Using data-tier applications (BACPAC) to migrate a database from Managed Instance to SQL Server

drware.com/using-data-tier-applications-bacpac-to-migrate-a-database-from-managed-instance-to-sql-server

Contributed May 14, 2021

This article is contributed. See the original author and article here.

Introduction

BACPAC is a ZIP file with an extension of BACPAC, containing both the **metadata** and **data** from SQL Server database. For more information you can refer to Data-tier Applications – SQL Server | Microsoft Docs. A BACPAC file can be exported in Azure Blob storage or in local file system in an on-premises location and later imported into SQL Server or Azure SQL Managed Instance, or Azure SQL Database. Although BACPAC file contains data and metadata and can be used to deploy a copy of the source database, it is not a backup and shouldn't be considered as such. However, in a situation where it's not possible to use a database backup, BACPAC may be an acceptable substitute.

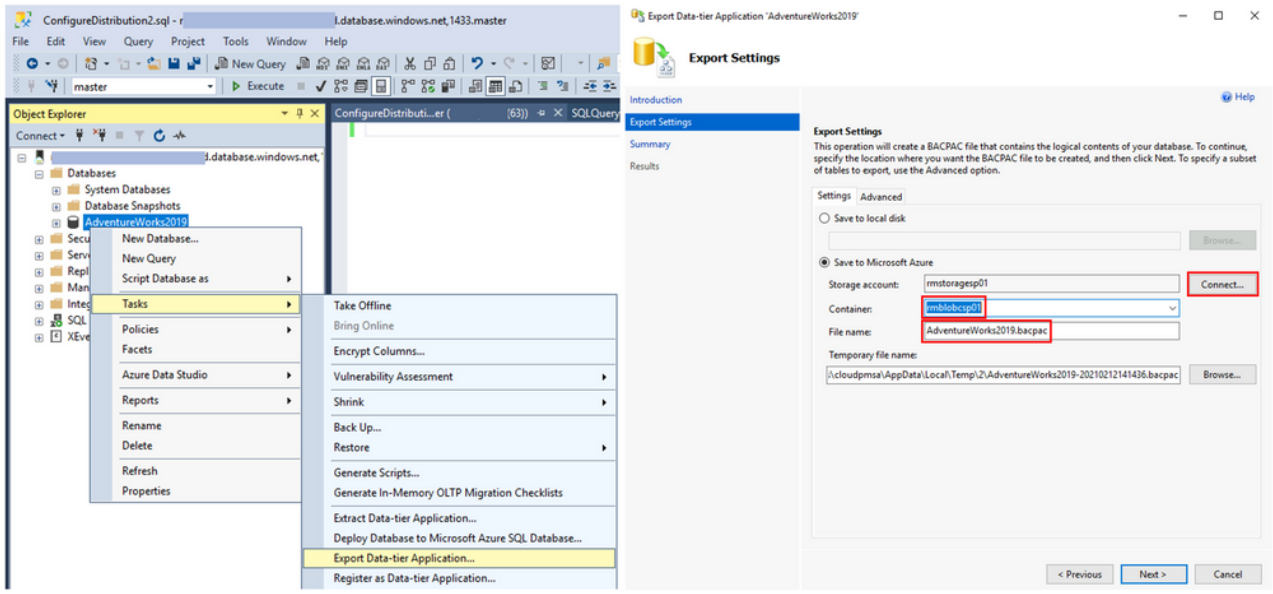
Note: With BACPACs, **data consistency is guaranteed only at the point of execution** against each individual database object, meaning that inflight transactions are missed. In other words, BACPAC does not guarantee transactional replication. You will find more details about this later in Transactional consistency paragraph.

