Migrating from Teradata to Azure SQL Data Warehouse

Prepared by:

Mike Boswell – Cloud Solution Architect

Nilabja Ball – Cloud Solution Architect

Reviewed by:

Jim Toland – Senior Program Manager

Mitch van Huuksloot – Solution Architect

Jim Rosengarth – Solution Architect

Copyright

Revision History

Release Date	Changes
July 12, 2018	Released the first version of this guide
July 18, 2018	Updated proof of concept approach
July 20, 2018	Corrected flow of document and explained POC workflow
July 23, 2018	Added Executive Summary
August 23, 2018	Removed Scripts to be hosted externally on GitHub and changes offline data migration workflow
September 15, 2018	Review by Centre of Excellence Team
September 18, 2018	Flow rewritten and Next Steps section
October 18, 2018	V1 General Release

Table of Contents

Executive Summary

This document gives insight into how to approach a Teradata to Azure SQL Data Warehouse migration.

This whitepaper is broken into sections which detail the migration phases, the preparation required for data migration including schema migration, migration of the business logic, the actual data migration approach and testing strategy.

The migration should pivot on six areas (one is optional but highly recommended) and, with the benefit of Azure, you can get ahead of the development curve by quickly being able to provision Azure SQL Data Warehouses for your development team to start business object migration before the data is migrated.

The use of an agile methodology, with focus on what must be delivered, to provide a working environment and any items that are “nice to haves” will achieve a foundation from which to build upon within Azure.

The illustration below shows the different areas and they all build on top of each other: -

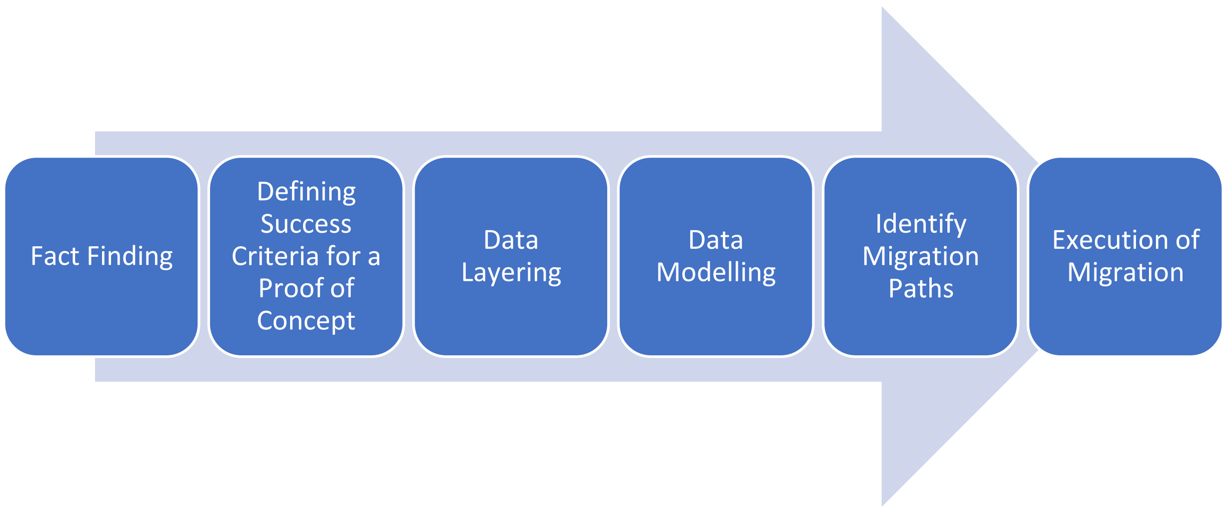


Figure 1: Migration Workflow

Phase One – Fact Finding

Through a question and answers session you can help define what your inputs and outputs are for the migration project. This is the collecting requirements in an Agile methodology.

Phase Two – Defining Success Criteria for Proof of Concept

Taking the answers from phase one, you decide upon running a POC to validate the outputs required and run the following phases as a POC. If you have decided upon a live migration, then you would skip the POC stage (not recommended). This should feed into the Agile Project Charter and establish the team norms.

Phase Three: Data layer Mapping Options

This phase is about mapping the data you have in Teradata to the data layout you will create in Azure SQL Data Warehouse.

Phase Four: Data Modelling

Once you've defined the data mappings, phase four concentrates on how to tune Azure SQL Data Warehouse to provide the best performance for the data you will be landing into it.

Depending on the state of the data warehouse you might need to deploy a team to adjust the schema into a star or snowflake data model. This layout can bring performance benefits to both the data and reporting layers.

Phase Five: Identify Migration Paths

What is the path of least resistance? What is the quickest path given your cloud maturity? Phase Five helps describe the options open to you and then for you to decide on the path you wish to take. With Azure you have two different paths – export to Azure Databox and ship to the Azure Datacentre or use Azure ExpressRoute to transfer from your data centre to Azure's datacentre.

The data export file types can make a difference to the export, transfer and import to Azure SQL Data Warehouse – these have been listed in the detailed section.

Phase Six: Execution of Migration

Migrating your Teradata data to SQL Data Warehouse involves a series of steps. These steps are executed in three logical stages: Preparation, Metadata migration and Data migration which have been defined in the previous phases and are combined into Phase Six: Execution of Migration.

Once migrated, you should have a robust testing plans to validate the solution in a test environment before going live in production.

What is the best approach?

- Run as an Agile Project
  - Establish a Project Charter
  - Track lessons learnt throughout project.
    - Prioritise high items then identify the root cause of the problem.
  - Do not start project until complete team is in place and the infrastructure pre-reqs have been met.
- Infrastructure pre-requisites in place
  - Azure Subscription
    - Agree on location upfront based on service availability
  - Team has access to Azure Subscription to create resources within a resource group
  - Contractors have appropriate credentials to Azure subscription, Teradata and virtual machines. This is to enable them to create database objects and run migration pipelines.
  - Decide upon Azure Databox or Azure ExpressRoute to transfer the data
  - Validate that the virtual machines, in your data centre, which will be used to run the export can communicate with Azure via ExpressRoute or VPN. Without some connectivity to Azure, you won't be able to create the schema in Azure SQL Data Warehouse. If the VPN pipe is small, then Azure Databox is your only route – or upgrade your connection to Azure ExpressRoute.
  - Bring Infrastructure and domain teams in earlier to validate design and remove blockers for connecting Azure Databox, creating virtual machines or connecting to Azure.

- Work with the Teradata DBAs to understand the data compression. For example, a table in Teradata could be 1TB but Teradata has achieved 4x compression. On export, to disk, the text file would be 4TB. This then affects the amount of space you require on a virtual machine within the Teradata datacentre.

Migration Phases

Phase One: Fact Finding

Phase 1 is all about fact finding and understanding if the workload is new or a migration project. This is the collecting requirements phase in Agile.

Before migrating, you want to be certain SQL Data Warehouse is the right solution for your workload. SQL Data Warehouse is a distributed system, designed to perform analytics on large volumes of data. Migrating to SQL Data Warehouse requires some design changes that are not too hard to understand, but might take some time to implement. If your business requires an enterprise-class data warehouse (DW), the benefits are worth the effort. However, if you don't need the power of SQL Data Warehouse, it is more cost-effective to use SQL Server or Azure SQL Database.

Consider using SQL Data Warehouse when you:

- Have one or more Terabytes of data
- Plan to run analytics on large amounts of data
- Need the ability to scale compute and storage
- Want to save costs by pausing compute resources when you don't need them.

Don't use SQL Data Warehouse for operational (OLTP) workloads that have:

- High frequency reads and writes
- Large numbers of singleton selects
- High volumes of single row inserts
- Row by row processing needs
- Incompatible formats (JSON, XML)

Strategic Questions

- Is this a Lift & Shift Migration of On-Premises Data Warehouse or a Green Field Project?
- What Data Size Raw Capacity (Storage Level) is required?
- What Eco-System Processes are in place to be migrated as part of Data Migration?
- What are Data Growth Expectations in 2 – 3 years' time?
- Would it be 2-Tier Data Warehouse or Traditional 3-Tier Data Warehouse with (Acquisition / Ingestion Layer, Integrated Data / Core Layer and Access / View / Semantic Layer)
- What are the expectations around?
  - Concurrency (Number of Users at Peak Times and Tactical Workload Queries for ingress and egress of Data alongside in-place Transformations, reference data management and BI Query Workload)
  - Scalability – Ties to the concurrency above + Total daily ingestion in bytes
  - Availability – Tech. and Business SLA driven
  - Resilience – Disaster Recovery and RTOs and RPOs (Off site cross region Active / Passive replicated / synced Database
  - Protection – Data Backup and Archival – How often, what Strategy and SLAs around
- Are there any thoughts around deploying Semantic Layer via Dimensional or Tabular Data Models (SSAS?)

- How many ETL jobs do you have?
- What BI Layer would be used (PowerBI?)
- Time Scales in Mind (for new deployment or migration)?
- Indicative Pricing / Cost in mind (Migration / Operating)?

After you have the above questions and some of the bottom ones answered we can then look to sizing the Data Warehouse.

Tactical Questions (Applicable only if there is an existing data warehouse to migrate)

Sizing the Data Storage Capacity

- Raw Storage Capacity
  - Data
  - Logs
  - Index Size
- What is the compression ratio on the source data warehouse? During export the data might be uncompressed and then recompressed so you will need to make sure you have enough space for each export phase for the data.
- Processed Data Storage Capacity in Each Tier (If a multi-tiered system is designed and needs to be kept, it is a DW Standard 3 Tier model, Acquisition, IDL and Semantic/Access/View Layer)
  - Number of Tables and their compression ratio by ROM (Data Design Modelled – Hybrid, Star or Snow Flake, query performance can be guessed easily from designed model that how it will project roughly)
  - Number of Indexes / type of indexes
  - Staging Area Reserved for the Data Landing / Acquisition

Sizing the Processing Capacity

- Workload Identification
  - Type of Workloads
    - ETL (with # of feeds and frequencies – nice to capture their complexity if possible.)
    - Analytic (Basic – Moderate Reporting)
    - Advanced Analytics
    - Discovery - Data Mining / Machine Learning
    - % of Ad Hoc Workloads
  - Type of Queries and their respective complexities
  - Within DW ETL Transformation (slice and dicing to prepare optimised nature of data)
  - Down-Stream systems / applications workloads
  - Peak and Off-Peak hours
  - Custom applications (if any)
  - Number of Users

        - Total
        - Average
        - Max concurrent users
- Backups and Archive (Alongside frequencies, retention)
    - Hot Archives within the DW
    - Warm Archives outside DW
    - Cold Archives outside (if any)
    - Disaster Recovery (DR and their SLAs)
    - RTO and RPOs?
- IT / System / Data Governance Compliance
    - Security
    - Data Protection
    - Data Privacy

Phase Two: Defining Success Criteria for a Proof of Concept (Optional)

If you have decided to run a Proof of Concept prior to the live migration, then phase one answers should give enough information to understand what success looks like along with who the stakeholders are for certain parts of the project.

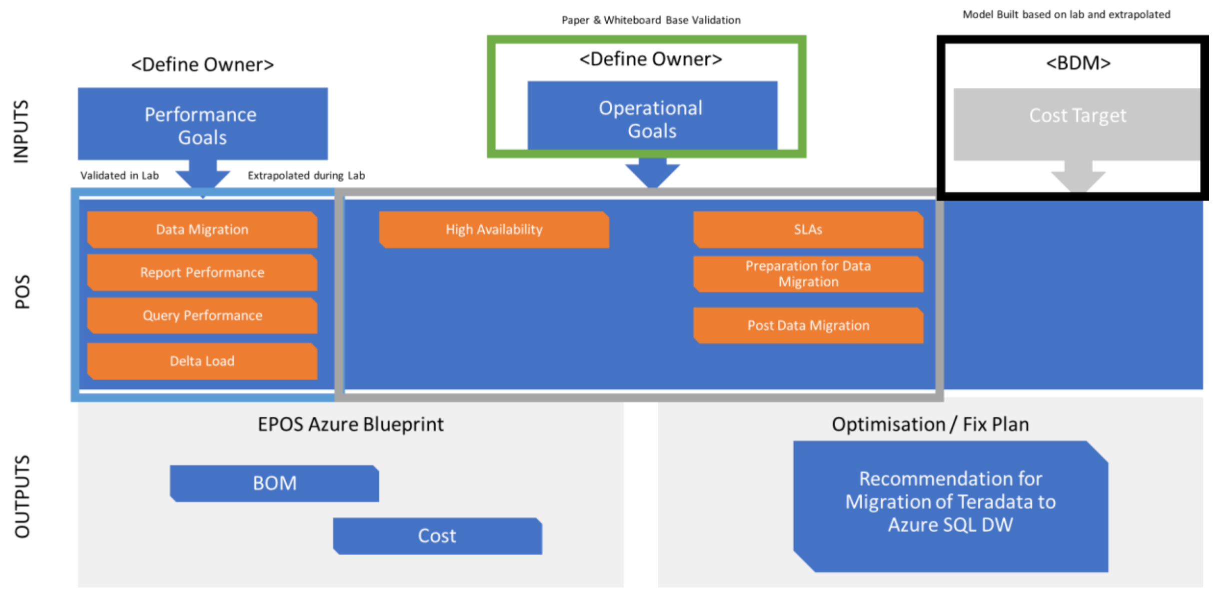


Figure 2: Example Proof of Concept

Taking those answers, you can map out what you want to validate in a POC lab and who the owners are for delivering that part. The owner(s) input should have had input into the success criteria in phase one.

In the POC above we have decided to validate data migration, report performance, query performance and delta loads. These tests will be compared against the current solution with the requirements to be as fast or faster than the current solution at a cost points which supports the business case.

As we are migrating to an Azure solution, some areas don't need to be POC lab tasks but can be extrapolated during the lab. For example, if we take the data migration task in the lab, this would drive an extrapolation of the tasks which would be required for the Preparation for Data Migration. Post Data Migration would come from the task carried out after data migration. The other two mentioned (High Availability and SLAs) can be verified based on the number of clients already running solutions in Azure.

The Business Development Manager is key at the end of the process as they are usually the person who signs off the investment to move to production. They must be involved from the start and should define the cost target at which it makes sense to migrate from Teradata to an Azure solution.

The outputs are driven from the overall proof of concept to provide a Bill of Materials (BOM) and the costs associated with running the solution.

Finally, through the course of the POC you will identify optimisation and fixes which will need to be applied to the migration. This all drives the cost to migration, but it is important to categorise and show the return on investment by implementing these changes. Some changes will be mandatory such as identity and networking infrastructure. These mandatory requirements are usually referred to as Cloud Operating Model.

Consider this as part of your Agile Project Charter and then take the information in this and the previous phase to build out your agile project. Run two weekly sprints with a well-defined backlog. The first sprint would concentrate on getting the infrastructure ready. From sprint two the team would be focussing on schema and data migration.

Phase Three: Data Layering Mapping Options

Phase three is about understanding Data Layering Mapping Options. Teradata has different data types and metadata options. Modify your data types to be compatible with SQL Data Warehouse. For a list of supported and unsupported data types, see data types. This source also provides a query to identify existing types that are not supported in SQL Data Warehouse.

If Teradata has several layers such as Staging, Operational Data Store and then a Semantic model you must decide if SQL Data Warehouse is the right location for all this data. There are multiple options in Azure, and the phase one fact finding will help you focus in on the right technology.

For example, a Teradata data warehouse might include a staging database, a data warehouse database, and some data mart databases. In this topology each database runs as a separate workload with separate security policies. By contrast, SQL Data Warehouse runs the entire data warehouse workload within one database. Cross database joins are not permitted. Therefore, SQL Data Warehouse expects all tables used by the data warehouse to be stored within the one database.

What is the best approach?

This is our approach for consolidating workloads, security, domain and functional boundaries by using user defined schemas:

- Consider consolidating your existing data warehouse environment to use one SQL Data Warehouse database if the workload formerly required cross-database joins.
- Leverage user-defined schemas to provide the boundary previously implemented using databases.
- If user-defined schemas have not been used previously then you have a clean slate. Simply use the old database name as the basis for your user-defined schemas in the SQL Data Warehouse database.

If schemas have already been used, then you have a few options:

- Remove the legacy schema names and start fresh
- Retain the legacy schema names by pre-pending the legacy schema name to the table name
- Retain the legacy schema names by implementing views over the table in an extra schema to re-create the old schema structure. This may seem like the most appealing option. However, the devil is in the detail. Views are read only in SQL Data Warehouse. Any data or table modification would need to be performed against the base table. Option 3 also introduces a layer of views into your system.

You might want to give this some additional thought if you are using views in your architecture already.

Phase Four: Data Modelling

SQL Data Warehouse uses different distribution types to Teradata and it is important to consider how you will distribute the tables you will be migrating.

Each table is distributed or replicated across the Compute nodes. There's a table option that lets you specify how to distribute the data. The choices are round-robin, replicated, or hash distributed. Each has pros and cons. If you don't specify the distribution option, SQL Data Warehouse will use round-robin as the default.

- Round-robin is the default. It is the simplest to use, and loads the data as fast as possible, but joins will require data movement which slows query performance.
- Replicated stores a copy of the table on each Compute node. Replicated tables are performant because they do not require data movement for joins and aggregations. They do require extra storage, and therefore work best for smaller tables.
- Hash distributed distributes the rows across all the nodes via a hash function. Hash distributed tables are the heart of SQL Data Warehouse since they are designed to provide high query performance on large tables. This option requires some planning to select the best column on which to distribute the data. However, if you don't choose the best column the first time, you can easily re-distribute the data on a different column.

To choose the best distribution option for each table, see Distributed tables.

What is the best approach?

For best performance, minimize the row length of your tables. Since shorter row lengths lead to better performance, use the smallest data types that work for your data.

For table row width, PolyBase has a 1 MB limit. When importing terabytes of data into SQL Data Warehouse, PolyBase is the fastest way to ingest data. Therefore, update your tables to have maximum row widths of less than 1 MB.

Depending on the state of the data warehouse you might need to deploy a team to adjust the schema into a star or snowflake data model. This methodology can bring performance benefits to both the data and reporting layers.

For example, you have a Sales fact table, a Retailer and Retailer_Store Dimensions tables. Your current reporting solution links Sales to Retailer on Sales.Retail_ID to Retailer.Retail_ID – this will work fine as it will bring back a retailer for each Sales line. However, you have linked Sales.Retail_ID to Retailer_Store. Retail_ID as opposed creating a unique surrogate key in Retailer_Store for StoreID and copying this value into a field in the Sales fact table.

The Retailer_Store has the layout of:

Retailer_ID	Store_ID
1	1
1	2
2	1
2	3

Table 1: Retail Store Table

As the table contain duplicates for Store you will have to write a query to join Sales on both Retail_ID and Store_ID. The data warehouse will have to scan considerably more data than if you have a join on single unique surrogate key.

A surrogate key on a table is a column with a unique identifier for each row. The key is not generated from the table data. Data modelers like to create surrogate keys on their tables when they design data warehouse models. You can use the IDENTITY property to achieve this goal simply and effectively without affecting load performance. See Using IDENTITY to create surrogate keys in Azure SQL Data Warehouse. The schema of

Retailer_ID	Store_ID	StoreKey
1	1	1
1	2	2
2	1	3
2	3	4

Table 2: Retail Store Table with surrogate key

Phase Five: Identify Migration Paths

The data migration steps usually affect the performance, maintainability and reliability of the migration. Approaches for migrating data to SQL Data Warehouse can be classified based on where the data migration is orchestrated from and based on whether the migration operations are individualized or combined. Typically you have four options:

- Source controlled pipeline - data export, transfer and import steps runs mostly from the source data system
- Azure controlled pipeline - data export, transfer and import steps runs mostly from Azure
- Differentiated Export - data export, transfer and import are distinctly executed with each reading from or writing to intermediate files
- Integrated Export - data export, transfer and import are combined, and entities are transferred directly from the source data system to Azure SQL Data Warehouse with no intermediate files created

What is the best approach?

For Teradata Migration you should use Azure Controlled, with Differentiated or Integrated history loads, depending on your Azure connectivity. These options are discussed under the Data Migration section.

Where possible, automate as much of the export as possible. This enables repeatable exports and can be used for both the proof of concept and the live migration.

Use Polybase to import the data to Azure SQL Data Warehouse. PolyBase is a technology that accesses data outside of the database via the T-SQL language. It is the best way to load data into SQL Data Warehouse. With PolyBase, the data loads in parallel from the data source directly to the compute nodes.

PolyBase loads data from UTF-8 and UTF-16 encoded delimited text files. In addition to the delimited text files, it loads from the Hadoop file formats RC File, ORC, and Parquet. PolyBase can load data from Gzip and Snappy compressed files. PolyBase currently does not support extended ASCII, fixed-width format, and nested formats such as WinZip, JSON, and XML.

Bringing it all together in the Execution of Migration

From Migrating data to Azure SQL Data Warehouse in practice.

After collecting information, combining and making the key decisions in phases one to five you must move onto the Execution of the Migration. Migrating your Teradata data to SQL Data Warehouse are executed in three logical stages: Preparation, Metadata migration and Data migration.

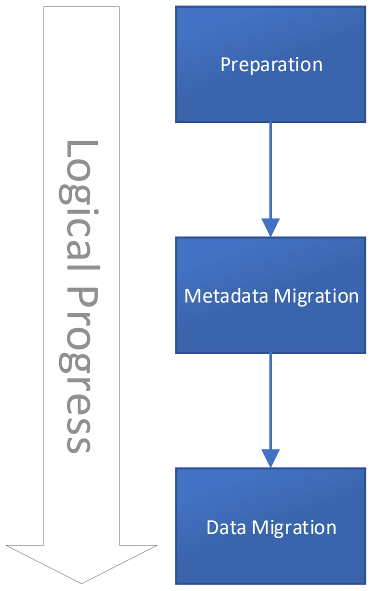


Figure 3: Logical Stages of Migration

In each stage, tasks to be executed involve the on-premises database system, the on-premises local storage, the network connecting the local system to Azure (either internet or a dedicated circuit) and SQL Data Warehouse.

This results in a physical data movement from the source database to Azure as shown below. (These steps are also similar in moving data from any other source system on cloud instead of on-premises to SQL Data Warehouse)

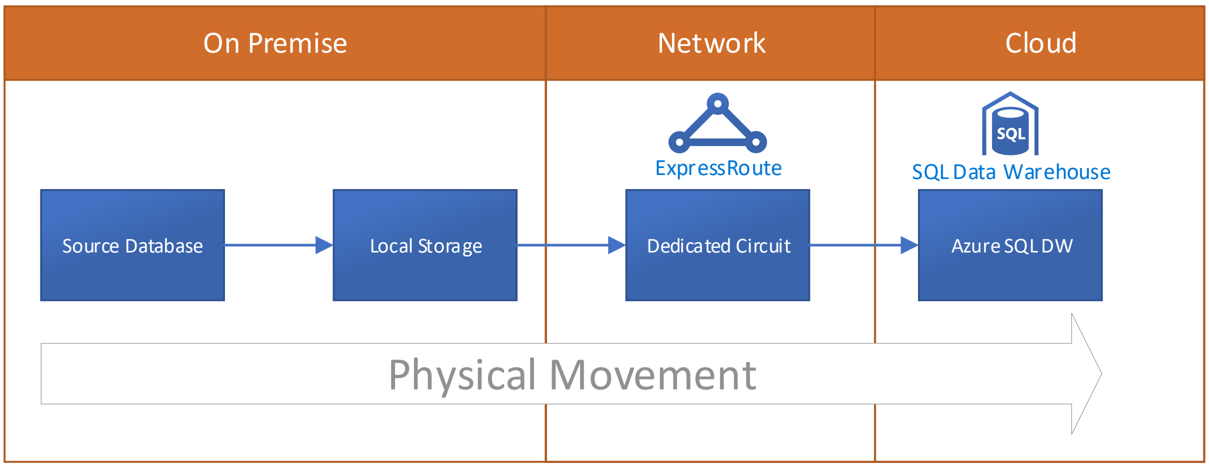


Figure 4: Physical Flow of Migration

Bringing these two diagrams together results in a logical flow from top to bottom and a physical flow from left to right.

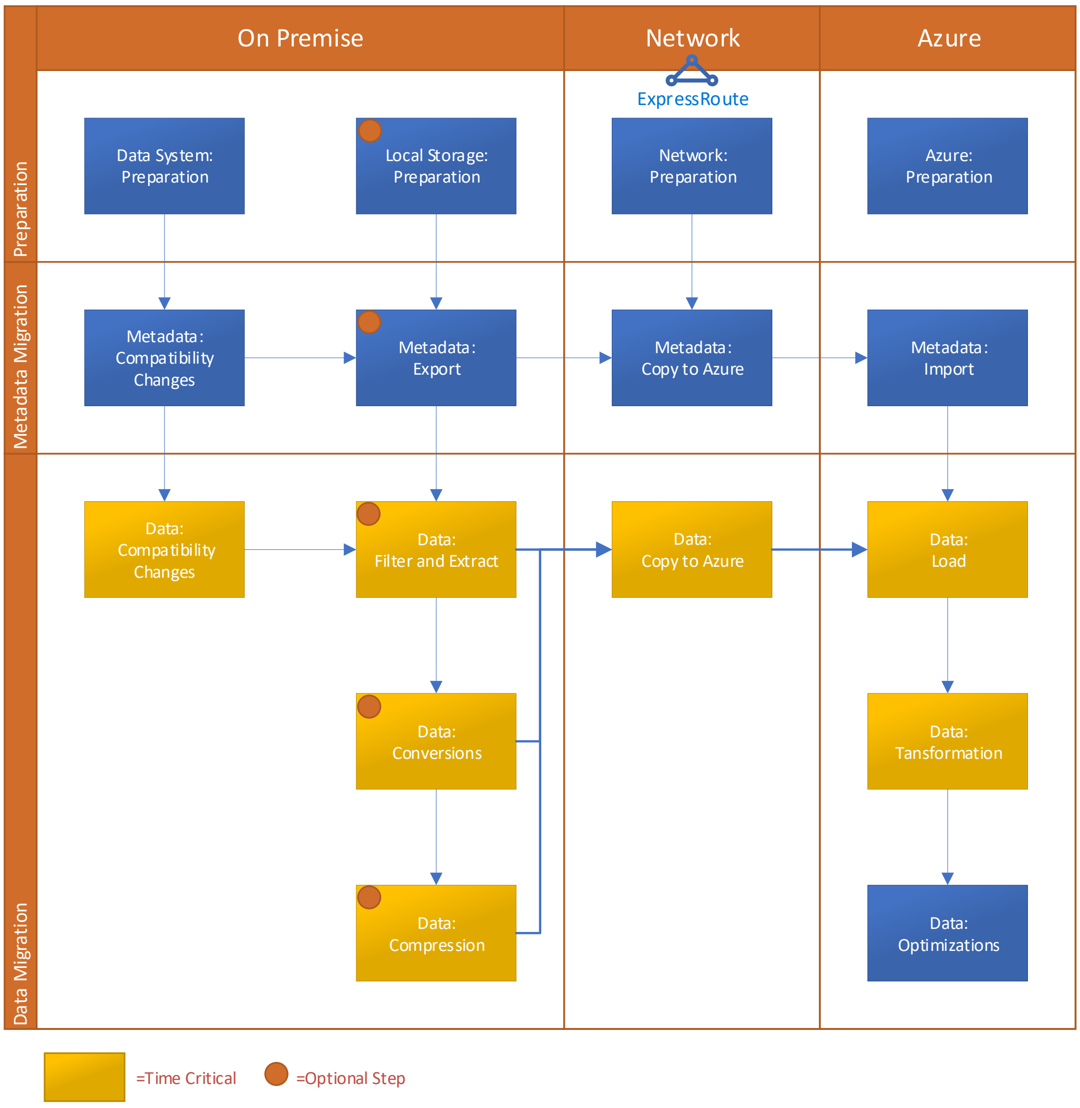


Figure 5: Logical and Physical Workflow

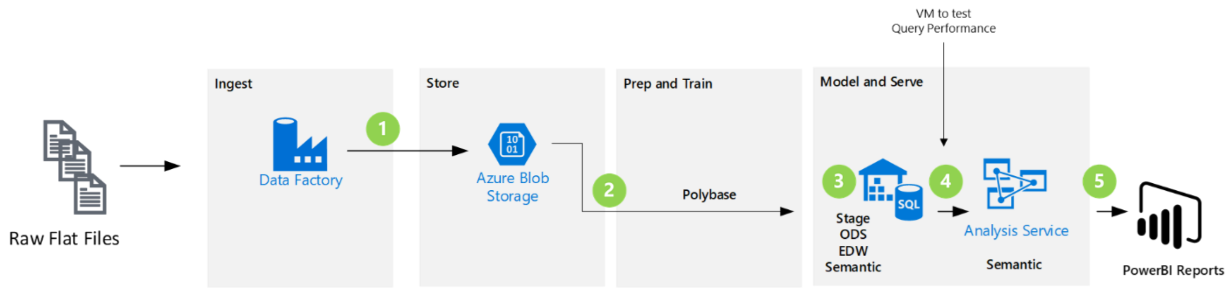
If the volume of the data to migrate is large, some steps can be time consuming. These steps are rate-determining because they influence the overall migration time. Such steps are shaded in yellow.

Some migration steps may be optional depending on the size of the data, the nature of the network, and the tools and services used for migration. Optional steps are shown with orange dots.

Architectures

Data Migration

To ingest historical data, you need a basic cloud Cloud Data Warehouse setup for moving data from your on-premise solution to Azure SQL Data Warehouse and to enable the development team to build Azure Analysis Cubes once the majority of the data is loaded.



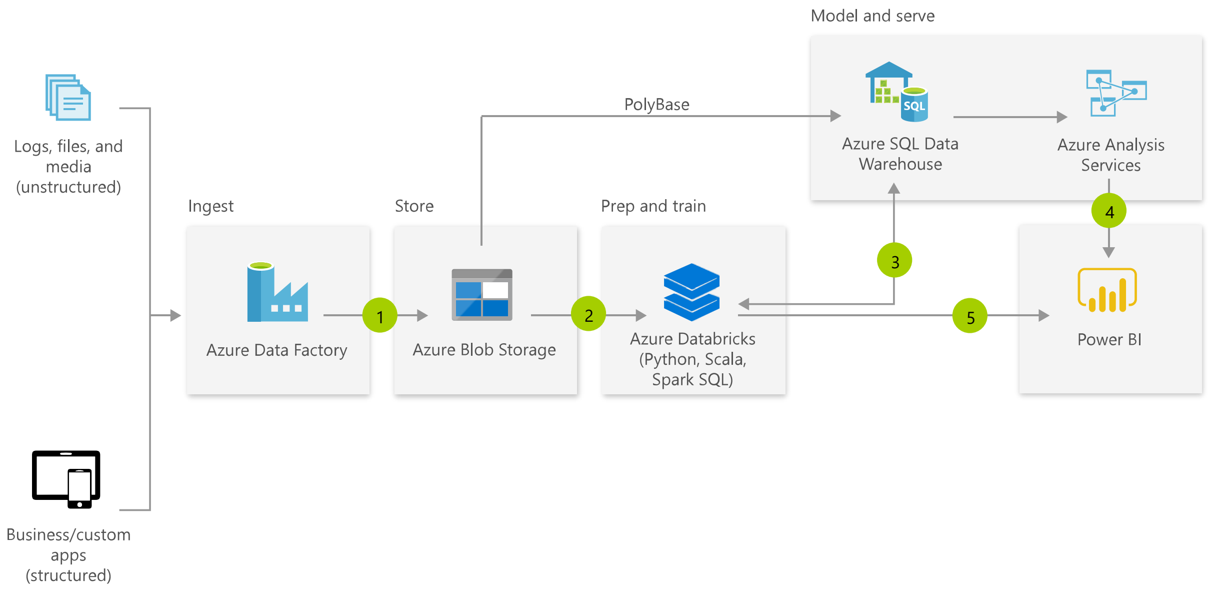
1. Azure Data Factory Pipeline is used to ingest and move data through the store, prep and train pipeline.
2. Extact and Load files, via Polybase, into the Staging Schema on Azure SQL DW.
3. Transform data through Staging, Source (ODS), EDW and Sematic Schemas on Azure SQL DW.
4. Azure Analysis Services will be used as the sematic layer to serve thousands of end users and scale out Azure SQL DW concurrency.
5. Build operational reports and analytical dashboards on top of Azure Analysis Services to serve thousands of end users via Power BI.

Note: You should provision a testing virtual machine in Azure, with accelerated networking enabled. This will be used to execute ad-hoc test queries. You can then look at the internals of Azure SQL DW to see the latency/execution times.

Modern Data Warehouse

After ingesting historical data, and depending on your data loading sizes, adding Azure Databrick alongside Azure Data Lake Storage Generation 2 lets you bring together all your data at any scale easily, and to get insights through analytical dashboards, operational reports, or advanced analytics for all your users.

Giving you the ability to easily add more data feeds which are then moved through the pipeline up to the Azure SQL Data Warehouse layer and to your analytics/reporting layer.



1. Combine all your structured, unstructured and semi-structured data (logs, files, and media) using Azure Data Factory to Azure Blob Storage.
2. Cleansed and transformed data can be moved to Azure SQL Data Warehouse to combine with existing structured data, creating one hub for all your data. Leverage native connectors between Azure Databricks and Azure SQL Data Warehouse to access and move data at scale.
3. Leverage data in Azure Blob Storage to perform scalable analytics with Azure Databricks and achieve cleansed and transformed data.
4. Build operational reports and analytical dashboards on top of Azure Data Warehouse to derive insights from the data and use Azure Analysis Services to serve thousands of end users. Run ad hoc queries directly on data within Azure Databricks.

What is the best approach?

To accelerate your development sprints, you should deploy the data migration architecture first to allow the team to start migrating schemas and historical information.

Once the team starts to work on incremental loads, the Modern Data Warehouse architecture should have been built, with focus on the Azure Data Bricks to cleanse and transform data into Azure SQL Data Warehouse.

See further details on Data Migration pipeline to understand how this architecture supports your migration.

Preparation for Data Migration

Schema Migration

Introduction

You can provision a Teradata Database (developer edition), in Azure, before migrating the data from your on-premises Teradata database. This can speed up migration efforts, as it often takes time to provision on-premise virtual machines, and go through internal approval processes, to obtain a virtual machine inside your datacenter.

By provisioning an environment in under an hour in Azure, it allows both activities to run in parallel. The key to getting the schema from Teradata to SQL Data Warehouse is to build a Teradata Database VM which we can recreate the on-premise schema on.

