Microsoft

Data Migration from on-premise relational Data Warehouse to Azure using Azure Data Factory

By Ye Xu (Senior Program Manager, Azure Data Factory Product Group)

Suzi Gubbels-Suguiyama (Principal Consultant, Data & AI, Microsoft Services)

Luka Lovosevic (Solution Architect, Data & AI, Microsoft Services)

Oct 2018

Contents

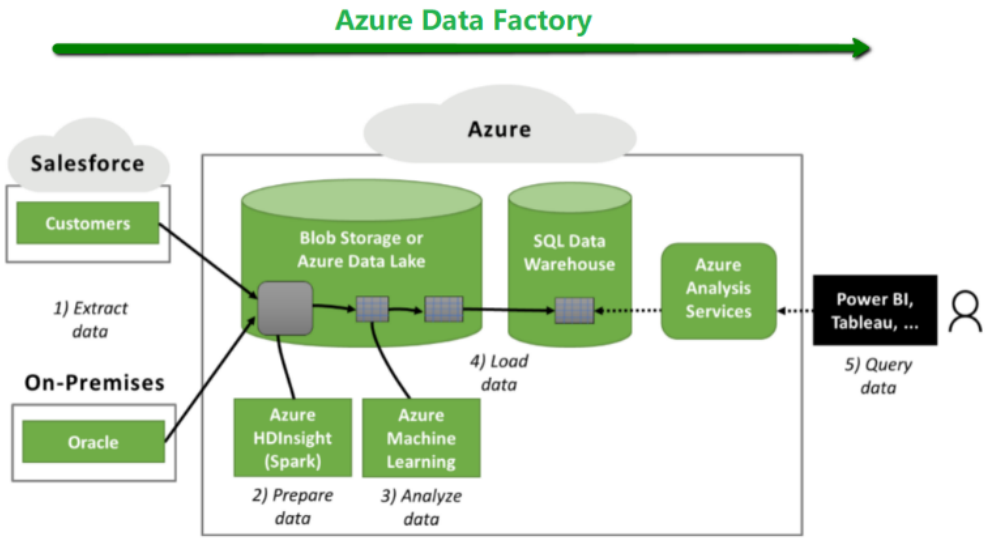
1 Background

A data warehouse is a system for an organization to store large amounts of historical data, and then analyze that data to better understand its customers, revenue, or other metrics related to business intelligence. Most data warehouses today are on premises, using technology such as SQL Server, Oracle, Netezza or Teradata etc. With the explosive size of data being generated due to widely used mobile devices, Internet of Things, e-commercial businesses, more and more organizations choose a modernized data warehouse in the cloud with low-cost data storage and massive amounts of processing power.

Azure is the place to build your modern data warehouse for its scalability, capacity, high performance, security, serverless compute & storage with a broad set of cloud data analysis services including:

- Azure Data Factory, providing scalable and serverless data integration service to orchestrate data analysis process including data ingestion (from both relational and unstructured data; both on-premises and cloud data services), data preparation, data transformation/advanced analysis and data egress for BI reporting.
- Azure SQL Data Warehouse, providing scalable relational data warehousing in the cloud.
- Azure Blob Storage, commonly called just Blobs, which provides low-cost cloud storage of binary data.
- Azure Data Lake Storage, implementing the Hadoop Distributed File System (HDFS) as a cloud service.
- Azure Data Lake Analytics, offering U-SQL, a tool for distributed analysis of data in Azure Data Lake Storage.
- Azure Analysis Services, a cloud offering based on SQL Server Analysis Services for building cubes.
- Azure HDInsight, with support for Hadoop technologies, such as Hive and Pig, along with Spark.
- Azure Databricks, a Spark-based analytics platform.
- Azure Machine Learning, a set of data science tools for finding patterns in existing data, then generating models that can recognize those patterns in new data.

The following figure is showing an example of a typical modern data warehouse on Azure.



The data are firstly extracted from a big variety of data sources, no matter they are on premise or cloud, into a data lake, a much less expensive form of storage implemented using either Blob Storage or Azure Data Lake Storage. Then, data cleaning and preparation are required to delete duplicates, convert the data type by a compute engine running on Azure Data Lake Analytics, Azure HDInsight or Azure Databricks. Given these jobs will be done concurrently through big data technology like Spark, it won't take too long time even if huge amount of data is required to process. After the data are clean and ready, some further advanced data analysis like machine learning can be applied on the data in order to get business insight from your data. Finally, the subset of aggregated data as the result of an analysis can be loaded into a relational database so that business analysts can easily query against that data.

