

# SQL Server Replication as a tool to migrate very large databases

*Prepared by*

Data SQL Ninja Engineering Team ([datasqlninjaex@microsoft.com](mailto:datasqlninjaex@microsoft.com))

## Disclaimer

The High-Level Architecture, Migration Dispositions and guidelines in this document is developed in consultation and collaboration with Microsoft Corporation technical architects. Because Microsoft must respond to changing market conditions, this document should not be interpreted as an invitation to contract or a commitment on the part of Microsoft.

Microsoft has provided generic high-level guidance in this document with the understanding that MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THE INFORMATION CONTAINED HEREIN.

This document is provided "as-is". Information and views expressed in this document, including URL and other Internet Web site references, may change without notice.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2019 Microsoft. All rights reserved.

**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 .....	4
2	Empirical Overview .....	4
3	Limitations .....	6
4	Configurations .....	6
5	References .....	12
6	Feedback and suggestions .....	12

# 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.

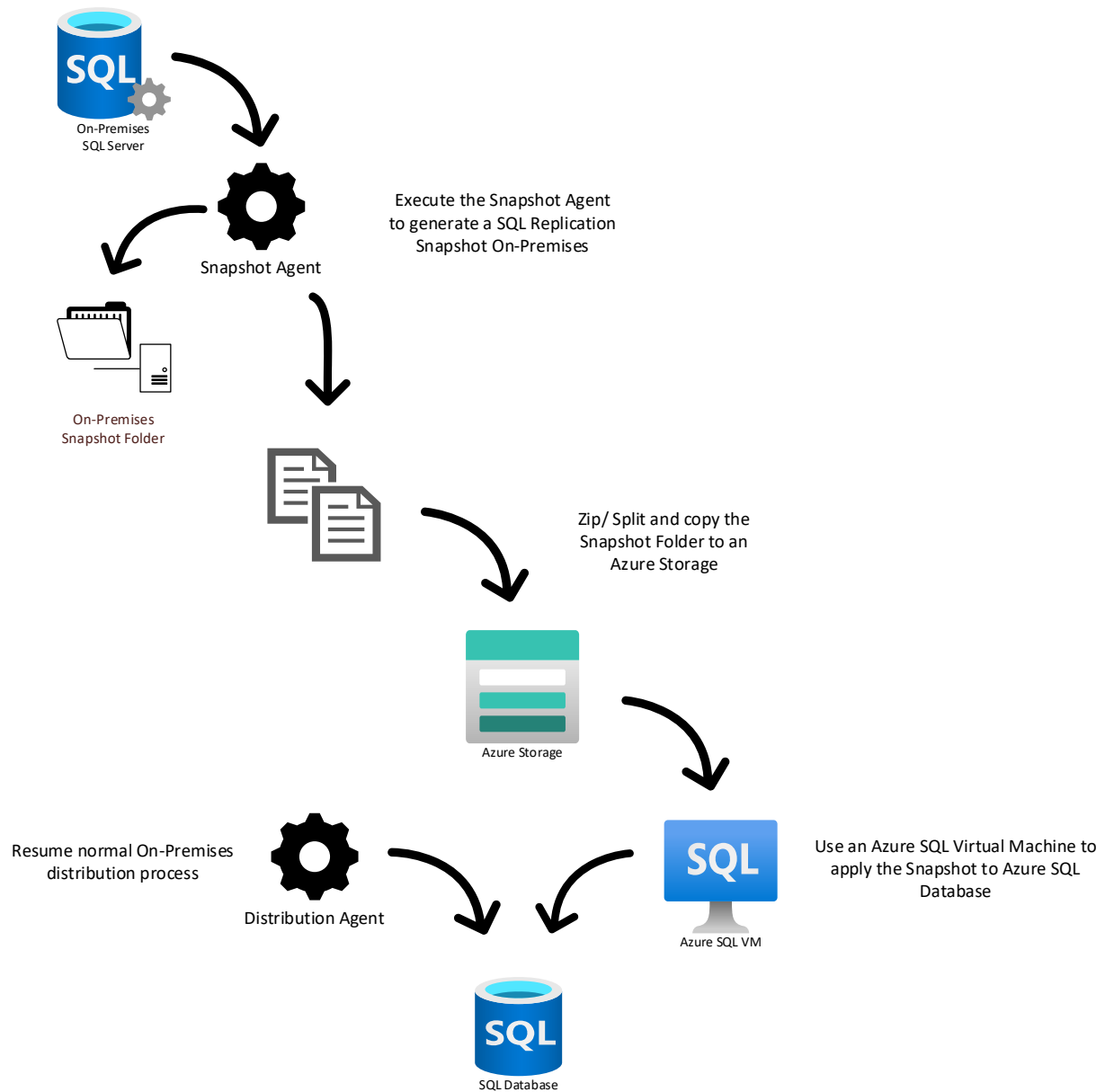
SQL Replication is a perfect use case for specific migration scenarios involving Azure SQL Database and Azure SQL Database Hyperscale as targets when the source database has the correct compatibility requirements for Transactional Replication.