The process of extracting the Teradata tables to Azure SQL Data Warehouse is:

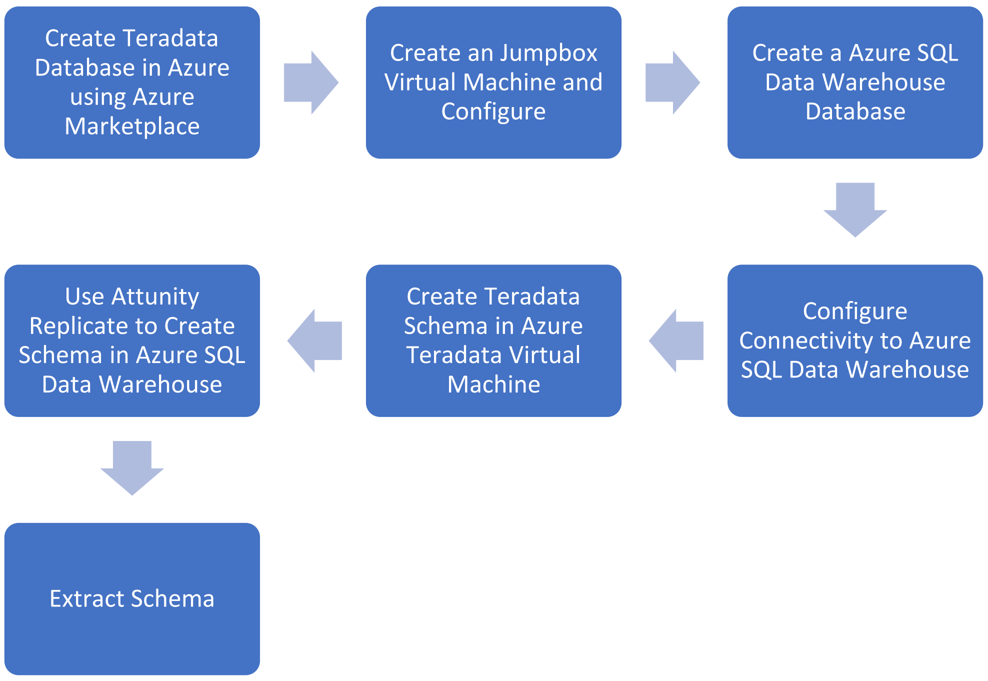

Create Teradata Database in Azure using Azure Marketplace
Create an Jumpbox Virtual Machine and Configure
Create a Azure SQL Data Warehouse Database
Configure Connectivity to Azure SQL Data Warehouse
Create Teradata Schema in Azure Teradata Virtual Machine
Use Attunity Replicate to Create Schema in Azure SQL Data Warehouse
Extract Schema

Figure 6: Workflow for Schema Migration

If you are able to provision a virtual machine quickly in your datacenter, close to the Teradata database, then this is the preferred process. This would allow you to start at "Create Jumpbox". However, in our experience, internal processes can slow this down and with a schema the project will be on hold for ages.

Create a Teradata Database Azure VM

A development version of Teradata Database which, as at 4th July 2018, is provided free (All you pay for is the Azure compute and storage). Teradata is licensed for use on specific Azure VMs. For latest information on supported software and VMs, visit www.teradata.com/azure/products/

Using Azure Marketplace, it is possible to deploy a 2-node development edition of Teradata Database 15.x and to then use create a copy of the schemas you wish to migrate. Only Teradata Database is required, no other Teradata components are required.

Pre-requisites:

You should have extracted a copy of the Teradata Schema scripts for the objects you wish to migrate. You will need one vNet with three subnets provisions and an Azure Subscription with contributor permission on at least one empty Azure resource group.

After the Teradata cluster has been created, you will need to open port 1025 to allow the Jumpbox VM to communicate with it.

Full details on creating a Teradata Azure VM can be found in the Appendix under "Create Teradata Database Azure VM".

Create Jumpbox VM

To facilitate copying the schema between the Azure Teradata Cluster and Azure SQL Data Warehouse you should create a Windows 2012 Azure VM (Size D4_v2). Using the Data Science Virtual Machine - Windows Server 2012 edition, from the Azure Marketplace, will save you time in installing SQL Server Management Studio and Visual Studio – since these components are pre-installed on the Azure Marketplace image.

Additional software which should be installed after this has been provisioned (also applies to on-premise virtual machine): -

- Visual Studio Code
- Teradata Utilities v15.x with
  - ODBC Driver
  - OLEDB Access Module
  - .Net Data Provider
  - TPT Base
  - TPT Stream
- Microsoft Connectors for Oracle and Teradata by Attunity for Integration Services (SSIS)
- Attunity Replicate for Microsoft Migrations

Full details on creating a Teradata Azure VM can be found in the Appendix under "Create Jumpbox VM"

Create an Azure SQL Data Warehouse Database

You will need a blank SQL Data Warehouse database to migrate the schema to and this needs to be setup prior to migrating the schema. Initially the performance level can be set Gen2 DWU500, as you will only be copying metadata at this stage – you can scale this up later when you need to load data.

Prior to using Attunity Replicate, to create the schema, you will need to create a master key for Azure SQL Data Warehouse.

To run loads with appropriate compute resources, we recommend that you create a load database user resource class to fit the resource class and load need (see Loading Best Practices).

Full details on Creating an Azure SQL Data Warehouse Database can be found in the Appendix under "Create a SQL DataWarehouse Dev DB"

Creating the Teradata Schema in Azure

To create the schema, you will need to export your on-premise Teradata Schema into .sql text files and then use the Teradata Developer Tools for Visual Studio, to recreate the schema on Azure Teradata Cluster.

By approaching the schema migration using a separate instance away from Production, you can remove several barriers around connecting to a production instance.

Full details on how you can create the Teradata schema can be found in the Appendix under "Create the Teradata Schema in Azure".

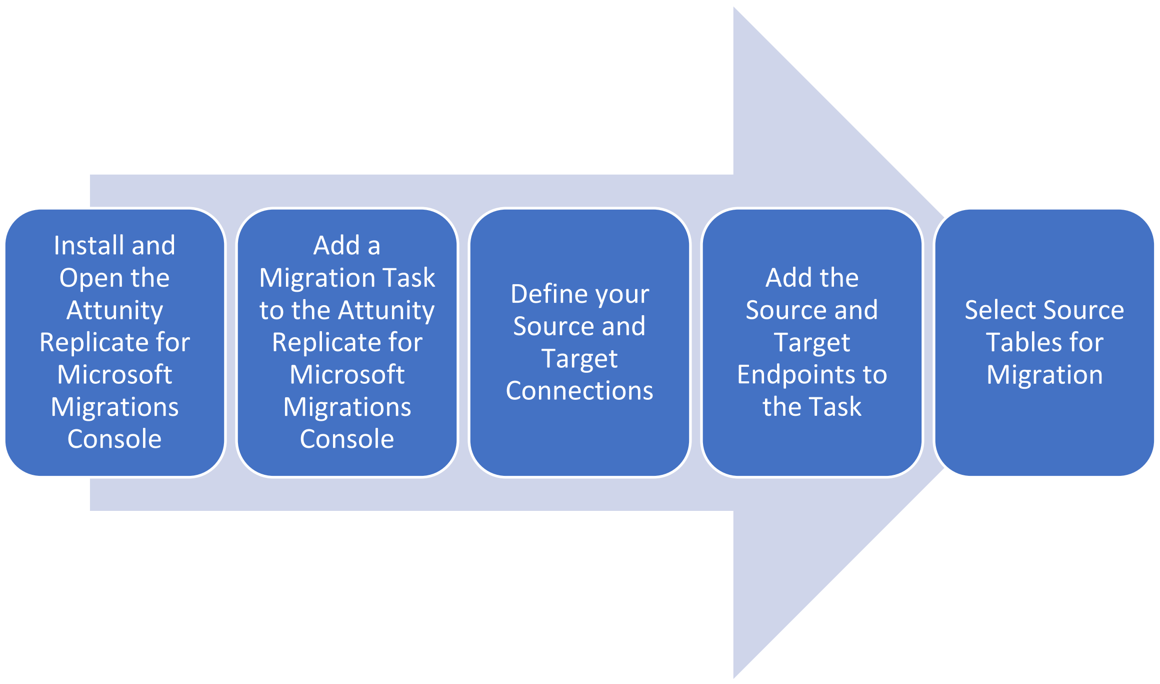
Use Attunity Replicate to Create Schema

Attunity Replicate for Microsoft Migrations is for Microsoft customers who want to migrate data from popular commercial and open-source databases to the Microsoft Data Platform, including Teradata to Azure SQL Data Warehouse. It can be obtained from https://www.attunity.com/products/replicate/attunity-replicate-for-microsoft-migration/.

Attunity Replicate for Microsoft Migrations is installed on the Jumpbox VM and is used transfer the Teradata schema to Azure SQL Data Warehouse.

The table schema will need to be performance tuned as each table is created as a ROUND ROBIN table.

This schema migration consists of the following stages:


Install and Open the Attunity Replicate for Microsoft Migrations Console
Add a Migration Task to the Attunity Replicate for Microsoft Migrations Console
Define your Source and Target Connections
Add the Source and Target Endpoints to the Task
Select Source Tables for Migration

Once migrated, you should check that the schema has been transferred. The full process is described in the Appendix under "Using Attunity Replicate to Create Schema".

Extract Schema

At this point in the schema creation process, it is advisable to extract the Azure SQL Data Warehouse schema and check it into your code store.

The easiest way to achieve this is with the mssql-scripter. mssql-scripter is the multiplatform command line equivalent of the widely used Generate Scripts Wizard experience in SSMS.

You can use mssql-scripter on Linux, macOS, and Windows to generate data definition language (DDL) and data manipulation language (DML) T-SQL scripts for database objects in SQL Server running anywhere, Azure SQL Database, and Azure SQL Data Warehouse. You can save the generated T-SQL script to a .sql file or pipe it to standard *nix utilities (for example, sed, awk, grep) for further transformations. You can edit the generated script or check it into source control and subsequently execute the script in your existing SQL database deployment processes and DevOps pipelines with standard multiplatform SQL command line tools such as sqlcmd.

Follow the installation guide on https://github.com/Microsoft/mssql-scripter/blob/dev/doc/installation_guide.md and launch a command prompt with the following script

```
mssql-scripter -S <servername>.database.windows.net -d <DatabaseName> -U <UserID> -
P <Password> -f ./CreateSQLDWTables.dsql --script-create --target-server-version
AzureDW --display-progress
```

This will extract the Azure SQL Data Warehouse Schema to allow you to recreate the database later. For a full command line reference please see https://github.com/Microsoft/mssql-scripter/blob/dev/doc/usage_guide.md.

Migrate Business Logic

This section runs through how you can take you Teradata business logic and migrate it through to SQL Data Warehouse.

Some Azure SQL Data Warehouse features can significantly improve performance as they are designed to work in a distributed fashion. However, to maintain performance and scale, some features are also not available. See Migrate your SQL code to SQL Data Warehouse.

Metadata Approaches

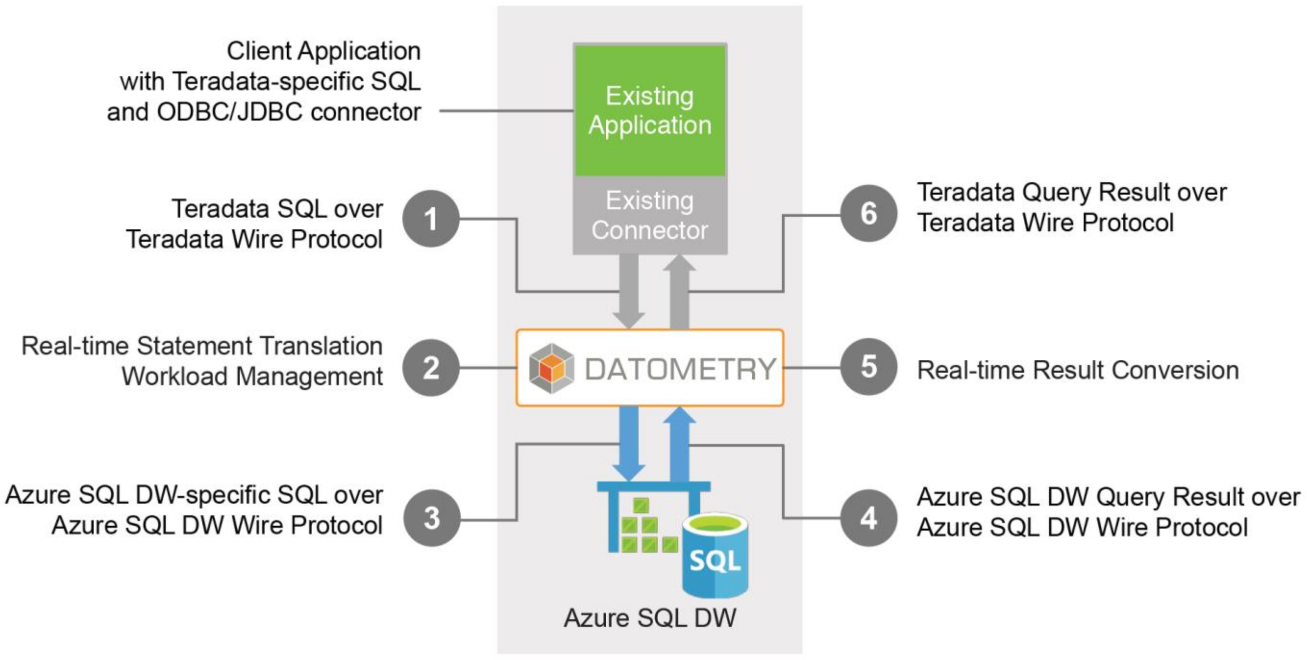
There are two approaches which you can take to migrating the stored procedures and views which reside in the Teradata Database.

Phase One: Lift and Shift

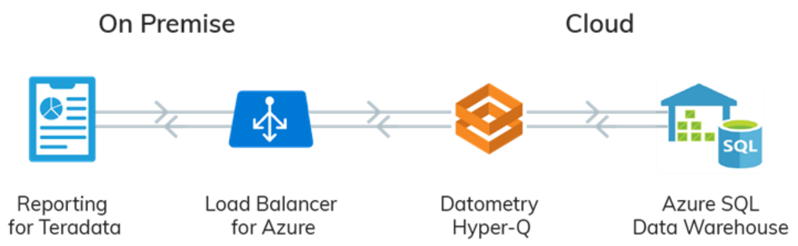
Datometry Hyper-Q (https://datometry.com/)

Datometry, a Microsoft Partner, should be considered in phase one (lift and shift) of any Teradata Migration where you have adhoc Teradata (TD) SQL, legacy applications and reporting which needs to be moved across quickly. It runs between the legacy applications and Azure SQL Data Warehouse to translate Teradata queries into Azure SQL Data Warehouse T-SQL at the network layer. They claim they can eliminate the rewriting of database applications and reduce migration cost and project time by up to 90% by letting existing applications, written originally for Teradata, to run natively and instantly on SQL Data Warehouse.

The application works, on a VM, within Azure, in the following manner:


Client Application
with Teradata-specific SQL
and ODBC/JDBC connector
Existing Application
Existing Connector
1 Teradata SQL over Teradata Wire Protocol
2 Real-time Statement Translation Workload Management
DATOMETRY
3 Azure SQL DW-specific SQL over Azure SQL DW Wire Protocol
4 Azure SQL DW Query Result over Azure SQL DW Wire Protocol
5 Real-time Result Conversion
6 Teradata Query Result over Teradata Wire Protocol
SQL
Azure SQL DW

Existing on-premise applications can continue to be used:



Phase Two: Rewrite Business Logic

If you have decided to go for a complete re-write of the application there are various toolkits and Microsoft Partners who will help you take complex business logic through to SQL Server from which you can add SQL Server Data Warehouse specifics.

What is the best approach?

If you have legacy applications and want to move quickly our recommended approach would be to use Datometry and then concentrating on recoding the application using a Microsoft Partner.

However, if you are migrating to different application stacks and have the timelines to be able to move business logic migration and data loads, then investment upfront in this will leave you with less technical debt.

Example of Business Logic Migration

If you are moving from Teradata to Azure SQL Data Warehouse, without Datometry, then you will have to do a certain amount of recoding. Take this view definition, from Teradata, as an example of what would need to be changed to be compatible with Azure SQL Data Warehouse.

```
REPLACE VIEW ADW.DIMDATE AS
LOCKING ROW FOR ACCESS
SELECT
       CAL.DayNumberofWeek AS WeekName,
       CAST(CAL.DateKey AS DATE FORMAT 'MM/DD/YYYY') AS CalDate,
       CASE WHEN CAL.DayNumberOfMonth = 5 THEN 1 ELSE 0 END AS Calc_CY,
FROM TESTTD.CALENDER AS CAL
WHERE CAL.DateKey BETWEEN (Current_Date- INTERVAL '3' YEAR) AND (Current_Date- INTERVAL '1' DAY);
```

Looking at this code there are multiple challenges with trying to execute this against SQL Data Warehouse. To convert these into T-SQL we would apply the following changes:

- CAST to a CONVERT with a date format number
- Change the way a column is alias in a case statement from “as columnname” to “columnname=case xyz”.
- Fully qualify the columns used in the case statement as opposed relying on column alias names in the statement.
- The keyword INTERVAL isn’t supported in T-SQL so to achieve the same functionality we use the dateadd functionality.

These are small changes and shows why our approach of Datometry followed by recoding eases the migration path to SQL Data Warehouse.

The finished T-SQL script, with changes highlighted in read, would look like:

```sql
CREATE VIEW [ADW].[DimDate] AS SELECT
       CAL.DayNumberofWeek AS WeekName,
       CONVERT(datetime,CAL.DateKey,101) AS CalDate,
       /*note the way that alias has changed from as columname to columname=case statement*/
       "MonthOfMay"=CASE WHEN CAL.DayNumberOfMonth = 5 THEN 1 ELSE 0 END,
FROM TESTTD.CALENDER AS CAL
WHERE CAL.DateKey BETWEEN dateadd(YEAR, -3,GETDATE()) and dateadd(DAY, -1, GETDATE());
```

Semantic Layer and Reporting Views

The pattern for semantic layer views is to create these in Azure SQL Data Warehouse and then to create an Azure Analysis Services Tabular Model over the top of them to improve scale and speed. For concurrency and scale, Azure Analysis Services should be included in the architecture. Ad-hoc queries can be run against the Azure SQL Data Warehouse.

Several great benefits exist when leveraging Azure Analysis Services (AAS) with SQL DW:

- Fewer to no queued queries while increasing the overall solution concurrency
- Improved short query read performance with AAS as a caching layer
- Reduce overall solution cost by lowering DWU levels for SQL DW with query offloading

What is the best approach?

Make sure the semantic layer conforms to a Star or Snowflake Schema.

Decide on your query model for Azure Analysis Services.

- Use Cached mode for maximum query performance, where you have complex measures, full cache model and to be able to pause/scale Azure SQL Data Warehouse up and down for loads. The challenge with this mode is reduced data freshness, data must fit into memory and it is read only (no writeback).
- Use Direct Query for sourcing data directly, optimal data freshness and query at base fact level. The challenge is inconsistent query performance, consumption of concurrency slots, no caching, requires Azure SQL Data Warehouse to be online and may require higher DWU

The Analysis Services Dimensions can be optimized by:

- Minimize the number of columns.
- Reduce cardinality (data type conversions).
- Filter out unused dimension values (unless a business scenario requires them).
- Integer Surrogate Keys (SK).
- Ordered by SK (to maximize Value encoding).
- Hint for VALUE encoding on numeric columns.
- Hint for disabling hierarchies on SKs.

Analysis Services Facts can be for optimized by:

- Handle early arriving facts. [Facts without corresponding dimension records.]
- Replace dimension IDs with their surrogate keys.

- Reduce cardinality (data type conversions).
- Consider moving calculations to the source (to use in compression evaluations).
- Ordered by less diverse SKs first (to maximize compression).
- Increased Tabular sample size for deciding Encoding, by considering segments and partitions.
- Hint for VALUE encoding on numeric columns.
- Hint for disabling hierarchies.

Microsoft have published a whitepaper on modeling for AS tabular scalability.

Data Migration

There multiple ways to approach data migration and your choice will depend on the connectivity you have from your datacenter to Azure.

Execution

Source-controlled pipeline

- Data export, transfer and import steps runs mostly from the source data system
- Reuse existing computer and storage resources at the source system
- May use custom scripts and programs or ETL tools like SSIS run from the source database server Azure Controlled

Azure Controlled pipeline

- Data export, transfer and import steps runs mostly from Azure
- Aim to reduce non-Azure assets for greater maintainability, and to do on-demand migrations by spinning up or scheduling Azure resources as needed
- May run migration logic in virtual machines running on Azure, with the virtual machines being allocated and deallocated on demand, or on-premise virtual machines close to the Teradata Database.

What is the best approach?

After assessing the Teradata export toolset and the effort required in creating BTEQ scripts to export all the data, the source-controlled pipeline approach isn't appropriate for a Teradata to Azure SQL Data Warehouse migration.

Adopting the Azure Controller pipeline approach, all the way from Azure down to the on-premise resources removes human intervention and reduces the number of mistakes. It relies on less on skills and more on the infrastructure you have enabled.

You would use Azure Data Factory v2 (https://docs.microsoft.com/en-us/azure/data-factory/introduction) with Self Hosted Runtimes (https://docs.microsoft.com/en-us/azure/data-factory/create-self-hosted-integration-runtime), to manage the pipeline, installed onto a single or multiple Windows 2012 R2+ Virtual Machine(s) located near the existing Teradata Cluster. Data Factory, is a cloud data integration service, to compose data storage, movement, and processing services into automated data pipelines. A self-hosted integration runtime can run copy activities between a cloud data store and a data store in a private network, and it can dispatch transform activities against compute resources in an on-premises network or an Azure virtual network.

Teradata ships with FASTEXPORT utilities which allow you to export data which is called by Teradata Parallel Transporter. Azure Data Factory v2 Teradata connector calls the Teradata Parallel Transporter.

An internet connection to the virtual machine is required to allow Azure Data Factory to connect to self-hosted runtimes which will orchestrate the export/import process.

If your data export is between 1TB and 2TB at a time, having at least two virtual machines, with self-hosted runtimes, will increase throughput by 1.5x.

The following items are required on the Virtual Machine located in the Teradata Datacenter: -

- Windows Server 64bit 2012 R2, 8 cores and minimum 256GB RAM.
- Enough disk space to export your data to. This depends on how much whitespace you have in your database tables. It is possible to compress a 1.2TB extract to 100GB using pipe delimited text, with GZIP, or ORC formats if most columns are NULL or have repeated values.
- Teradata Tools and Utilities 15.1 (https://downloads.teradata.com/download/tools/teradata-tools-and-utilities-windows-installation-package).
- Configure Virtual Machine as per the instructions on Configuring Data Science VM earlier in this document.
- PowerShell with the Azure PowerShell module 3.6 or later (see install documentation at https://docs.microsoft.com/en-us/powershell/azure/install-azurerm-ps?view=azurermps-6.8.1)
- SQL Server Management Studio (https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-2017)
- Microsoft Connectors for Oracle and Teradata by Attunity for Integration Services (SSIS) (https://docs.microsoft.com/en-us/sql/integration-services/attunity-connectors?view=sql-server-2017)

Differentiated or Integrated Export Process

Integrated Export

- Data export, transfer and import are combined, and entities are transferred directly from the source data system to Azure SQL Data Warehouse with no intermediate files created
- Fewer moving pieces and tends to be more maintainable
- Does not compress data in bulk, and can result in slower data transfer to Azure
- Typically realized by ETL tools like Attunity Replicate for Microsoft Migrations, SSIS or with Azure Data Factory with the Data Management Gateway, which is an on-premises installable agent to enable data movement from on premise to Azure.

For the integrated history load process, we would suggest you create a Windows 2012 R2+ Virtual Machine located as near to the existing Teradata Cluster and follow the instructions on Configuring Jumpbox VM earlier in this document.

You should then follow the Attunity Replicate for Microsoft Migrations instructions for migrating data which are maintained outside this document.

The integrated approach relies on Azure ExpressRoute being in place between the source Teradata location datacenter and Azure datacenter location, then using Attunity Replicate for Microsoft Migrations to ease the historical load process.

Attunity Replicate for Microsoft Migrations can be scaled out onto multiple nodes and use one node to co-ordinate the load process. It is recommended that you build VMs, based on their recommended specifications, in the Teradata source location datacenter. Attunity will query data locally and then upload to blob storage before creating external tables and importing data using a Create Table as Select to the base tables.

Differentiated Export

- Data export, transfer and import are distinctly executed with each reading from or writing to intermediate files
- File compression is used to reduce the cost and time in transferring files from the source system to Azure

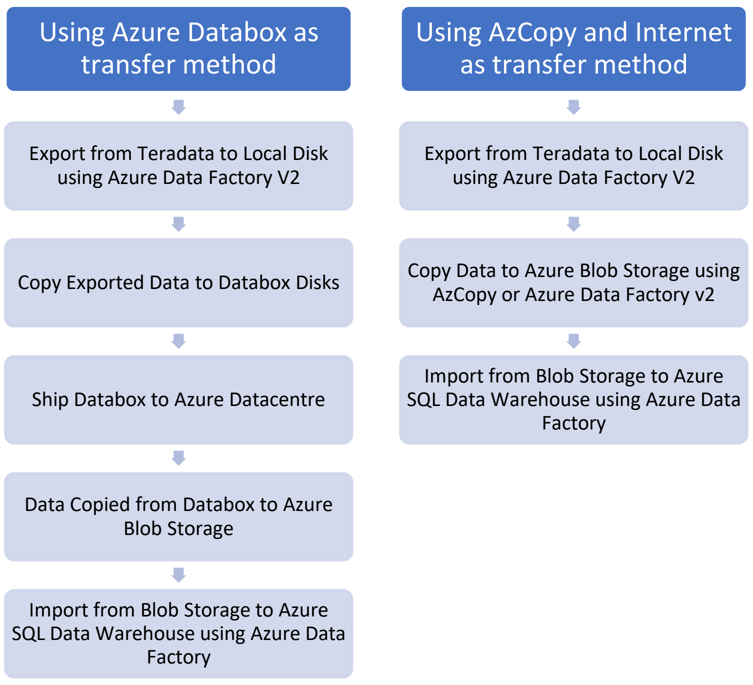
- Compressed files are transferred to Azure-based storage before import
- When the connectivity from source to Azure is lower in bandwidth or reliability, this approach may turn out to be more feasible.
- Typically realized by custom programs and scripts working independently, using tools such as bcp.exe, AzCopy, and compression libraries

The data is transferred to local storage which can then be copied to Azure Databox Disks and shipped back to the Azure datacenter. Full details on Azure Databox (https://docs.microsoft.com/en-us/azure/databox/data-box-disk-overview).

The time to transfer data over the internet or Azure Express route will drive the decision to use Azure Data Factory or AzCopy to upload the export files instead of the Azure Databox service.

For example a 100TB export, over a 70MBps connection, assuming you have 100% bandwidth all of the time, would take almost 140 days to upload. Anything more than 14 days to upload should make use of the Azure Databox service.

To summarize, the process flow, of the two methods described would be broken into: -


Using Azure Databox as transfer method
Export from Teradata to Local Disk using Azure Data Factory V2
Copy Exported Data to Databox Disks
Ship Databox to Azure Datacentre
Data Copied from Databox to Azure Blob Storage
Import from Blob Storage to Azure SQL Data Warehouse using Azure Data Factory
Using AzCopy and Internet as transfer method
Export from Teradata to Local Disk using Azure Data Factory V2
Copy Data to Azure Blob Storage using AzCopy or Azure Data Factory v2
Import from Blob Storage to Azure SQL Data Warehouse using Azure Data Factory

What is the best approach?

Where possible use ORC file format over Text Format. Delimited files lose any text formatting that has been applied and will fail if there are special characters or the delimiter appears within the data.

If you have Azure ExpressRoute or performant internet connection:

- For large databases following the different differentiated export approach, with Azure Data Factory, and export tables to Pipe Delimited Text with gzip or ORC from Teradata – AZCOPY to Azure Blob Storage or Azure Data Lake Store. Use Azure Data Factory to PolyBase into Azure SQL DW.
- To force Polybase instead of BULKCOPY, in Azure Data Factory, make sure:

- – The input dataset type is AzureBlob or AzureDataLakeStoreFile. The format type under type properties is OrcFormat, or TextFormat (Pipe Delimited Text is fastest), with the following configurations:
    - fileName doesn't contain wildcard filter.
    - rowDelimiter must be \n.
    - nullValue is either set to empty string ("") or left as default, and treatEmptyAsNull is not set to false.
    - encodingName is set to utf-8, which is the default value.
    - escapeChar, quoteChar and skipLineCount aren't specified. PolyBase supports skip header row which can be configured as firstRowAsHeader in ADF.
    - compression can be no compression, GZip, or Deflate – our results showed that 1.5TB can be compressed down to 100GB using GZIP if the data contains a lot of similar values. If using GZIP compression, set to Optimal.
- If data is relatively small;
    - Following the integrated approach by using Attunity for Microsoft Migrations and consider scaling this out with a couple of nodes.
    - Use your favorite ETL tool and parallel copy several tables at once from source to target

If you have no Azure ExpressRoute or less than 70MBps VPN to Azure, then:

- Using Azure Data Factory, export tables to Pipe Delimited Text or ORC from Teradata and use DataBox to transfer to Azure Data Center – Polybase into Azure SQL DW.

If your POC doesn’t have Azure ExpressRoute or you need to load data in stages, then you will need to look at alternative ways to load data from Teradata to SQL Data Warehouse, using the differentiated approach and your favorite ETL toolkit.

Suggested Azure Data Factory Pipeline Configuration

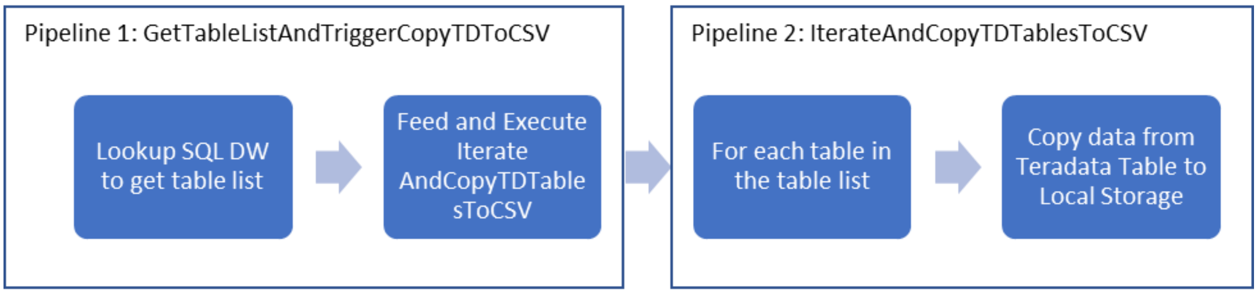
To load SQL Data Warehouse, you would make use of Azure Data Factory v2 to extract data from Teradata to local storage then copy the data from the blob storage, once the data has been transferred either by Azure Databox or AzCopy.

All configuration activities are executed from the Virtual Machine located nearest to the Teradata Cluster.

At a high level, this export involves following steps:

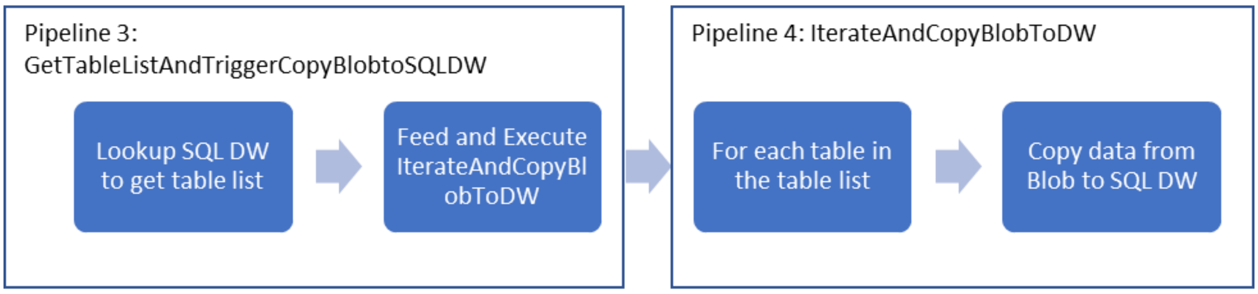
- Configure Azure Data Factory Pipeline Pre-requisites.
- Create Azure SQL Data Warehouse metadata.
- Create an Azure Data Factory.
- Create a self-hosted integration runtime.
- Create a pipeline to look up the tables to be copied and another pipeline to perform the actual copy operation to local disk
- Create a pipeline to look up the tables to be copied, from blob, and another pipeline to perform the actual copy operation to Azure SQL Data Warehouse

End-to-End Workflow


Pipeline 1: GetTableListAndTriggerCopyTDToCSV
Lookup SQL DW to get table list
Feed and Execute Iterate AndCopyTDTable sToCSV
Pipeline 2: IterateAndCopyTDTablesToCSV
For each table in the table list
Copy data from Teradata Table to Local Storage

- The first pipeline looks up the list of tables that needs to be copied over to the sink data stores. Then, the pipeline triggers another pipeline, which iterates over each table in the database and performs the data copy operation.
- The second pipeline performs the actual copy. It takes the list of tables as a parameter. For each table in the list, copy the specific table in Teradata to the local storage.

Note: Depending on bandwidth, you would copy the local storage to Azure Data Box disks, and ship Azure Datacenter, or Use AzCopy with multiple threads (no more than the number of processor cores on the VM) to copy to blob storage.


Pipeline 3: GetTableListAndTriggerCopyBlobtoSQLDW
Lookup SQL DW to get table list
Feed and Execute IterateAndCopyBl obToDW
Pipeline 4: IterateAndCopyBlobToDW
For each table in the table list
Copy data from Blob to SQL DW

- The third pipeline looks up the list of tables that needs to be copied over to the sink data stores. Then, the pipeline triggers another pipeline, which iterates over each table in the database and performs the data copy operation.
- The second pipeline performs the actual copy. It takes the list of tables as a parameter. For each table in the list, copy the specific table in blob storage to the Azure SQL Data Warehouse.

Azure Data Factory Metadata Pattern

To allow us to dynamically feed Azure Data Factory with the tables which need to be exported from Teradata, we create metadata tables in the Azure SQL Data Warehouse, which ADF v2 loops through for both the extract and import process.

You could keep these metadata tables simple BUT you must keep in mind that if you have some very large tables, you will want to split these into smaller export files, as opposed creating one large export file, which will cause out of memory issues during export and are much harder to reload should the import fail at some point.

During our testing, we always ran a smaller 1.5TB test export a couple of times, to make sure that the multiple file sizes for one table were less than 2GB.

The simple metadata table, called TablesToProcess, has schemaname and tablename columns. If you wanted to add intelligence to this, you could add an imported smallint column to allow ADF to update this column to 1 upon successful import or 3 upon failure.

The data in the simple metadata table would look like:

schemaname	tablename	Imported
Dimension	City	0
Integration	City_Staging	0
Dimension	Customer	0
Integration	Customer_Staging	0
Dimension	Date	0
Dimension	Employee	0

3. Simple metadata table

A more advanced metadata table structure would still need one table called TablesToProcess which has schemaname, tablename, filtercol, FilterVal and imported columns.

schemaname	tablename	Imported	filtercol	Imported	FilterValue
FACT	Orders	0	Orderdate	0	2015
FACT	Orders	0	Orderdate	0	2016
Dimension	Customers	0	CustomerType	0	Retail

4. Advanced metadata table

In the advanced metadata table, you will see multiple entries for the same table. As Azure Data Factory accepts parameters, we can build dynamic SQL where clauses. The filtercol contains the column we will filter on and the filtervalue is the value we will compare to.

When the pipeline runs, it will use the advance metadata table to export data into smaller files. The import will use the same table to move the data into Azure SQL Data Warehouse.

