

Migrate SQL Server workloads to Azure SQL Database

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-sql-databases/1-introduction>

Introduction

Completed

Azure SQL Database is a Platform as a Service (PaaS) database engine that provides a complete development and deployment environment in the cloud, which can be used for simple cloud-based applications and for advanced enterprise applications.

Migrating to SQL Database allows you to modernize your application by taking advantage of its PaaS capabilities. This enables you to eliminate dependencies on technical components that are scoped at the instance level, such as SQL Agent jobs. Azure SQL Database offers a low-maintenance solution that can be an excellent choice for certain workloads.

You may have specific requirements that are better suited for Azure SQL Database rather than Azure SQL Managed Instance in the following scenarios:

- You need to simplify deployment for databases with intermittent, unpredictable usage.
- New databases without usage history where compute sizing is difficult or not possible to estimate prior to deployment.
- The complexity of deployment and development is a concern.
- Your storage requirements are higher than what Azure SQL Managed Instance offers, and database consolidation isn't an option.

The process of migrating a SQL Server database running on an Azure virtual machine to Azure SQL Database is similar to the steps we'll learn in this module.

Note

Before proceeding, it is important to ensure that you have reviewed the [Assess SQL Server databases for migration to Azure SQL](#). This module will introduce you to the assessment tools

and help you discover new features in the target SQL Server platform that your database can benefit from after an upgrade.

Use case scenario

Throughout this module, we're using an example scenario to explain key data migration concepts.

Suppose you work for a company that builds bikes and bicycle parts. You have several legacy database servers that you want to upgrade, including a product database, a parts stock database, and a human resources database. You also want to move from a capital expenditure model to an operational expenditure model, and benefit from the scalability and availability of Azure services. You plan to migrate your SQL Server databases to Azure SQL Database. Your board of directors has asked you to plan the migration project and has made you responsible for the execution of the migration tasks.

You'll learn how to migrate SQL Server databases to Azure SQL Database. You'll begin by exploring the pre-migration considerations you need to take into account before a migration, and how to create an Azure SQL database. You'll then explore the different methods for offline and online migrations, and look at ways to move data to Azure SQL Database.

2. Choose the right Azure SQL Database feature

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-sql-databases/2-choose-right-sql-database-option-azure>

Choose the right Azure SQL Database feature

Completed

In our bicycle manufacturing scenario, you've already identified and profiled the databases that you want to migrate to Azure SQL Database. Now, you want to plan the migration, considering data recoverability, disaster recovery, security, and other implementation details.

You'd like to know the tools and features available to support with the migration process to Azure SQL Database.

Benefits of Azure SQL Database

The following summarize the benefits of deploying single and elastic pool databases:

Category	Feature
Backup and recovery	Automatic backup
	Point-in-time restore
	Backup retention 7 days+
	Long-term backup retention stores backups for up to 10 years
High availability	99.99% availability guarantee
	Built-in availability with three secondary replicas
	Zone redundancy via Azure availability zones
Disaster recovery	Geo-restore of database backups
	Active-geo replication between Azure regions
Service scalability	Dynamic scale-up and scale-down
	Scale out with multiple shards
	Share compute resources between databases using elastic pools
Security	Support for Microsoft Entra authentication
	Cloud-only security features such as Advanced Threat Protection
	Transparent data encryption (TDE) enabled by default
	Support for dynamic and static data masking, row-level security, and Always Encrypted
	Firewall allowlist
Licensing	DTU purchasing model for predictive costing
	vCore purchasing model, enabling storage to be scaled independently of compute
	Combine the vCore purchasing model with Azure Hybrid Benefit for SQL Server to realize cost savings of up to 30 percent

Tip

To review the benefits of migrating to Azure SQL Database and the features available, please refer to [Deploy PaaS solutions with Azure SQL](#) module.

Exclusive features of Azure SQL Database

Some features are supported in Azure SQL Database that aren't available in other Azure SQL offerings:

Feature	Definition
Hyperscale	Cloud-native architecture that allows for independently scalable compute and storage, providing greater flexibility and resources than other tiers.
Auto-scale	With serverless compute tier
Automatic tuning (indexes)	This built-in feature automatically identifies and creates indexes that can improve the performance of your workload. It also verifies that query performance has improved and removes unused or duplicate indexes.
Elastic query	Allows you to run T-SQL queries that bridge multiple databases in SQL Database. This feature is useful for applications using three- and four-part names that can't be changed.
Elastic jobs	The elastic job feature is the SQL Server Agent replacement for Azure SQL Database. To some extent, elastic job is equivalent to the Multi Server Administration feature available on SQL Server instance.
Query Performance Insights (QPI)	This tool helps find the queries to optimize to improve overall workload performance and efficiently use the resource that you're paying for.

Important

To understand additional feature differences between SQL Database, SQL Server, and Azure SQL Managed Instance, as well as the differences among different Azure SQL Database options, see [SQL Database features](#).

Migration options supported

There are two modes of migration to Azure SQL Database: **Online** and **Offline**. The online mode has minimal or no downtime, while the offline mode experiences downtime during the migration process.

Tool	Migration mode
Azure Database Migration Service	Offline
Transactional replication	Online
Azure Migrate	Offline
Import Export Wizard/BACPAC	Offline
Bulk copy (bcp utility)	Offline
Azure Data Factory	Offline

* Can have a higher performance impact, depending on the workload.

Note

We recommend that you use the [Azure Database Migration Service](#) for large migrations and enhanced overall experience.

Migration performance

Consider the following recommendations when migrating to Azure SQL Database:

- Monitor data file I/O and latency on the source, and mitigate any bottlenecks.
- Scale up the target Azure SQL database to Business Critical Gen5 8 vCore or use the Hyperscale service tier to minimize latency for log files.
- Ensure that your network bandwidth can accommodate the maximum log ingestion rate.
- Choose the highest service tier and compute size for maximum transfer performance, and scale down after migration.
- Minimize the distance between BACPAC files and the destination data center.
- Disable auto update and auto create statistics during migration.
- Partition tables and indexes, drop indexed views, and recreate them after migration.
- Consider migrating rarely queried historical data to a separate database in Azure SQL Database, and query it using elastic queries.