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 apply 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, therefore reducing the required time for snapshot delivery. Transactional replication will then maintain the target database in close sync with the source database allowing for reduced cutoff requirements.

Visual description of the overall process:



## 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.

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 times of lower activity on the Publisher.

The snapshot files require significant storage space what can in certain cases 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 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 from where we will then execute from there the distribution agent to deliver the snapshot.

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 and are free to delete the Azure Virtual Machine and Storage Account.

Additional optimization agent parameter settings should optimize subscriber initialization and continuous synchronization performance. Plus leveraging the replication of stored procedures feature 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](#)

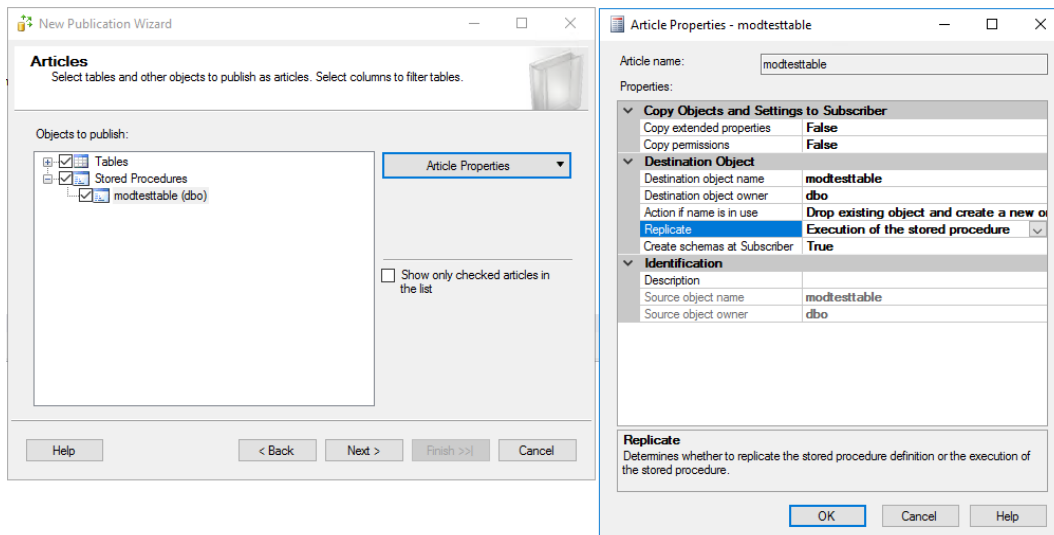
6. Make sure the Log Reader Agent is running.

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](#)



Set the article option to replicate the stored procedure execution.

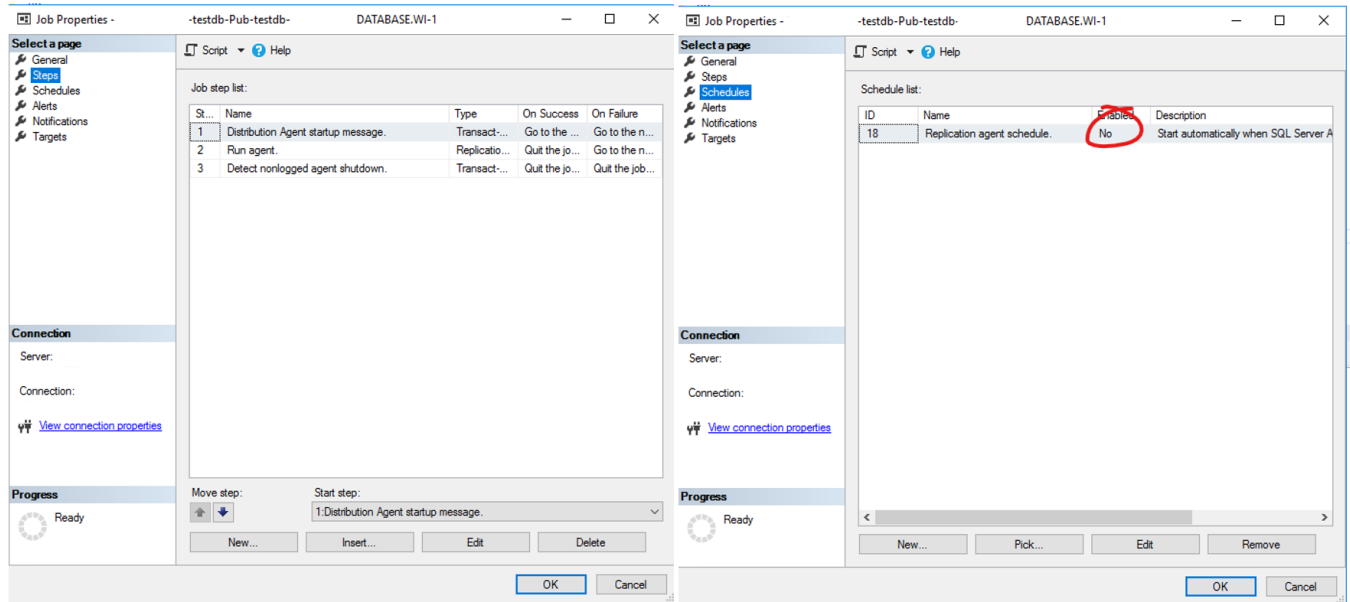
**Do not check the box to execute the Snapshot Agent on the next screen.**