Performing export/import via SSMS and BACPAC

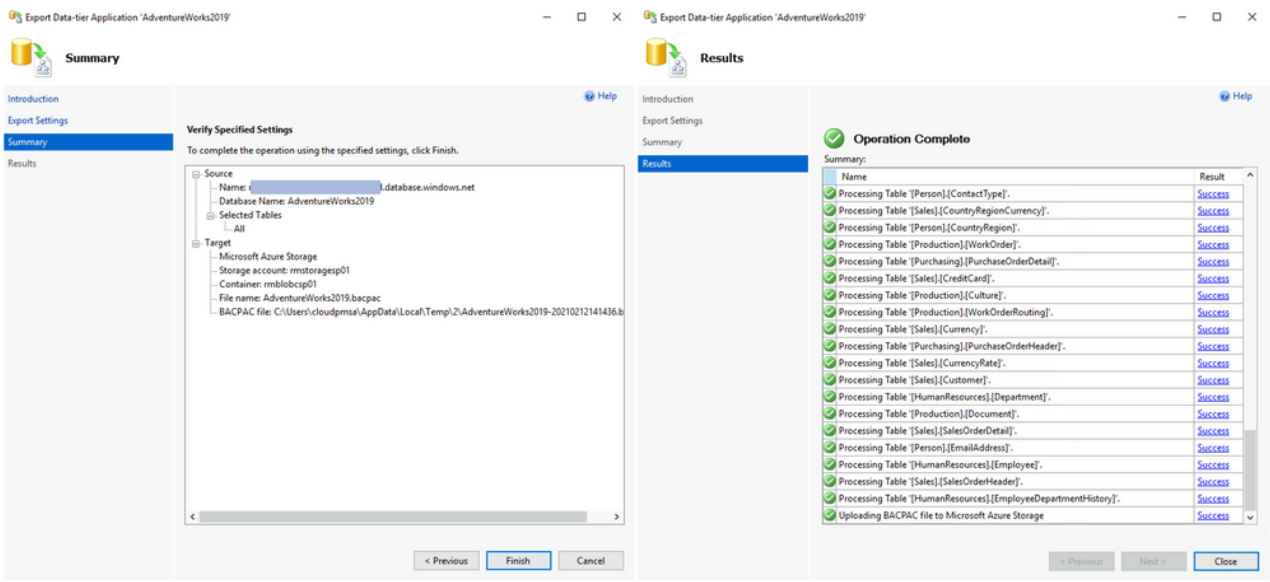
To migrate user database from Managed Instance to SQL Server, first you would need to export the database to a BACPAC file. That can be done from variety of tools: Azure Portal, SqlPackage command line utility, SSMS, Azure Data Studio or PowerShell. The export process is explained in the Azure documentation. The second step is creating a user database on SQL Server by importing created BACPAC. The process is covered in the documentation.

Let's quickly go through the steps in SSMS. To migrate the database, we will to export a BACPAC to Azure Storage (other options are available as well) and then import it from Azure Storage to a SQL Server. Here's is what it looks like. In SSMS, from the context

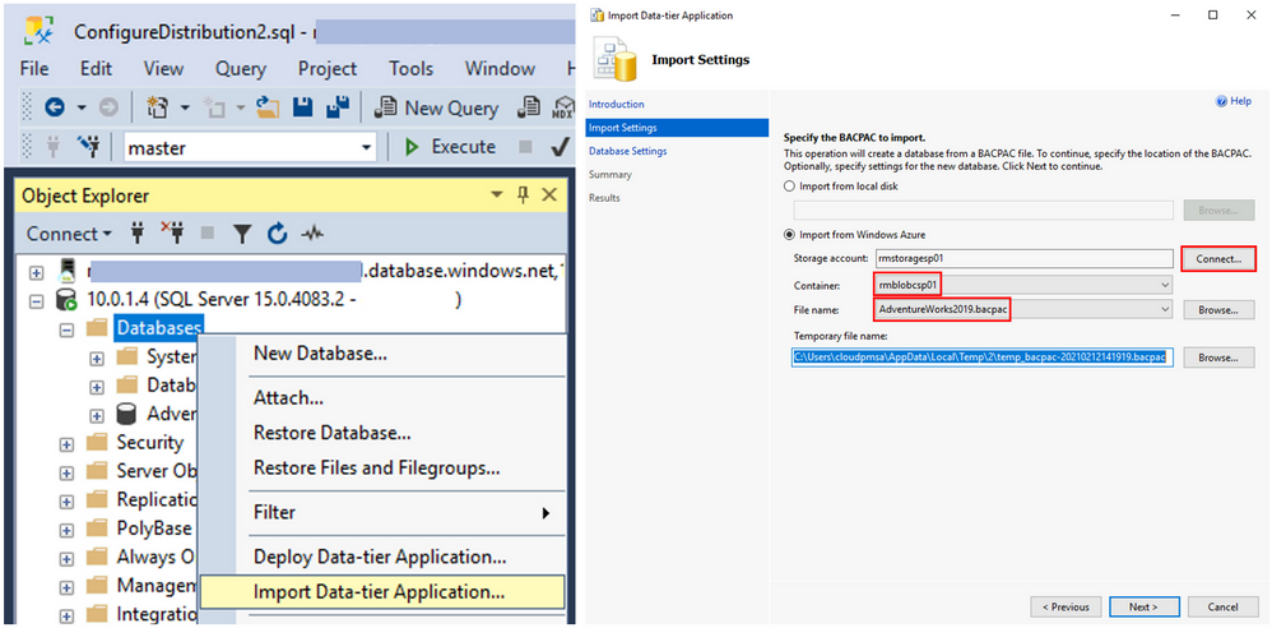
menu of a database you want to migrate chose Tasks, Export Data-tier Application. In Export settings, click Connect, connect with your Azure account, then chose target Container and File name and click Next.