Retry application connections

When migrating to Azure SQL Database, it's important to anticipate occasional transient failures when connecting to the database resource, and implement a proper retry logic method. Setting a maximum number of retries before the program terminates is also important.

We recommend waiting for 5 seconds at a minimum on your first retry. Each subsequential retry should increase the delay exponentially, up to a maximum of 60 seconds.

Note

If a SELECT statement fails with a transient error for SQL Database, don't directly retry it. Instead, retry the SELECT statement in a new connection.

To learn more about the connection retry principals, see [Troubleshoot transient connection errors in SQL Database and SQL Managed Instance](#).

3. Use Azure Database Migration Service to migrate to Azure SQL Database

Use Azure Database Migration Service to migrate to Azure SQL Database

Completed

If you can afford to take the database offline while you migrate to Azure, you have a few tools you can use.

In our bicycle manufacturing scenario, the HR database is considered business-critical but is rarely used at weekends. You've planned to execute an offline migration between Friday evening and Monday morning, but you want to assess the best migration method.

It's assumed that all pre-migration checks have been done with [Azure Migrate](#). This process ensures that feature and compatibility issues are addressed.

Migrate using Azure Database Migration Service with Azure CLI

[Azure Database Migration Service](#) is a fully managed service designed to enable seamless migrations from multiple database sources to Azure data platforms with minimal downtime. You can use Azure CLI or PowerShell to automate database migrations, making it ideal for migrating databases at scale.

The Azure CLI `az datamigration` extension provides commands to create and manage database migrations to Azure SQL Database. This approach is particularly useful for:

- Automating migrations as part of CI/CD pipelines
- Migrating multiple databases at scale
- Scripting repeatable migration processes

Prerequisites

Before starting the migration, ensure you have:

1. **Azure CLI installed** with the `datamigration` extension
2. **Azure Database Migration Service** created in your subscription
3. **Target Azure SQL Database** provisioned with the schema already deployed
4. **Self-hosted integration runtime** configured for connectivity to your source SQL Server

To install the Azure CLI `datamigration` extension, run:

```
az extension add --name datamigration
```

Create the Database Migration Service

First, create an Azure Database Migration Service to orchestrate your migration activities:

```
# Create the Azure Database Migration Service
az datamigration sql-service create \
  --resource-group "<YourResourceGroup>" \           # Name of your Azure resource group
  --sql-migration-service-name "<YourMigrationService>" \ # Name for the migration service
  --location "<YourLocation>"                        # Azure region (e.g., eastus)
```

Migrate the database schema

Before migrating data, you need to migrate the database schema from the source to the target. Use the `az datamigration sql-server-schema` command:

```
# Migrate schema from source to target database
az datamigration sql-server-schema \
  --action "MigrateSchema" \
  --src-sql-connection-str "Server=<YourSourceServer>;Initial Catalog=<YourSourceDatabase>" \
  --tgt-sql-connection-str "Server=<YourTargetServer>.database.windows.net;Initial Catalog=<YourTargetDatabase>"
```

Start the database migration

Create a new database migration to copy data from the source to target:

```
# Create a database migration to Azure SQL Database
az datamigration sql-db create \
  --resource-group "<YourResourceGroup>" \           # Name of your Azure resource group
  --sqldb-instance-name "<YourTargetServer>" \        # Name of the target Azure SQL Database
  --target-db-name "<YourTargetDB>" \                 # Name of the target database
  --source-database-name "<YourSourceDB>" \           # Name of the source database
  --source-sql-connection authentication="SqlAuthentication" \
    data-source="<YourSourceServer>" \               # Source SQL Server hostname
    user-name="<YourSourceUser>" \                   # Source SQL Server username
    password="<YourSourcePassword>" \               # Source SQL Server password
    encrypt-connection=true \
    trust-server-certificate=true \
  --target-sql-connection authentication="SqlAuthentication" \
    data-source="<YourTargetServer>.database.windows.net" \ # Target Azure SQL Database
    user-name="<YourTargetUser>" \                   # Target database username
    password="<YourTargetPassword>" \               # Target database password
    encrypt-connection=true \
    trust-server-certificate=true \
  --scope "/subscriptions/<YourSubscription>/resourceGroups/<YourResourceGroup>/providers/Microsoft.DataMigration/migrationServices/<YourMigrationService>" \
  --migration-service "/subscriptions/<YourSubscription>/resourceGroups/<YourResourceGroup>/providers/Microsoft.DataMigration/migrationServices/<YourMigrationService>"
```

Migrate specific tables

To migrate only specific tables, use the `--table-list` parameter:

```
# Create a database migration for specific tables
az datamigration sql-db create \
  --resource-group "<YourResourceGroup>" \
  --sqldb-instance-name "<YourTargetServer>" \
  --target-db-name "<YourTargetDB>" \
  --source-database-name "<YourSourceDB>" \
  --source-sql-connection authentication="SqlAuthentication" \
    data-source="<YourSourceServer>" \
    user-name="<YourSourceUser>" \
    password="<YourSourcePassword>" \
    encrypt-connection=true \
    trust-server-certificate=true \
  --target-sql-connection authentication="SqlAuthentication" \
    data-source="<YourTargetServer>.database.windows.net" \
    user-name="<YourTargetUser>" \
    password="<YourTargetPassword>" \
    encrypt-connection=true \
    trust-server-certificate=true \
  --table-list "[Person].[Person]" "[Person].[EmailAddress]" "[Sales].[Customer]" \
  --scope "/subscriptions/<YourSubscription>/resourceGroups/<YourResourceGroup>/providers/Microsoft.DataMigration/dmigrations/<YourMigration>" \
  --migration-service "/subscriptions/<YourSubscription>/resourceGroups/<YourResourceGroup>/providers/Microsoft.DataMigration/dmigrations/<YourMigration>"
```

The Database Migration Service optimizes the migration process by skipping empty tables, even if you select them.

Migration status

There are a few statuses that keep you updated on the progress of the migration.