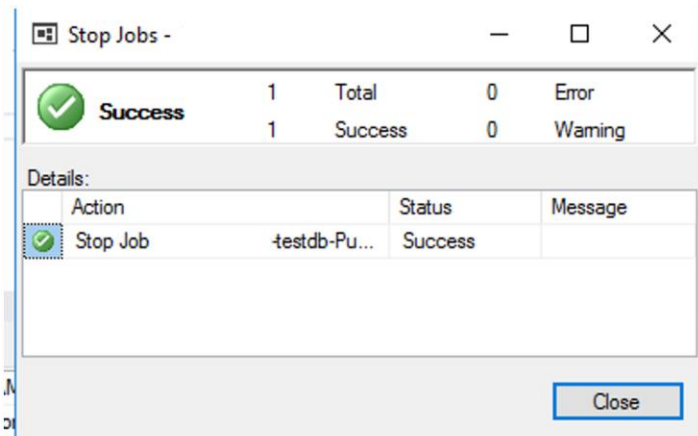
8. Add your Azure SQL Database Hyperscale as a subscriber to your replication.

Select the option to run the distribution agent at the distributor

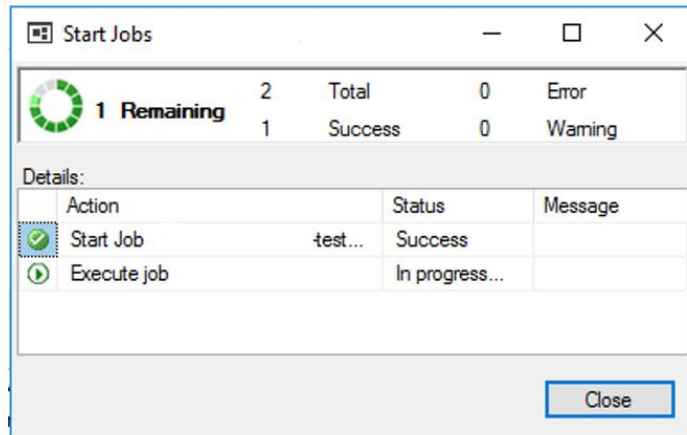
9. **Disable** the Distribution Agent Job.