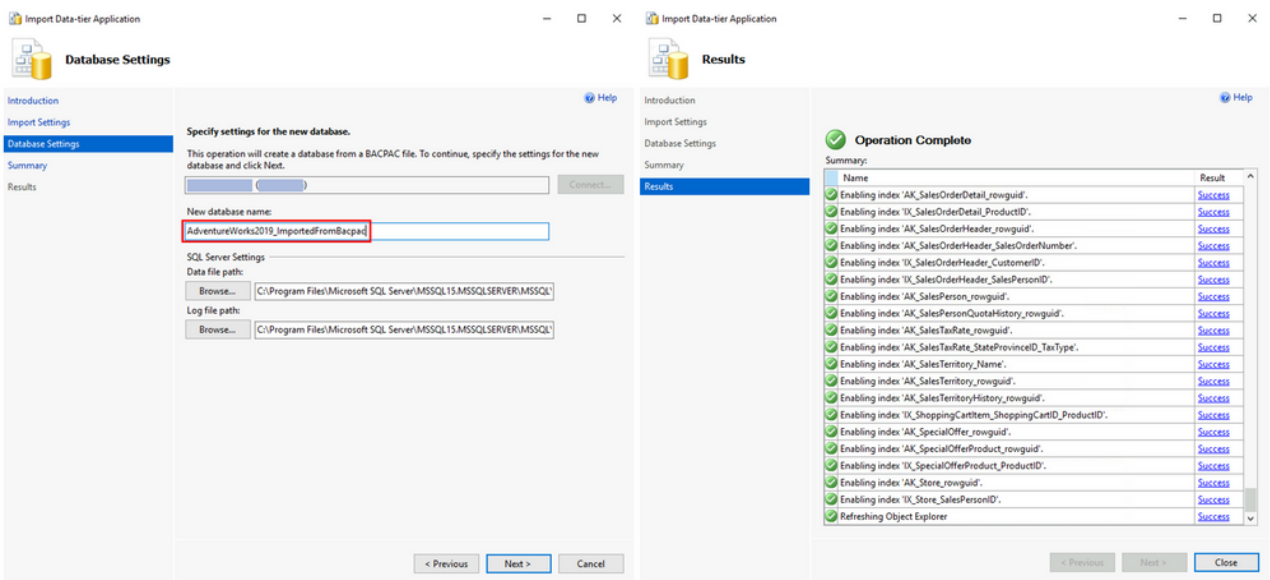
Follow through the Wizard. The final step will show the operation progress. When it reaches Operation Complete, your BACPAC will be saved in the Azure Storage account.