To best understand this pipeline we would suggest running through the detailed setup on the Appendix.

Next Steps

This section details the next areas you need to put into your migration plan once the schema and data have to been moved to Azure SQL Data Warehouse.

Basic Data Optimization

The tables you created to load data into will have been distributed as ROUND_ROBIN tables, without any partitions, which aren't always the most optimal for Azure SQL Data Warehouse.

Data Distribution

Data Distribution is an important concept in SQL Data Warehouse. It allows breaking down data processing requirements for efficient parallel processing across all hardware nodes. When creating any table in SQL DW, the distribution key is used to determine the node on which each row of data is stored. That is, any table will be distributed across multiple nodes based upon the distribution key. An important factor in SQL DW database design is selecting good distribution keys for fact tables.

The most expensive queries in MPP systems such as SQL DW, are those that involve a lot of data movement. When you execute a query, SQL DW will redistribute rows across the compute nodes as necessary to fulfill the query such as on JOINs and GROUP BY statements. Users should create tables with distribution styles that minimize data movement, reflective of queries that would occur in their normal workload.

SQL Data Warehouse supports three primary distribution styles:

- Hash Distribution – Based on the result of a hashing algorithm on the chosen key, SQL Data Warehouse will place the row in one of sixty distributions.
- Round Robin – SQL Data Warehouse will distribute each incoming row in a round robin fashion
- Replicated – SQL Data Warehouse will replicate the table on each of the compute nodes, backed by a round robin table for insert performance.

Once data has been loaded, you should redistribute tables to a more appropriate distribution type.

Indexing

SQL Data Warehouse offers several indexing options including clustered columnstore indexes, clustered indexes and nonclustered indexes, and a non-index option also known as heap.

Clustered columnstore tables offer both the highest level of data compression as well as the best overall query performance. Clustered columnstore tables will generally outperform clustered index or heap tables and are usually the best choice for large tables. Clustered columnstore is the best place to start when you are unsure of how to index your table.

There are a few scenarios where clustered columnstore may not be a good option:

- Columnstore tables do not support varchar(max), nvarchar(max) and varbinary(max). Consider heap or clustered index instead.
- Columnstore tables may be less efficient for transient data. Consider heap and perhaps even temporary tables.
- Small tables with less than 100 million rows. Consider heap tables.

For futher information on Indexing tables in SQL Data Warehouse please read https://docs.microsoft.com/en-us/azure/sql-data-warehouse/sql-data-warehouse-tables-index

Partitioning

Fact tables are typically millions or billions of rows in a data warehouse and using table partitioning during loading and querying can improve performance.

Benefits to loads

The primary benefit of partitioning in SQL Data Warehouse is to improve the efficiency and performance of loading data by use of partition deletion, switching and merging see (https://docs.microsoft.com/en-us/azure/sql-data-warehouse/sql-data-warehouse-tables-partition). In most cases data is partitioned on a date column that is closely tied to the order in which the data is loaded into the database. One of the greatest benefits of using partitions to maintain data is the avoidance of transaction logging. While simply inserting, updating, or deleting data can be the most straightforward approach, with a little thought and effort, using partitioning during your load process can substantially improve performance.

Partition switching can be used to quickly remove or replace a section of a table. For example, a sales fact table might contain just data for the past 36 months. At the end of every month, the oldest month of sales data is removed from the table. This data could be removed by using a delete statement to delete the data for the oldest month. However, removing a large amount of data row-by-row with a delete statement can take too much time, as well as create the risk of large transactions that take a long time to rollback if something goes wrong. A more optimal approach is to drop the oldest partition of data. Where deleting the individual rows could take hours, dropping an entire partition could take seconds.

Benefits to queries

Partitioning can also be used to improve query performance. A query that applies a filter to partitioned data can limit the scan to only the qualifying partitions. Parition elimination can avoid a full table scan and only scan a smaller subset of data. With the introduction of clustered columnstore indexes, the predicate elimination performance benefits are less beneficial, but in some cases, there can be a benefit to queries. For example, if the sales fact table is partitioned into 36 months using the sales date field, then queries that filter on the sale date can skip searching in partitions that don't match the filter.

What is the best approach?

Table Distribution

Hash distribute keys on common join conditions. Each table can only have one distribution key. As a rule of thumb, you should look at your typical query patterns and find the most common join conditions such as those between fact and the largest dimension tables as candidates for distribution. This will ensure that data is generally collocated at query time.

Choose an integer type for your distribution key if possible. Integer types such as INT or BIGINT can be used in query optimizations better than data types such as char, varchar, decimal, date, etc.

For even data skew, choose columns with large numbers of unique values for distribution keys. If you choose a date column as your distribution key, you may get a relatively even skew. However, if your query filters on a narrow date range, most of the data will land on a single compute node, thereby losing benefits of parallelization. Remember that distribution of data is more relevant to which data is chosen during a filtered query than how the data is at rest. As a rule of thumb, try to avoid any more than 30% skew across distributions.

Avoid a NULLABLE column for your distribution key. Rows where the distribution key is NULL or an "unknown" value such as -1 will be distributed in a single distribution which can lead to very skewed data and bottleneck performance.

For type 1 and type 2 dimensions.

- Since the table has multiple copies, replicated tables work best when the table size is less than 2 GB compressed. If you have dimension tables that are frequently used in join conditions, with simple query predicates, such as equality or inequality, and are rarely updated, consider replicating these tables. This will create a copy of the table on each compute node, preventing any data movement from occurring on join. If data changes frequently you should consider the performance impact and consult https://docs.microsoft.com/en-us/azure/sql-data-warehouse/design-guidance-for-replicated-tables#performance-considerations-for-modifying-replicated-tables.
- If table size is greater than 2GB user or your have queries with complex query predicates, such as LIKE or NOT LIKE. Use Distributed tables.

Consider using ROUND_ROBIN tables for tables that cannot be replicated, have no consistent join column, or have very low cardinality (uniqueness) across all columns.

Indexing

Clustered columnstore should be your default option for most table types. Clustered columnstore is a highly efficient compressed storage format that reduces IO greatly during query time.

Clustered indexes enforce a physical row sorting of your data on disk. Consider this type of index only on relatively small tables (< 60 million rows) such as (replicated) dimension tables that benefit from range filtering such as on date columns.

Sizing Partitioning

While partitioning can be used to improve performance in some scenarios, creating a table with too many partitions can hurt performance in others. These concerns are especially true for clustered columnstore tables. For partitioning to be helpful, it is important to understand when to use partitioning and the number of partitions to create. There is no hard-fast rule as to how many partitions are too many, it depends on your data and how many partitions you are loading simultaneously. A successful partitioning scheme usually has tens to hundreds of partitions, not thousands.

When creating partitions on clustered columnstore tables, it is important to consider how many rows belong to each partition. For optimal compression and performance of clustered columnstore tables, a minimum of 1 million rows per distribution and partition is needed. Before partitions are created, SQL Data Warehouse already divides each table into 60 distributed files. Any partitioning added to a table is in addition to the distributions created behind the scenes. Using this example, if the sales fact table contained 36 monthly partitions, and given that SQL Data Warehouse has 60 distributions, then the sales fact table should contain 60 million rows per month, or 2.1 billion rows when all months are populated. If a table contains fewer than the recommended minimum number of rows per partition, consider using fewer partitions in order to increase the number of rows per partition. The Indexing article on Microsoft docs includes queries that can assess the quality of cluster columnstore indexes.

Data Validation

Once the performance tuning of the schema has been completed. It is important to check that all the data has been transferred across.

You should implement a robust testing plan which includes comparing row counts, data types and checking that queries between on the source and on Azure SQL Data Warehouse return the same amount of data.

A full testing plan is outside the scope of this whitepaper but consider defining a repeatable process.

Creating the Azure Analysis Services Model

We discussed selecting the correct model type for the size of your data under Semantic Layer and Reporting Views.

Microsoft have publish a whitepaper called Performance Tuning of Tabular Models in SQL Server 2012 Analysis Services. This will help you optimise and performance tune your model.

Optimise the processessing of Azure Analysis Services by using partitioning, in your model, which align with the partitions in Azure SQL Data Warehouse.Large datasets normally require table partitioning to accelerate and optimize the data-load process. Azure Analysis Services partitioning enables incremental loads, increases parallelization, and reduces memory consumption. See Automated Partition Management with Azure Analysis Services and Azure Analysis Service Partitions.

Reporting

After creating your Azure Analysis Service Model your should recreate your reports. For a Proof of Concept, choose no more than five reports and decide if you will keep your existing reporting technology or adopt a new solution.

If the reports are complex and are going to take time to transform, then Datometry is a great solution for being able to lift and shift your reporting solution whilst you evaluate your path.

If you are moving to a Microsoft reporting solution such as Power BI it can still make sense to repoint your current reporting solution, via Datometry, whilst you redevelop your reports. This is a much quicker way to get users working with your modern data warehouse.

Appendix – Detailed Setup

Create Teradata Database Azure VM

Pre-requisites:

- Azure Subscription with contributor permission
- One Azure vNet with three subnets provisioned
- Teradata Schema Creation Scripts

1. Open Microsoft Edge or Chrome and go to the Azure Marketplace (https://azuremarketplace.microsoft.com/en-us/marketplace/) to start off the provisioning of the Teradata Database Azure VM.

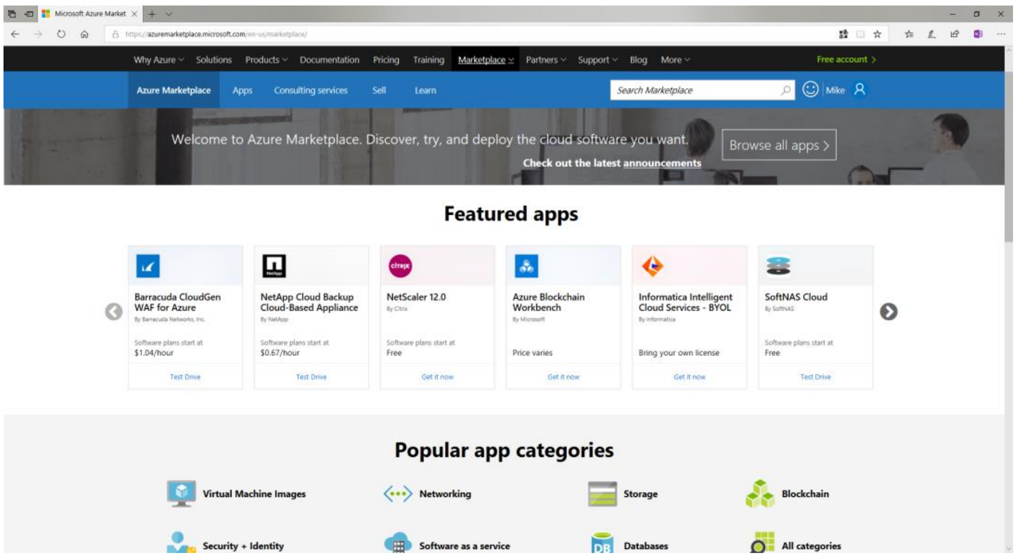
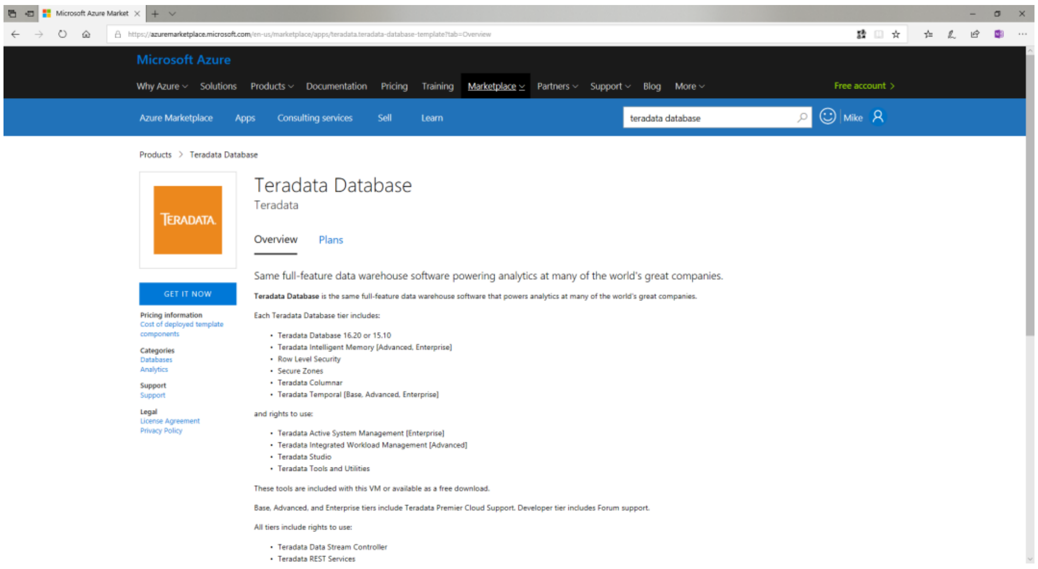


Figure 7: Azure Marketplace

2. Using the Search function, in the top right corner, search the market place for Teradata Database and click the selection as it appears in the drop-down list.



Teradata Database
Teradata

Same full-feature data warehouse software powering analytics at many of the world's great companies.

Teradata Database is the same full-feature data warehouse software that powers analytics at many of the world's great companies.

Each Teradata Database tier includes:

- Teradata Database 16.20 or 15.10
- Teradata Intelligent Memory [Advanced, Enterprise]
- Row Level Security
- Secure Zones
- Teradata Columnar
- Teradata Temporal [Base, Advanced, Enterprise]

and rights to use:

- Teradata Active System Management [Enterprise]
- Teradata Integrated Workload Management [Advanced]
- Teradata Studio
- Teradata Tools and Utilities

These tools are included with this VM or available as a free download.

Base, Advanced, and Enterprise tiers include Teradata Premier Cloud Support. Developer tier includes Forum support.

All tiers include rights to use:

- Teradata Data Stream Controller
- Teradata REST Services

Pricing information
Cost of deployed template components

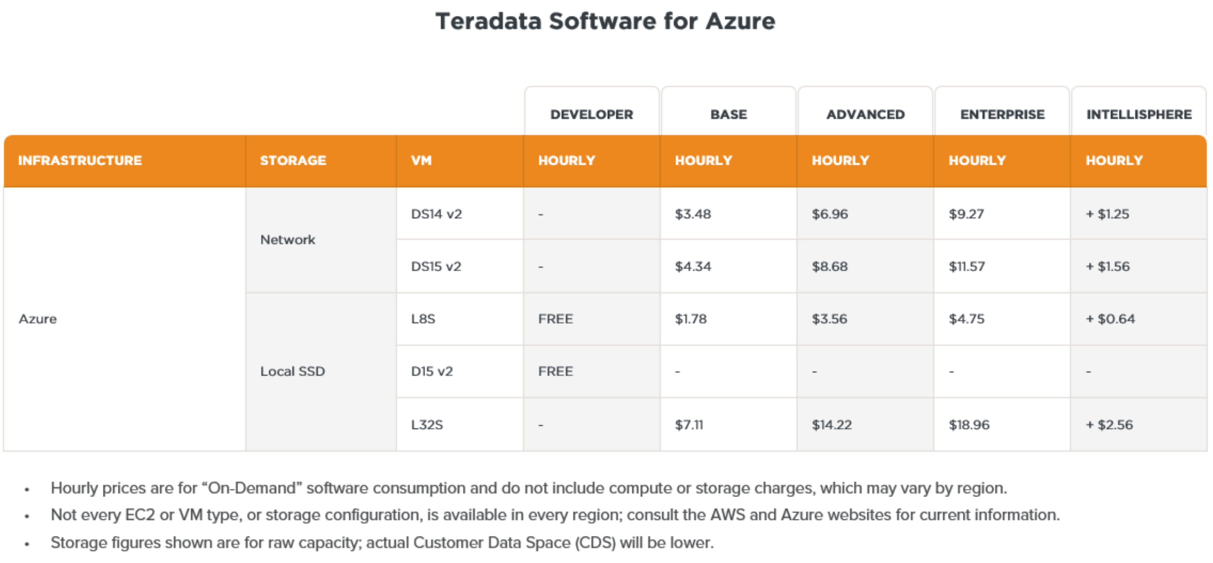
Categories
Databases
Analytics

Support
Support

Legal
License Agreement
Privacy Policy

Figure 8: Teradata Marketplace Offering

3. Before continuing check the price plans. For the purpose of this document we will be creating a Development version of Teradata Database which, as at 4th July 2018, is provided free (All you pay for is the Azure compute and storage). Teradata is licensed for use on specific Azure VMs. For latest information on supported software and VMs, visit www.teradata.com/azure/products/.



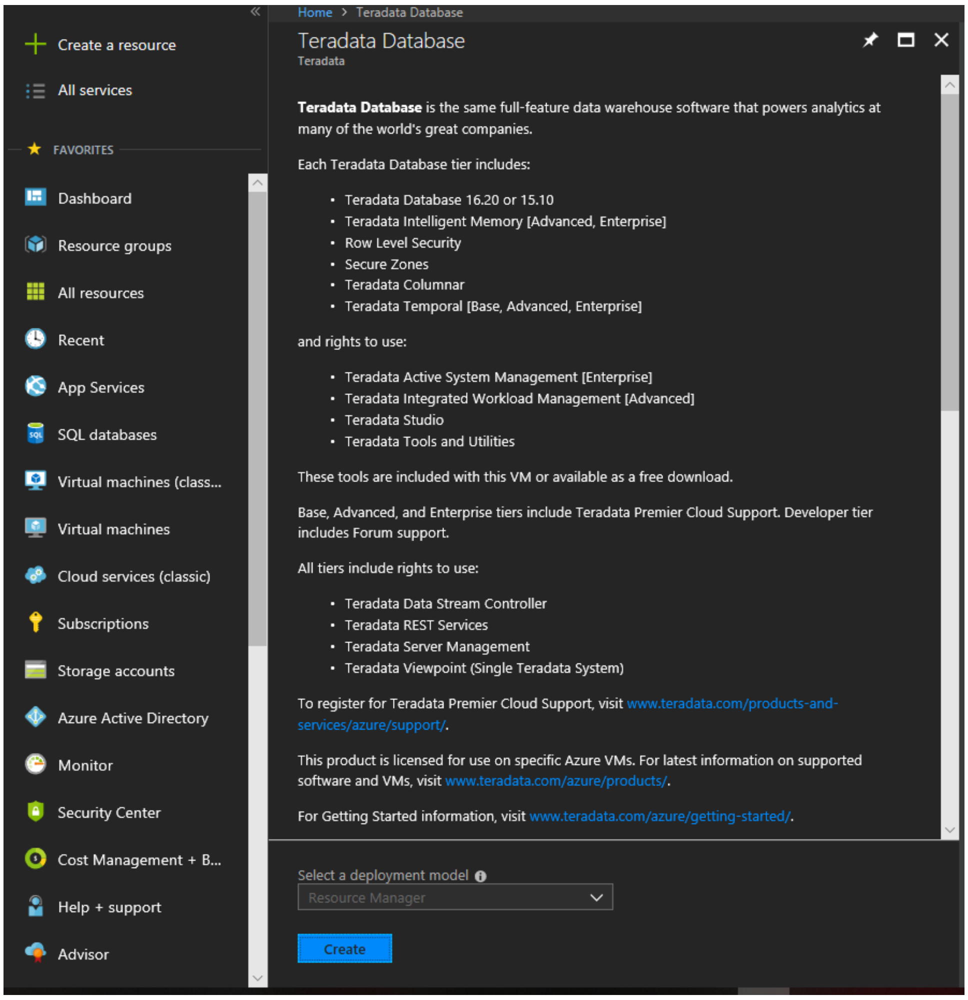
Teradata Software for Azure

			DEVELOPER	BASE	ADVANCED	ENTERPRISE	INTELLISPHERE
INFRASTRUCTURE	STORAGE	VM	HOURLY	HOURLY	HOURLY	HOURLY	HOURLY
Azure	Network	DS14 v2	-	$3.48	$6.96	$9.27	+ $1.25
		DS15 v2	-	$4.34	$8.68	$11.57	+ $1.56
	Local SSD	L8S	FREE	$1.78	$3.56	$4.75	+ $0.64
		D15 v2	FREE	-	-	-	-
		L32S	-	$7.11	$14.22	$18.96	+ $2.56

- Hourly prices are for "On-Demand" software consumption and do not include compute or storage charges, which may vary by region.
- Not every EC2 or VM type, or storage configuration, is available in every region; consult the AWS and Azure websites for current information.
- Storage figures shown are for raw capacity; actual Customer Data Space (CDS) will be lower.

Figure 9: Teradata Costs as at July 2018

4. Click Get Now and fill in the details on the form, click continue, and this will then take you through to provisioning Teradata for Azure.



Teradata Database
Teradata

Teradata Database is the same full-feature data warehouse software that powers analytics at many of the world's great companies.

Each Teradata Database tier includes:

- Teradata Database 16.20 or 15.10
- Teradata Intelligent Memory [Advanced, Enterprise]
- Row Level Security
- Secure Zones
- Teradata Columnar
- Teradata Temporal [Base, Advanced, Enterprise]

and rights to use:

- Teradata Active System Management [Enterprise]
- Teradata Integrated Workload Management [Advanced]
- Teradata Studio
- Teradata Tools and Utilities

These tools are included with this VM or available as a free download.

Base, Advanced, and Enterprise tiers include Teradata Premier Cloud Support. Developer tier includes Forum support.

All tiers include rights to use:

- Teradata Data Stream Controller
- Teradata REST Services
- Teradata Server Management
- Teradata Viewpoint (Single Teradata System)

To register for Teradata Premier Cloud Support, visit www.teradata.com/products-and-services/azure/support/.

This product is licensed for use on specific Azure VMs. For latest information on supported software and VMs, visit www.teradata.com/azure/products/.

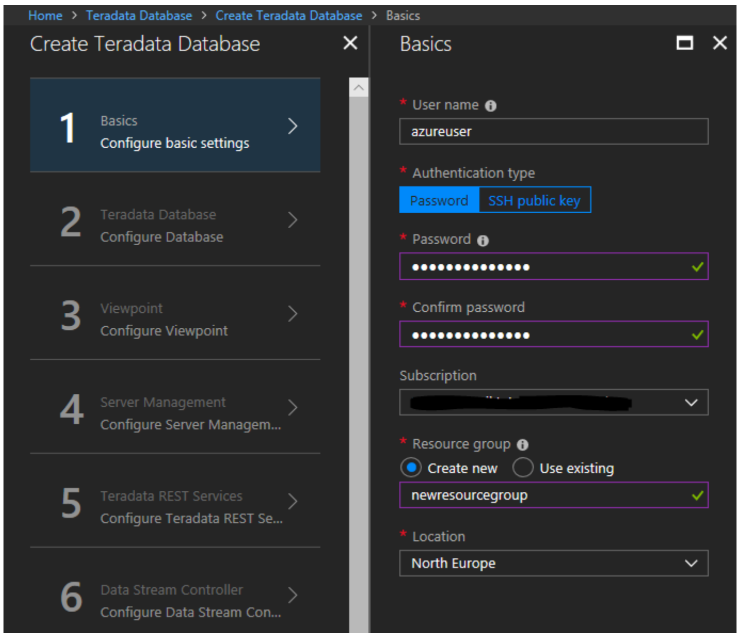
For Getting Started information, visit www.teradata.com/azure/getting-started/.

Select a deployment model: Resource Manager

Create

Figure 10: Create Teradata Cluster

5. Click Create to take you through to the next Azure blade.
6. Fill out the user name, password and make sure that you chose to create a new resource group. Failure to use a new resource group will see the deployment fail.
7. Click OK



Home > Teradata Database > Create Teradata Database > Basics

Create Teradata Database

1 Basics
Configure basic settings

2 Teradata Database
Configure Database

3 Viewpoint
Configure Viewpoint

4 Server Management
Configure Server Managem...

5 Teradata REST Services
Configure Teradata REST Se...

6 Data Stream Controller
Configure Data Stream Con...

Basics

* User name: azureuser

* Authentication type: [✓ Password] SSH public key

* Password: ••••••••••••••

* Confirm password: ••••••••••••••

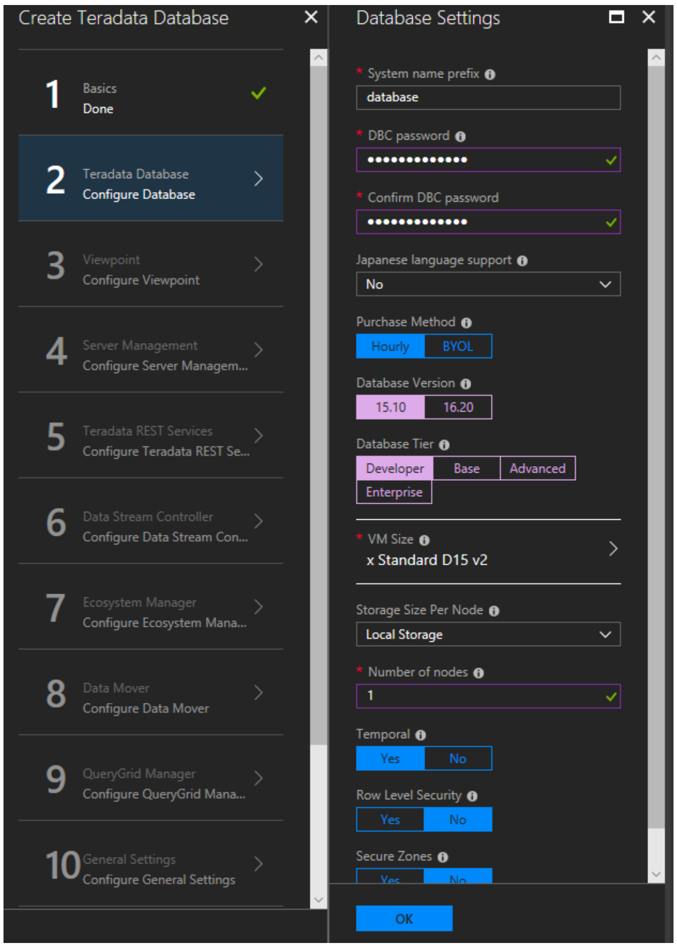
Subscription: [redacted]

* Resource group: [✓ Create new] Use existing
newresourcegroup

* Location: North Europe

Figure 11: Teradata Basic Settings

8. On the Teradata database blade, fill out the DBC password (this is the password you will use to connect to Teradata).
    a. Choose your Purchase method. For our lab we selected Hourly.
    b. Choose your Database version and Tier. As most migrated systems are from 15.10 we chose this for our lab and the development tier as we are only created a schema.
    c. Leave the VM size at Standard D15 v2
    d. Change the number of nodes from 2 to 1.
9. Click OK


Create Teradata Database

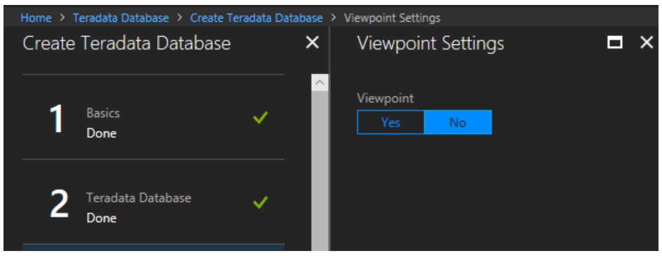
1 Basics — Done ✓
2 Teradata Database — Configure Database
3 Viewpoint — Configure Viewpoint
4 Server Management — Configure Server Managem...
5 Teradata REST Services — Configure Teradata REST Se...
6 Data Stream Controller — Configure Data Stream Con...
7 Ecosystem Manager — Configure Ecosystem Mana...
8 Data Mover — Configure Data Mover
9 QueryGrid Manager — Configure QueryGrid Mana...
10 General Settings — Configure General Settings

Database Settings

* System name prefix: database
* DBC password: ••••••••••••• ✓
* Confirm DBC password: ••••••••••••• ✓
Japanese language support: No
Purchase Method: [✓ Hourly] BYOL
Database Version: [✓ 15.10] 16.20
Database Tier: [✓ Developer] Base Advanced Enterprise
* VM Size: x Standard D15 v2
Storage Size Per Node: Local Storage
* Number of nodes: 1 ✓
Temporal: [✓ Yes] No
Row Level Security: Yes [✓ No]
Secure Zones: Yes [✓ No]

Figure 12: Teradata Database Configuration

10. On the Viewpoint Blade select No and click OK.


Home > Teradata Database > Create Teradata Database > Viewpoint Settings

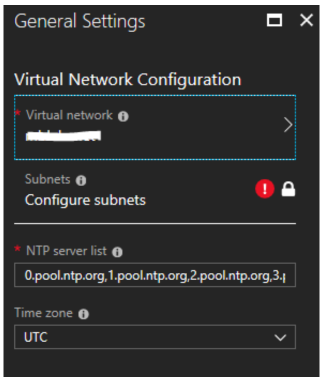
Create Teradata Database

1 Basics — Done ✓
2 Teradata Database — Done ✓

Viewpoint Settings

Viewpoint: Yes [✓ No]

11. On the Server Management Blade, click OK.
12. On the Teradata REST Services Settings, click OK.
13. On the Data Stream Controller, click OK.
14. On the Ecosystem Manager, click OK.
15. On the Data Mover, click OK.
16. On the Query Grid Manager, Click OK.
17. On the General Setting Tab click onto Virtual Networks and select the Azure vNet, with the three subnets, which you have already created.


General Settings

Virtual Network Configuration

* Virtual network: [redacted]

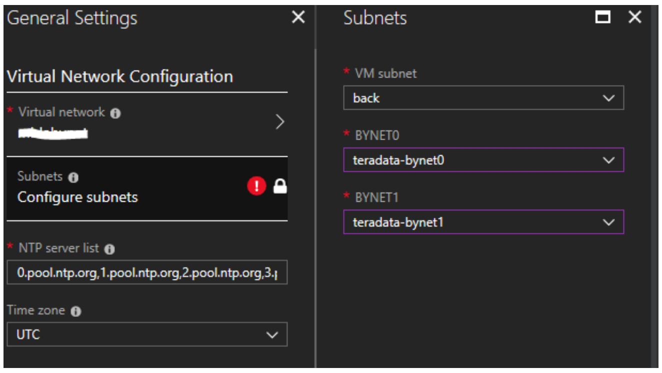
Subnets: Configure subnets

* NTP server list: 0.pool.ntp.org,1.pool.ntp.org,2.pool.ntp.org,3.|

Time zone: UTC

Figure 13: Teradata General Settings

18. On the General Setting Tab click onto subnets and chose the three subnet you wish to use. Make sure you allocate the subnet you wish to communicate on to the VM subnet as you won't be able to communicate over the other two networks.
19. Click OK.


General Settings

Virtual Network Configuration

* Virtual network: [redacted]

Subnets: Configure subnets

* NTP server list: 0.pool.ntp.org,1.pool.ntp.org,2.pool.ntp.org,3.|

Time zone: UTC

Subnets

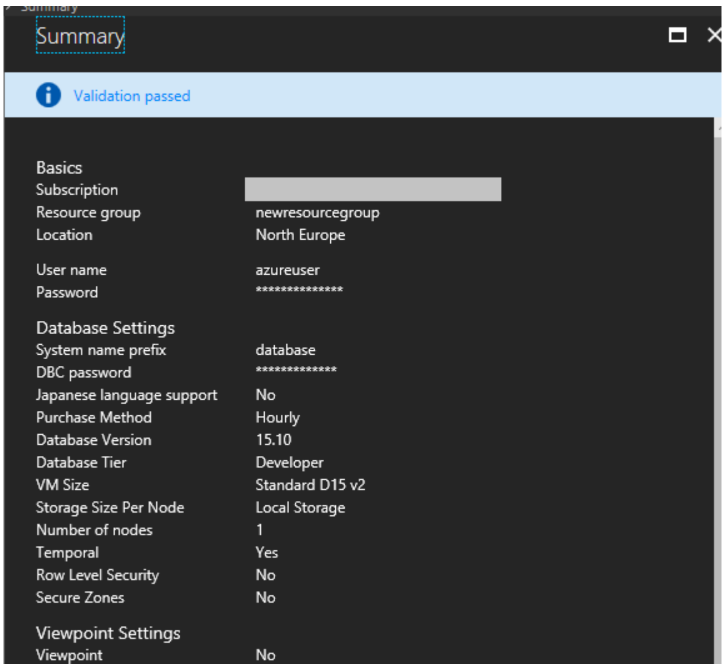
* VM subnet: back

* BYNET0: teradata-bynet0

* BYNET1: teradata-bynet1

Figure 14: Teradata Virtual Network Settings

20. Click OK
21. On the Summary page you will see the information which will be passed to the ARM template.
22. Review the summary information
23. Click OK.



Summary

Validation passed

Basics

Subscription	[redacted]
Resource group	newresourcegroup
Location	North Europe
User name	azureuser
Password	**************

Database Settings

System name prefix	database
DBC password	*************
Japanese language support	No
Purchase Method	Hourly
Database Version	15.10
Database Tier	Developer
VM Size	Standard D15 v2
Storage Size Per Node	Local Storage
Number of nodes	1
Temporal	Yes
Row Level Security	No
Secure Zones	No

Viewpoint Settings

Viewpoint	No

Figure 15: Teradata Azure Validation

24. Review the Terms of use and Privacy Policy.

25. If agree to the terms, click Create.

Figure 16:Create Teradata Cluster

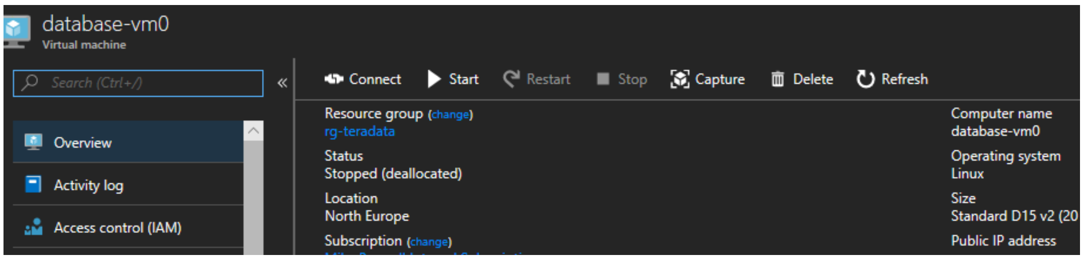
Note: Provisioning of the Teradata Database (Developer Edition) can take up to 30 minutes.

Important: once the Teradata database VM has been provision. do not shut this VM down until you have finished the schema conversion, or it will fail to come back online.

Opening Teradata Port 1025

To allow access to the Teradata Database VM we need to enable port 1025, on the firewall, for the virtual network. Without this our Data Science VM won't be able to connect to create and then extract the schema.