- **Preparing for copy:** The service is in the process of disabling autostats, triggers, and indexes in the target table.
- **Copying:** The data copy from the source database to the target database is in progress.
- **Copy finished:** Data copy is finished, and the service is waiting on other tables to finish copying.
- **Rebuilding indexes:** The service is rebuilding indexes on target tables.
- **Succeeded:** All data is copied and the indexes are rebuilt.

Monitor migration using Azure CLI

You can check the status of your migration using the `az datamigration sql-db show` command:

```
# Check the status of the database migration
az datamigration sql-db show \
  --resource-group "<YourResourceGroup>" \
  --sqldb-instance-name "<YourTargetServer>" \
```

```
--target-db-name "<YourTargetDB>" \  
--expand "MigrationStatusDetails" # Include detailed migration
```

This command returns detailed information about the migration, including the current status and any errors encountered.

Wait for migration completion

You can use the `wait` command to pause script execution until the migration completes:

```
# Wait for the migration to complete before continuing  
az datamigration sql-db wait \  
  --resource-group "<YourResourceGroup>" \  
  --sqldb-instance-name "<YourTargetServer>" \  
  --target-db-name "<YourTargetDB>" \  
  --created # Wait until migration is c
```

Cancel a migration

If you need to stop an in-progress migration:

```
# Cancel an in-progress migration  
az datamigration sql-db cancel \  
  --resource-group "<YourResourceGroup>" \  
  --sqldb-instance-name "<YourTargetServer>" \  
  --target-db-name "<YourTargetDB>" \  
  --migration-operation-id "<YourMigrationOperationId>" # ID from the migration
```

Monitor migration from the Azure portal

You can also monitor the migration activity using Azure Database Migration Service in the Azure portal.

To monitor your database migration, go to the Azure portal and find your instance of the Database Migration Service. Once you've located the service, you can view its instance overview. Select **Monitor migrations** to access detailed information about your ongoing database migration.

AdventureWorks

[Cancel migration](#)
[Delete migration](#)
[Retry](#)
[Copy migration details](#)
[Refresh](#)

Essentials

Source database name : AdventureWorks

Source server : localhost

Migration status : In progress

Target database name : my-db

Target server : courseware

Target version : Azure SQL Database

Table name ↑↓	Status ↑↓	Data read ↑↓	Data written ↑↓	Rows read ↑↓	Rows copied ↑↓	Copy throughput ↑↓	Copy duration ↑↓	Parallel copy type ↑↓	Used parallel copies ↑↓	Copy start ↑↓
[Sales].[SalesReason]	Copying	0.00 MB	0.00 MB	0	0	0.00 MB	00:00:07	None	1	09/01/2023, 10:39:46 AM
[Sales].[SalesTaxRate]	Copying	0.00 MB	0.00 MB	0	0	0.00 MB	00:00:11	None	1	09/01/2023, 10:39:42 AM
[Sales].[SpecialOfferProduct]	Copy finished	0.02 MB	0.02 MB	538	538	0.00 MB	00:00:14	None	1	09/01/2023, 10:39:34 AM
[Sales].[SalesPersonQuotaHistory]	Copy finished	0.01 MB	0.01 MB	163	163	0.00 MB	00:00:16	None	1	09/01/2023, 10:39:24 AM
[Sales].[SpecialOffer]	Copy finished	0.00 MB	0.00 MB	16	16	0.00 MB	00:00:15	None	1	09/01/2023, 10:39:23 AM
[Sales].[SalesTerritoryHistory]	Copy finished	0.00 MB	0.00 MB	17	17	0.00 MB	00:00:28	None	1	09/01/2023, 10:39:11 AM
[Sales].[Store]	Copy finished	0.61 MB	0.61 MB	701	701	0.03 MB	00:00:26	None	1	09/01/2023, 10:39:06 AM
[Sales].[SalesTerritory]	Copy finished	0.00 MB	0.00 MB	10	10	0.00 MB	00:00:30	None	1	09/01/2023, 10:39:04 AM
[Sales].[ShoppingCartItem]	Copy finished	0.00 MB	0.00 MB	3	3	0.00 MB	00:00:28	None	1	09/01/2023, 10:38:44 AM
[Sales].[SalesPerson]	Copy finished	0.00 MB	0.00 MB	17	17	0.00 MB	00:00:30	None	1	09/01/2023, 10:38:33 AM

Showing 1 - 10 of 70 results.

< 1 2 3 4 5 >

After the migration status is **Succeeded**, navigate to the target server, and validate the target database. Check the database schema and data.

Performance considerations

Migration speed heavily depends on the target Azure SQL Database SKU and the self-hosted Integration Runtime host. We strongly recommend that you scale up your Azure SQL Database compute resources before initiating the migration process for an optimal migration experience.

When deciding on the server to install the self-hosted integration runtime, make sure this machine can handle the CPU and memory load of the data copy operation.

Azure SQL Database migration can be slow with a large volume of tables due to the time Azure Data Factory (ADF) takes to start activities, even for small tables.

Tables with large blob columns may fail to migrate due to timeout.

We recommend up to 10 concurrent database migrations per self-hosted integration runtime on a single computer. Scale out the self-hosted runtime or create separate instances on different computers to increase the concurrent database migrations.

Migrate at scale using PowerShell

You can also perform an offline migration of the database from SQL Server on-premises to an Azure SQL Database by using PowerShell.

The following example migrates the *AdventureWorks* database to Azure SQL Database.

```
# Set up secure credentials for source and target connections
$sourcePass = ConvertTo-SecureString "<YourSourcePassword>" -AsPlainText -Force
$targetPass = ConvertTo-SecureString "<YourTargetPassword>" -AsPlainText -Force

# Start the database migration to Azure SQL Database
```

```
New-AzDataMigrationToSqlDb `
  -ResourceGroupName "<YourResourceGroup>" `           # Name of your Azure resource group
  -SqlDbInstanceName "<YourTargetServer>" `             # Name of the target Azure SQL Database
  -Kind "SqlDb" `
  -TargetDbName "<YourTargetDB>" `                     # Name of the target database
  -SourceDatabaseName "<YourSourceDB>" `               # Name of the source database
  -SourceSqlConnectionAuthentication SQLAuthentication `
  -SourceSqlConnectionDataSource "<YourSourceServer>" ` # Source SQL Server host name
  -SourceSqlConnectionUserName "<YourSourceUser>" `    # Source SQL Server user name
  -SourceSqlConnectionPassword $sourcePass `
  -Scope "/subscriptions/<YourSubscription>/resourceGroups/<YourResourceGroup>/providers/Microsoft.DataMigration/migrations/<YourMigration>" `
  -TargetSqlConnectionAuthentication SQLAuthentication `
  -TargetSqlConnectionDataSource "<YourTargetServer>.database.windows.net" `
  -TargetSqlConnectionUserName "<YourTargetUser>" `    # Target database user name
  -TargetSqlConnectionPassword $targetPass `
  -MigrationService "/subscriptions/<YourSubscription>/resourceGroups/<YourResourceGroup>/providers/Microsoft.DataMigration/migrations/<YourMigration>"
```