In order to build your modern data warehouse on Azure, the first important step at the beginning of the journey is to migrate huge amounts of data from existing on-premise relational data warehouse (e.g. Netezza, Oracle, Teradata, SQL server) to a storage implemented using either Blob Storage or Azure Data Lake Storage. The data migration process normally contains 2 parts:

1. Migrating the historical data from on-premise relational data warehouse (Netezza, Oracle, Teradata, SQL server etc.) to Azure (Blob Storage, Azure Data Lake Storage etc.);
2. Keeping the data synchronization between existing relational data warehouse and Azure Storage before the switchover.

This paper is targeting to address the complexity of the first part, which is migrating tens of TB data from an on-premise relational data warehouse (Netezza, Oracle, Teradata, SQL server etc.) to Azure Data Lake Storage by Azure Data Factory.

Azure Data Factory is a single Azure cloud service for data integration on your data, no matter they are on Azure, on premises, or on another public cloud, such as Amazon Web Services (AWS). It provides a single set of tools and a common management experience for all of your data integrations. When you use Azure Data Factory* to migrate historical data from on-premise

relational data warehouse to Azure, you will get benefit of its scalability, security, performance and high availability on data movement.

* This article refers to the Azure Data Factory v2

2 Get started

For migrating historical data (tens of TB) from on-premise relational data warehouse (Netezza, Oracle, Teradata, SQL server etc.) to Azure Data Lake Storage, we recommended you go through the steps as following:

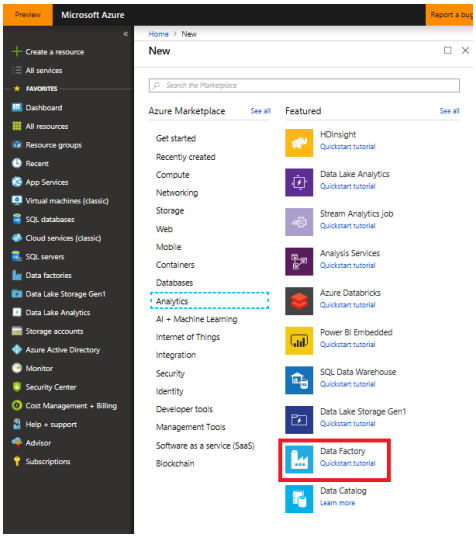
1. Do a POC test by loading a small size of data from on-premise system to Azure Data Lake Storage using Azure Data Factory, in order to prove the end-to-end data migration solution.
2. Plan the data migration and partition the data on on-premise data warehouse.
3. Do the data migration using Azure Data Factory.
4. Monitor the data migration progress and re-run the failure jobs.

3 Do a test to verify the solution

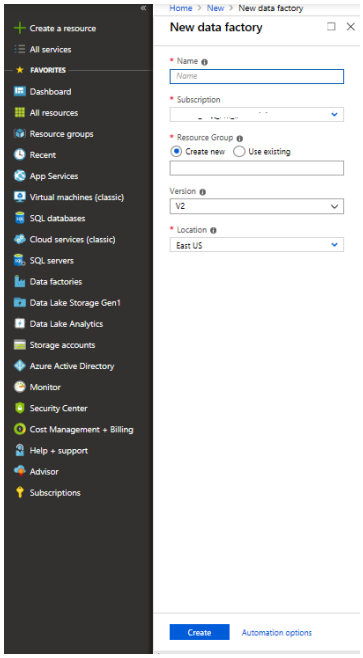
A small test is highly encouraged before you really start the data migration work, so that you can remove all the dependencies and issues, resolve the throughput bottleneck, and obtain a high-confidential timeline to complete the data migration.

3.1 Create your Azure Data Factory

1. Go to https://ms.portal.azure.com/ to create a resource, you will find "**Data Factory**" in the **Analytics** category.