1. In the Azure Portal go to Virtual Machines and click on the Teradata Database VM you created.



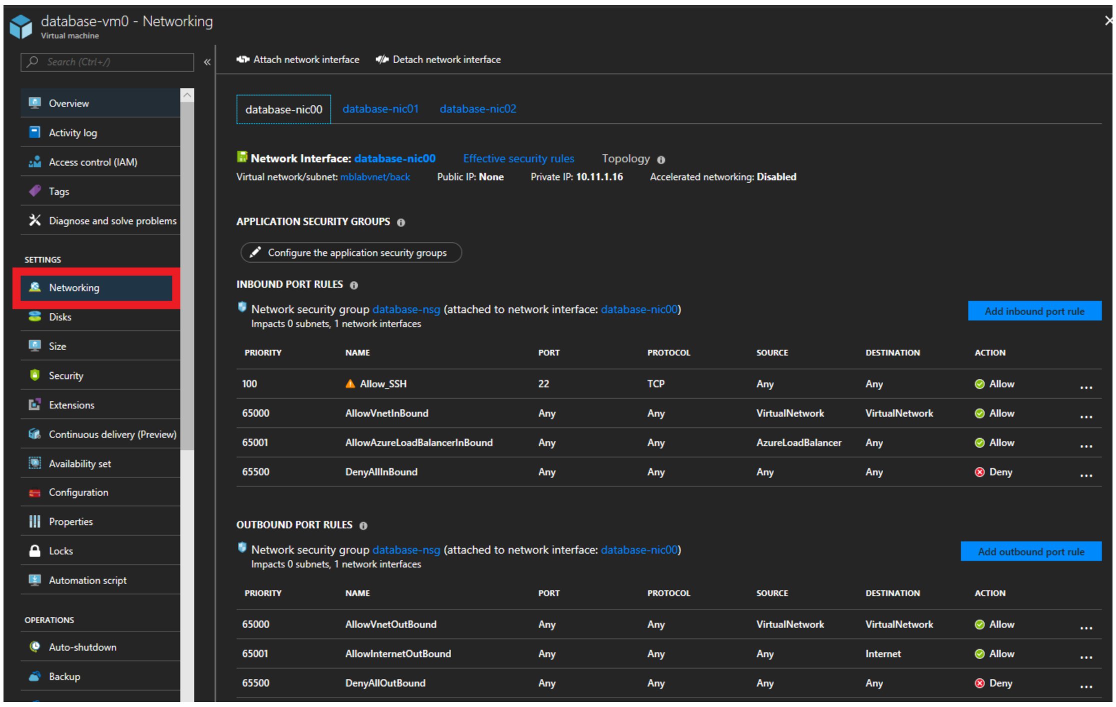
Resource group (change)
rg-teradata
Status
Stopped (deallocated)
Location
North Europe
Subscription (change)

Computer name
database-vm0
Operating system
Linux
Size
Standard D15 v2 (20
Public IP address

Figure 17: Virtual Machine Blade

2. In the VM Blade click on Networking to display the current port rules.

3. You will only add a port rule to the first network card. Usually end with "-NIC00"
4. Click Add Inbound Port Rule.



Network Interface: database-nic00

Virtual network/subnet: mblabvnet/back Public IP: None Private IP: 10.11.1.16 Accelerated networking: Disabled

INBOUND PORT RULES

Network security group database-nsg (attached to network interface: database-nic00)
Impacts 0 subnets, 1 network interfaces

PRIORITY	NAME	PORT	PROTOCOL	SOURCE	DESTINATION	ACTION
100	Allow_SSH	22	TCP	Any	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

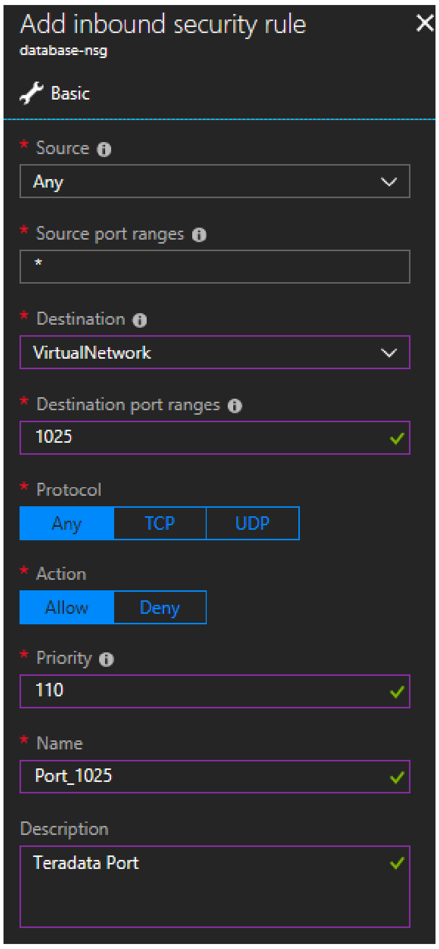
OUTBOUND PORT RULES

Network security group database-nsg (attached to network interface: database-nic00)
Impacts 0 subnets, 1 network interfaces

PRIORITY	NAME	PORT	PROTOCOL	SOURCE	DESTINATION	ACTION
65000	AllowVnetOutBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowInternetOutBound	Any	Any	Any	Internet	Allow
65500	DenyAllOutBound	Any	Any	Any	Any	Deny

Figure 18: Networking Firewall Rules

5. On the Inbound Port Rule blade set: -
   a. Source: Any
   b. Source port ranges: *
   c. Destination: Virtual Network
   d. Destination Port: 1025
   e. Protocol: Any
   f. Action: Allow
   g. Priority: 110
   h. Name: Port_1025
   i. Click Add
6. It will take about 4 minutes to create the port rule.


Add inbound security rule
database-nsg
Basic

* Source: Any
* Source port ranges: *
* Destination: VirtualNetwork
* Destination port ranges: 1025
* Protocol: [✓ Any] TCP UDP
* Action: [✓ Allow] Deny
* Priority: 110
* Name: Port_1025
Description: Teradata Port

Figure 19: Add inbound security rule

Create Jumpbox VM

We will use this to connect to the Teradata Azure VM and load some additional software onto this machine to assist with the schema migration.

1. Open Microsoft Edge or Chrome and go to the Azure Marketplace (https://azuremarketplace.microsoft.com/en-us/marketplace/) to start off the provisioning of the Data Science Virtual Machine - Windows 2012.

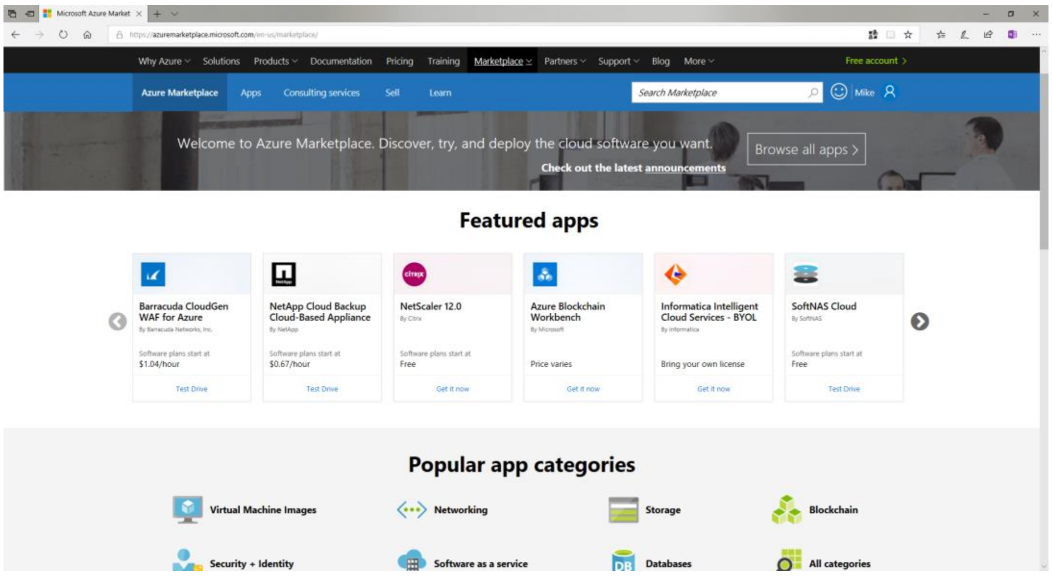
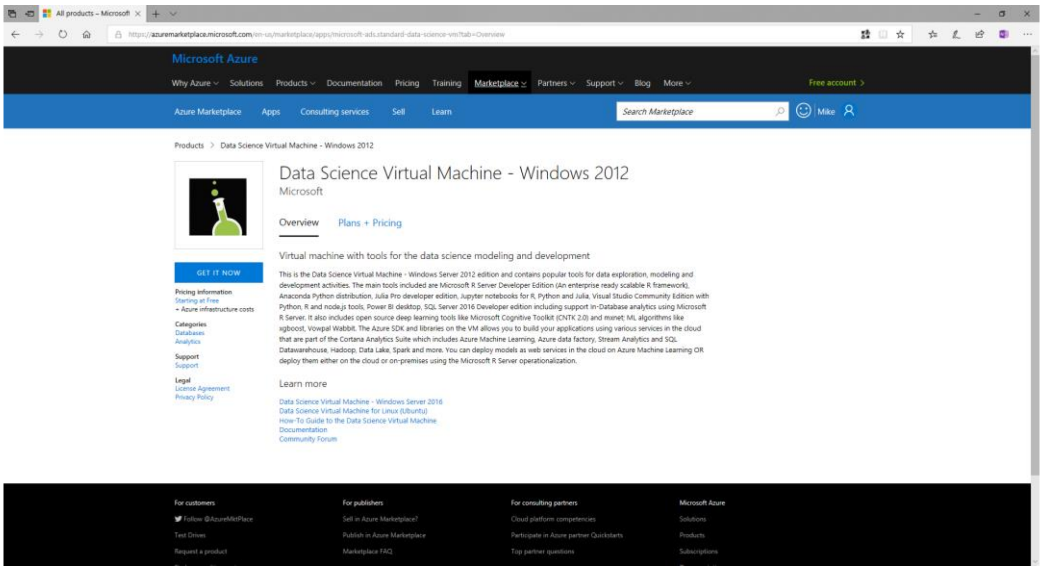


Figure 20: Azure Marketplace

2. Using the Search function, in the top right corner, search the market place for Data Science Virtual Machine - Windows 2012 and click the selection as it appears in the drop-down list.



Data Science Virtual Machine - Windows 2012
Microsoft

Pricing information
Starting at Free
+ Azure infrastructure costs

Categories
Databases
Analytics

Support
Support

Legal
License Agreement
Privacy Policy

Virtual machine with tools for the data science modeling and development

This is the Data Science Virtual Machine - Windows Server 2012 edition and contains popular tools for data exploration, modeling and development activities. The main tools included are Microsoft R Server Developer Edition (An enterprise ready scalable R framework), Anaconda Python distribution, Julia Pro developer edition, Jupyter notebooks for R, Python and Julia, Visual Studio Community Edition with Python, R and node.js tools, Power BI desktop, SQL Server 2016 Developer edition including support In-Database analytics using Microsoft R Server. It also includes open source deep learning tools like Microsoft Cognitive Toolkit (CNTK 2.0) and mxnet; ML algorithms like xgboost, Vowpal Wabbit. The Azure SDK and libraries on the VM allows you to build your applications using various services in the cloud that are part of the Cortana Analytics Suite which includes Azure Machine Learning, Azure data factory, Stream Analytics and SQL Datawarehouse, Hadoop, Data Lake, Spark and more. You can deploy models as web services in the cloud on Azure Machine Learning OR deploy them either on the cloud or on-premises using the Microsoft R Server operationalization.

Learn more

Data Science Virtual Machine - Windows Server 2016
Data Science Virtual Machine for Linux (Ubuntu)
How-To Guide to the Data Science Virtual Machine
Documentation
Community Forum

For customers
Follow @AzureMktPlace
Test Drives
Request a product

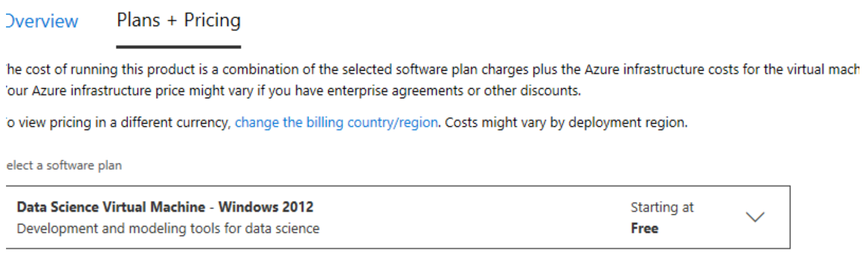
For publishers
Sell in Azure Marketplace?
Publish in Azure Marketplace
Marketplace FAQ

For consulting partners
Cloud platform competencies
Participate in Azure partner Quickstarts
Top partner questions

Microsoft Azure
Solutions
Products
Subscriptions

Figure 21: Data Science VM in Azure Marketplace

3. Before continuing check the price plans. For the purposes of this document we will be creating a DS3_v2.



Ↄverview Plans + Pricing

he cost of running this product is a combination of the selected software plan charges plus the Azure infrastructure costs for the virtual mach
'our Azure infrastructure price might vary if you have enterprise agreements or other discounts.

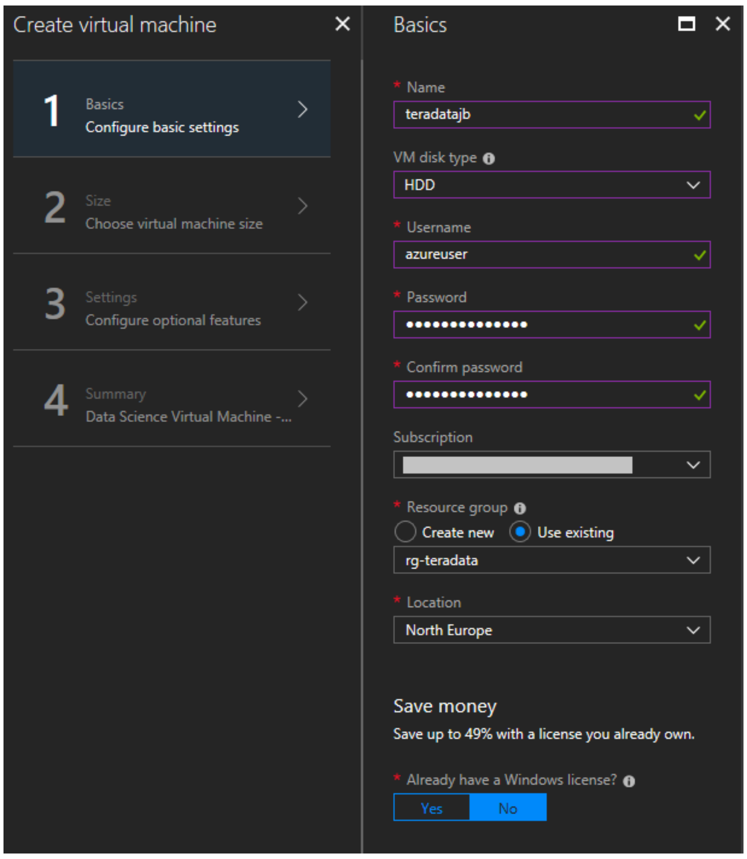
o view pricing in a different currency, change the billing country/region. Costs might vary by deployment region.

elect a software plan

Data Science Virtual Machine - Windows 2012 Starting at
Development and modeling tools for data science **Free**

Figure 22: Data Science VM Pricing

4. Click Get Now
5. Click continue, and this will then take you through to provisioning.
6. Click Create to take you through to the next Azure blade.
   a. Chose a VM Name
   b. Change the VM Disk Type from SSD to HDD.
   c. Fill out the user name, password and make sure that you chose the resource group you used when you created the Teradata Database
   d. Place the VM in the same location as the Teradata Database.
7. Click OK



Create virtual machine

1 Basics
Configure basic settings

2 Size
Choose virtual machine size

3 Settings
Configure optional features

4 Summary
Data Science Virtual Machine -...

Basics

* Name: teradatajb

VM disk type: HDD

* Username: azureuser

* Password: ••••••••••••••

* Confirm password: ••••••••••••••

Subscription: [redacted]

* Resource group: Create new [✓ Use existing]
rg-teradata

* Location: North Europe

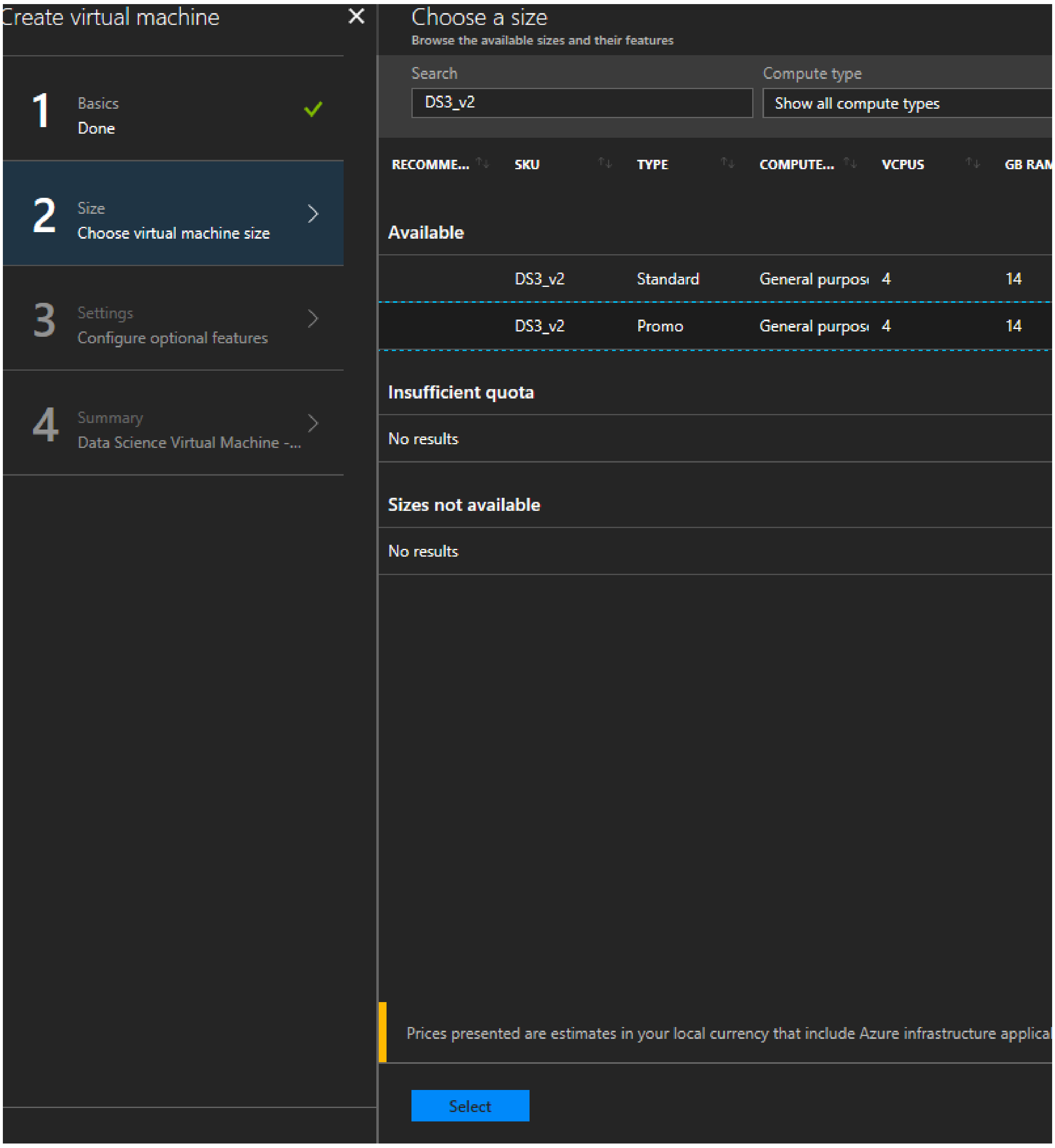
Save money
Save up to 49% with a license you already own.

* Already have a Windows license? Yes [✓ No]

Figure 23: Create virtual machine

8. Under choose virtual machine size type DS3_v2 into the search box.
9. Click on the VM size

10. Click Select



Create virtual machine

1 Basics Done

2 Size Choose virtual machine size

3 Settings Configure optional features

4 Summary Data Science Virtual Machine -...

Choose a size

Browse the available sizes and their features

Search: DS3_v2

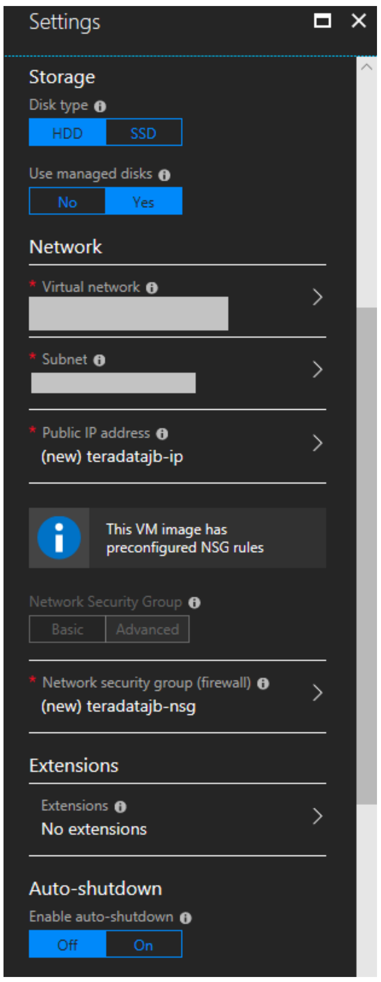
Compute type: Show all compute types

RECOMME...	SKU	TYPE	COMPUTE...	VCPUS	GB RAM
Available					
	DS3_v2	Standard	General purpos	4	14
	DS3_v2	Promo	General purpos	4	14
Insufficient quota					
No results					
Sizes not available					
No results					

Prices presented are estimates in your local currency that include Azure infrastructure applical

Figure 24: Virtual Machine Size

11. Under Settings checked the disk type is HDD and that managed disk are being used.
12. Click on Virtual Network and select the network which you previously used for the Teradata Database VM to connect to.
13. Click on Subnet and select the subnet which you allocated to the Teradata Database VM connection.
14. To allow public access, then leave the Public IP address allow. Otherwise select public IP and then select none.
15. Accept the other defaults and Click OK



Settings

Storage

Disk type: [✓ HDD] SSD

Use managed disks: No [✓ Yes]

Network

* Virtual network: [redacted]

* Subnet: [redacted]

* Public IP address: (new) teradatajb-ip

This VM image has preconfigured NSG rules

Network Security Group: Basic Advanced

* Network security group (firewall): (new) teradatajb-nsg

Extensions

Extensions: No extensions

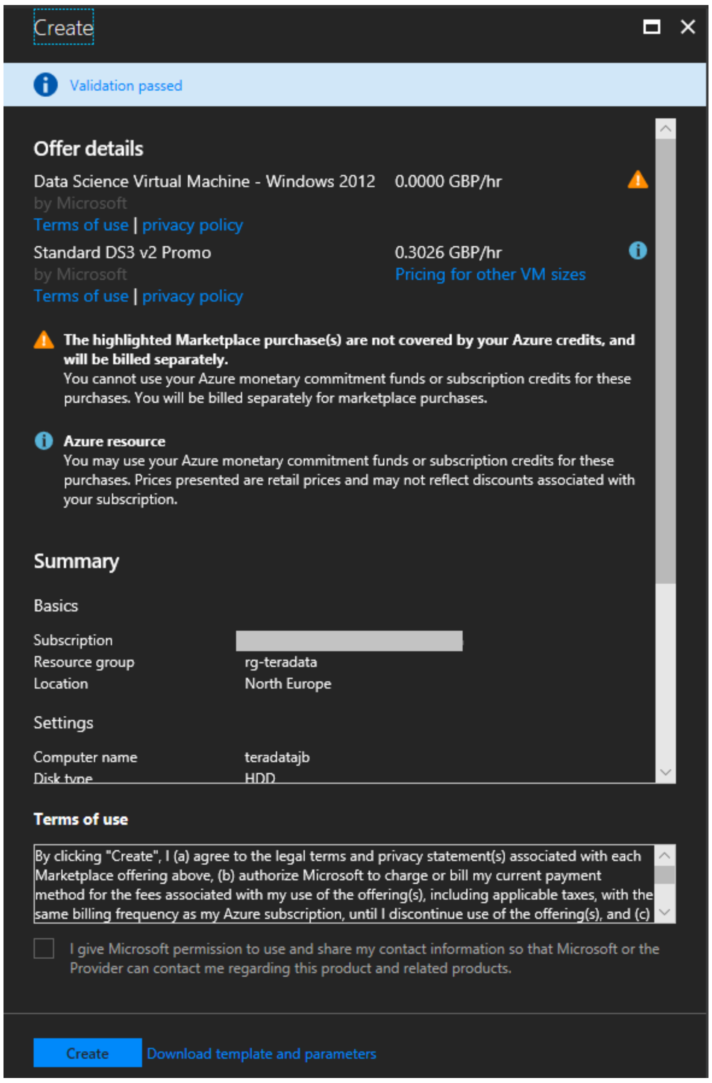
Auto-shutdown

Enable auto-shutdown: [✓ Off] On

Figure 25: Virtual Machine Settings

16. Review the Create Blade.

17. Click Create and allow up to 30 minutes for the VM to be provisioned.


Create

Validation passed

Offer details

Data Science Virtual Machine - Windows 2012 0.0000 GBP/hr
by Microsoft
Terms of use | privacy policy

Standard DS3 v2 Promo 0.3026 GBP/hr
by Microsoft Pricing for other VM sizes
Terms of use | privacy policy

The highlighted Marketplace purchase(s) are not covered by your Azure credits, and will be billed separately.
You cannot use your Azure monetary commitment funds or subscription credits for these purchases. You will be billed separately for marketplace purchases.

Azure resource
You may use your Azure monetary commitment funds or subscription credits for these purchases. Prices presented are retail prices and may not reflect discounts associated with your subscription.

Summary

Basics

Subscription	[redacted]
Resource group	rg-teradata
Location	North Europe

Settings

Computer name	teradatajb
Disk type	HDD

Terms of use

By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with each Marketplace offering above, (b) authorize Microsoft to charge or bill my current payment method for the fees associated with my use of the offering(s), including applicable taxes, with the same billing frequency as my Azure subscription, until I discontinue use of the offering(s), and (c)

☐ I give Microsoft permission to use and share my contact information so that Microsoft or the Provider can contact me regarding this product and related products.

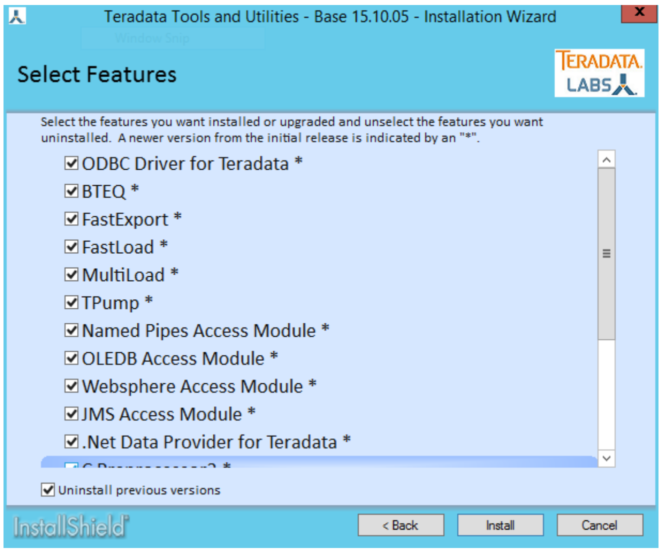
Figure 26: Create Virtual Machine

Configure Jumpbox VM

The jump box needs to be configured with software to connect to the Teradata database, to create your schema, and has the migration software installed which will take your Teradata schema across to SQL Data Warehouse.

1. Remote Desktop to the Azure Data Science VM you created in the previous step.
2. Download the Teradata Tools and Utilities v15.xx from http://downloads.teradata.com/download/tools/teradata-tools-and-utilities-windows-installation-package.
3. Extract the ZIP file to C:\InstallFiles\TeradataTools
4. Go to C:\InstallFiles\TeradataTools\TTUExpress
5. Launch TTU_BASE to install core Teradata Components

6. Select every Feature and Click Install.


Teradata Tools and Utilities - Base 15.10.05 - Installation Wizard

Select Features

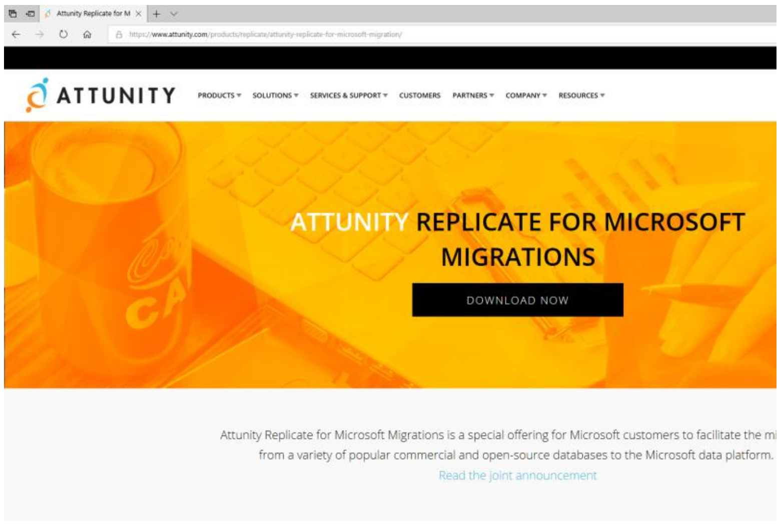
Select the features you want installed or upgraded and unselect the features you want uninstalled. A newer version from the initial release is indicated by an "*".

- ☑ ODBC Driver for Teradata *
- ☑ BTEQ *
- ☑ FastExport *
- ☑ FastLoad *
- ☑ MultiLoad *
- ☑ TPump *
- ☑ Named Pipes Access Module *
- ☑ OLEDB Access Module *
- ☑ Websphere Access Module *
- ☑ JMS Access Module *
- ☑ .Net Data Provider for Teradata *

☑ Uninstall previous versions

Figure 27: TTU Tools Installation

7. Download Attunity Replicate for Microsoft Migrations from https://www.attunity.com/products/replicate/attunity-replicate-for-microsoft-migration/ to C:\InstallFiles.

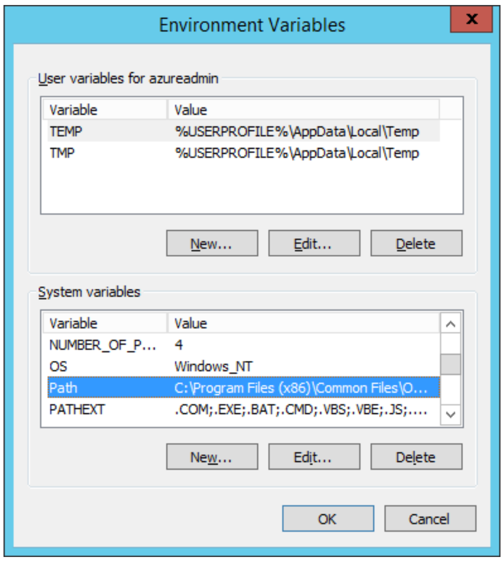


Attunity Replicate for Microsoft Migrations is a special offering for Microsoft customers to facilitate the mi
from a variety of popular commercial and open-source databases to the Microsoft data platform.
Read the joint announcement

Figure 28: Attunity Website

8. Download the Attunity Replicate for Microsoft Migrations User and Setup Guide from https://attunity.jiveon.com/servlet/JiveServlet/downloadBody/1013-102-5-1048/AttunityReplicateMSM_5_5_0_User_Guide.pdf
9. Open PDF and follow installation instructions from section three to four.
10. Chose “Install local Replicate for Microsoft Migrations Server”.
11. Download Java SE Runtime Environment 8 and install from http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html
12. Right-click on My Computer, and then select Properties.
13. Click on Advance Settings.
14. Click on Environment Variables.

15. Add C:\Program Files\Teradata\Client\15.10\ODBC Driver for Teradata to the System Path.
16. Click OK.
17. Click OK



User variables for azureadmin

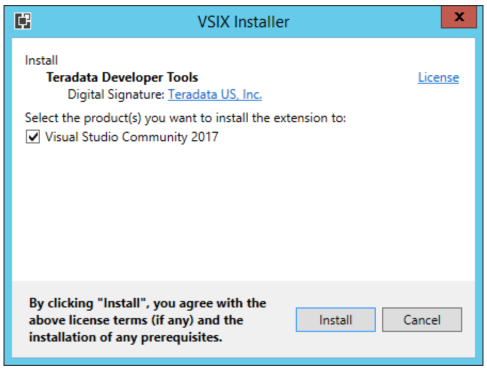
Variable	Value
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
NUMBER_OF_P...	4
OS	Windows_NT
Path	C:\Program Files (x86)\Common Files\O...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....

Figure 29: Virtual Machine Environment Varibles

18. Download, Open and Install the Teradata Developer tools from Visual Studio from https://marketplace.visualstudio.com/items?itemName=MikeDempsey.TeradataDeveloperTools



Install
Teradata Developer Tools
Digital Signature: Teradata US, Inc.
Select the product(s) you want to install the extension to:
☑ Visual Studio Community 2017

By clicking "Install", you agree with the above license terms (if any) and the installation of any prerequisites.

Figure 30: Teradata Visual Studio Install

19. Launch Visual Studio 2017.
20. Sign into Visual Studio Community Edition.

Close all applications and log off.

Create a SQL Data Warehouse Dev DB

You will need a blank SQL Data Warehouse database to migrate the schema to and this needs to be setup prior to running Attunity Replicate.