The following example migrates a subset of tables from the *AdventureWorks* database.

```
# Migrate specific tables from source to target database
New-AzDataMigrationToSqlDb `
  -ResourceGroupName "<YourResourceGroup>" `
  -SqlDbInstanceName "<YourTargetServer>" `
  -Kind "SqlDb" `
  -TargetDbName "<YourTargetDB>" `
  -SourceDatabaseName "<YourSourceDB>" `
  -SourceSqlConnectionAuthentication SQLAuthentication `
  -SourceSqlConnectionDataSource "<YourSourceServer>" `
  -SourceSqlConnectionUserName "<YourSourceUser>" `
  -SourceSqlConnectionPassword $sourcePass `
  -Scope "/subscriptions/<YourSubscription>/resourceGroups/<YourResourceGroup>/providers/Microsoft.DataMigration/migrations/<YourMigration>" `
  -TargetSqlConnectionAuthentication SQLAuthentication `
  -TargetSqlConnectionDataSource "<YourTargetServer>.database.windows.net" `
  -TargetSqlConnectionUserName "<YourTargetUser>" `
  -TargetSqlConnectionPassword $targetPass `
  -TableList "[Person].[Person]", "[Person].[EmailAddress]" ` # Specify tables to migrate
  -MigrationService "/subscriptions/<YourSubscription>/resourceGroups/<YourResourceGroup>/providers/Microsoft.DataMigration/migrations/<YourMigration>"
```

To learn more about database migration commands, refer to the following links: [PowerShell module for data migration](#) and [Azure CLI for data migration](#).

4. Migrate to Azure SQL Database using BACPAC

Migrate to Azure SQL Database using BACPAC

Completed

A SQL Server database can be imported into an Azure SQL database using a **.bacpac** file.

A **.bacpac** file is a compressed file containing the metadata and data from the database. The data can be imported from the Azure Blob Storage or from a local storage in an on-premises environment.

For optimal scale and performance in production environments, it's recommended to use the **SQLPackage** utility. Running multiple **SqlPackage** commands in parallel for subsets of tables can significantly accelerate import/export operations.

Import from a BACPAC file in the Azure portal

You can follow these steps to import a **.bacpac** file in Azure SQL Database.

1. To import from a BACPAC file into a new single database using the Azure portal, open the appropriate database server page and then, on the toolbar, select **Import database**.
2. Select the storage account and container for the BACPAC file, and then select the BACPAC file from which to import.
3. Specify the new database size (usually the same as the origin) and provide the destination SQL Server credentials, and then select **OK**.
4. To monitor an import's progress, open the database server page and, under **Settings**, select **Import/Export history**. When successful, the import has a **Completed** status.

Additionally, you can also use **SqlPackage** to import a BACPAC file as it's faster than using the Azure portal. The following command imports the **AdventureWorks2019** database from local storage to an Azure SQL Database server called `<server-name>`. It creates a new database called **myMigratedDatabase** with a **Premium** service tier and a **P6** service objective.

Change these values as appropriate for your environment.

```
SqlPackage.exe /a:import /tcs:"Data Source=<server-name>.database.windows.net;Init:
```

Tip

To increase the speed of the import process, you can scale your database to a higher service tier and compute size, providing more and faster resources. Once the import is complete, you can scale down to your desired service tier and compute size.

5. Use an online method to migrate to Azure SQL Database

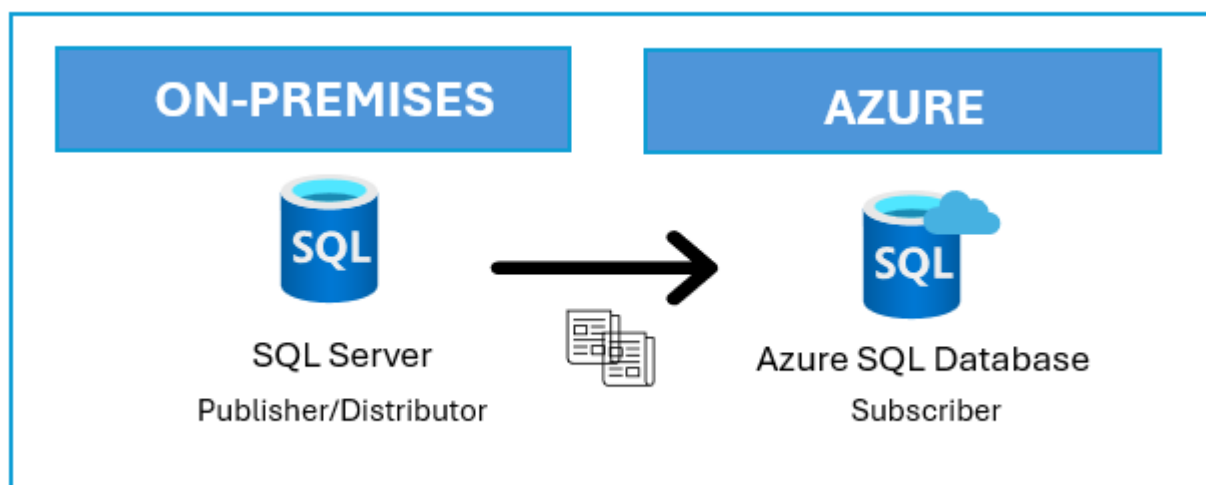
<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-sql-databases/6-migrate-sql-server-azure-sql-database-online>

Use an online method to migrate to Azure SQL Database

Completed

If you need a database to remain online to users throughout the migration process, you can use transactional replication to move the data. Transactional replication is the only online method available for migrating to Azure SQL Database.