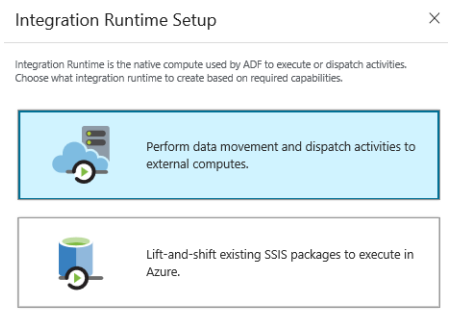
2. In the **New data factory** page, create your own Data Factory.



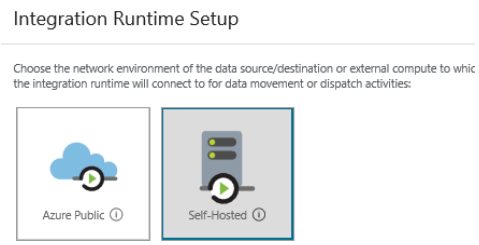
3.2 Create and install ADF self-hosted Integration Runtime in your local windows machine

Now you have ADF created on Azure. Given that you want to load data from on-premise database to Azure, it is highly possible that the on-premise system does not have direct connection to the public cloud due to firewalls. ADF can solve this problem by installing a software agent in your local windows machine behind your corporate firewall or inside the virtual private network. The software agent is called ADF self-hosted integration runtime (IR), which performs the data movement job from on-premise data warehouse to Azure. One of the most important features is that it only requires outbound HTTP-based connections so that you will not have security issues.

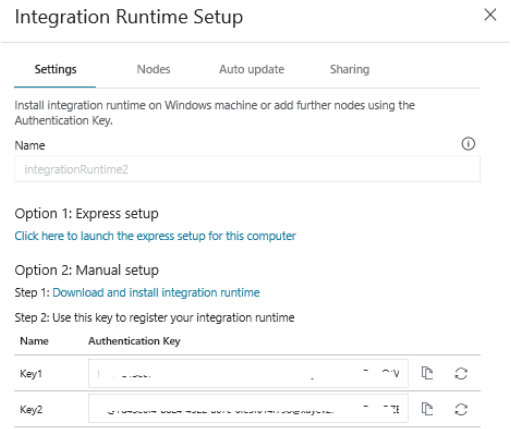
1. Go to ADF UI and navigate to **Integration Runtime Setup** page to create Integration Runtime. Select "**Perform data movement"**.



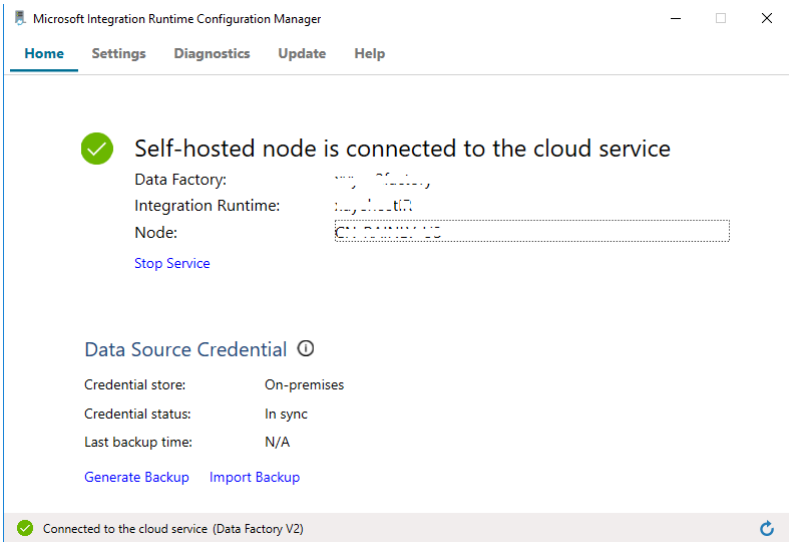
2. Select the **Self-hosted** type.



3. After you create self-hosted integration runtime from ADF UI, it will lead you to install the app in your local windows machine. Please make sure the local windows machine having direct connection to your on-premise database.