1. Log into the Azure Portal (https://portal.azure.com)
2. Click Create a Resource.
3. Click Databases.

4. Click SQL Datawarehouse to create a new Data Warehouse database.

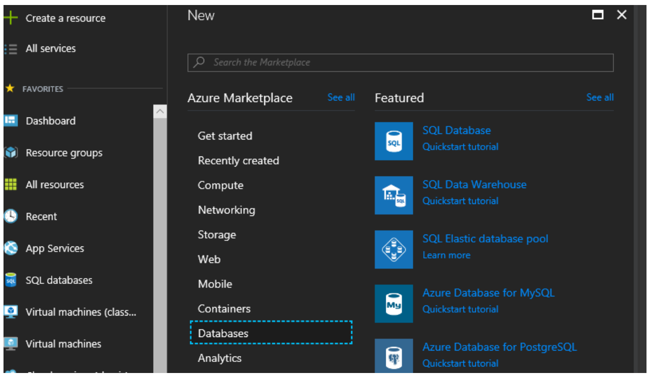
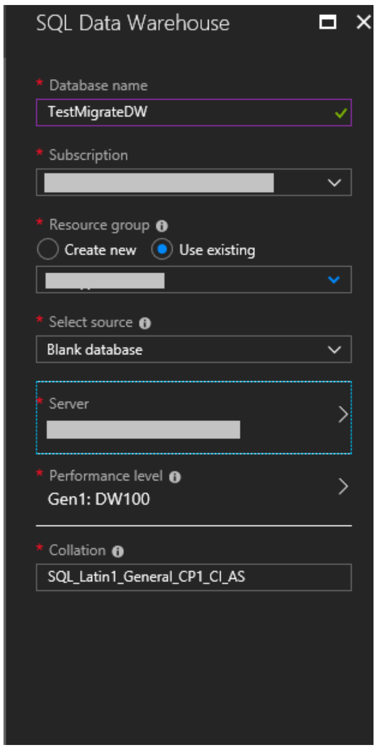


Figure 31: Azure New Service Blade

5. Enter a Database Name.
6. Use the Teradata resource group.



SQL Data Warehouse

* Database name
TestMigrateDW

* Subscription
[redacted]

* Resource group
Create new [✓ Use existing]
[redacted]

* Select source
Blank database

* Server
[redacted]

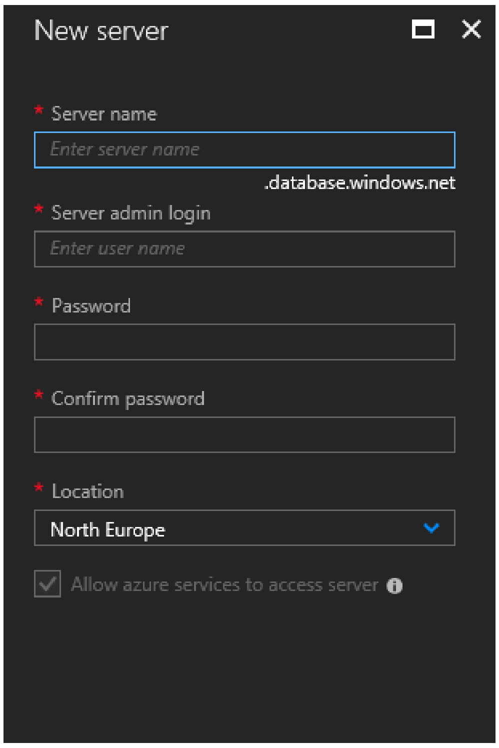
* Performance level
Gen1: DW100

* Collation
SQL_Latin1_General_CP1_CI_AS

Figure 32: Azure SQL Data Warehouse Settings

7. Click Server and create a new server (make a note of the login, password and server name).

8. Click Select.



New server

* Server name

Enter server name

.database.windows.net

* Server admin login

Enter user name

* Password

* Confirm password

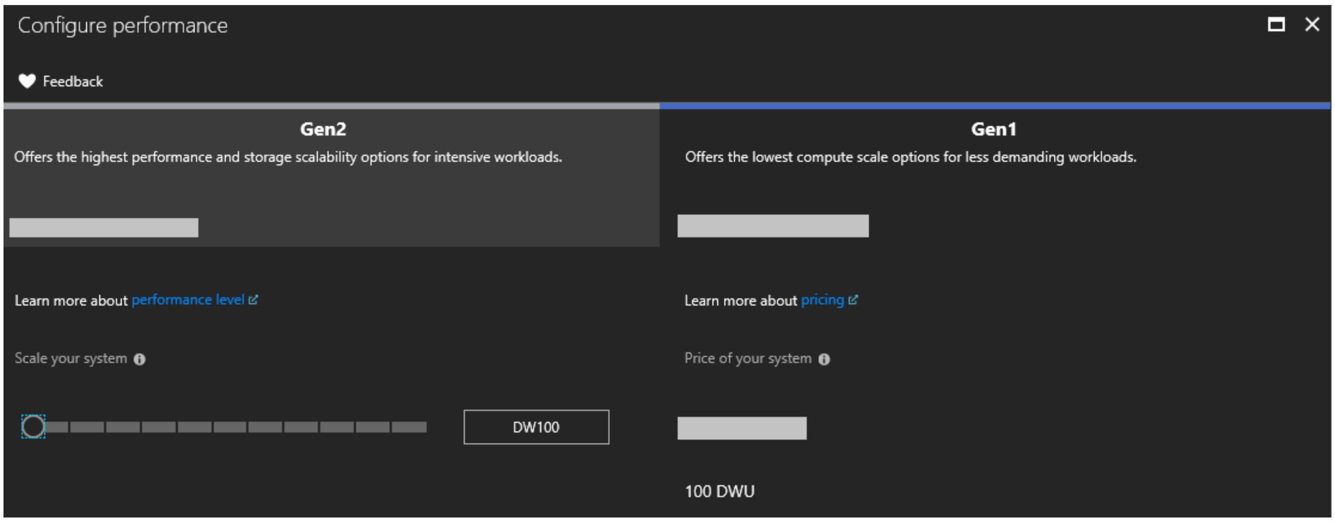
* Location

North Europe

✓ Allow azure services to access server

Figure 33: Add New Server

9. Click Performance level and configure for Gen1 DWU100 – you can scale this up later when you need to load data.
10. Click Apply.



Configure performance

Feedback

Gen2

Offers the highest performance and storage scalability options for intensive workloads.

Gen1

Offers the lowest compute scale options for less demanding workloads.

Learn more about performance level

Learn more about pricing

Scale your system

DW100

Price of your system

100 DWU

Figure 34:Performance Tier

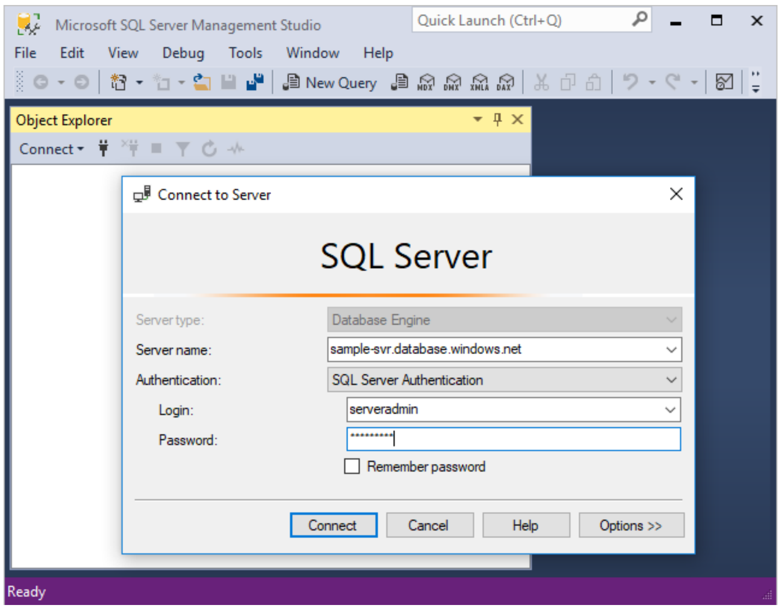
11. Click Create.

Configure SQL Data Warehouse

Prior to deploying the Teradata Schema, you will need to check connectivity from the Data Science VM; Create a login and user that is dedicated for loading data; Create a MASTER KEY.

1. Remote Desktop to the Azure Data Science VM you created in the previous step.
2. Open SQL Server Management Studio.

3. In the Connect to Server dialog box enter:
    a. Server type
    b. Database engine
    c. Server name: The fully qualified server name
    d. Authentication: SQL Server Authentication
    e. Login: The server admin account (This is the account that you specified when you created the server)
    f. Password
    g. The password for your server admin account
4. Click Connect to open the Object Explorer Windows in SSMS.



Server type: Database Engine
Server name: sample-svr.database.windows.net
Authentication: SQL Server Authentication
Login: serveradmin
Password: *********
Remember password

Figure 35: Connect to Azure SQL Data Warehouse

5. In Object Explorer, expand Databases. Then expand System databases and master to view the objects in the master database. Expand the Azure SQL Data Warehouse Database you created to view the objects in your new database.

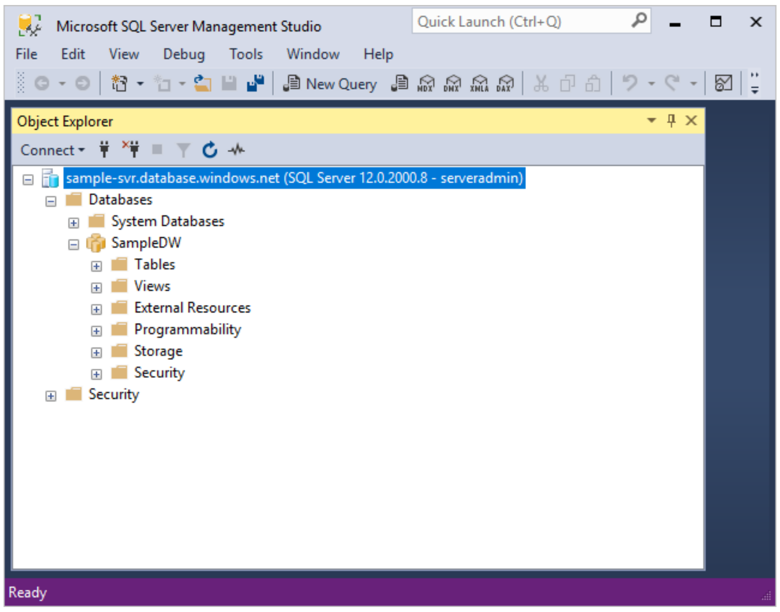
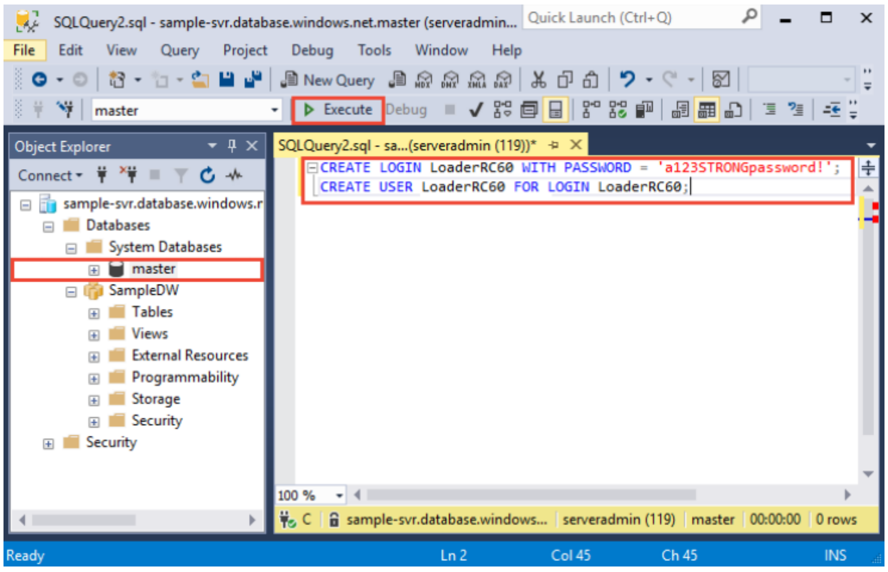


Figure 36: Object View

6. To create the user for loading data, in SSMS right click master to show a drop-down menu and choose New Query.
7. In the query window, enter these T-SQL commands to create a login and user named LoaderRC60, substituting your own password for 'a123STRONGpassword!'.
   a. CREATE LOGIN LoaderRC60 WITH PASSWORD = 'a123STRONGpassword!';
   b. CREATE USER LoaderRC60 FOR LOGIN LoaderRC60;
8. Click Execute.



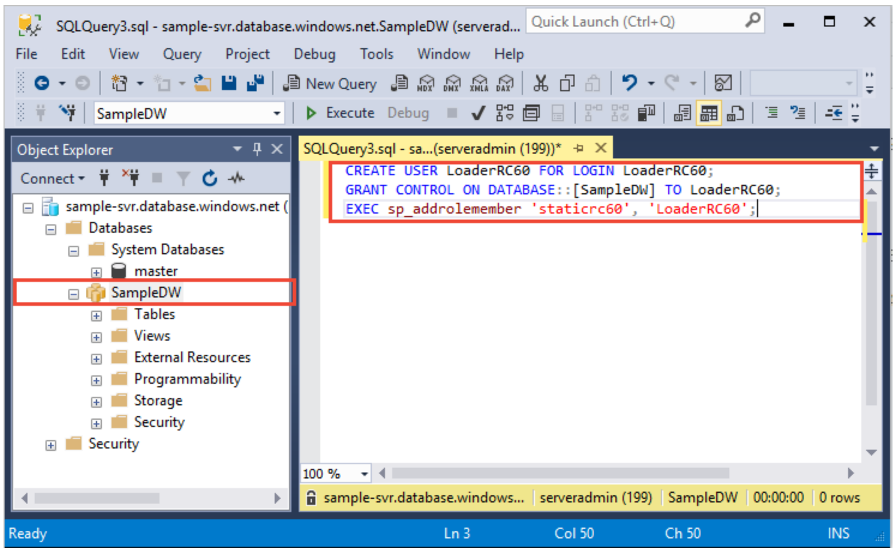
```
CREATE LOGIN LoaderRC60 WITH PASSWORD = 'a123STRONGpassword!';
CREATE USER LoaderRC60 FOR LOGIN LoaderRC60;
```

SQLQuery2.sql - sample-svr.database.windows.net.master (serveradmin...
Execute
master
Object Explorer
sample-svr.database.windows.r
Databases
System Databases
master
SampleDW
Tables
Views
External Resources
Programmability
Storage
Security
Security
sample-svr.database.windows... serveradmin (119) master 00:00:00 0 rows
Ready Ln 2 Col 45 Ch 45 INS

Figure 37: New Query

9. Right-click your Azure SQL Data Warehouse Database and choose New Query. A new query Window opens.

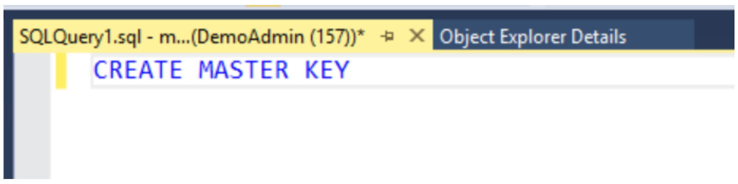
10. Enter the following T-SQL commands to create a database user named LoaderRC60 for the LoaderRC60 login. The second line grants the new user CONTROL permissions on the new data warehouse. These permissions are similar to making the user the owner of the database. The third line adds the new user as a member of the 'xlargerc' resource class.
    a. CREATE USER LoaderRC60 FOR LOGIN LoaderRC60;
    b. GRANT CONTROL ON DATABASE::[<YourSQLDWDB>] to LoaderRC60;
    c. EXEC sp_addrolemember 'largerc', 'LoaderRC60';
11. Click Execute



```
CREATE USER LoaderRC60 FOR LOGIN LoaderRC60;
GRANT CONTROL ON DATABASE::[SampleDW] TO LoaderRC60;
EXEC sp_addrolemember 'staticrc60', 'LoaderRC60';
```

Figure 38: Create User

12. To create the user for loading data, in SSMS right click master to show a drop-down menu and choose New Query. In the query window, enter this T-SQL commands
13. CREATE MASTER KEY



```
CREATE MASTER KEY
```

Figure 39: Create Master Key

14. Click Execute
15. Close all applications and log off.

Creating the Teradata Schema in Azure

This describes the process for creating your Teradata Database Schema in Azure Teradata VM. It presumes that you are a DBA or have Teradata skills.

1. Export you on-premise Teradata Schema into .sql text files.
2. Log on Azure Data Science Machine
3. Launch Visual Studio 2017
4. From menu select Teradata and click New Query.

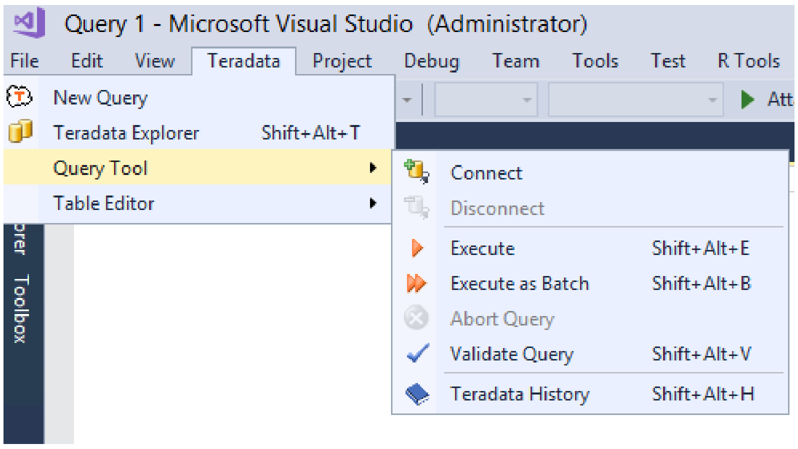
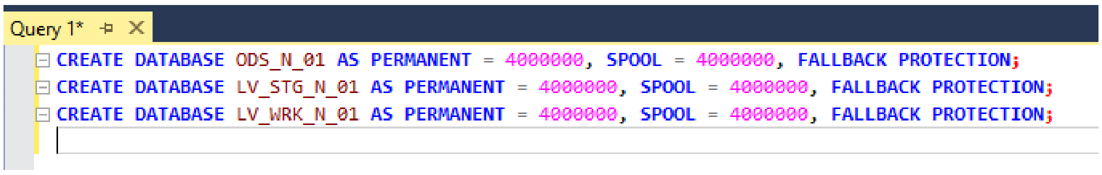

Query 1 - Microsoft Visual Studio (Administrator)
File Edit View Teradata Project Debug Team Tools Test R Tools
New Query
Teradata Explorer Shift+Alt+T
Query Tool
Table Editor
Connect
Disconnect
Execute Shift+Alt+E
Execute as Batch Shift+Alt+B
Abort Query
Validate Query Shift+Alt+V
Teradata History Shift+Alt+H

Figure 40: New Teradata Query

5. In the new query window, type the statements to recreate your tablespace(s) on Teradata



```
CREATE DATABASE ODS_N_01 AS PERMANENT = 4000000, SPOOL = 4000000, FALLBACK PROTECTION;
CREATE DATABASE LV_STG_N_01 AS PERMANENT = 4000000, SPOOL = 4000000, FALLBACK PROTECTION;
CREATE DATABASE LV_WRK_N_01 AS PERMANENT = 4000000, SPOOL = 4000000, FALLBACK PROTECTION;
```

Figure 41: Recreate Teradata Tablespace

6. Before executing you need to connect to the Azure Teradata instance. From menu select Teradata and go to Query Tool > Connect

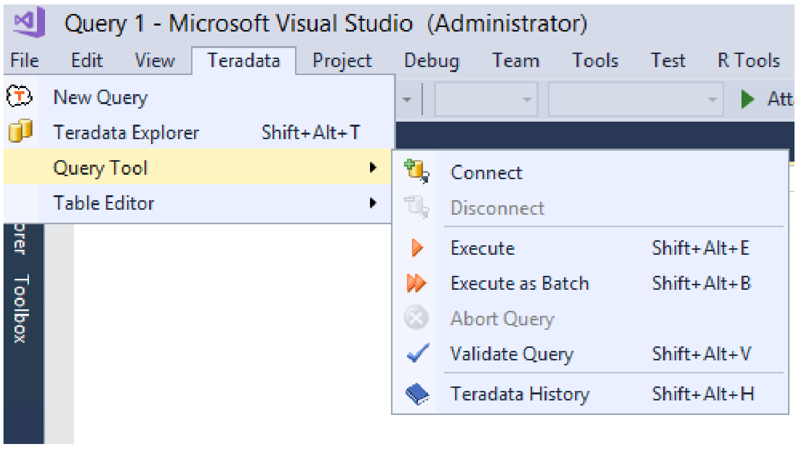
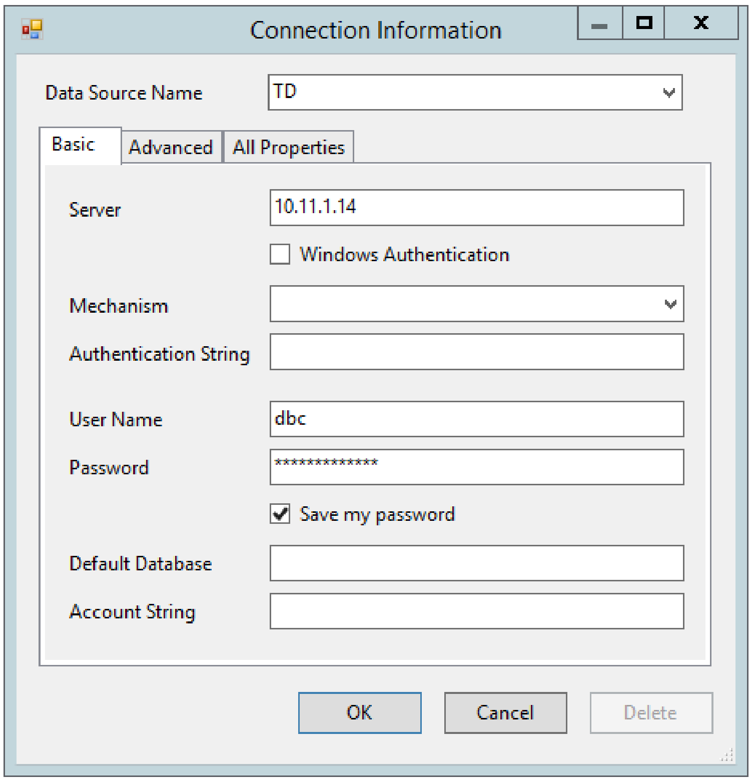

Query 1 - Microsoft Visual Studio (Administrator)
File Edit View Teradata Project Debug Team Tools Test R Tools
New Query
Teradata Explorer Shift+Alt+T
Query Tool
Table Editor
Connect
Disconnect
Execute Shift+Alt+E
Execute as Batch Shift+Alt+B
Abort Query
Validate Query Shift+Alt+V
Teradata History Shift+Alt+H

Figure 42: Connect to Teradata

7. The Connection Information dialogue allows you to connect to the instance.
   a. Data Source Name: TD
   b. Server: IP of Teradata Network -NIC00 interface
   c. User Name: dbc
   d. Password: <Password you set>



Connection Information

Data Source Name: TD

Basic | Advanced | All Properties

Server: 10.11.1.14

☐ Windows Authentication

Mechanism:

Authentication String:

User Name: dbc

Password: *************

☑ Save my password

Default Database:

Account String:

OK | Cancel | Delete

Figure 43: Connection Information Dialogue

8. Click OK
9. From menu select Teradata and go to Query Tool >Execute

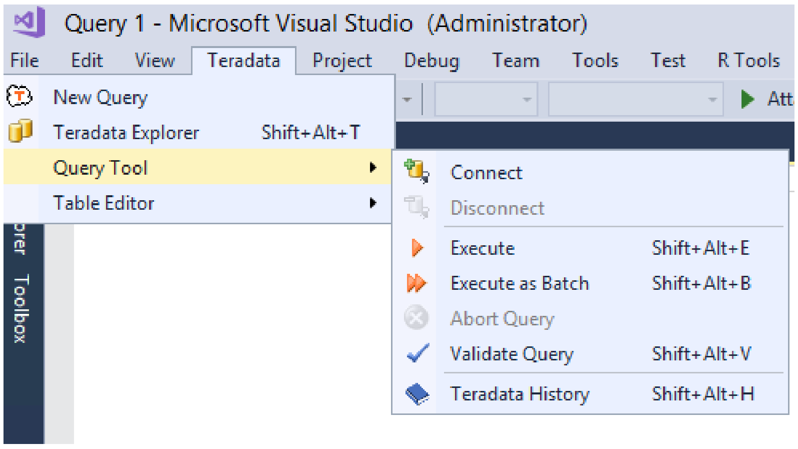
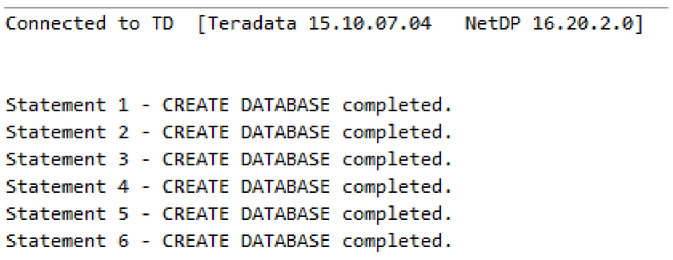

Query 1 - Microsoft Visual Studio (Administrator)
File Edit View Teradata Project Debug Team Tools Test R Tools
New Query
Teradata Explorer Shift+Alt+T
Query Tool
Table Editor
Connect
Disconnect
Execute Shift+Alt+E
Execute as Batch Shift+Alt+B
Abort Query
Validate Query Shift+Alt+V
Teradata History Shift+Alt+H
Toolbox

Figure 44: Execute Query

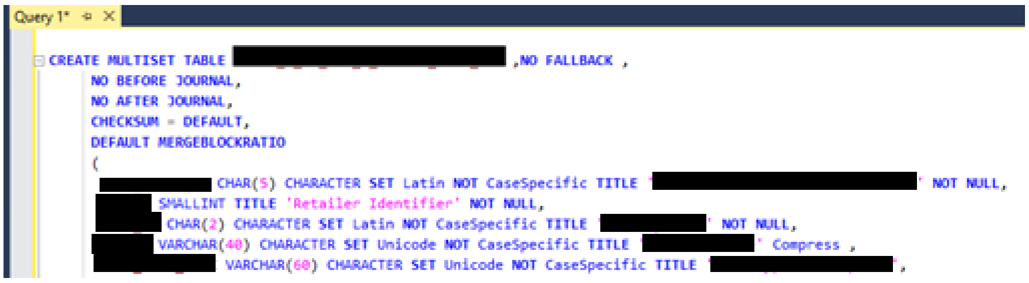


```
Connected to TD  [Teradata 15.10.07.04   NetDP 16.20.2.0]


Statement 1 - CREATE DATABASE completed.
Statement 2 - CREATE DATABASE completed.
Statement 3 - CREATE DATABASE completed.
Statement 4 - CREATE DATABASE completed.
Statement 5 - CREATE DATABASE completed.
Statement 6 - CREATE DATABASE completed.
```

Figure 45: Query Output

10. Repeat the above to create the tables in the Teradata Database



```
CREATE MULTISET TABLE [redacted] ,NO FALLBACK ,
     NO BEFORE JOURNAL,
     NO AFTER JOURNAL,
     CHECKSUM = DEFAULT,
     DEFAULT MERGEBLOCKRATIO
     (
      [redacted] CHAR(5) CHARACTER SET Latin NOT CaseSpecific TITLE '[redacted]' NOT NULL,
      [redacted] SMALLINT TITLE 'Retailer Identifier' NOT NULL,
      [redacted] CHAR(2) CHARACTER SET Latin NOT CaseSpecific TITLE '[redacted]' NOT NULL,
      [redacted] VARCHAR(40) CHARACTER SET Unicode NOT CaseSpecific TITLE '[redacted]' Compress ,
      [redacted] VARCHAR(60) CHARACTER SET Unicode NOT CaseSpecific TITLE '[redacted]',
```

Figure 46: Create Tables

Using Attunity Replicate to Create Schema

Launching Attunity Replicate

1. Open the Attunity Replicate for Microsoft Migrations Console
2. From the Windows Start menu, select All Programs > Attunity Replicate for Microsoft Migrations > Attunity Replicate for Microsoft Migrations Console.

   To access the Console from a remote type the following address in the address bar of your Web browser:

   http://<computer name>/attunityreplicatemsm

   where <computer name> is the name or IP address of the computer

3. Add a Teradata Migration Task to the Attunity Replicate for Microsoft Migrations Console.

   After opening the console, the next step is to add a migration task.

To add a replication task:

1. Make sure that you are in TASKS view. The word "TASKS" should be displayed in the upper left corner of the Attunity Replicate for Microsoft Migrations Console.

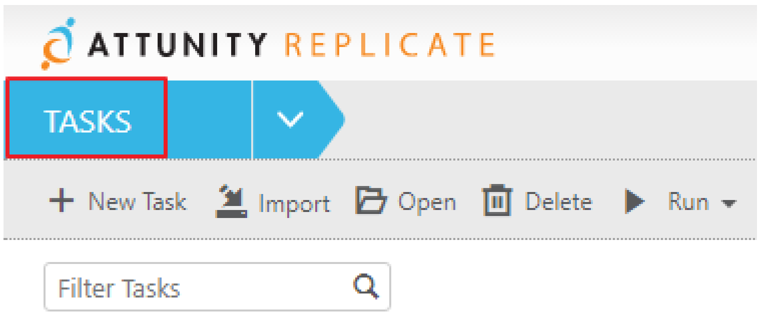


Figure 47: Attunity Task

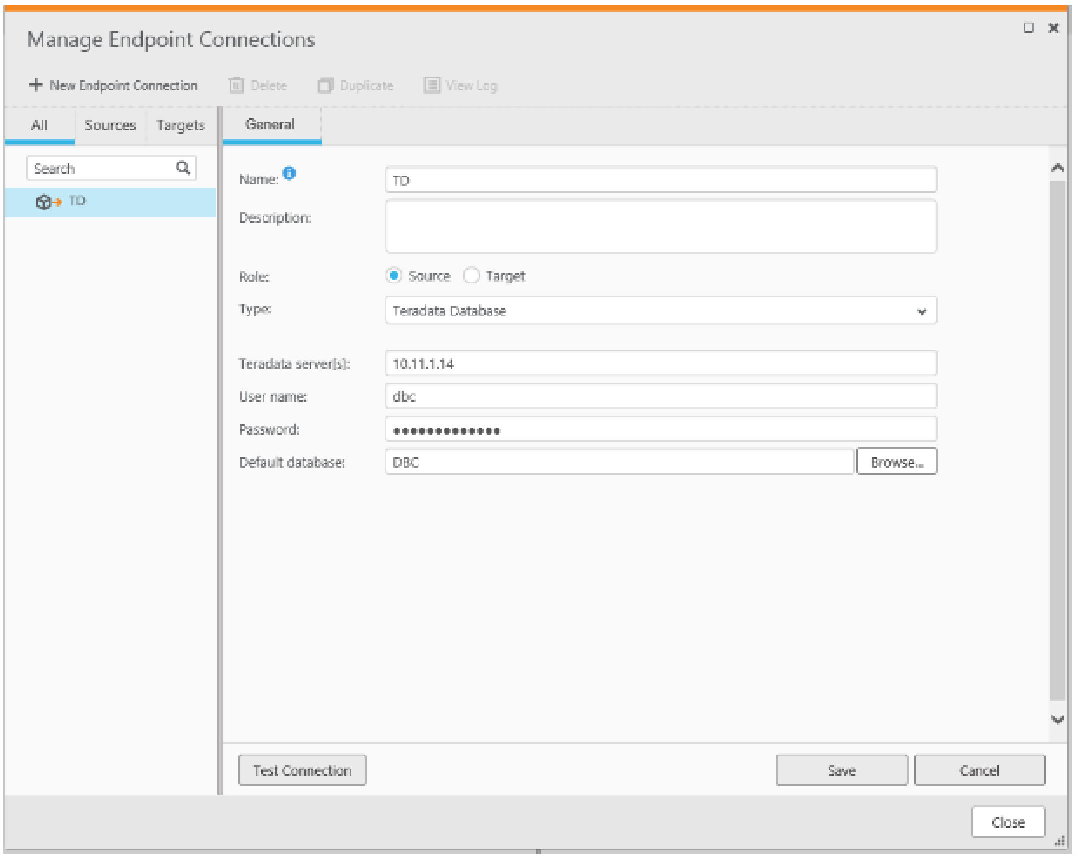
2. Click New Task to open the New Task dialog box.
3. In the Name field, enter either TeradatatoAzureSQL and then click OK.
4. Click the Apply Changes button to disable this task option. When this option is enabled, Replicate for Microsoft Migrations detects any changes to the source tables and automatically applies them to the target tables.
5. Click OK to add the task (the task is displayed in a new tab)

Define your Source and Target Connections

After adding a new task, the next step is to set up your source and target endpoints in Attunity Replicate for Microsoft Migrations. An endpoint is a set of connection parameters that enables Replicate for Microsoft Migrations to extract data from your source and migrate it to your target. Connection parameters specify information such as the database server name, the database name, the user name, and so on.

To add a Teradata Database source endpoint to Attunity Replicate for Microsoft:

1. In the Attunity Replicate for Microsoft Migrations console, click Manage Endpoint Connections to open the Manage Endpoint Connections dialog box.
2. Click New Endpoint Connection.
3. In the Name field, type a name for your Teradata database. This can be any name that will help to identify the database being used.
4. In the Description field, type a description that helps to identify the Teradata database. This is optional.
5. Select SOURCE as the database role.



Name: TD
Description:
Role: ● Source ○ Target
Type: Teradata Database
Teradata server(s): 10.11.1.14
User name: dbc
Password: ••••••••••••••
Default database: DBC

6. Select Teradata Database as the database Type.
7. Type the Server IP address. This is the name of the computer with the Teradata Database instance which you've created in Azure/
8. Type the Teradata Database authentication information (Username, Password) for the authorized user for this Teradata Database. If you do not know this information, see your Teradata Database system manager.
9. Set the default database to the database you wish to migrate.