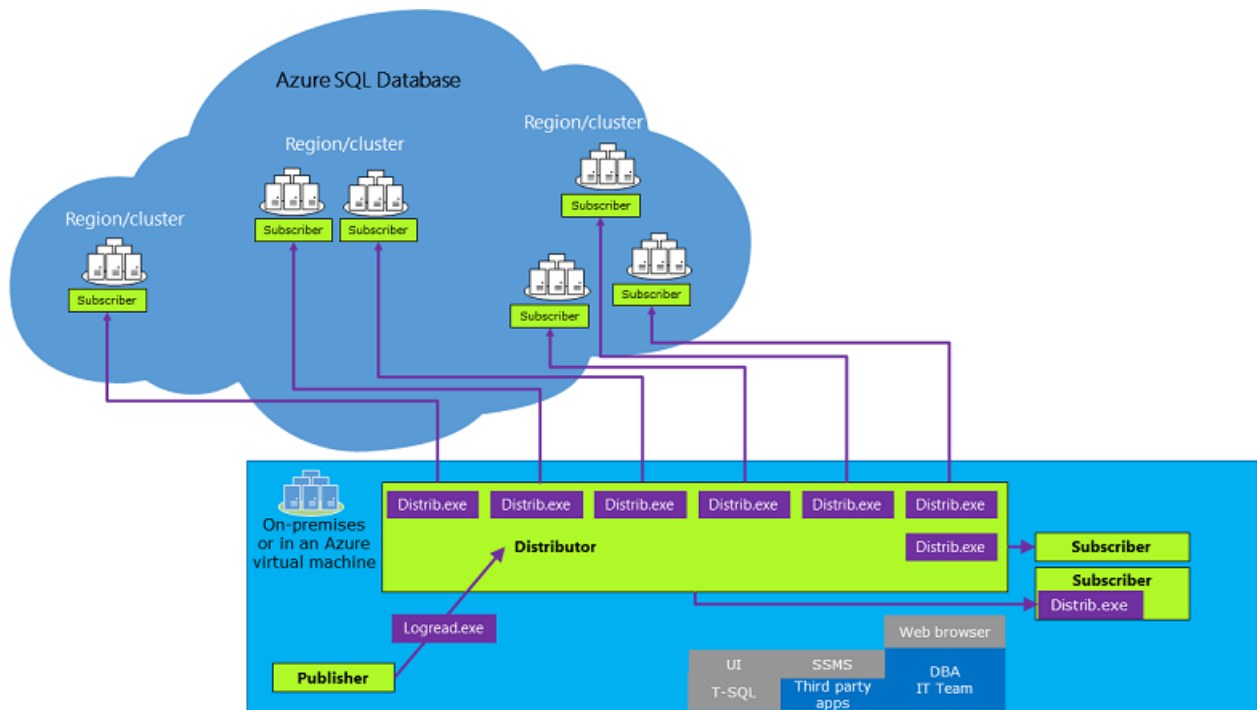
In our bicycle manufacturer scenario, warehouses run on a 24 hour, 7 days a week basis, and there are no periods of inactivity. Your board of directors wants to be sure that the inventory database is constantly available, even during the migration to Azure SQL Database.



What is transactional replication?

Transactional replication is a way to move data between continuously connected database servers.

The process begins with a snapshot of the publication database objects and data. Once the initial snapshot is taken, any subsequent changes to the data or schema at the Publisher are typically delivered to Azure SQL Database in near real-time as they occur.



Azure SQL Database supports both transactional and snapshot replication as a push subscriber. That means Azure SQL Database can receive and apply changes from a publisher using either a transactional or snapshot replication.

The publisher and/or distributor can be an instance of SQL Server that is either running on-premises, on an Azure virtual machine in the cloud, or as an Azure SQL Managed Instance.

You can configure transactional replication through SQL Server Management Studio, or by executing Transact-SQL statements on the publisher. Transactional replication can't be configured from the Azure portal.

Transactional replication requires the following components:

Role	Definition
Publisher	A database instance that hosts the data to be replicated (source).
Subscriber	Receives the data being replicated by the <i>Publisher</i> (target).
Distributor	Collects changes in the articles from a <i>Publisher</i> and distributes them to the <i>Subscribers</i> .
Article	A database object; for example, a table that's included in the <i>Publication</i> .
Publication	A collection of one or more articles from the database being replicated.
Subscription	A request from a <i>Subscriber</i> for a <i>Publication</i> .

Set up a transactional replication

Follow the steps below to migrate the table `[Person].[Person]` from *AdventureWorks* database to Azure SQL Database with no downtime. Transactional replication can only use SQL Server authentication logins to connect to Azure SQL Database.

Parameter	Definition
<code>@distributor</code>	Source instance name.
<code>@publisher</code>	Source instance name.
<code>@subscriber</code>	Azure SQL Database in the format: <code><server>.database.windows.net</code> . The Azure SQL Database must exist before running the script.
<code>@dbname</code>	Database name at the source.
<code>@publisher_login</code>	SQL user with required permissions at the source.
<code>@publisher_password</code>	Password for the SQL user.
<code>@destination_db</code>	Database name at the destination.
<code>@subscriber_login</code>	SQL user with required permissions at the destination.
<code>@subscriber_password</code>	Password for the SQL user.
<code>@working_directory</code>	Replication working directory, change this location as appropriate.

Adjust the parameters above according to your own environment when running the script.