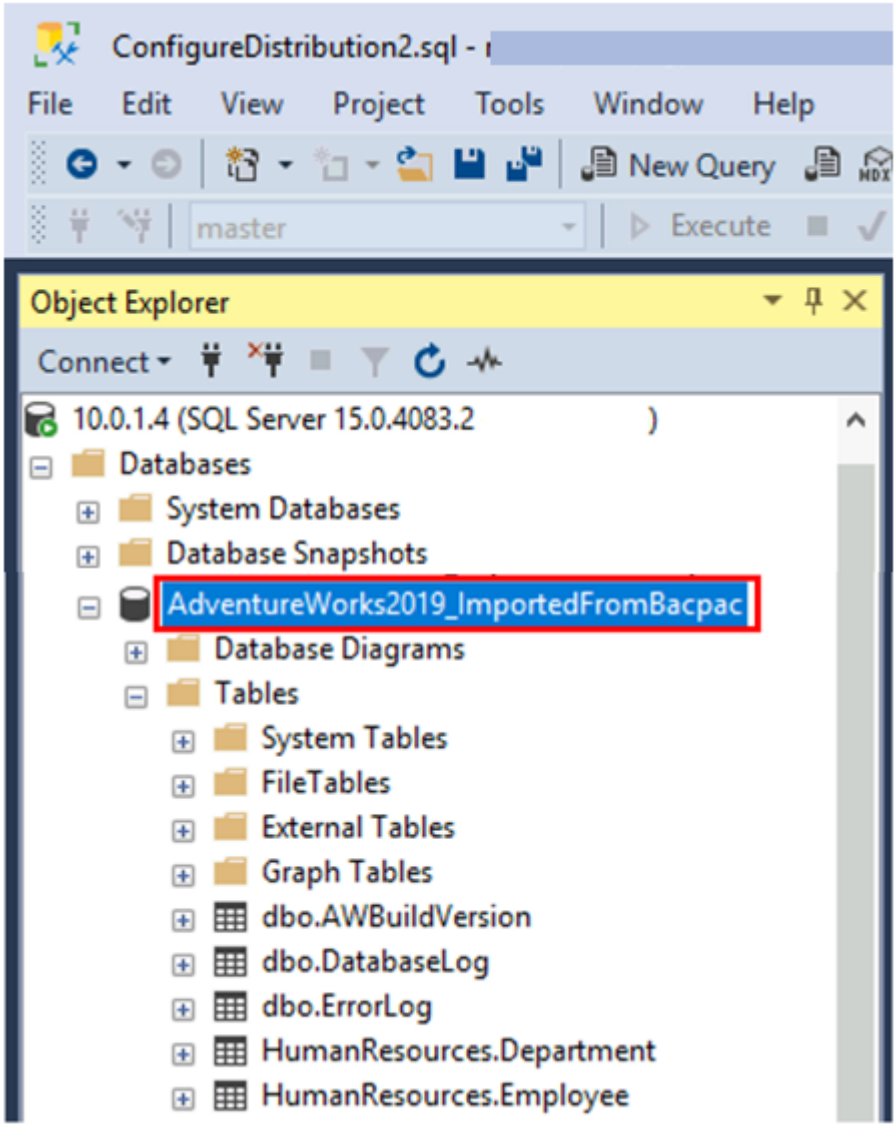
Now, connect to the target SQL Server and from the Databases context menu chose Import Data-tier Application. On the Import Settings window click Connect, connect to your Azure account, then select Container and File name of the BACPAC previously saved and click Next.



On Database Settings window, set the New database name. Follow the Wizard. The final step will show the operation progress and once it reaches Operation Complete the database will be ready.



In SSMS, connect to the target SQL Server and you will find the database with schema and data.



Performing BACPAC export/import via SqlPackage utility

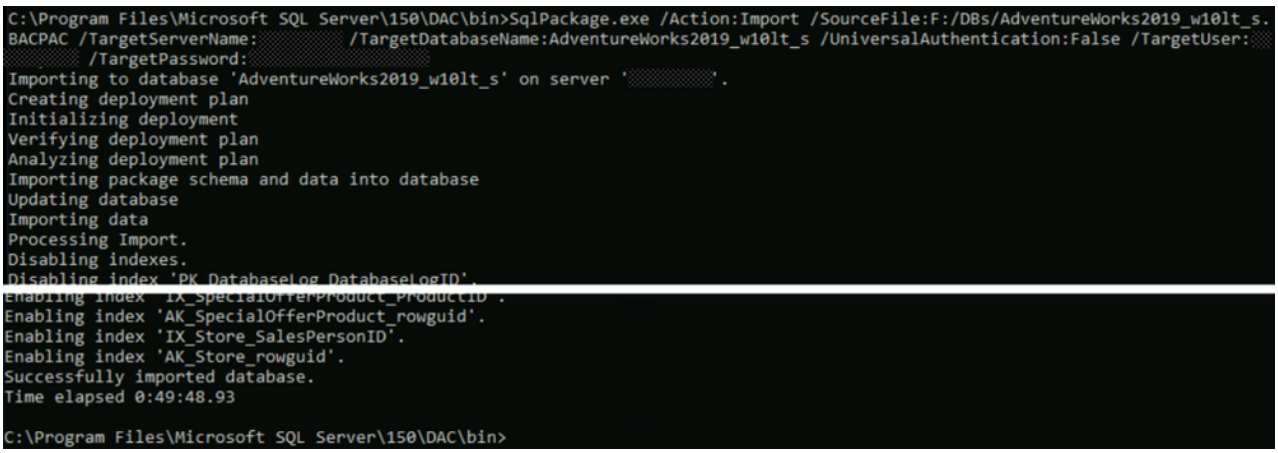
If you need you need to migrate more databases, using a command line tool might be more convenient than SSMS. The utility also offers more parameters than SSMS, so it offers more flexibility and options for fine-tuning. SqlPackage utility can be downloaded from here. Once you install it on a client machine you will be able to run commands to export a BACPAC from Managed Instance, and then to import it into a SQL Server. Here is an example of that process.