To add a SQL Data Warehouse Target to Attunity Replicate for Microsoft:

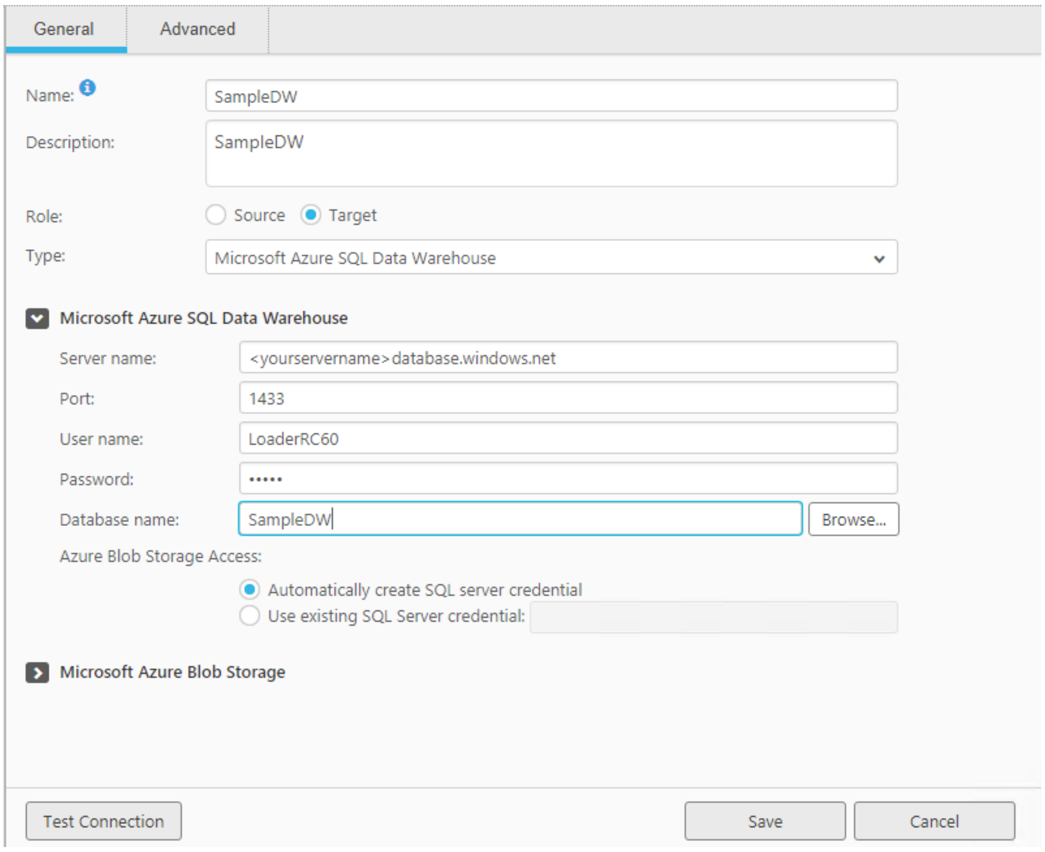
1. In the Attunity Replicate for Microsoft Migrations Console, click Manage Endpoint Connections to open the Manage Endpoints Connections dialog box.
2. In the Manage Endpoint Connections dialog box, click New Endpoint Connection.
3. In the Name field, type a name for your SQL Data Warehouse data warehouse [service]. This can be any name that will help to identify your SQL Data Warehouse database.
4. In the Description field, type a description that helps to identify the SQL Data Warehouse target database. This is optional.
5. Select TARGET as the role.
6. Select SQL Data Warehouse as the Type.
7. Enter the following SQL Data Warehouse information:
8. Server name: Specify the name of the SQL Data Warehouse server you created earlier
   a. Port: Specify port 1433
   b. User name: LoaderRC60

c. Password: Specify the password for the user entered in the LoaderRC60 User.

d. Database name: Specify the target database name.

e. Azure Blob Storage Access: During a migration task, SQL Data Warehouse authenticates itself to Azure Blob Storage using an SQL Server Credential. You can either configure Replicate for Microsoft Migrations to create the Credential automatically during runtime (the default) or use an existing Credential.

f. Select Automatically create SQL Server Credential



Name:	SampleDW
Description:	SampleDW
Role:	Source [✓ Target]
Type:	Microsoft Azure SQL Data Warehouse

Microsoft Azure SQL Data Warehouse

Server name:	<yourservername>database.windows.net
Port:	1433
User name:	LoaderRC60
Password:	•••••
Database name:	SampleDW

Azure Blob Storage Access:

[✓ Automatically create SQL server credential]
Use existing SQL Server credential:

Microsoft Azure Blob Storage

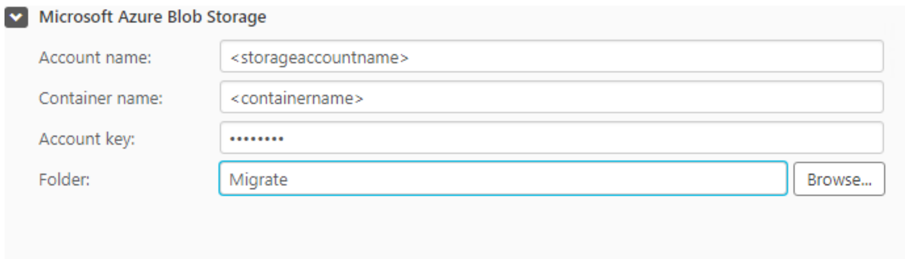
g. Enter the following Microsoft Azure Blob Storage information. You may need to click the Microsoft Azure Blob Storage header to see the information.

h. Account name: Specify the name of the Azure Blob Storage account to which you created earlier.

i. Container name: Specify the name of the Azure Blob Storage container which you created earlier.

j. Account key: Specify the key for your Azure Blob Storage account.

k. Folder: Specify the name of the Azure Blob Storage folder to which you want the files copied.



Microsoft Azure Blob Storage

Account name:	<storageaccountname>
Container name:	<containername>
Account key:	••••••••
Folder:	Migrate

9. In the Advanced tab, set the following properties: Max file size (MB): Change the file size to 2000MB.

Add the Source and Target Endpoints to the Task

After defining your source and target endpoints, the next step is to add them to the migration task.

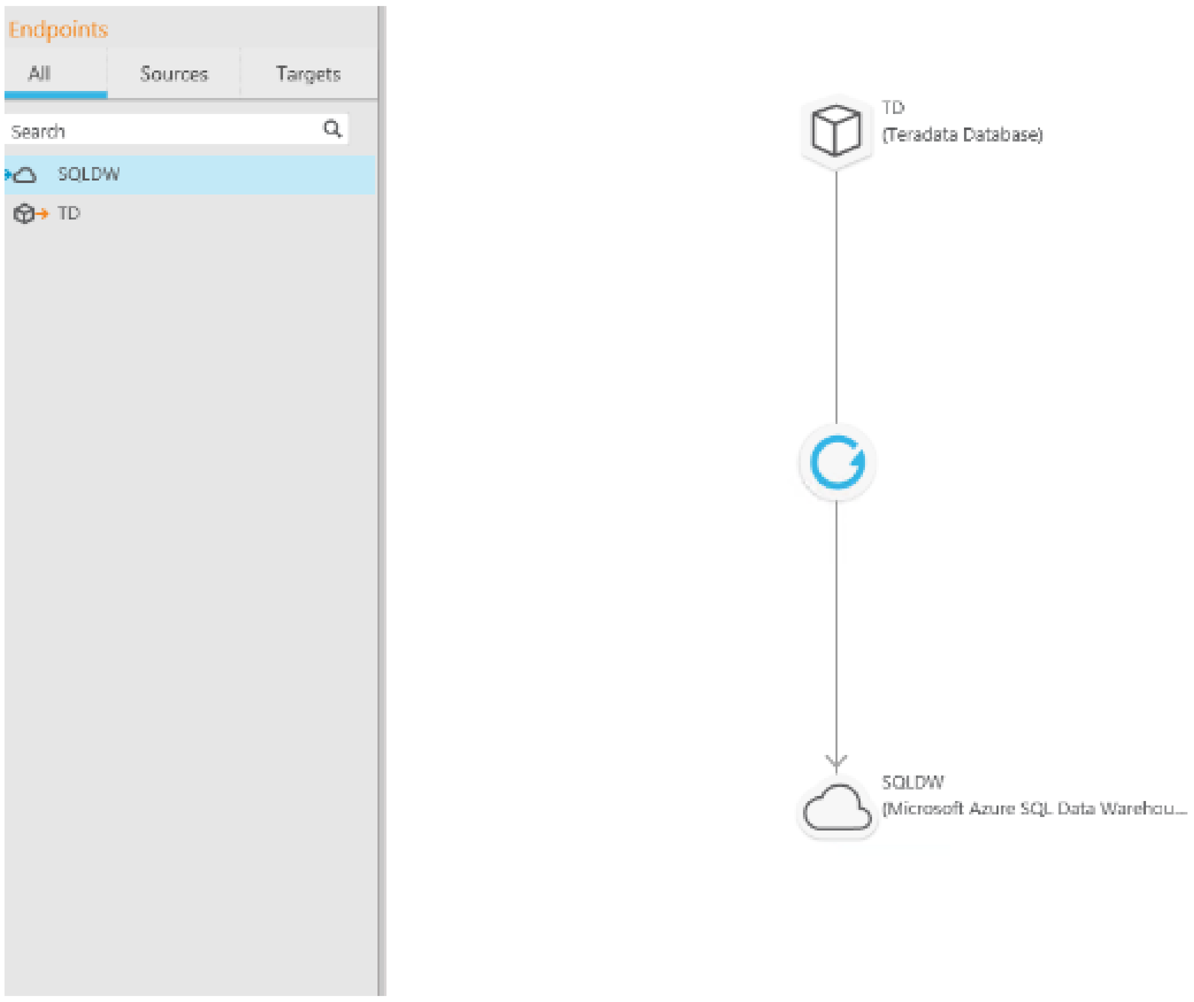
The Endpoints pane is in the left of the console. Replicate for Microsoft Migrations supports several endpoint types (database, Hadoop, data warehouse, file, NoSQL) with each endpoint type having its own icon. As both of our endpoints are databases, the following icons will be displayed:

The source endpoint.

The target endpoint.

To add the source and target endpoints to the task

1. In the Endpoints pane, select the All tab.
2. Add the Teradata source endpoint using any of the following methods:
   - Drag the endpoint to the Drop source endpoint here area in the endpoints diagram to the right of the Endpoints pane.
   - Hover your mouse cursor over the endpoint name and click the arrow that appears to its right.
3. Add the Microsoft SQL Data Warehouse Database target endpoint using any of the following methods:
   - Drag the endpoint to the Drop target endpoint here area in the endpoints diagram to the right of the Endpoints pane.
   - Hover your mouse cursor over the endpoint name and click the arrow that appears to its right.
4. The endpoint diagram should now appear like this:

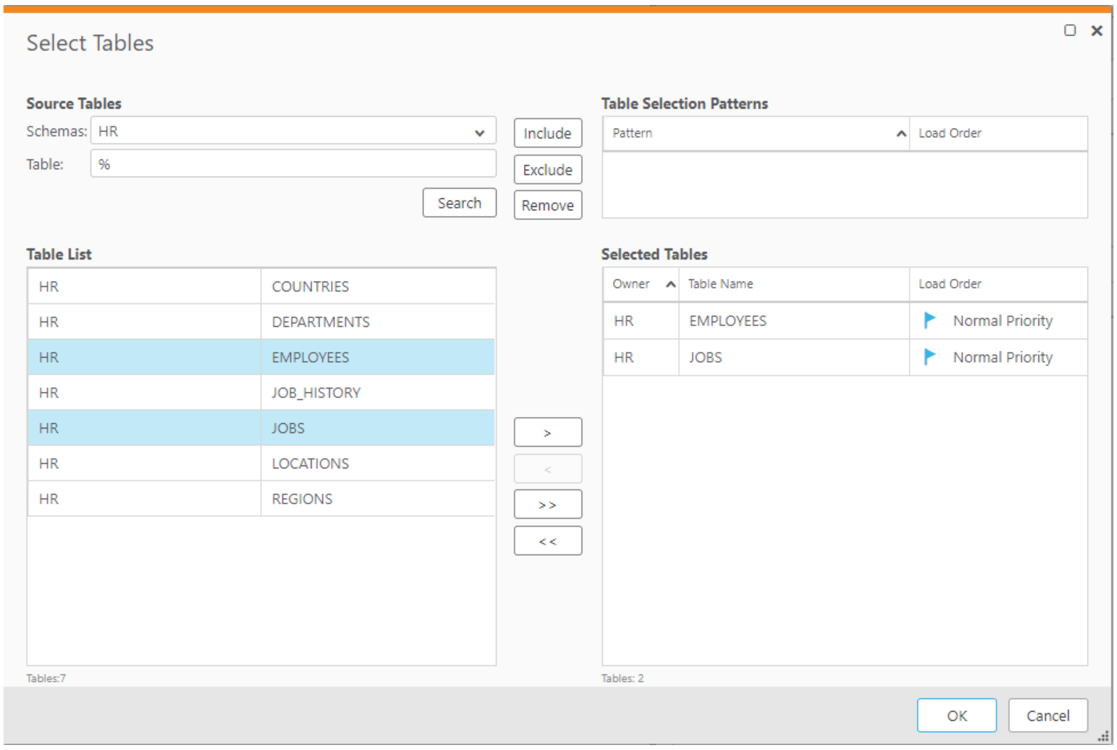


Select Source Tables for Migration

After adding the source and target endpoints, you now need to select the Teradata source tables that you want to migrate to Microsoft Azure SQL Database.

To add Teradata tables to the migration task:

1. In the right pane of the Attunity Replicate for Microsoft Migrations Console, click the
2. Table Selection button.
3. The Select Tables dialog box opens.
4. In the Select Tables dialog box, do the following:
5. From the Schema list, select the schema containing your source tables, and then click Search.
6. From the Table List, either:
   - Hold down the [Ctrl] button and select the tables you want to migrate. Then, click the > button to move the tables to the Selected Tables list.
   - Click the >>> button to move all the tables to the Selected Tables list.


Select Tables

Source Tables

Schemas:	HR
Table:	%

Table Selection Patterns

Pattern	Load Order

Table List

HR	COUNTRIES
HR	DEPARTMENTS
HR	EMPLOYEES
HR	JOB_HISTORY
HR	JOBS
HR	LOCATIONS
HR	REGIONS

Tables:7

Selected Tables

Owner	Table Name	Load Order
HR	EMPLOYEES	Normal Priority
HR	JOBS	Normal Priority

Tables: 2

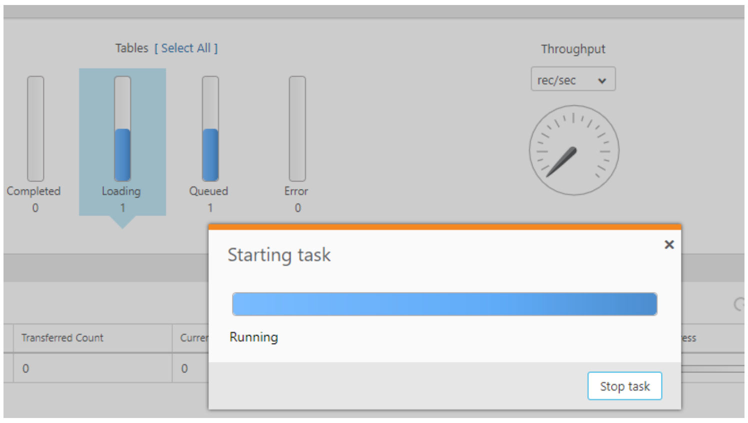
7. Click OK.
8. On the main toolbar, click Save. The task is now ready to run.

Run and Monitor the Migration Task

To run and monitor the replication task

1. On main task toolbar, click Run.

   The Starting task dialog is displayed, and the console automatically switches to Monitor view:



Monitor view contains the following tabs:

– Full Load tab: Uses status bars to provide a visual indication of the loading status during the full load operation.

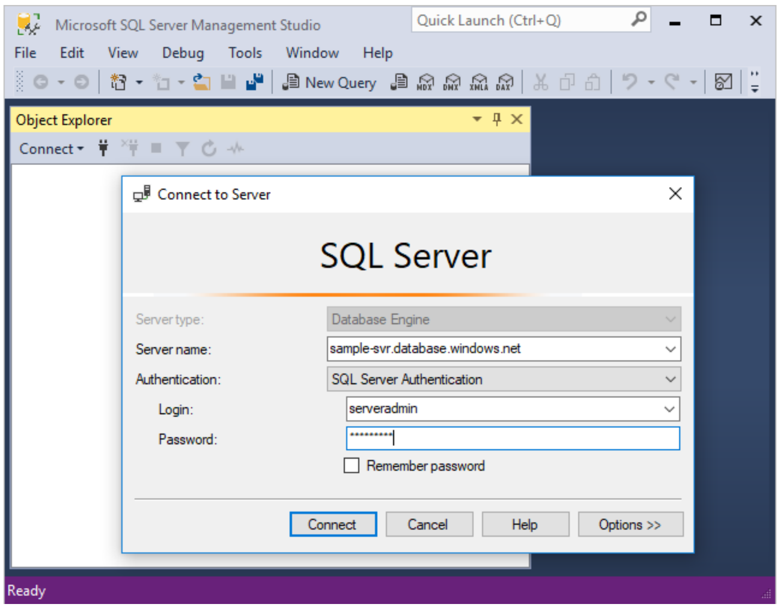
– Change Processing tab: Shows any changes that occur after the full load completes. Since this a Full Load only task, this tab is not relevant.

2. Click the Completed status bar. Replicate for Microsoft Migrations displays a table below the graphs with statistics related to the migrated tables.
3. Click the Total Completion bar to see a summary of the task.

Check Schema Table Migration

After you have run the Attunity Migration you should check the SQL Data Warehouse Database to see the schema which has been transferred.

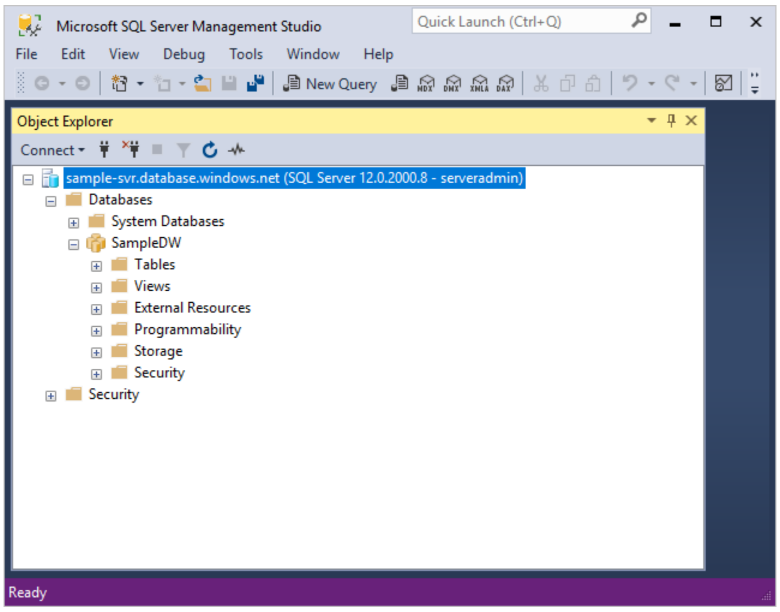
1. Remote Desktop to the Azure Data Science VM you created in the previous step.
2. Open SQL Server Management Studio
3. In the Connect to Server dialog box enter
   a. Server type
   b. Database engine
   c. Server name: The fully qualified server name
   d. Authentication: SQL Server Authentication
   e. Login: The server admin account (This is the account that you specified when you created the server)
   f. Password
4. The password for your server admin account
5. Click Connect. The Object Explorer Windows opens in SSMS.



Server type:	Database Engine
Server name:	sample-svr.database.windows.net
Authentication:	SQL Server Authentication
Login:	serveradmin
Password:	*********

☐ Remember password

6. In Object Explorer, expand Databases. Expand the AZURE SQL DATA WAREHOUSE Database, you created, then tables to view the objects in your new database.



Setup Azure Data Factory for Historical Loads

Azure Data Factory Prerequisites

Download all of the Powershell and SQL Scripts, from the GitHub Repository to "C:\ADF" on the local VM.

Using PowerShell, with the Azure PowerShell module 3.6 or later, open ".\02-ADF\01-Pre-Requisites.ps1" and step through the script.

Sign in to Azure

Sign in to your Azure subscription with the Connect-AzureRmAccount command and follow the on-screen directions.

```
Connect-AzureRmAccount
```

If you don't know which location you want to use, you can list the available locations. After the list is displayed, find the one you want to use. This example uses eastus. Store this in a variable and use the variable so you can change it in one place.

```
Get-AzureRmLocation | select-object Location
$location = "eastus"
```

Create a resource group

Create an Azure resource group with New-AzureRmResourceGroup. A resource group is a logical container into which Azure resources are deployed and managed.

```
$resourceGroup = "myResourceGroup"
New-AzureRmResourceGroup -Name $resourceGroup -Location $location
```

Create a storage account

Create a standard general-purpose storage account with LRS replication using New-AzureRmStorageAccount, then retrieve the storage account context that defines the storage account to be used. When acting on a storage account, you reference the context instead of repeatedly providing the credentials. This example creates a storage account called mystorageaccount with locally redundant storage (LRS) and blob encryption (enabled by default).

```
$storageAccount = New-AzureRmStorageAccount -ResourceGroupName $resourceGroup `
  -Name "mystorageaccount" `
  -Location $location `
  -SkuName Standard_LRS `
  -Kind Storage

$ctx = $storageAccount.Context
```

Create a container

Blobs are always uploaded into a container. You can organize groups of blobs like the way you organize your files on your computer in folders.

Set the container name, then create the container using New-AzureStorageContainer, setting the permissions to 'blob' to allow public access of the files. The container name in this example is import

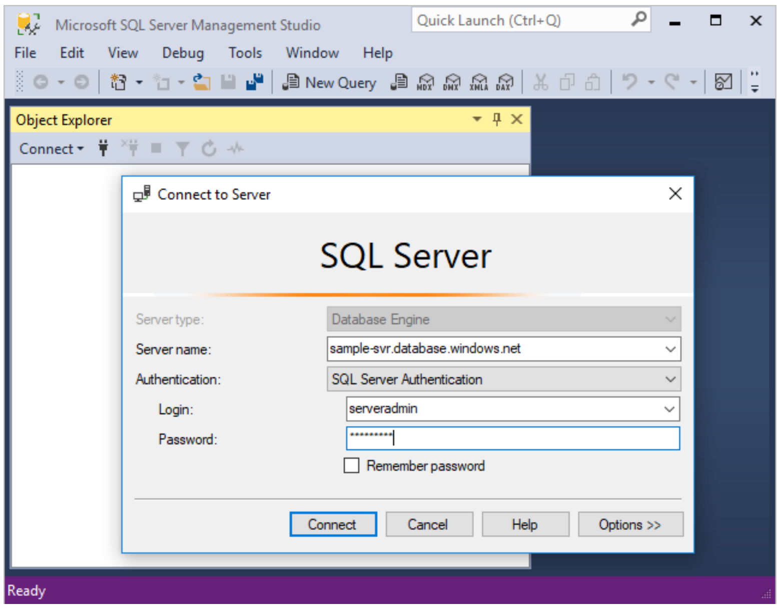
```
$containerName = "import"
New-AzureStorageContainer -Name $containerName -Context $ctx -Permission blob
```

Creating Metadata Tables

To import data from blob storage you need to create external tables which points to the container you created earlier. You cannot create these external tables until you have uploaded data to the blob store but for now we can create the DDL. Creating the DDL before the data is uploaded allows us to then create a local directory structure on the Teradata export machine to copy the files to.

Create and Populate Metadata Tables

1. Remote Desktop to the VM you've created in the Teradata Datacentre.
2. Open SQL Server Management Studio
3. In the Connect to Server dialog box enter
    - Server type
    - Database engine
    - Server name: The fully qualified server name
    - Authentication: SQL Server Authentication
    - Login: The server admin account (This is the account that you specified when you created the server)
    - Password
4. The password for your server admin account
5. Click Connect. The Object Explorer Windows opens in SSMS.



Server type: Database Engine
Server name: sample-svr.database.windows.net
Authentication: SQL Server Authentication
Login: serveradmin
Password: *********
Remember password

6. In Object Explorer, expand Databases. Right click on the database you created earlier and select new query.
7. In the query window, enter this T-SQL commands

```
CREATE TABLE [dbo].[TablesToProcess]
(
	[schemaname] [varchar](255) NULL,
	[tablename] [varchar](255) NULL
)
WITH
(
	DISTRIBUTION = ROUND_ROBIN,
	CLUSTERED COLUMNSTORE INDEX
)
GO
INSERT INTO TablesToProcess (schemaname, tablename)
    select top 30000 sc.name, so.name
    from sys.tables so
    join sys.schemas sc on so.schema_id = sc.schema_id
    left join sys.external_tables et on so.object_id = et.object_id
    where et.name is NULL and so.type = 'U' order by so.name
GO
```

8. Click Execute
9. Close all applications and log off.

Create a data factory

With the metadata tables created we are going to create a new data factory for your historical migration.

1. Start PowerShell on your machine, and keep it open through completion of this section. If you close and reopen it, you'll need to run these commands again.
   a. Run the following command, and enter the user name and password that you use to sign in to the Azure portal:

```
Connect-AzureRmAccount
```

   b. Run the following command to view all the subscriptions for this account:

```
Get-AzureRmSubscription
```

   c. Run the following command to select the subscription that you want to work with. Replace SubscriptionId with the ID of your Azure subscription:

```
Select-AzureRmSubscription -SubscriptionId "<SubscriptionId>"
```

2. Run the .\02-ADF\04-Create-ADF.ps1 script to create a data factory.
   a. Enter a name for your data factory (this must be unique)
   b. Select an Azure Datacenter Region and click ok.
   c. Enter a Resource Group Name (this can be a new or existing one)

Note the following points:

   a. The name of the Azure data factory must be globally unique. If you receive the following error, change the name and try again.
   b. To create Data Factory instances, you must be a Contributor or Administrator of the Azure subscription.

Create a self-hosted runtime

To enable the VM to export data from Teradata to local storage we need to create you a self-hosted integration runtime and associate it with an on-premises machine with the Teradata database and local storage. The self-hosted integration runtime is the component that copies data from the Teradata database on your machine to local storage.

1. Run the .\02-ADF\ 05aSelfHostedRuntime.ps1 script to create an integrated runtime.
   a. Select the Azure Resource Group you created or used in the previous step
   b. Select the Azure Data Factory you created in the previous step.
   c. Enter a name for your integration runtime (this must be unique)

Here is the sample output:

```
Id                : /subscriptions/<subscription
ID>/resourceGroups/ADFTutorialResourceGroup/providers/Microsoft.DataFactory/factori
es/onpremdf0914/integrationruntimes/myonpremirsp0914
Type              : SelfHosted
ResourceGroupName : ADFTutorialResourceGroup
DataFactoryName   : onpremdf0914
Name              : myonpremirsp0914
Description       : selfhosted IR description
```

2. To retrieve the status of the created integration runtime, run the .\02-ADF\ 05bSelfHostedRuntime.ps1 script.

Here is the sample output:

```
Nodes                       : {}
CreateTime                  : 9/14/2017 10:01:21 AM
InternalChannelEncryption :
Version                     :
Capabilities                : {}
ScheduledUpdateDate         :
UpdateDelayOffset           :
LocalTimeZoneOffset         :
AutoUpdate                  :
ServiceUrls                 : {eu.frontend.clouddatahub.net,
*.servicebus.windows.net}
ResourceGroupName           : <ResourceGroup name>
DataFactoryName             : <DataFactory name>
Name                        : <Integration Runtime name>
State                       : NeedRegistration
```

3. To retrieve the authentication keys for registering the self-hosted integration runtime with the Data Factory service in the cloud, run the following command. Copy one of the keys (excluding the quotation marks) for registering the self-hosted integration runtime that you install on your machine in the next step.

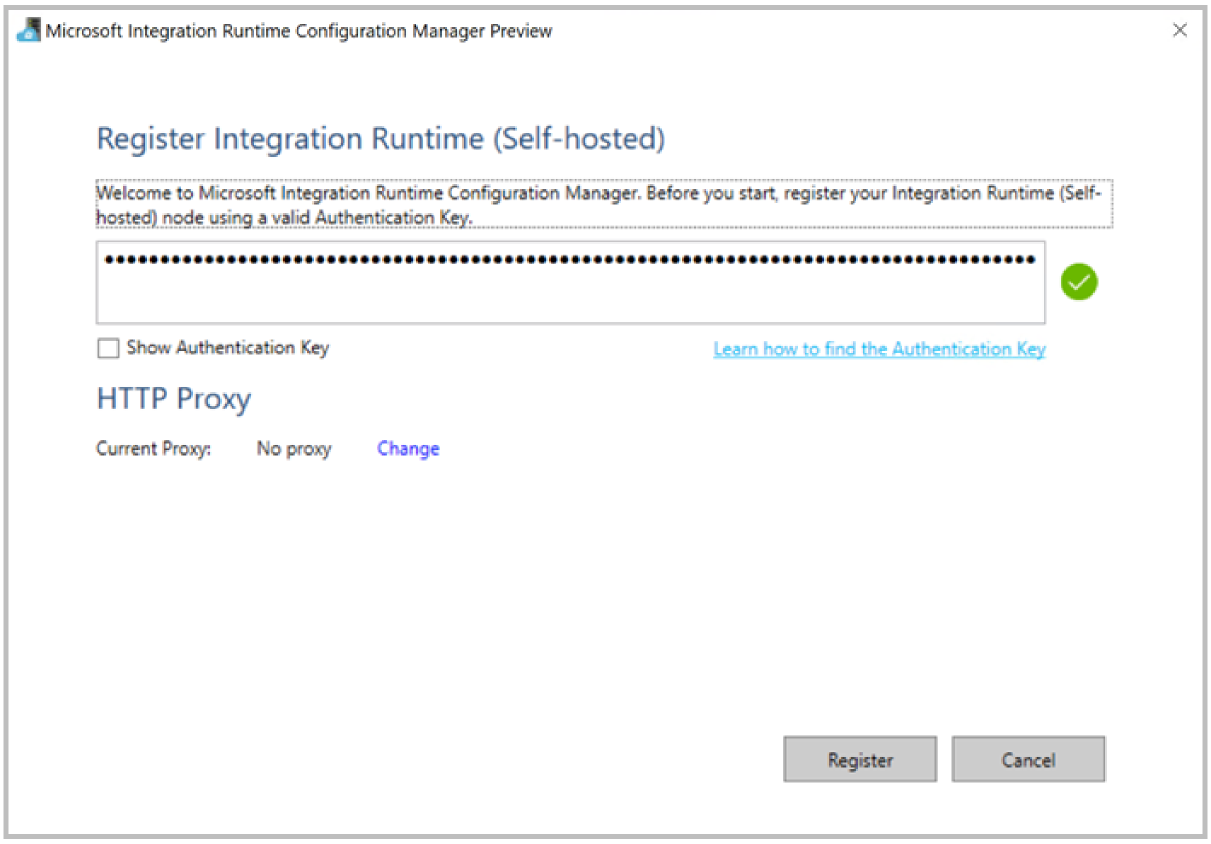
```
Get-AzureRmDataFactoryV2IntegrationRuntimeKey `
    -Name $integrationRuntimeName `
    -DataFactoryName $dataFactoryName.DataFactoryName `
    -ResourceGroupName $resourceGroupName.ResourceGroupName | `
    ConvertTo-Json
```

Here is the sample output:

```
{
    "AuthKey1":  "IR@0000000000-0000-0000-0000-
000000000000@xy0@xy@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx=",
    "AuthKey2":  "IR@0000000000-0000-0000-0000-
000000000000@xy0@xy@yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy="
}
```

Install the Integration Runtime on the Virtual Machine

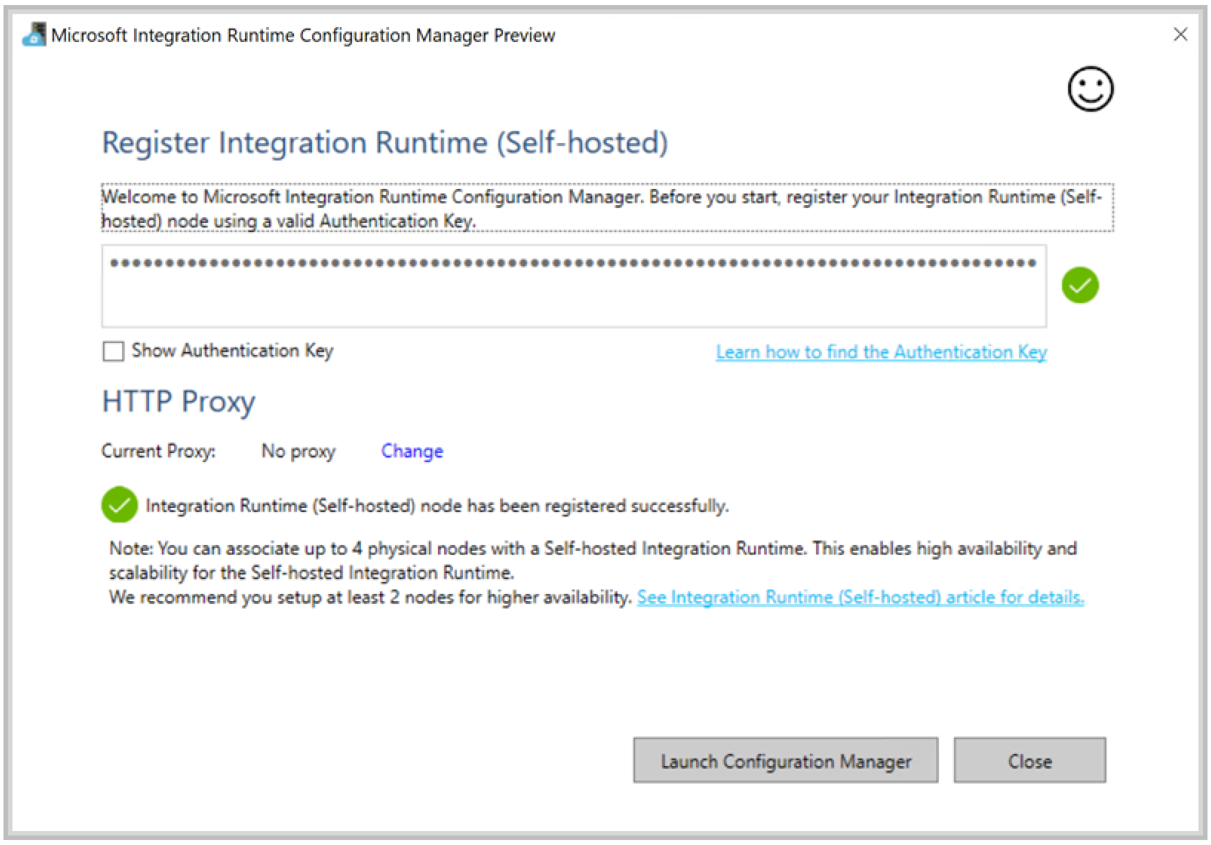
1. Download Azure Data Factory Integration Runtime on a local Windows machine, and then run the installation.
2. In the Welcome to Microsoft Integration Runtime Setup wizard, select Next.
3. In the End-User License Agreement window, accept the terms and license agreement, and select Next.
4. In the Destination Folder window, select Next.
5. In the Ready to install Microsoft Integration Runtime window, select Install.
6. If you see a warning message about the computer being configured to enter sleep or hibernate mode when not in use, select OK.
7. If a Power Options window is displayed, close it, and switch to the setup window.
8. In the Completed the Microsoft Integration Runtime Setup wizard, select Finish.
9. In the Register Integration Runtime (Self-hosted) window, paste the key you saved in the previous section, and then select Register.


Microsoft Integration Runtime Configuration Manager Preview

Register Integration Runtime (Self-hosted)

Welcome to Microsoft Integration Runtime Configuration Manager. Before you start, register your Integration Runtime (Self-hosted) node using a valid Authentication Key.