10. Stop the Distribution Job on the Distributor

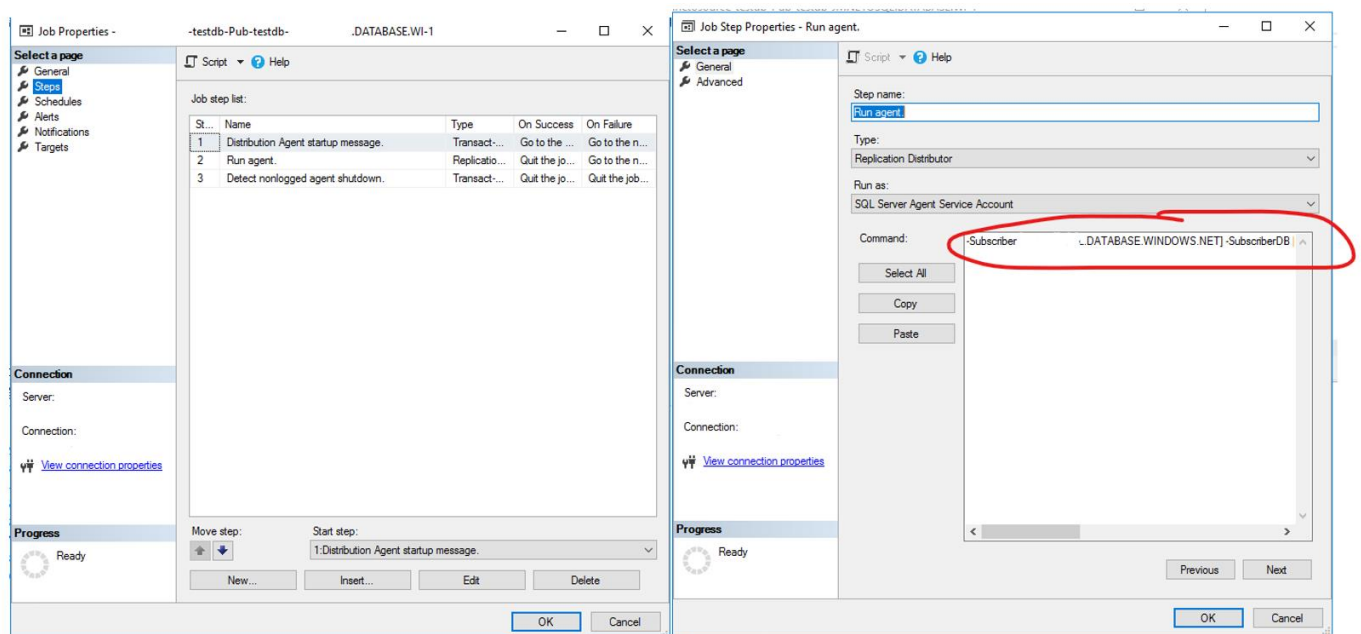


11. Select an off-business-peak hour and manually execute the Snapshot Agent.



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.



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
```

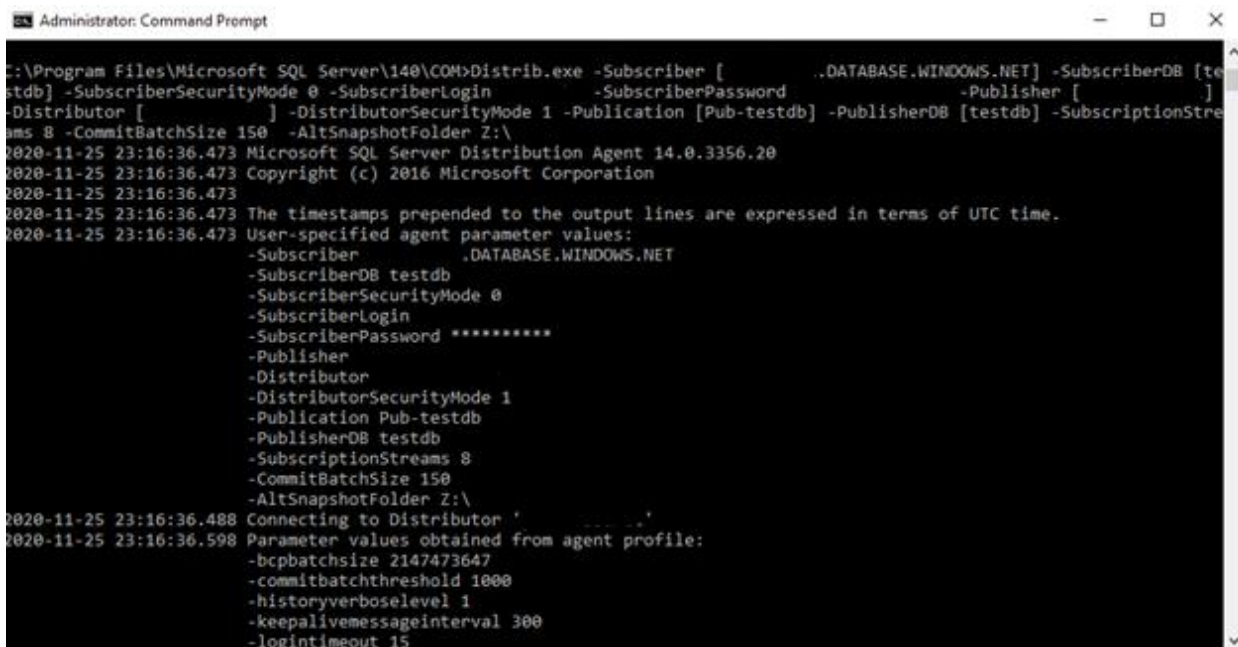
14. On the Virtual Machine from step 1, open a command line and execute the SQL Replication Agent Program Distrib.exe to deliver the Snapshot Agent to Azure SQL Database. For this step we are going to use the parameters from we collected in step 13 plus some adjustments:

- [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

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



```
Administrator: Command Prompt
C:\Program Files\Microsoft SQL Server\140\COM>Distrib.exe -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 23:16:36.473 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 References

- [Transactional Replication - SQL Server | Microsoft Docs](#)
- [Publish Data and Database Objects - SQL Server | Microsoft Docs](#)
- [Replication Distribution Agent - SQL Server | Microsoft Docs](#)

## 6 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](mailto:datasqlninja@microsoft.com)). Thanks for your support!

Note: For additional information about migrating various source databases to Azure, see the [Azure Database Migration Guide](#).