubscriberDB [te
stdb] -SubscriberSecurityMode 0 -SubscriberLogin          -SubscriberPassword                 -Publisher [           ]
-Distributor [            ] -DistributorSecurityMode 1 -Publication [Pub-testdb] -PublisherDB [testdb] -SubscriptionStre
ams 8 -CommitBatchSize 150  -AltSnapshotFolder Z:\
2020-11-25 23:16:36.473 Microsoft SQL Server Distribution Agent 14.0.3356.20
2020-11-25 23:16:36.473 Copyright (c) 2016 Microsoft Corporation
2020-11-25 23:16:36.473
2020-11-25 23:16:36.473 The timestamps prepended to the output lines are expressed in terms of UTC time.
2020-11-25 23:16:36.473 User-specified agent parameter values:
                        -Subscriber          .DATABASE.WINDOWS.NET
                        -SubscriberDB testdb
                        -SubscriberSecurityMode 0
                        -SubscriberLogin
                        -SubscriberPassword **********
                        -Publisher
                        -Distributor
                        -DistributorSecurityMode 1
                        -Publication Pub-testdb
                        -PublisherDB testdb
                        -SubscriptionStreams 8
                        -CommitBatchSize 150
                        -AltSnapshotFolder Z:\
2020-11-25 23:16:36.488 Connecting to Distributor '      .'
2020-11-25 23:16:36.598 Parameter values obtained from agent profile:
                        -bcpbatchsize 2147473647
                        -commitbatchthreshold 1000
                        -historyverboselevel 1
                        -keepalivemessageinterval 300
                        -logintimeout 15
```

15. After the snapshot is delivered, we do not need the Virtual machine and the Snapshot Folder anymore. You can delete those resources.

16. From the publisher server enable and start the distribution job so replication resumes from there.

5 Workaround for tables without a Primary Key

SQL Transaction replication requires that tables must have a primary key to be included in as articles in a transactional replication setup. However, it can also replicate views, indexed views, stored procedures, user-defined functions, and stored procedure execution. As we cannot replicate tables without primary keys, we will create indexed views and published these instead.

The solution will be:

- Create **indexed views** for the tables without primary key that we need to publish. There views require a specific setup of SET options.
- Create a unique clustered index on each of the views.
- Manually create the destination tables on the subscriber with table names the same as the source tables.
- Create publication articles on the indexed views that will replicate data to the pre-created tables on the subscriber.
- These articles must be created with T-SQL only. To replicate the data using indexed views we will need to use the sp_addarticle stored procedure with @type parameter set to the "*indexed view logbased*". Log-based indexed view article. Not supported for Oracle Publishers. For this type of article, the base table does not need to be published separately.
- Use sp_addarticle @destination_table parameter to point to the real destination table we want to replicate the data to.

Example:

```
--Set the options to support indexed views.
SET NUMERIC_ROUNDABORT OFF;
SET ANSI_PADDING, ANSI_WARNINGS, CONCAT_NULL_YIELDS_NULL, ARITHABORT,
   QUOTED_IDENTIFIER, ANSI_NULLS ON;

-- Create views with schemabinding
CREATE VIEW vw_repltablenopk
WITH SCHEMABINDING
AS
SELECT c1, c2, c3, c4, c5 FROM originalsourcetable
GO
```

```sql
-- Create the unique clustered index on the Indexed View
CREATE UNIQUE CLUSTERED INDEX CI_vw_repltablenopk ON vw_repltablenopk
( c1 ASC) ON [PRIMARY]
GO

-- create destination table on the subscriber
USE subscriber
GO

CREATE TABLE destinationtable(
  c1 int NOT NULL,
  c2 varchar(20) NULL,
  c3 varchar(20) NULL,
  c4 varchar(20) NULL,
  c5 varchar(20) NULL)
GO

-- Create the replication article using TSQL
EXEC sys.sp_addarticle @publication = N'test', @article = N'art_ReplNoPk',
   @source_owner = N'dbo', @source_object = N'vw_repltablenopk',
   @destination_owner = N'dbo', @destination_table = N'destinationtable',
   @type ='indexed view logbased', @pre_creation_cmd = N'truncate'

GO
```

6 References

- Transactional Replication - SQL Server | Microsoft Docs
- Publish Data and Database Objects - SQL Server | Microsoft Docs
- Replication Distribution Agent - SQL Server | Microsoft Docs
- Configure replication with availability groups - SQL Server Always On | Microsoft Docs

7 Feedback and suggestions

If you have feedback or suggestions for improving this data migration asset, please contact the Data SQL Ninja Engineering Team (datasqlninja@microsoft.com). Thanks for your support!

Note: For additional information about migrating various source databases to Azure, see the Azure Database Migration Guide.