When the self-hosted integration runtime is registered successfully, the following message is displayed:


Microsoft Integration Runtime Configuration Manager Preview

Register Integration Runtime (Self-hosted)

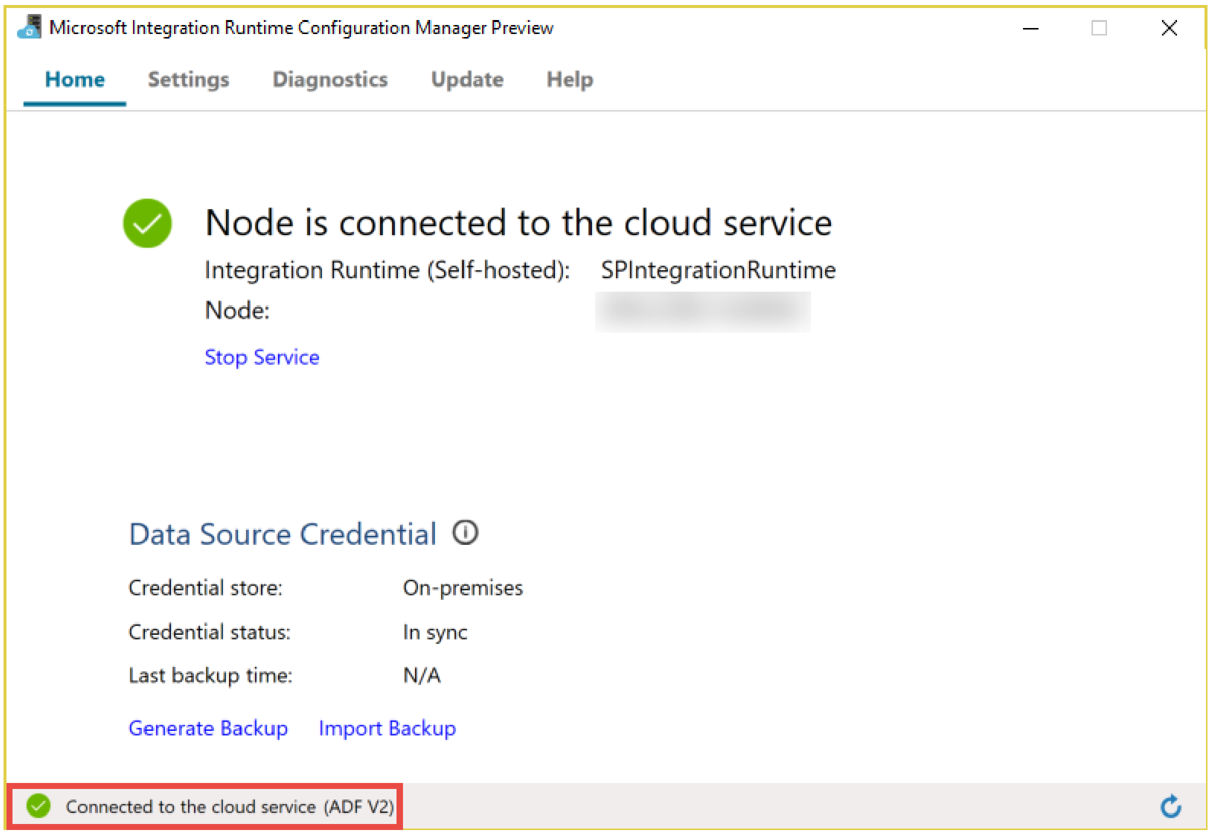
Welcome to Microsoft Integration Runtime Configuration Manager. Before you start, register your Integration Runtime (Self-hosted) node using a valid Authentication Key.

Integration Runtime (Self-hosted) node has been registered successfully.

Note: You can associate up to 4 physical nodes with a Self-hosted Integration Runtime. This enables high availability and scalability for the Self-hosted Integration Runtime.
We recommend you setup at least 2 nodes for higher availability. See Integration Runtime (Self-hosted) article for details.

10. In the New Integration Runtime (Self-hosted) Node window, select Next.

11. In the Register Integration Runtime (Self-hosted) window, select Launch Configuration Manager.
12. When the node is connected to the cloud service, the following message is displayed:



Node is connected to the cloud service

Integration Runtime (Self-hosted): SPIntegrationRuntime

Node: [redacted]

Data Source Credential

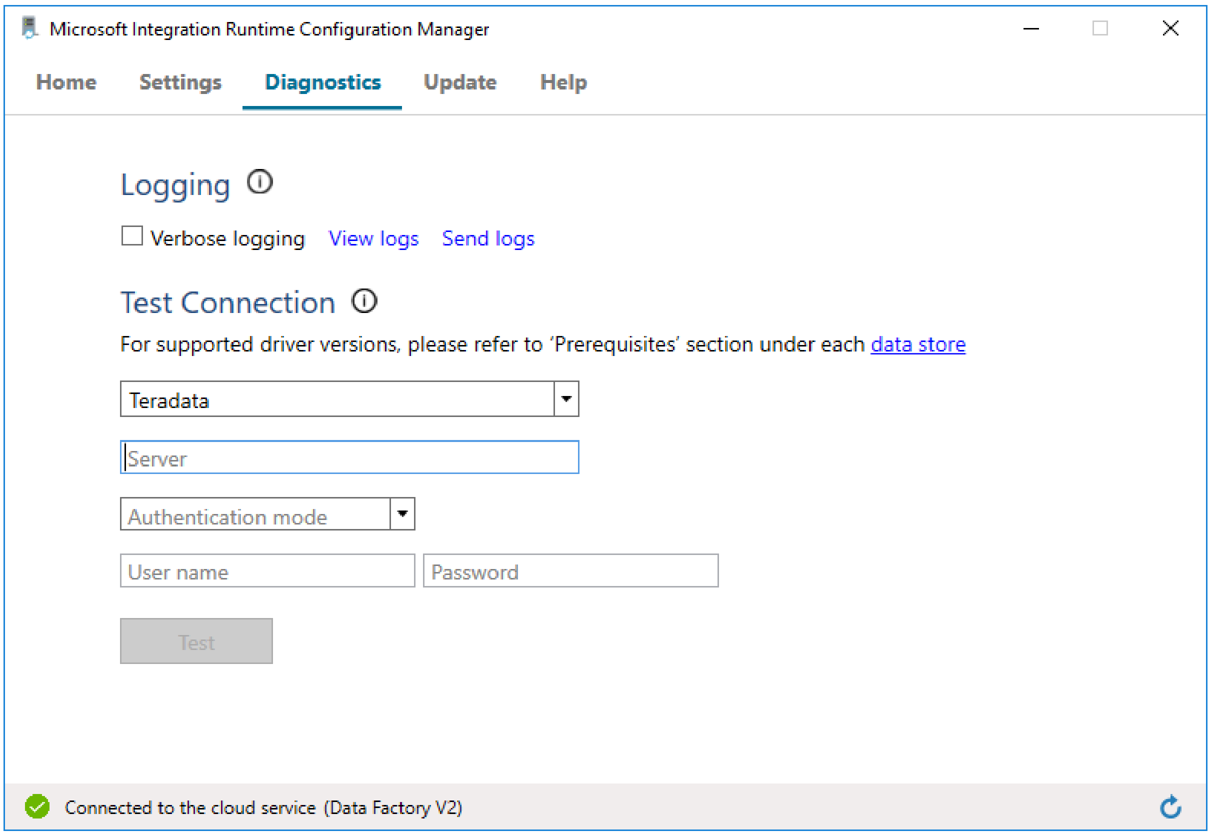
Credential store: On-premises

Credential status: In sync

Last backup time: N/A

Connected to the cloud service (ADF V2)

13. In the Intranet Communication Channel window, select Skip.
14. Test the connectivity to your Teradata database by doing the following:



Test Connection

For supported driver versions, please refer to 'Prerequisites' section under each data store

Connected to the cloud service (Data Factory V2)

a. a. In the Configuration Manager window, switch to the Diagnostics tab.
b. b. In the Data source type box, select Teradata.
c. Enter the server name.
d. Select the authentication mode.
e. Enter the username.
f. Enter the password that's associated with the username.
g. To confirm that integration runtime can connect to the SQL Server, select Test. If the connection is successful, a green checkmark icon is displayed. Otherwise, you'll receive an error message associated with the failure. Fix any issues and ensure that the integration runtime can connect to your Teradata instance.

Note all the preceding values for later use.

Create linked services

To link your data stores and compute services to the data factory create linked services in the data factory. You will need to create links to Teradata, local storage, your Azure storage account and Azure SQL Data Warehouse. The linked services have the connection information that the Data Factory service uses at runtime to connect to them.

Create and encrypt a Teradata linked service (source)

In this step, you link your on-premises Teradata instance to the data factory.

1. Open the JSON file named TeradataLinkedService.json in the C:\ADF\Connectors folder by using the following code:
    a. Replace <integration runtime name> with the name of your integration runtime.
    b. Before you save the file, replace <servername>, <username>, and <password> with the values of your Teradata instance.
    c. If you need to use a backslash (\) in your user account or server name, precede it with the escape character (\). For example, use mydomain\\myuser.

Create and encrypt a local storage linked service (source)

In this step, you link your on-premises Teradata instance to the data factory.

1. Open the JSON file named FileLinkedService.json in the C:\ADF\Connectors folder by using the following code:
    a. Replace <server> and <ShareName> with the values of your file path either fileshare or x:\ExportFiles
    b. Before you save the file, replace <Domain>, <UserName>, and <password> with the values of user which has permission to write files to the local storage.
    c. If you need to use a backslash (\) in your user account or server name, precede it with the escape character (\). For example, use mydomain\\myuser.
    d. Replace <accountName> and <accountKey> with name and key of your Azure storage account before saving the file.

Create the Azure Storage linked service

1. Open the JSON file named AzureStorageLinkedService.json in the C:\ADF\Connectors folder by using the following code:

   a. Replace <accountName> and <accountKey> with name and key of your Azure storage account before saving the file.

Creating the Azure SQL Data Warehouse linked service

1. Open the JSON file named AzureStorageLinkedService.json in the C:\ADF\Connectors folder by using the following code:
   a. Replace <servername>, <databasename>, <username> and <password> with values of your Azure SQL Database before saving the file. You should use the user, with the largerc, you created earlier.

Create All Linked Services

1. Run the .\02-ADF\ 06-CreateLinkedServers.ps1 script to create an all of the linked services
   a. Select the Azure Resource Group you created or used in the previous step.
   b. Select the Azure Data Factory you created in the previous step.
   c. Select the Self Hosted Integration Runtime created in the previous step.

Linked Services Notes

The file share should be on the local virtual machine as opposed pushing out file copies from the VM to another location.

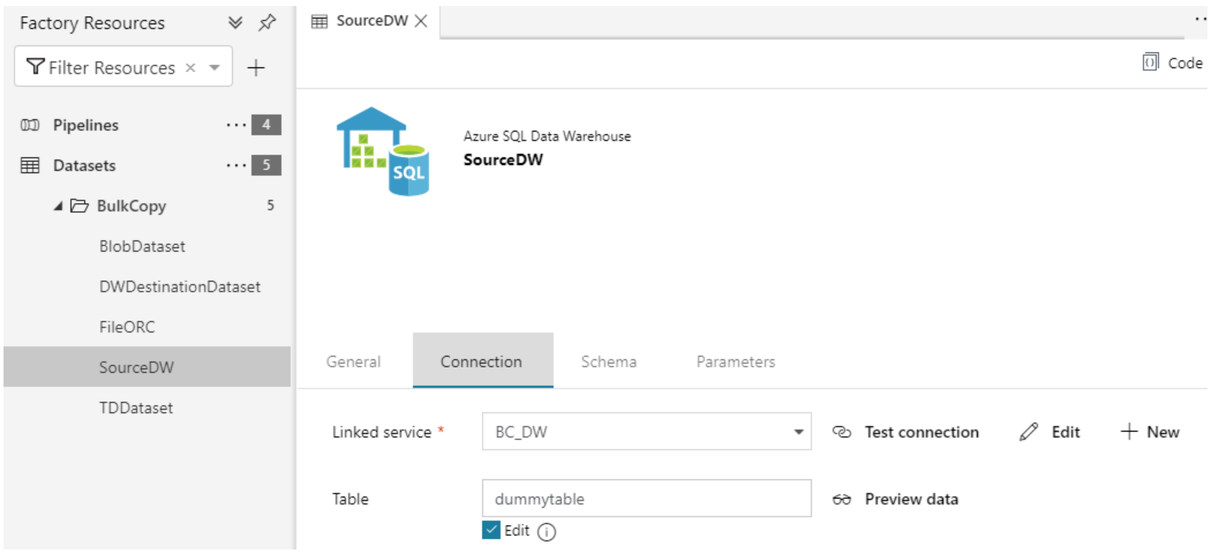
Create Datasets

1. Run the .\02-ADF\07-CreateDatasets.ps1 script to create an all of the linked services
   a. Select the Azure Resource Group you created or used in the previous step.
   b. Select the Azure Data Factory you created in the previous step.

This will create all the datasets required for the four pipelines to use.

Dataset Notes

In SourceDW.json, in the C:\ADFv2\02-ADF\DataSets folder, “tableName" is a dummy one which the pipeline will use the SQL query in copy activity to retrieve data. This can be checked or changed in the Azure Data Factory User Interface SourceDW dataset.

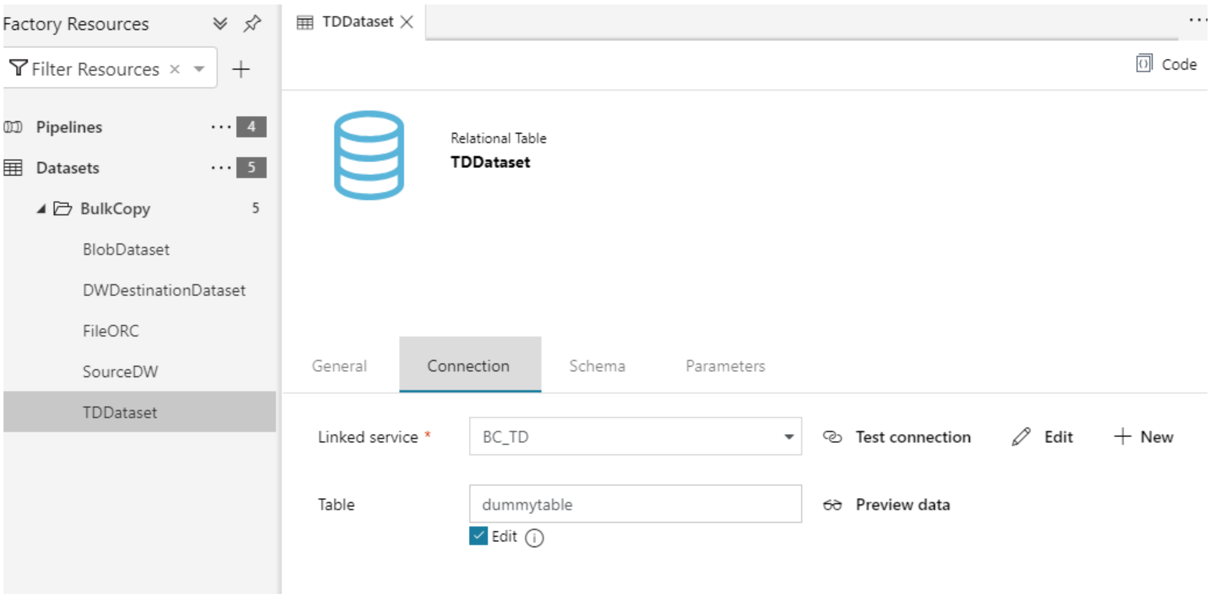


Azure SQL Data Warehouse
SourceDW

Linked service * BC_DW
Table dummytable
☑ Edit

Figure 48: SourceDW Connection Properties

In TDDataset.json, in the C:\ADFv2\02-ADF\DataSets folder, “tableName" is a dummy one which the pipeline will use the SQL query in copy activity to retrieve data. This can be checked or changed in the Azure Data Factory User Interface TDDataset dataset.

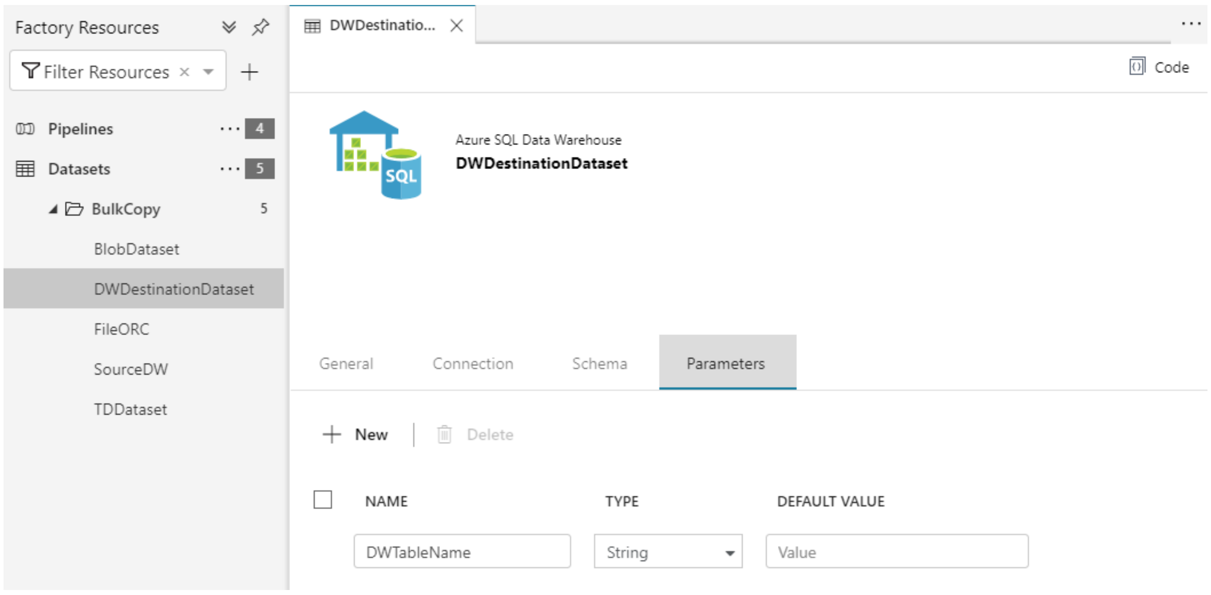


Relational Table
TDDataset

Linked service * BC_TD
Table dummytable
☑ Edit

Figure 49: TDDataset Connection Properties

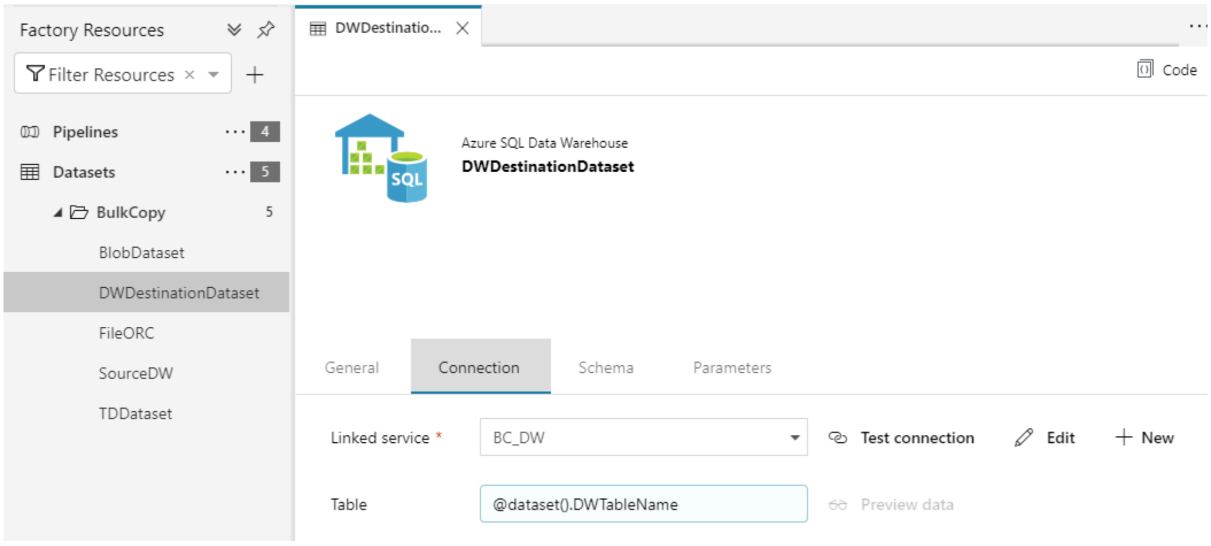
In DWDestinationDataset.json, in the C:\ADFv2\02-ADF\DataSets folder the "tableName" is set as a parameter. The copy activity that references this dataset dynamically passes the actual value into the dataset. This can be checked or changed in the Azure Data Factory User Interface DWDestinationDataset dataset.



NAME	TYPE	DEFAULT VALUE
DWTableName	String	Value

Figure 50: DWDestinationDataset Parameters

These are dynamically referenced in table name.

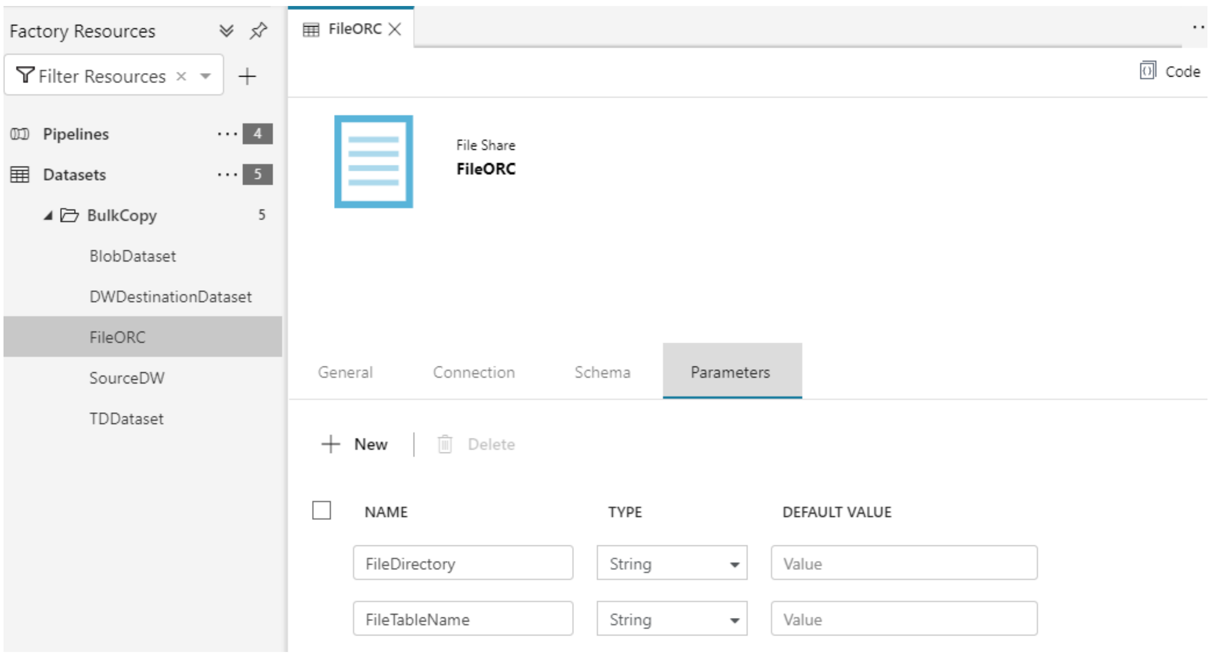


Linked service * BC_DW

Table @dataset().DWTableName

Figure 51: DWDestinationDataset Connection Properties

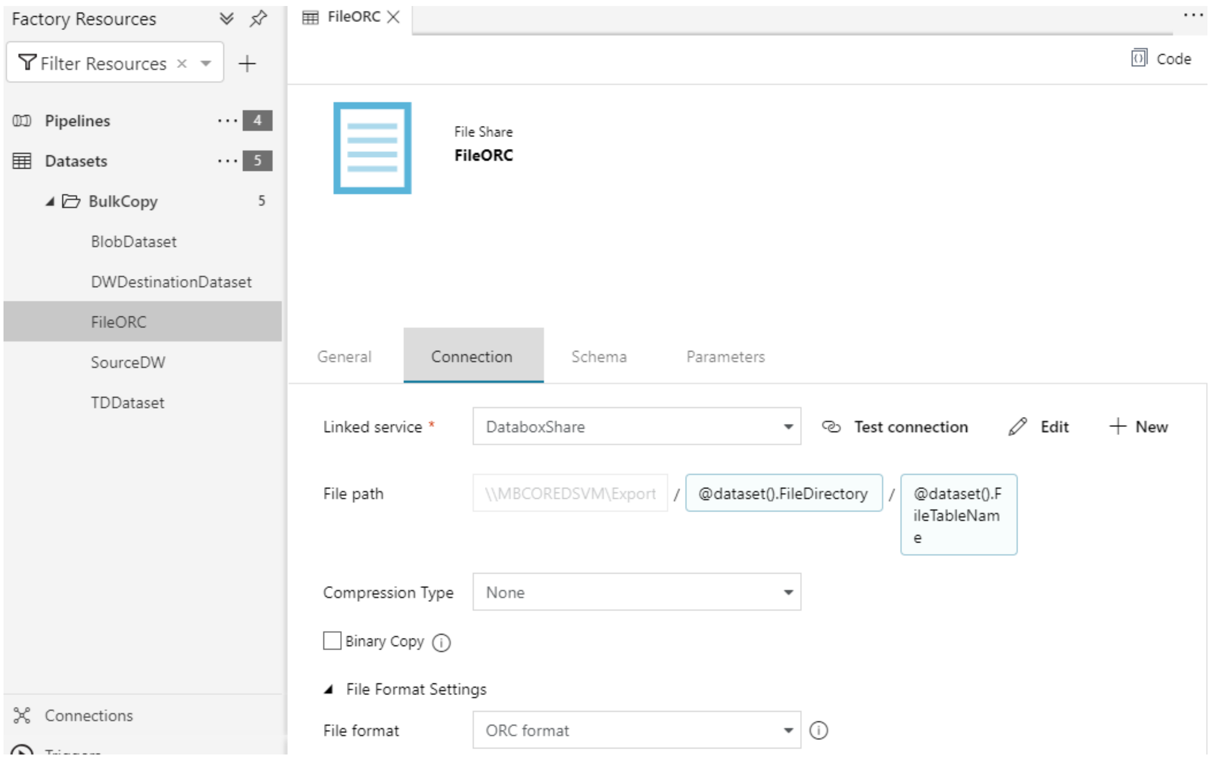
In FileORC.json, in the C:\ADFv2\02-ADF\DataSets folder the "FileDirectory" and “FileTableName” are set as a parameter. The copy activity that references this dataset passes the actual directory and filename into the dataset. This can be checked or changed in the Azure Data Factory User Interface FileORC dataset.



NAME	TYPE	DEFAULT VALUE
FileDirectory	String	Value
FileTableName	String	Value

Figure 52: FIleORC Parameters

These are dynamically referenced in the file path and the file format is set to ORC format.



Linked service * DataboxShare

File path \\MBCOREDSVM\Export / @dataset().FileDirectory / @dataset().FileTableName

Compression Type None

File format ORC format

Figure 53: FileOrc Connection Properties

Create Pipelines

1. Run the .\02-ADF\ 08-CreatePipelines.ps1 script to create an all of the linked services
   a. Select the Azure Resource Group you created or used in the previous step.

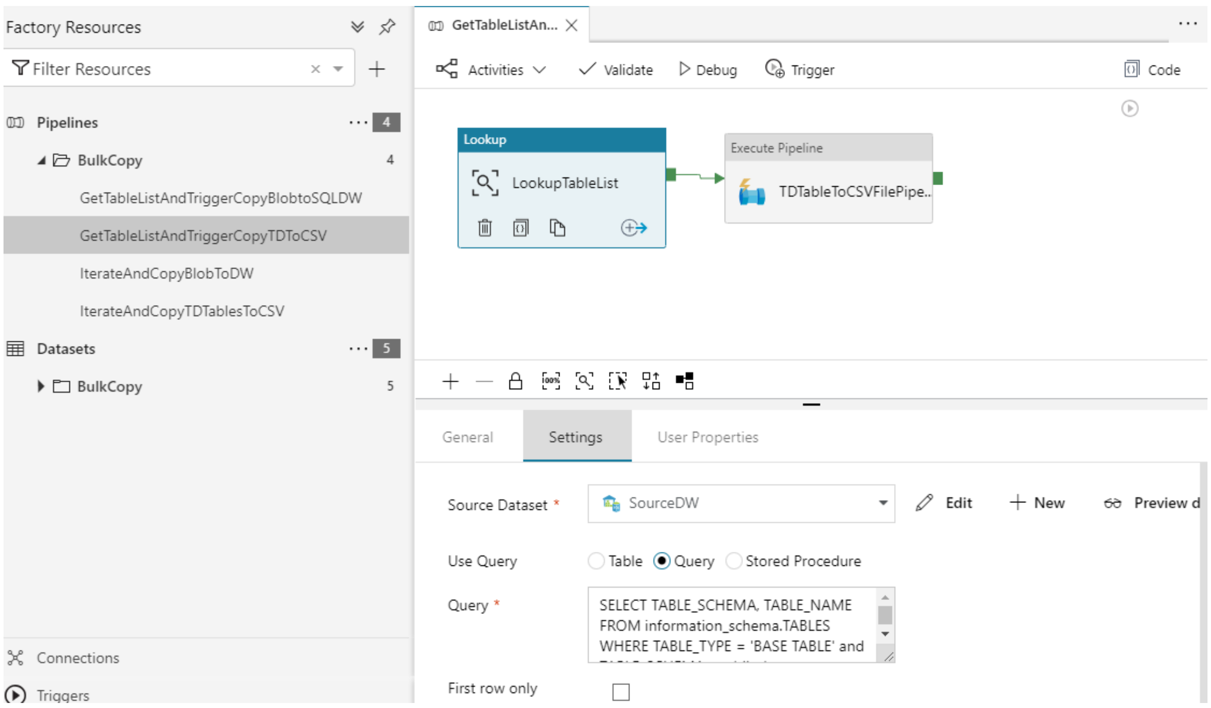
b. Select the Azure Data Factory you created in the previous step.

This will create all four pipelines to use for migration (GetTableListAndTriggerCopyTDToCSV, IterateAndCopyTDTablesToCSV, GetTableListAndTriggerCopyBlobtoSQLDW and IterateAndCopyTDTablesToCSV)

Pipeline Notes

Teradata to Local Storage

The GetTableListAndTriggerCopyTDToCSV pipeline looks up the Azure SQL Data Warehouse system table to get the list of tables to be copied from Teradata to the local storage. This can be checked or changed in the Azure Data Factory User Interface GetTableListAndTriggerCopyTDToCSV pipeline by selecting the LookupTableList activity.



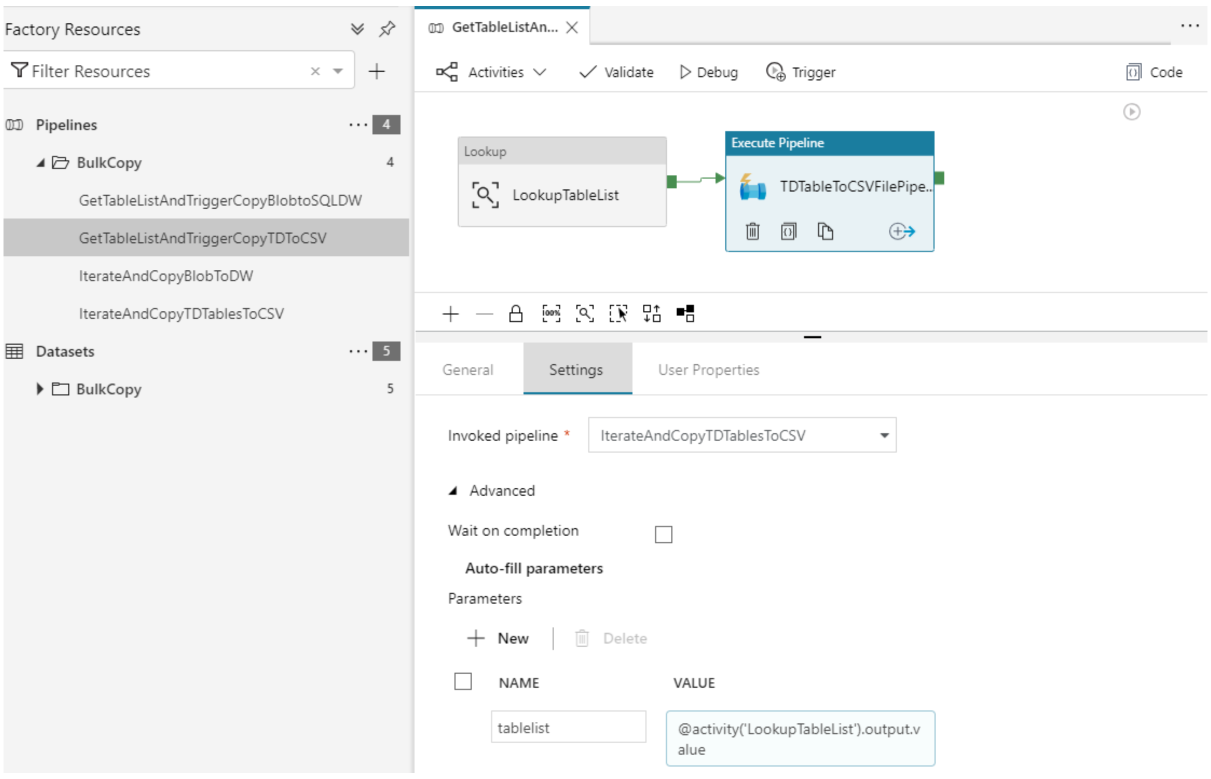
Source Dataset * SourceDW

Use Query: Table [✓ Query] Stored Procedure

Query *
```
SELECT TABLE_SCHEMA, TABLE_NAME
FROM information_schema.TABLES
WHERE TABLE_TYPE = 'BASE TABLE' and
```

First row only ☐

Selecting the TDTableToCSVFilePipeline activity and setting will show how we pass the output of LookupTableList to the IterateAndCopyTDTablesToCSV pipeline.


Factory Resources

Pipelines 4
BulkCopy 4
GetTableListAndTriggerCopyBlobtoSQLDW
GetTableListAndTriggerCopyTDToCSV
IterateAndCopyBlobToDW
IterateAndCopyTDTablesToCSV
Datasets 5
BulkCopy 5

GetTableListAn...
Activities | Validate | Debug | Trigger | Code

Lookup: LookupTableList → Execute Pipeline: TDTableToCSVFilePipe...

General | Settings | User Properties

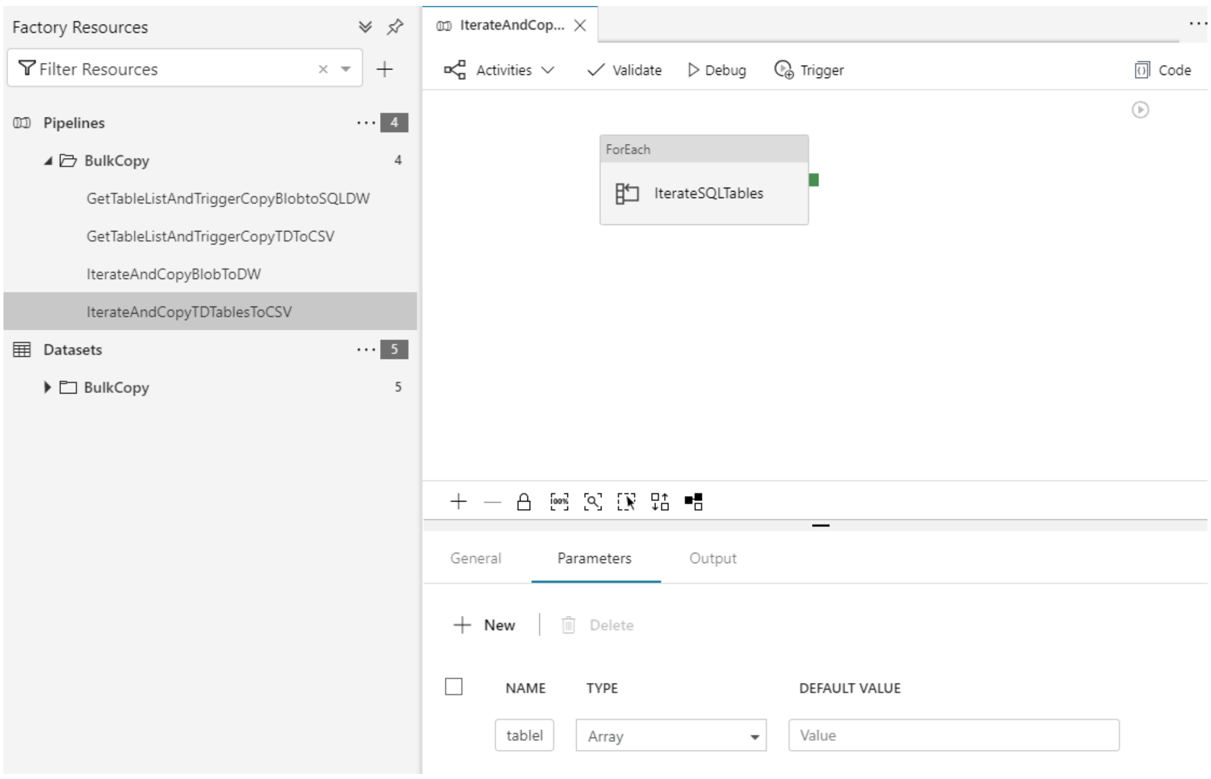
Invoked pipeline *: IterateAndCopyTDTablesToCSV

Advanced
Wait on completion: ☐
Auto-fill parameters
Parameters

NAME	VALUE
tablelist	@activity('LookupTableList').output.value

GetTableListAndTriggerCopyTDToCSV pipeline triggers the pipeline " IterateAndCopyTDTablesToCSV" to do the actual data copy.