Create the distributor

The following script creates the distributor database, distributor publishers, and the agents.

```
USE [master]
GO

EXEC sp_adddistributor @distributor = N'CONTOSO-SRV', @password = N''
GO

EXEC sp_adddistributiondb
    @database = N'distribution',
    @data_folder = N'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQL
    @data_file = N'distribution.MDF',
    @data_file_size = 13,
    @log_folder = N'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLS
    @log_file = N'distribution.LDF',
    @log_file_size = 9,
    @min_distretention = 0,
    @max_distretention = 72,
    @history_retention = 48,
    @deletebatchsize_xact = 5000,
    @deletebatchsize_cmd = 2000,
```

```

                @security_mode = 1
GO

-- Adding the distribution publishers
exec sp_adddistpublisher
    @publisher = N'CONTOSO-SRV',
    @distribution_db = N'distribution',
    @security_mode = 1,
    @working_directory = N'C:\REPL',
    @trusted = N'false',
    @thirdparty_flag = 0,
    @publisher_type = N'MSSQLSERVER'
GO

exec sp_addsubscriber
    @subscriber = N'contoso.database.windows.net',
    @type = 0,
    @description = N'Azure SQL Database (target)'
GO

-- Enabling the replication database
use master
exec sp_replicationdboption
    @dbname = N'AdventureWorks',
    @optname = N'publish',
    @value = N'true'
GO

--Adds a Log Reader agent for the AdventureWorks database.
exec [AdventureWorks].sys.sp_addlogreader_agent
    @publisher_security_mode = 1
GO

--Adds a Queue Reader agent for the distributor.
exec [AdventureWorks].sys.sp_addqreader_agent
    @frompublisher = 1
GO

```

Create the transactional publication

The following script creates the transactional publication of the `AdventureWorks` database from the publisher.

```

USE [AdventureWorks]
GO

EXEC sp_addpublication
    @publication = N'REPL-AdventureWorks',
    @description = N'Transactional publication of database ''AdventureWorks'' f:
    @sync_method = N'concurrent',
    @retention = 0,
    @allow_push = N'true',
    @allow_pull = N'true',

```

```

        @allow_anonymous = N'true',
        @enabled_for_internet = N'false',
        @snapshot_in_defaultfolder = N'false',
        @alt_snapshot_folder = N'C:\REPL',
        @compress_snapshot = N'true',
        @ftp_port = 21,
        @ftp_login = N'anonymous',
        @allow_subscription_copy = N'false',
        @add_to_active_directory = N'false',
        @repl_freq = N'continuous',
        @status = N'active',
        @independent_agent = N'true',
        @immediate_sync = N'true',
        @allow_sync_tran = N'false',
        @autogen_sync_procs = N'false',
        @allow_queued_tran = N'false',
        @allow_dts = N'false',
        @replicate_ddl = 1,
        @allow_initialize_from_backup = N'false',
        @enabled_for_p2p = N'false',
        @enabled_for_het_sub = N'false'
GO

exec sp_addpublication_snapshot
        @publication = N'REPL-AdventureWorks',
        @frequency_type = 1,
        @frequency_interval = 0,
        @frequency_relative_interval = 0,
        @frequency_recurrence_factor = 0,
        @frequency_subday = 0,
        @frequency_subday_interval = 0,
        @active_start_time_of_day = 0,
        @active_end_time_of_day = 235959,
        @active_start_date = 0,
        @active_end_date = 0,
        @publisher_security_mode = 0,
        @publisher_login = N'sqladmin',
        @publisher_password = N'<pwd>'

```

Create the article for the publication

The following script creates the article for the `[Person].[Person]` table.

```

USE [AdventureWorks]
GO

EXEC sp_addarticle
        @publication = N'REPL-AdventureWorks',
        @article = N'Person',
        @source_owner = N'Person',
        @source_object = N'Person',
        @type = N'logbased',
        @description = N'',

```

```

        @creation_script = N'',
        @pre_creation_cmd = N'drop',
        @schema_option = 0x000000000803509F,
        @identityrangemanagementoption = N'none',
        @destination_table = N'Person',
        @destination_owner = N'Person',
        @status = 24,
        @vertical_partition = N'false',
        @ins_cmd = N'CALL [sp_MSins_PersonPerson]',
        @del_cmd = N'CALL [sp_MSdel_PersonPerson]',
        @upd_cmd = N'SCALL [sp_MSupd_PersonPerson]'
GO