4. After downloading, installing and registering the self-hosted integration runtime, you will see the self-hosted integration runtime works well as shown below:

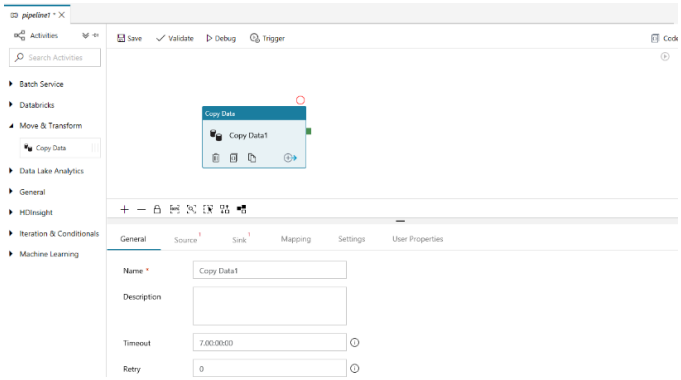


3.3 Create one pipeline in ADF to load a small size of data from on-premise database to Azure Data Lake Storage

Now you can start to build your first ADF pipeline to load a small size of data from on-premise data warehouse to Azure Data Lake Storage. Again, the purpose for this step is to:

- Verify the end-to-end workflow of loading data from your on-premise environment to Azure.
- Get the throughput so that you can calculate the estimated date (timeline) to complete the whole data migration.

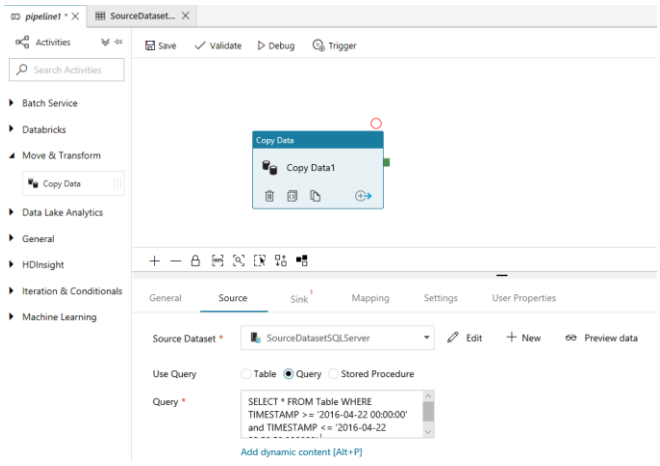
1. Go to ADF UI to **create pipeline**, and drag one **copy activity** in your pipeline



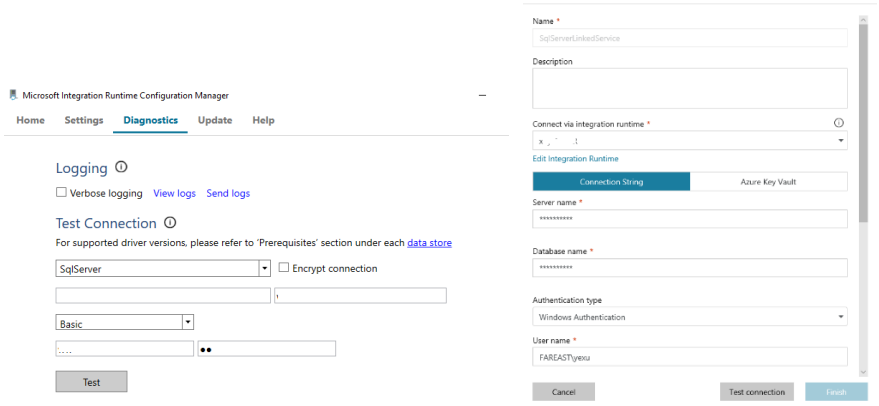
2. In the copy activity, put your on-premise data warehouse as **source store**, which includes the connection string, table name, self-hosted integration runtime name etc.