Export command:

```
C:Program FilesMicrosoft SQL Server150DACbin>SqlPackage.exe /Action:Export
/TargetFile:F:DBsAdventureWorks2019_w10lt_s.BACPAC
/SourceServerName:"****.****.database.windows.net"
/SourceDatabaseName:AdventureWorks2019_w10lt_s00
/UniversalAuthentication:False /SourceUser:******* /SourcePassword:*******
```

Import command:

```
C:Program FilesMicrosoft SQL Server150DACbin>SqlPackage.exe /Action:Import
/SourceFile:F:/DBs/AdventureWorks2019_w10lt_s.BACPAC
/TargetServerName:******* /TargetDatabaseName:AdventureWorks2019_w10lt_s
/UniversalAuthentication:False /TargetUser:******* /TargetPassword:*******
```



Limitations and workarounds

Export/Import performance

Overall performance of the export import process depends on the resources involved in the process. Source Managed Instance, its tier (GP or BC), number of cores, available IOPS, and resource utilization during the export will impact the performance as well as the network bandwidth between the source Managed Instance and the target storage. And finally, network bandwidth of the BACPAC storage and target SQL Server, and its resources (CPU and storage IO) will influence the import duration.

An example of expected performance for the export from GP Managed Instance, with default settings is:

- StackOverflow 10GB sample database, was exported in ~20min (exported BACPAC had 1.7GB).
- StackOverflow 50GB sample database, was exported in ~65min (exported BACKAC had 9.2GB).

Note: If the export operation exceeds 20 hours, it may be canceled, so optimizing for performance can make a difference between failure and success.

Here are some options for improving performance of export/import:

- If you're exporting from General Purpose Managed Instance (remote storage), you can increase remote storage database files to improve IO performance and speed up the export.
- Temporarily increase your compute size.

- Limit usage of database during export (like in Transactional consistency scenario consider using dedicated copy of the database to perform the export operation)

- Consider using a clustered index with non-null values on all large tables. With clustered index, export can be parallelized, hence much more efficient. Without clustered indexes, export service needs to perform table scan on entire tables in order to export them, and this can lead to time-outs after 6-12 hours for very large tables.
  Hint: A good way to determine if your tables are optimized for export is to run DBCC SHOW_STATISTICS and make sure that the RANGE_HI_KEY is not null and its value has good distribution. For details, see DBCC SHOW_STATISTICS.

Database size

Although export to BACPAC file is one of the easiest options, it's not possible to use it in every scenario. In case of exporting to blob storage, the maximum size of a BACPAC file is 200 GB. To archive a larger BACPAC file, export to local storage.

Transactional consistency

As mentioned in the introduction, BACPAC does not guarantee transactional replication thus if a child table is modified after the parent table is exported, the database won't be consistent. To avoid this, you must ensure either that **no write activity** is occurring during the export. You can achieve this by setting read-only mode to your source database or restoring your database to the same instance (under different name) or to a different instance and using that copy as a source for export.

Cross-database references

DacFx (framework used to create and manage BACPAC files) was designed to block Export/Import when object definitions like views, procedures, etc. contain external references. This includes blocking Export/Import for databases with three-part references

to themselves.

Possible resolutions for this situation are:

- Modify your database schema, removing all self-referencing three-part name references, reducing them to a two-part name.
- There are many 3rd party tools/mechanisms which can be used to fix schema and remove these external references. One option is to use SQL Server Data Tools (SSDT). In SSDT, you can create a database project from your Managed Instance database, setting the target platform of the resulting project to “SQL Azure”. This will enable Azure-specific validation of your schema which will flag all three-part name/external references as errors. Once all external reference errors identified in the error list have been remedied, you can publish your project back to your Managed Instance database and export/import your database.
- Finally, if you have only a few database objects using three-part names, you can script them out, delete them, use BACPAC to migrate the database and in the end, use created scripts to manually recreate objects that are using three-part names.

Conclusion

Data-tier applications (BACPAC) offers a simple way to migrate a database from Managed Instance to SQL Server, with some outstanding limitations. For more information on the migration options, see Moving databases from Azure SQL Managed Instance to SQL Server.

Brought to you by Dr. Ware, Microsoft Office 365 Silver Partner, Charleston SC.