```

Create the subscription and the subscription agent

The following script creates the push subscription to the Azure SQL Database subscriber.

```

USE [AdventureWorks]
GO

EXEC sp_addsubscription
    @publication = N'REPL-AdventureWorks',
    @subscriber = N'contoso.database.windows.net',
    @destination_db = N'my-db',
    @subscription_type = N'Push',
    @sync_type = N'automatic',
    @article = N'all',
    @update_mode = N'read only',
    @subscriber_type = 0

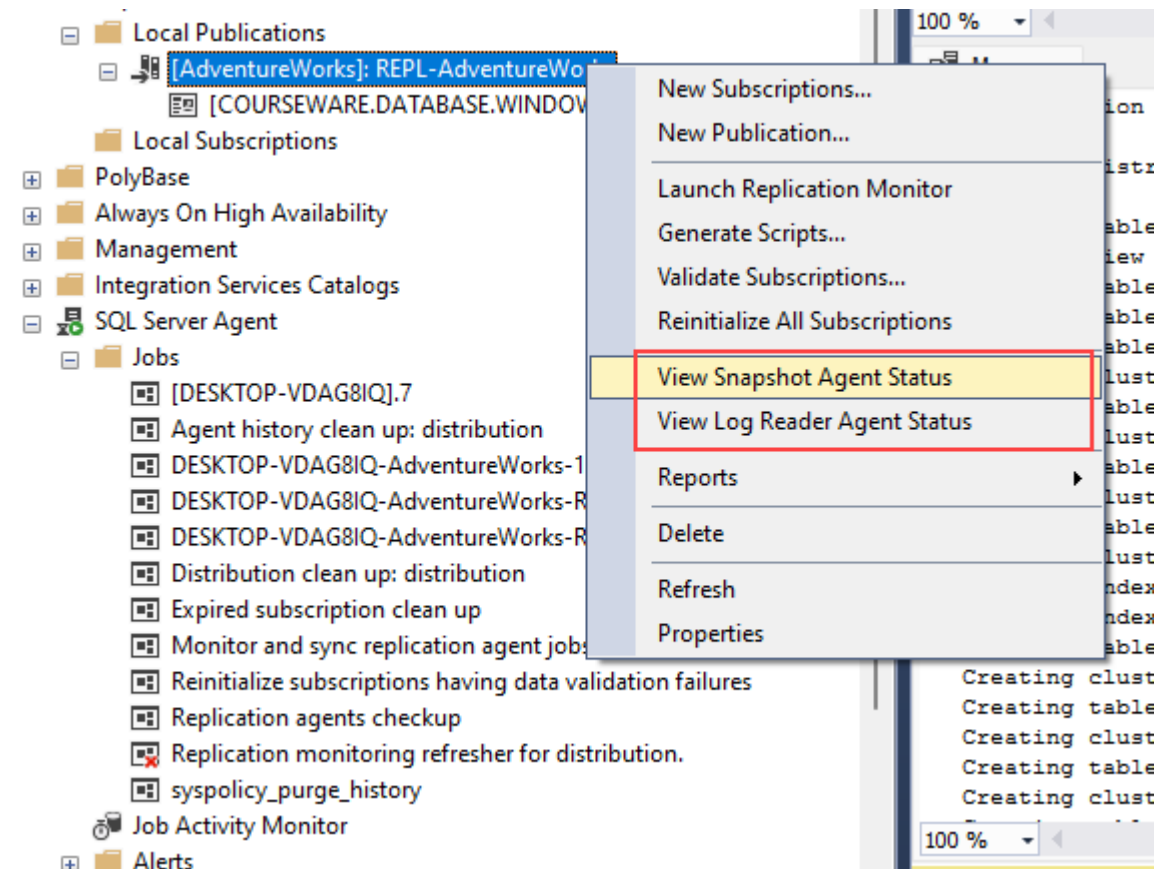
exec sp_addpushsubscription_agent
    @publication = N'REPL-AdventureWorks',
    @subscriber = N'contoso.database.windows.net',
    @subscriber_db = N'my-db',
    @job_login = null,
    @job_password = null,
    @subscriber_security_mode = 0,
    @subscriber_login = N'sqladmin',
    @subscriber_password = '<pwd>',
    @frequency_type = 64,
    @frequency_interval = 1,
    @frequency_relative_interval = 1,
    @frequency_recurrence_factor = 0,
    @frequency_subday = 4,
    @frequency_subday_interval = 5,
    @active_start_time_of_day = 0,
    @active_end_time_of_day = 235959,
    @active_start_date = 0,
    @active_end_date = 0,
    @dts_package_location = N'Distributor'
GO

```

Initiate and monitor replication

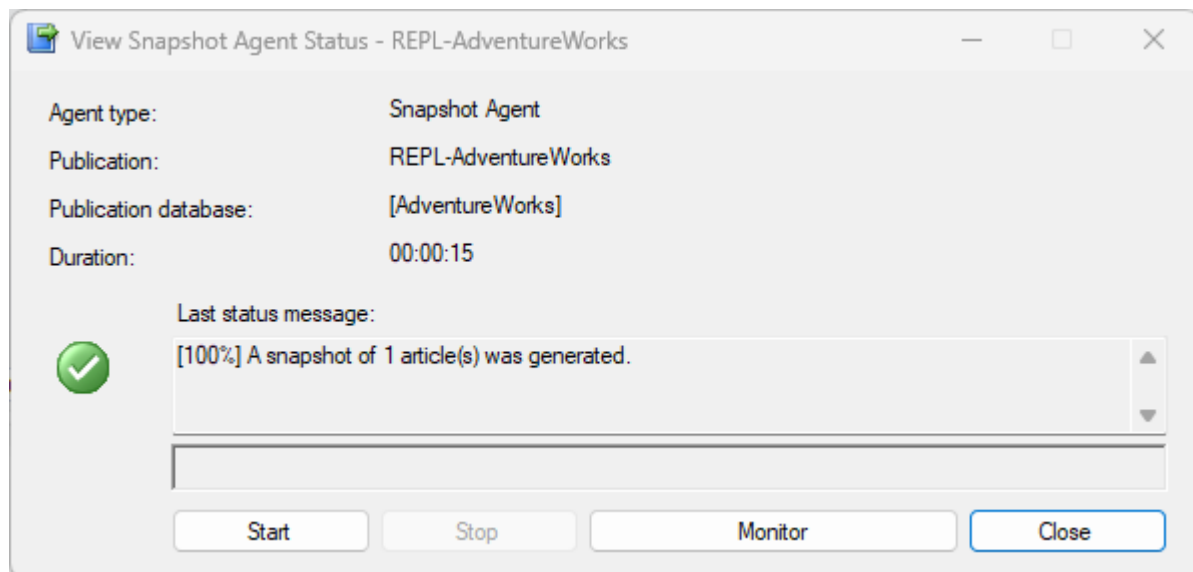
Replication management and monitoring are not supported from Azure SQL Database. Instead, perform these activities from SQL Server. To initiate replication, start the snapshot job, log reader job, and distributor job.

You can monitor the **Snapshot Agent** and **Log Reader Agent** by right-clicking on the publication and selecting the appropriate option. If the agents are not running, start them.