   Some recommended practices:

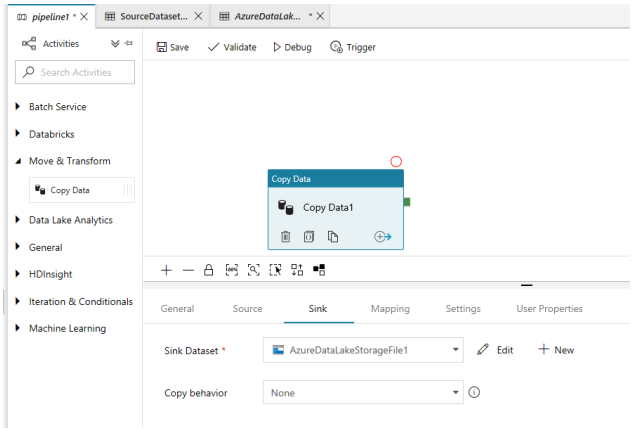
   a. When selecting the table in the source database, you can either write the query to filter the data to be loaded or input the table name to load the entire table. We would suggest you write a query like this: `SELECT * FROM Table WHERE TIMESTAMP >= '2016-04-22 00:00:00' and TIMESTAMP <= '2016-04-22 23:59:59.999999'` in order to easily control the size of the data to be loaded every time.



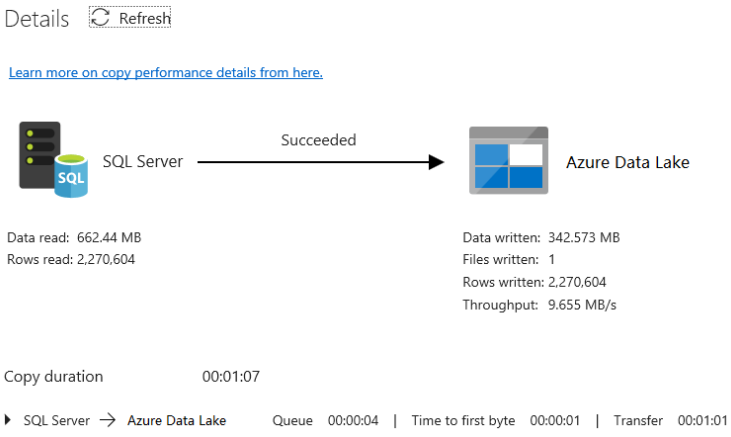
b. Please make sure you select the right self-hosted integration runtime which can connect to your on-premise data warehouse directly. You can do the test connection to verify.



3. In the copy activity, put Azure Data Lake Storage as the **destination store**, which includes Azure Data Lake Storage account name, folder name etc.



4. Run the pipeline.

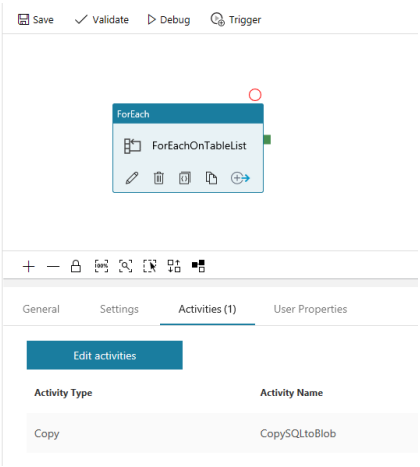


From the monitoring page of the ADF UI, you will see the pipeline running successfully with the concrete statistic data which contains number of rows being copied, throughput, etc. And you should also see a new file created in your Azure Data Lake Storage.

If you see the throughput is very low, you can do the performance tuning by figuring out the bottleneck which impact the copy throughput. Basically, the factors to impact the copy throughput can possibly be the network bandwidth, the loads on your on-premise database, the capacity of your Windows machine hosting self-hosted IR, or the schema of data in database, etc. For more details, you can refer to: https://docs.microsoft.com/en-us/azure/data-factory/copy-activity-performance#sample-scenario-copy-from-an-on-premises-sql-server-to-blob-storage .

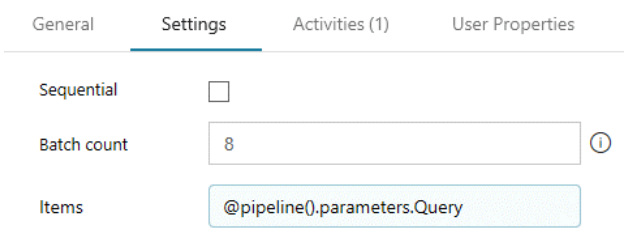
After getting the throughput of loading data from your on-premise data warehouse to Azure by one ADF copy activity, you can multiply the throughput by multiple ADF copy activities running in parallel. Let's take an example that the throughput is 5 MBps per ADF copy activity.