The output of the lookup activity is passed into an array parameter which is used by the ForEach loop.


Factory Resources

Pipelines 4
BulkCopy 4
GetTableListAndTriggerCopyBlobtoSQLDW
GetTableListAndTriggerCopyTDToCSV
IterateAndCopyBlobToDW
IterateAndCopyTDTablesToCSV
Datasets 5
BulkCopy 5

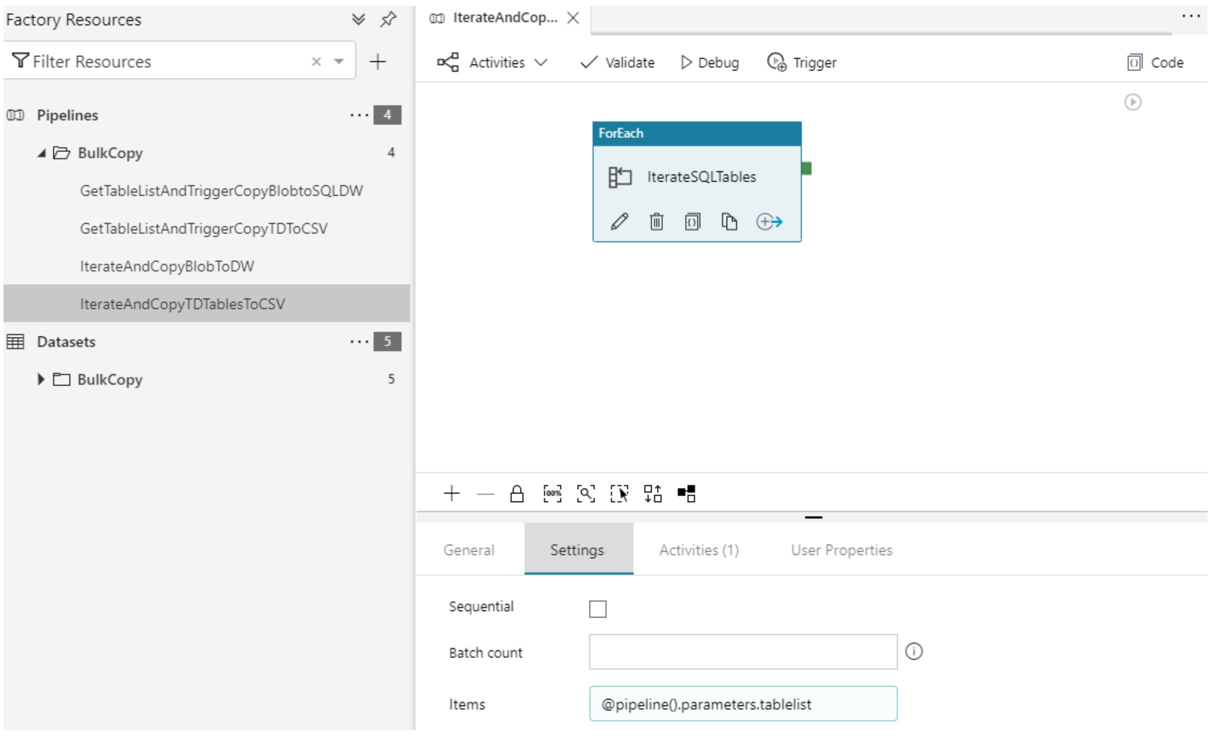
IterateAndCop...
Activities | Validate | Debug | Trigger | Code

ForEach: IterateSQLTables

General | Parameters | Output

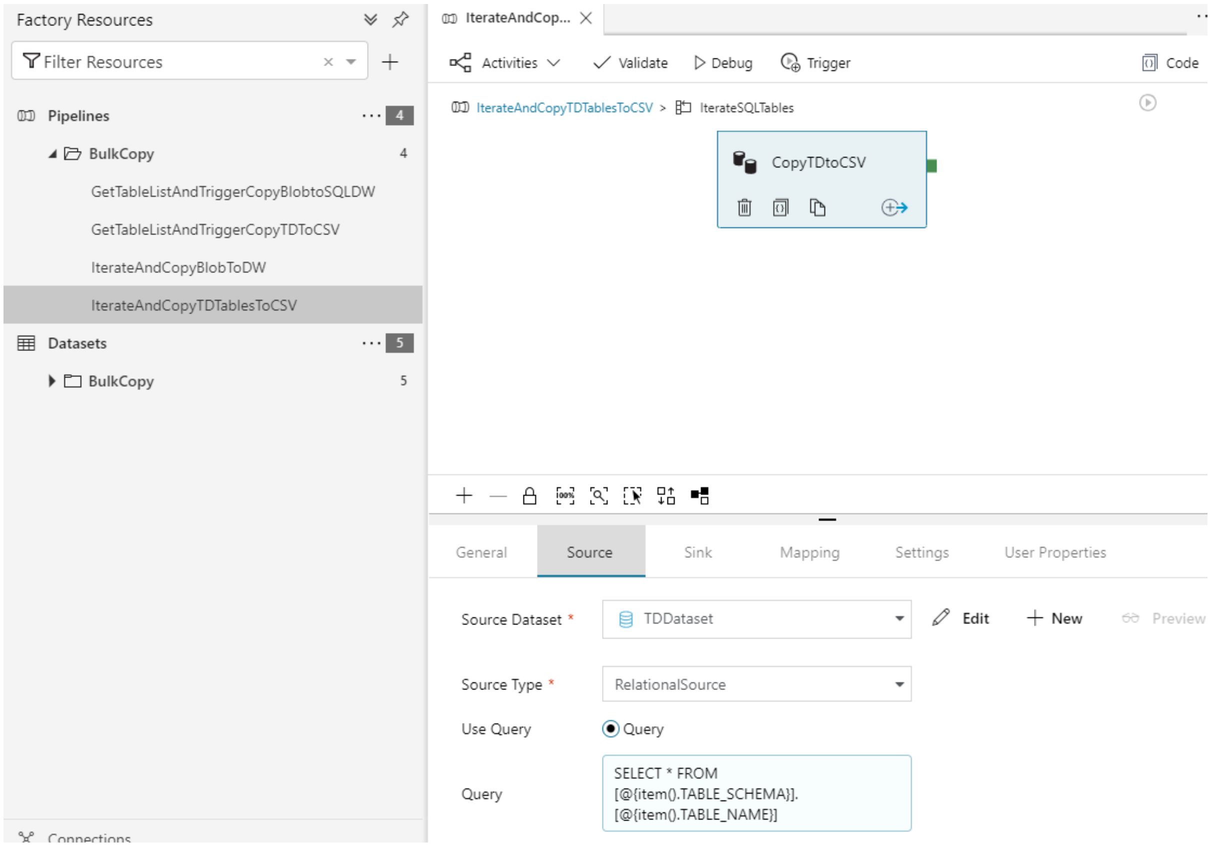
NAME	TYPE	DEFAULT VALUE
tablel	Array	Value

By selecting the ForEach activity you will see that this array to passed to another parameter associated with the activity.



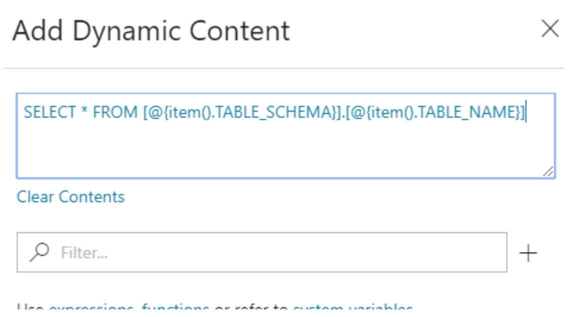
Sequential: ☐
Batch count:
Items: @pipeline().parameters.tablelist

Editing the activities for IterateAndCopyTDTablesToCSV and selecting CopyTDtoCSV you can investigate how Teradata data is copied to local storage using a dynamic query.



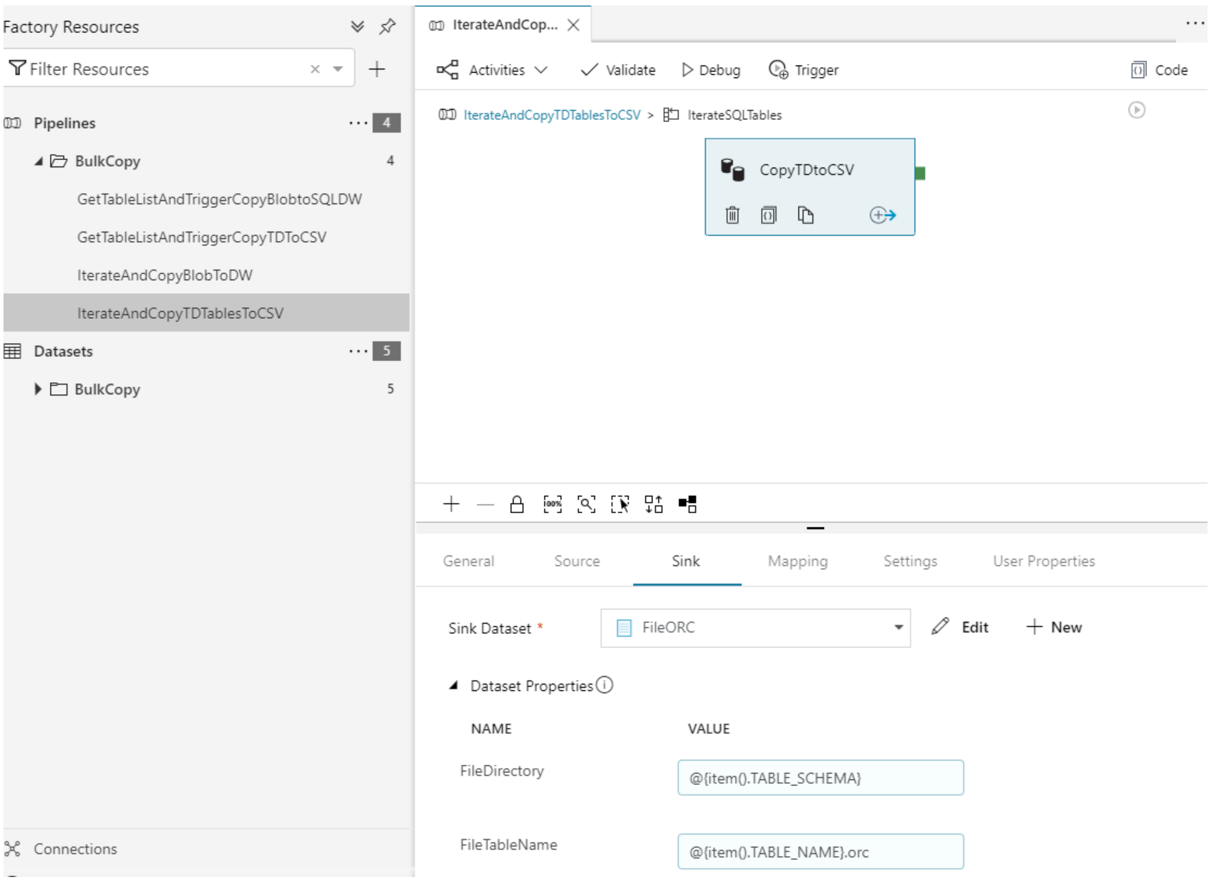
Source Dataset *: TDDataset
Source Type *: RelationalSource
Use Query: ● Query
Query:

```
SELECT * FROM
[@{item().TABLE_SCHEMA}].
[@{item().TABLE_NAME}]
```


Add Dynamic Content

```
SELECT * FROM [@{item().TABLE_SCHEMA}].[@{item().TABLE_NAME}]
```

The local storage is passed a directory and filename based on the schema and tablename. This example is a basic extract process. You will know your data and we highly recommend that modify this query to partition the data export based on the advanced metadata approach above.

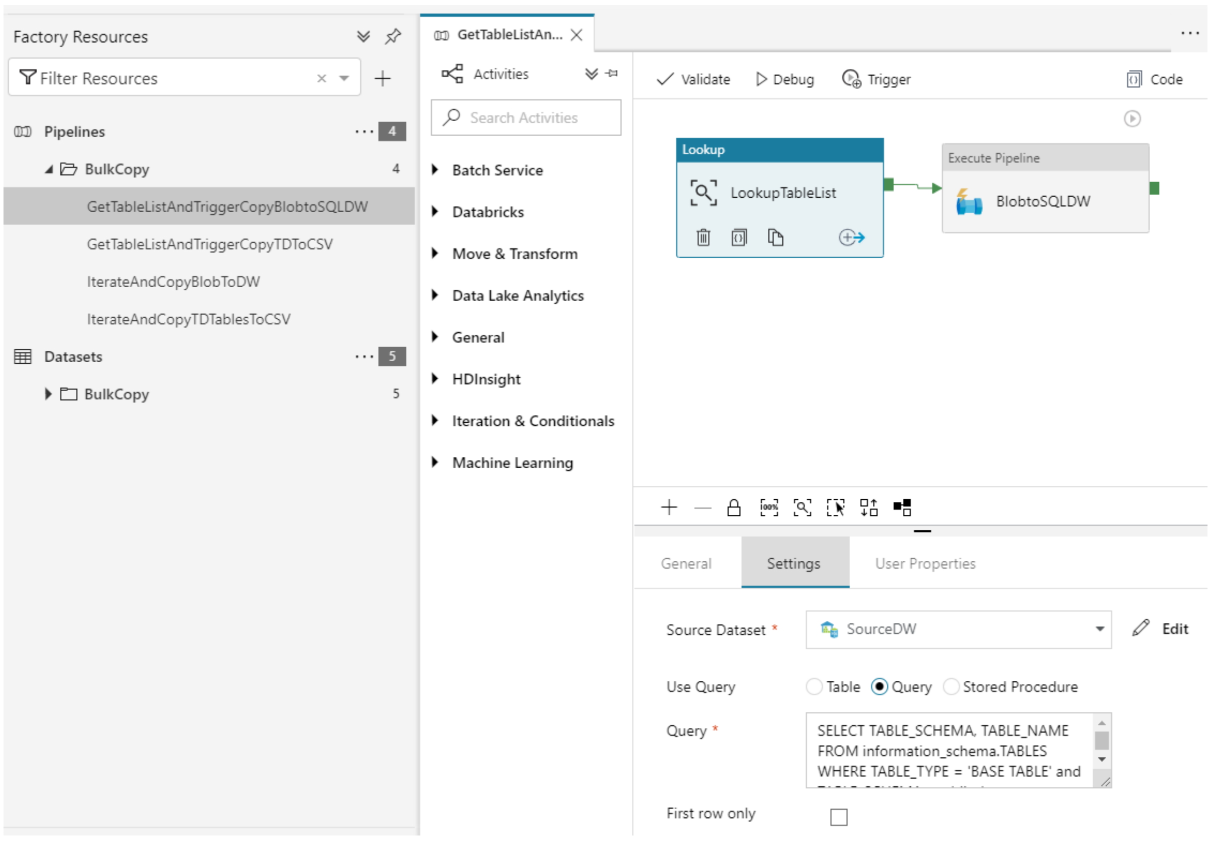


Sink Dataset * FileORC

NAME	VALUE
FileDirectory	@{item().TABLE_SCHEMA}
FileTableName	@{item().TABLE_NAME}.orc

Blob to Azure SQL Data Warehouse

The GetTableListAndTriggerCopyBlobtoSQLDW pipeline looks up the Azure SQL Data Warehouse system table to get the list of tables to be copied from Teradata to the local storage. This can be checked or changed in the Azure Data Factory User Interface GetTableListAndTriggerCopyBlobtoSQLDW pipeline by selecting the LookupTableList activity.



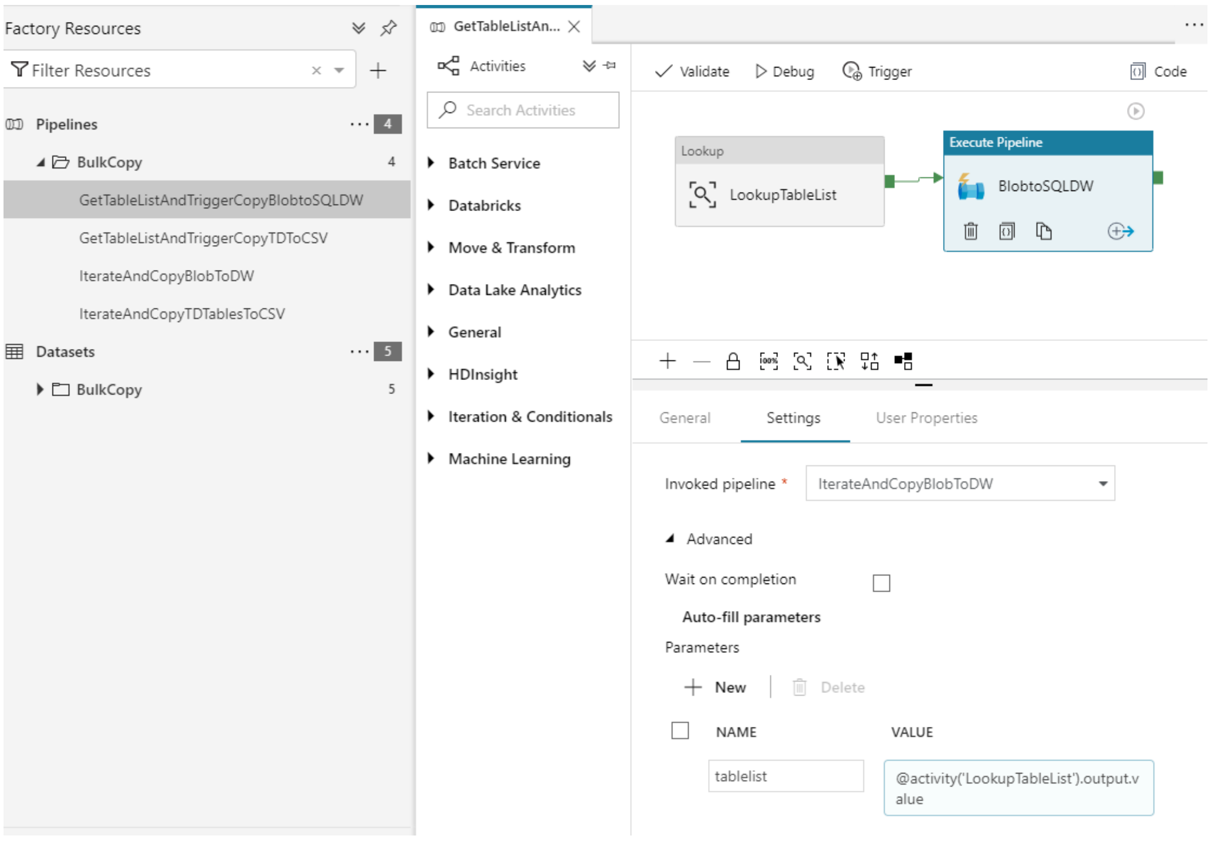
Source Dataset * SourceDW Edit

Use Query ( ) Table [✓ Query] ( ) Stored Procedure

Query *
```
SELECT TABLE_SCHEMA, TABLE_NAME
FROM information_schema.TABLES
WHERE TABLE_TYPE = 'BASE TABLE' and
```

First row only ☐

Selecting the BlobtoSQLDW activity and setting will show how we pass the output of LookupTableList to the IterateAndCopyBlobToDW pipeline.



Invoked pipeline * IterateAndCopyBlobToDW

Advanced

Wait on completion ☐

Auto-fill parameters

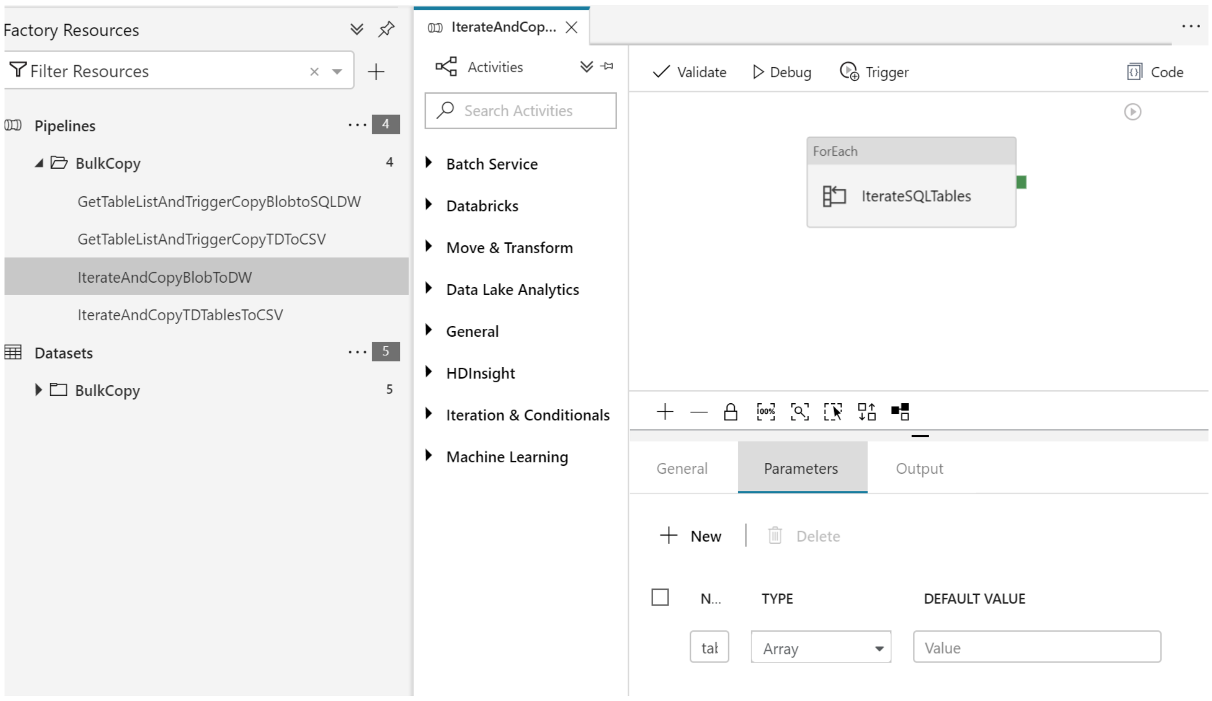
Parameters

New Delete

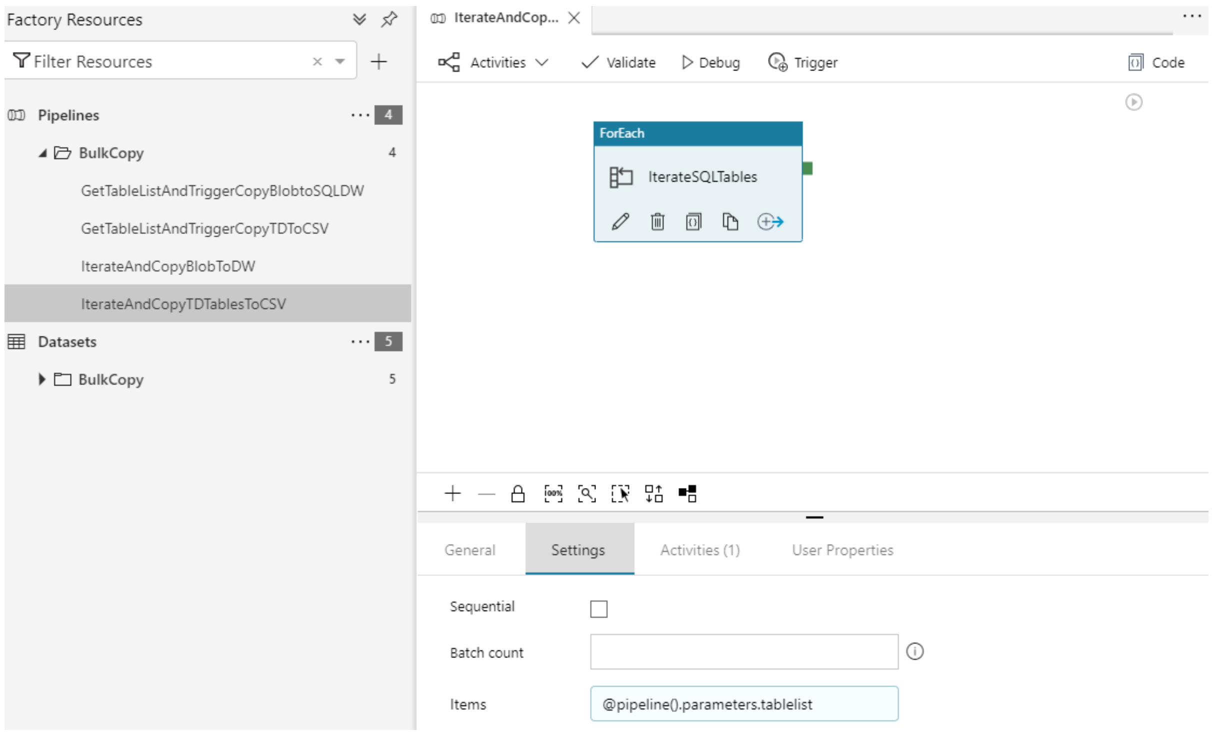
NAME	VALUE
tablelist	@activity('LookupTableList').output.value

GetTableListAndTriggerCopyBlobToDW pipeline triggers the pipeline IterateAndCopyBlobToDW to do the actual data copy.

The output of the lookup activity is passed into an array parameter which is used by the ForEach loop.

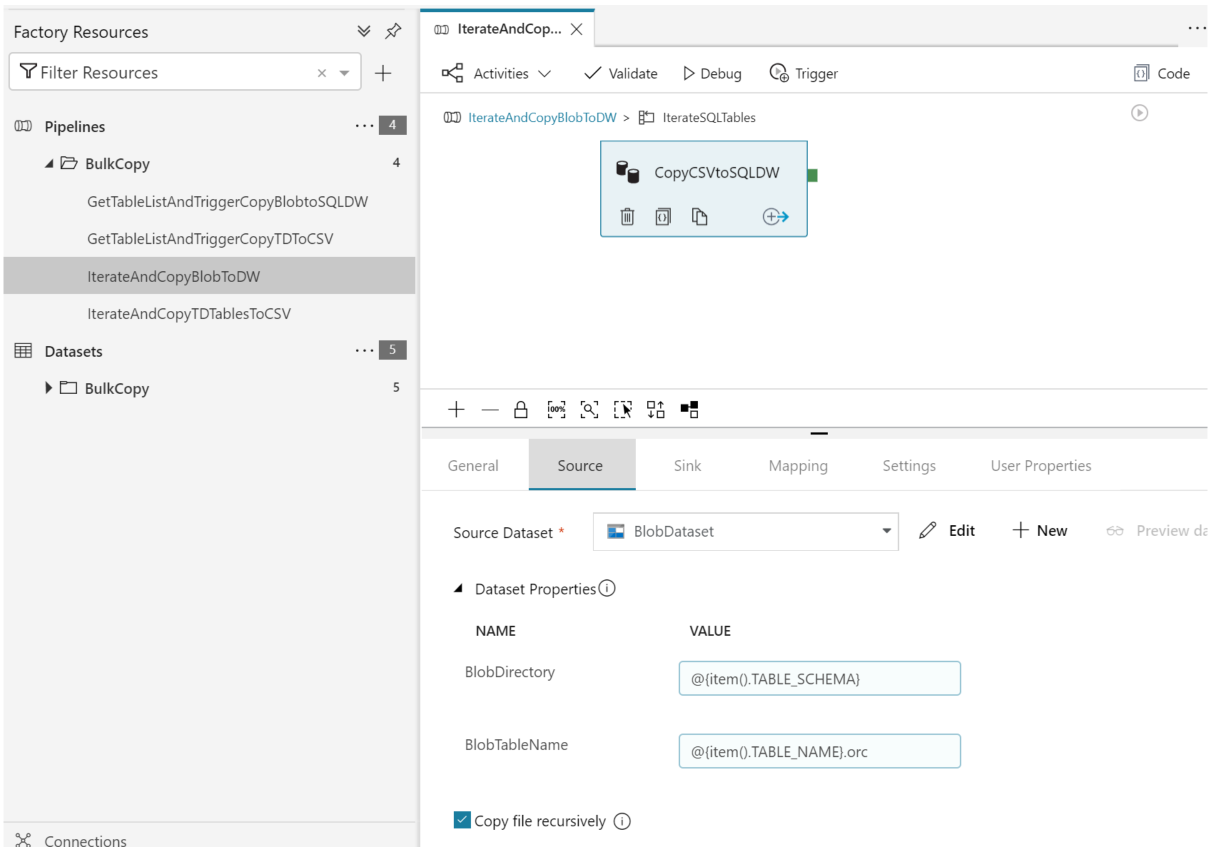


By selecting the ForEach activity you will see that this array to passed to another parameter associated with the activity.



Sequential	☐
Batch count	
Items	@pipeline().parameters.tablelist

Editing the activities for IterateAndCopyBlobToDW and selecting CopyCSVtoSQLDW you can investigate how the blob storage is passed a directory and filename based on the schema and tablename.



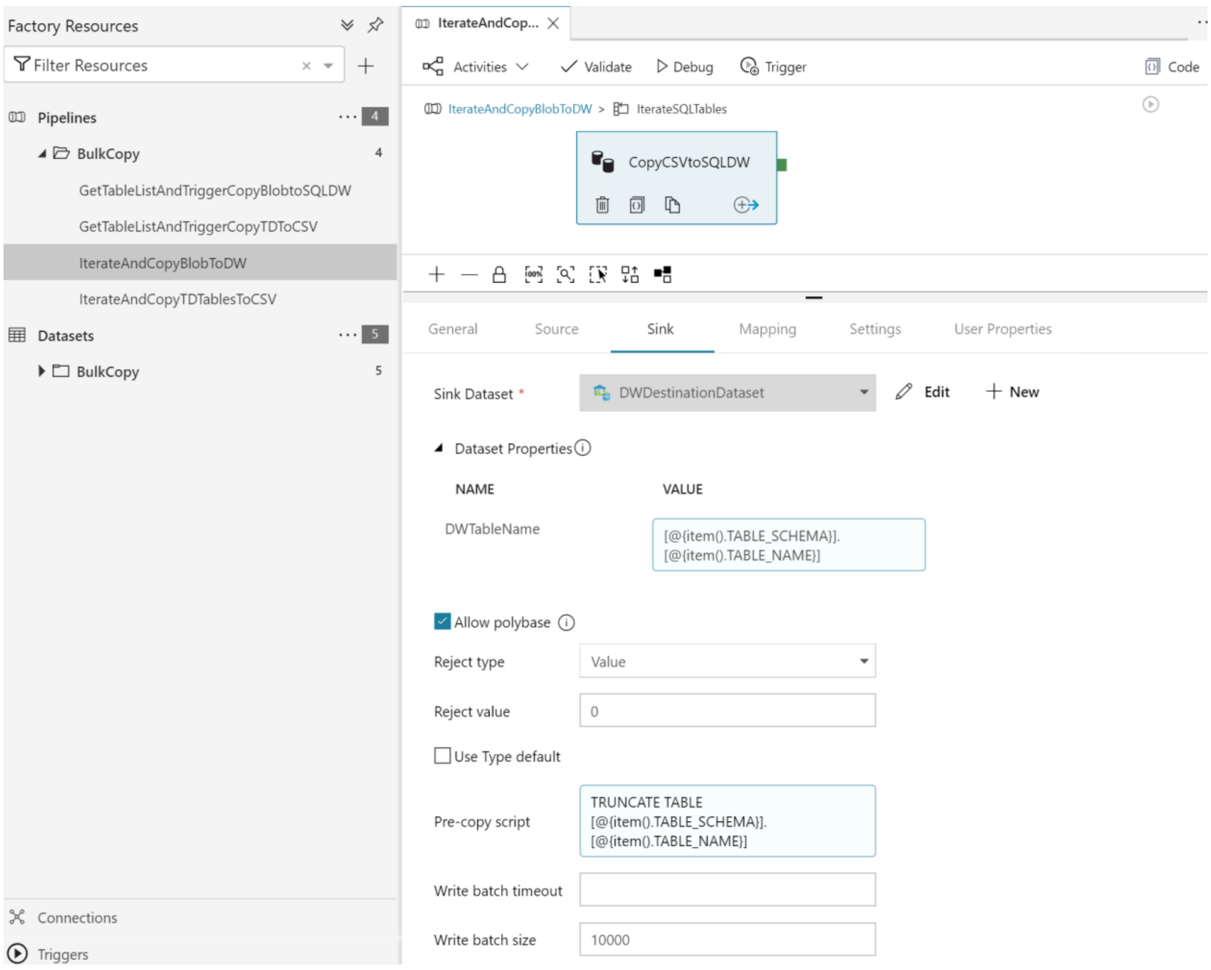
Source Dataset * BlobDataset

Dataset Properties

NAME	VALUE
BlobDirectory	@{item().TABLE_SCHEMA}
BlobTableName	@{item().TABLE_NAME}.orc

✓ Copy file recursively

Azure SQL Data Warehouse is setup as the sink and makes use of PolyBase, parameters for the destination and truncates the table before loading. In an advanced metadata approach, you would remove the pre-copy script and make use of Azure SQL Data Warehouse snapshot backups which are retained for seven days.



Sink Dataset * DWDestinationDataset

Dataset Properties

NAME	VALUE
DWTableName	[@{item().TABLE_SCHEMA}].[@{item().TABLE_NAME}]

☑ Allow polybase

Reject type	Value
Reject value	0

☐ Use Type default

Pre-copy script	TRUNCATE TABLE [@{item().TABLE_SCHEMA}].[@{item().TABLE_NAME}]
Write batch timeout	
Write batch size	10000

References

- Why Move to SQL Data Warehouse?
  https://datometry.com/solutions/replatforming/migrate-teradata-to-azure
- Migrate your solution to Azure SQL Data Warehouse
  https://docs.microsoft.com/en-us/azure/sql-data-warehouse/sql-data-warehouse-overview-migrate
- Migrating data to Azure SQL Data Warehouse in practice
  https://blogs.msdn.microsoft.com/sqlcat/2016/08/18/migrating-data-to-azure-sql-data-warehouse-in-practice/

Feedback and suggestions

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

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