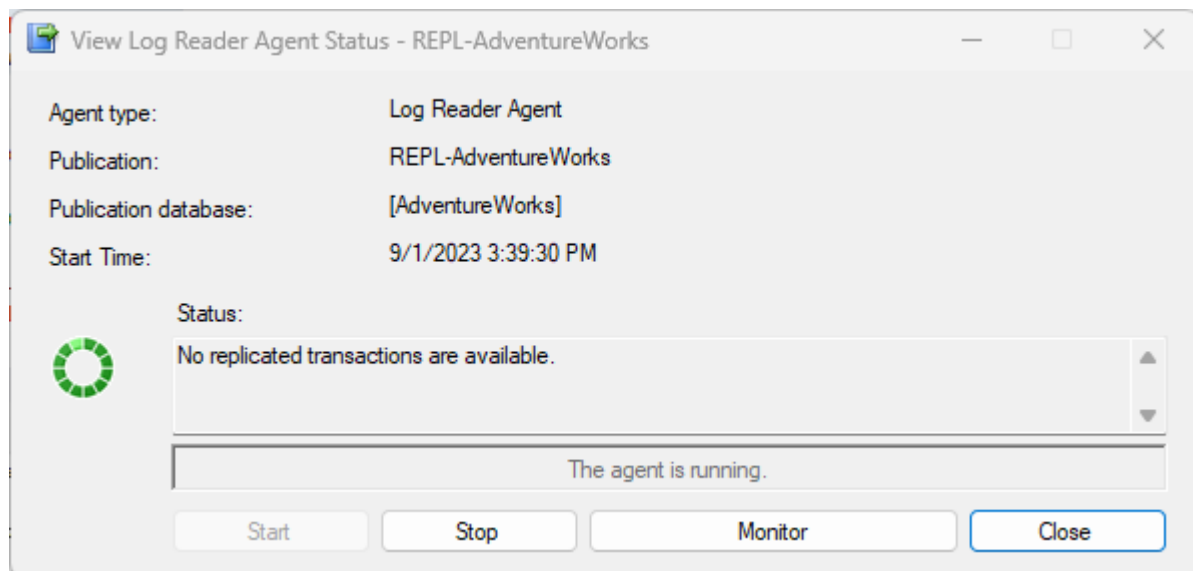


To view the synchronization status, right-click on the subscription, select **View Synchronization Status**, and then start the agent. If you encounter any error messages, check the agent jobs history on the SQL Server Agent. If the agents are running as expected, you should see the following results.

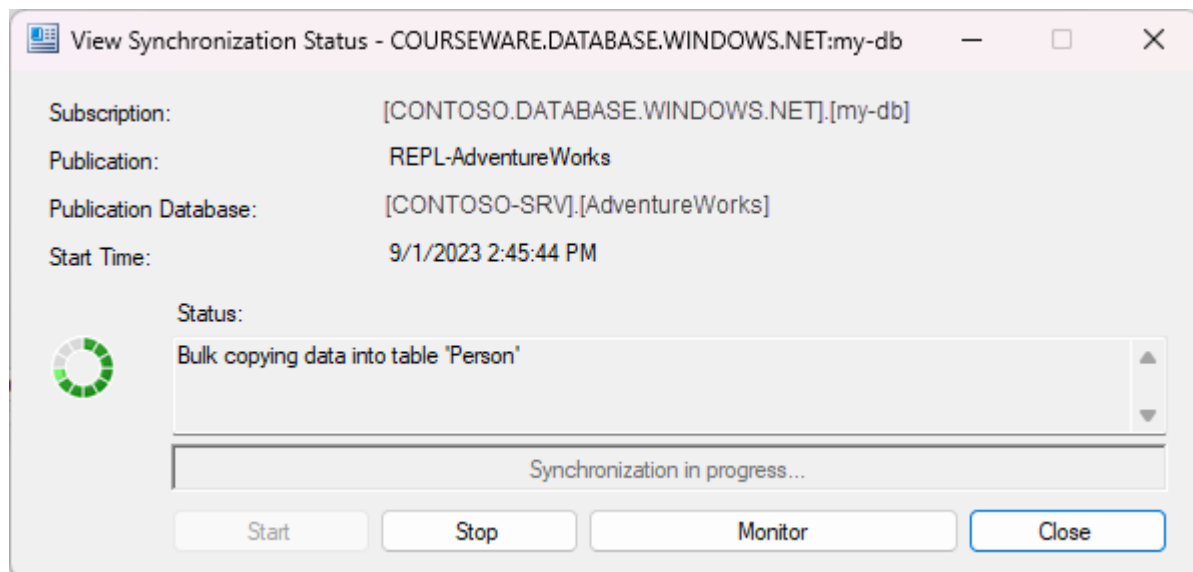
Snapshot Agent:



Log Reader Agent:



Synchronization Status:



After the data is fully replicated to Azure SQL Database, you can direct the connections to the subscriber database, and then stop and remove the replication.

To learn more about supported configurations, see [Replication to Azure SQL Database](#).

6. Move data to Azure SQL Database

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-sql-databases/7-move-data-sql-database>

Move data to Azure SQL Database

Completed

While there are methods available for migrating an entire schema and its data, there are also cases where only a subset of the database is needed. Fortunately, many of the methods we've seen support partial data migration, and we'll learn about a few others.

In our bicycle manufacturer scenario, suppose the company has an on-premises SQL Server database that contains several years' worth of sales, customer, and product data. The company wants to migrate to an Azure SQL Database to take advantage of the scalability and flexibility of the cloud. However, they only need to migrate the customer and product tables, as they want to keep their sales data on-premises for security reasons.

Bulk copy

The [bcp utility](#) allows bulk exporting of data from a SQL Server table into a data file and vice versa. The utility is versatile, enabling data transfer between SQL Server and other programs or databases.

Understanding the schema and data types of the table is essential for using the bcp command effectively, unless a preexisting format file is available.

Azure Data Factory

You can use [Azure Data Factory](#) for data migration rather than entire database migration. Azure Data Factory can migrate and transform data from source SQL Server databases, and it's commonly used for business intelligence workloads (BI).

7. Exercise - Migrate a SQL Server database to Azure SQL Database

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-sql-databases/8-exercise-migrate-sql-database>

Exercise - Migrate a SQL Server database to Azure SQL Database

Completed

Now it's your chance to migrate a SQL Server database using a few of the tools and features available.

In this exercise, you learn how to migrate a SQL Server database to Azure SQL Database.

Note

To complete this exercise, you need an [Azure subscription](#).

Launch the exercise and follow the instructions.

[Launch Exercise](#)

8. Module assessment

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-sql-databases/9-knowledge-check>

Module assessment

Completed

9. Summary

<https://learn.microsoft.com/en-us/training/modules/migrate-sql-workloads-azure-sql-databases/10-summary>

Summary

Completed

In our bike manufacturing scenario, you created and executed a plan to migrate your servers to Azure SQL Database using both offline and online methods.

Now, with the migration complete, your users have the best availability and scalability for their data and you're confident that you can respond to future changes in demand quickly. The method chosen to migrate the database is typically dependent on how much time the SQL Server databases can be offline.

For additional reading, you can refer to the following resources:

- [Migration overview: SQL Server to Azure SQL Database](#)
- [Azure Database Migration Service](#)
- [Transactional Replication with Azure SQL Database](#)