5. Add foreach activity in the pipeline to enable multiple copy activities running concurrently.

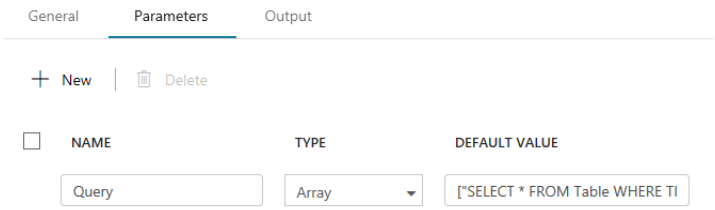


Some recommended practices:

a. Talk with the owners of on-premise data warehouse to see what the maximum sessions are allowed to use for the data migration work. Due to the normal ETL or BI work are still working on the on-premise data warehouse during the data migration, you will not be allowed to use many sessions to read data from the on-premise data warehouse in most cases. For example, let's say you can get maximum 8 concurrent sessions to load data from the on-premise data warehouse. Then you can write 8 into the batch count setting in ADF foreach activity (as shown in the picture below), so that ideally you will get 40 MBps (by the math of 5 MPps * 8) for data migration in total.



b. Create parameters for the pipeline, so that you can make your pipeline dynamically load different datasets without updating the pipeline. The picture below parametrized the query used to filter rows to be loaded from on-premise data warehouse to Azure. By doing that, you can run the pipeline with a list of queries as parameters to indicate which rows you want to load into Azure.



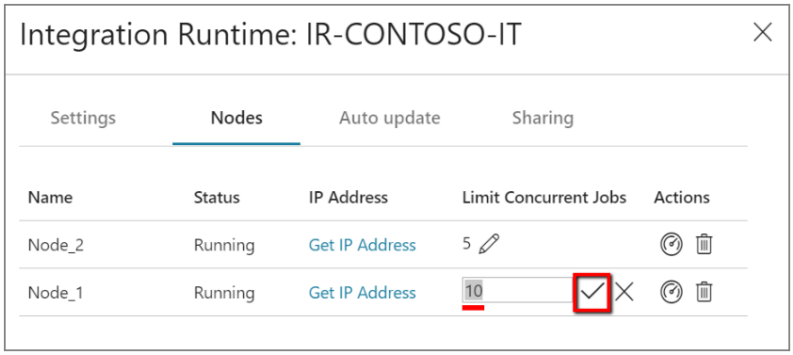
The parameter values can be as in this example:

```
[“SELECT * FROM Table WHERE TIMESTAMP >= '2016-04-22 00:00:00' and TIMESTAMP <= '2016-04-22 23:59:59.999999'”, “SELECT * FROM Table WHERE TIMESTAMP >= '2016-04-23 00:00:00' and TIMESTAMP <= '2016-04-23 23:59:59.999999'”, “SELECT * FROM Table WHERE TIMESTAMP >= '2016-04-24 00:00:00' and TIMESTAMP <= '2016-04-24 23:59:59.999999'”, “SELECT * FROM Table WHERE TIMESTAMP >= '2016-04-25 00:00:00' and TIMESTAMP <= '2016-04-25 23:59:59.999999'”, “SELECT * FROM Table WHERE TIMESTAMP >= '2016-04-26 00:00:00' and TIMESTAMP <= '2016-04-26 23:59:59.999999'”]
```

c. Enable the scalability/HA for self-hosted integration runtime.

   i. Given that you will load huge amounts of data from the on-premise data warehouse to Azure, having multiple nodes (windows machines with self-hosted integration runtime installed) for one logical self-hosted integration runtime can not only avoid single point of failure but also improve the throughput by scaling out the loads across machines if you found the capacity of a single node is the bottleneck to impact copy throughput.

   ii. If the maximum sessions of the on-premise data warehouse allowed for data migration is 8, you need to check the concurrent jobs limit of self-hosted integration runtime across all nodes to make sure it is bigger than 8. Otherwise, you cannot fully leverage the bandwidth of on-premise data warehouse. (e.g. The total number in the picture below is 15)



***Note**: The concurrent job limit for each node is also restricted by the capacity (CPU & memory) of computer node. If you cannot set the higher limit from ADF UI, you need either increase the capacity of the node (scale-up) or add more nodes into the logical self-hosted integration runtime (scale-out).*

6. Run the pipeline again to see if you can get 40 MBps throughput.

4 Create data migration plan

Now you have completed the solution design for the data migration, it is the time to create the data migration plan.

For example, let's say you have 20 tables with 25 TB data in on-premise data warehouse to be migrated to Azure, where 5 out of 20 are fact tables with relatively big data volume. You have 8 sessions from on-premise data warehouse which can be used to load data concurrently. The migration window is from 19:00 to 07:00 every Monday to Friday.

Based on the situation above, you can partition your data on the on-premise data warehouse. The reasons to do that is that:

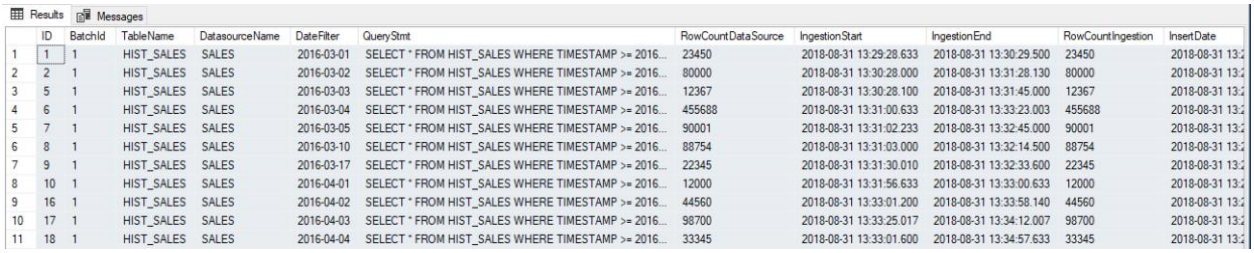
1. The timing window for migration is 12 hours every day. When loading the tables that require more than 12 hours to complete, you have to find a way to split them into multiple pieces to make sure each piece can be completed within 12 hours.

2. You have huge size of data volume to be loaded into Azure. If you directly load the entire tables, you cannot get much benefit of parallel loading without partitions, especially for the fact tables with relatively big data volume.

3. The time spent for each individual copy job would be much longer if you load the entire tables without partitions. So you will have more chances to meet unexpected interruptions like network transient issues, which could impact your migration timeline.

One of the practices to plan the partitions is to calculate the time for each copy job to complete loading one partition from your on-premise data warehouse to Azure. For example, it would be good if you can complete copying one partition to Azure within a couple of hours. Given the throughput for one copy activity to load data is 5 MBps, you can try 25 GB data volume as one partition, so that you can complete loading one partition into Azure in 1.42 hours (by the math of 25 * 1024 / (5 * 3600)). By doing that, you will have 1024 partitions in total (by the math of 25 TB / 25 GB). You can do further customizations on partitioning your data based on specific business logic, environment situation, etc.

After you come up with the partitions, you can write the metadata of the partitions into a control table. The example of control table can be as following:



Results | Messages

	ID	BatchId	TableName	DatasourceName	DateFilter	QueryStmt	RowCountDataSource	IngestionStart	IngestionEnd	RowCountIngestion	InsertDate
1	1	1	HIST_SALES	SALES	2016-03-01	SELECT * FROM HIST_SALES WHERE TIMESTAMP >= 2016...	23450	2018-08-31 13:29:28.633	2018-08-31 13:30:29.500	23450	2018-08-31 13:
2	2	1	HIST_SALES	SALES	2016-03-02	SELECT * FROM HIST_SALES WHERE TIMESTAMP >= 2016...	80000	2018-08-31 13:30:28.000	2018-08-31 13:31:28.130	80000	2018-08-31 13:
3	5	1	HIST_SALES	SALES	2016-03-03	SELECT * FROM HIST_SALES WHERE TIMESTAMP >= 2016...	12367	2018-08-31 13:30:28.100	2018-08-31 13:31:45.000	12367	2018-08-31 13:
4	6	1	HIST_SALES	SALES	2016-03-04	SELECT * FROM HIST_SALES WHERE TIMESTAMP >= 2016...	455688	2018-08-31 13:31:00.633	2018-08-31 13:33:23.003	455688	2018-08-31 13:
5	7	1	HIST_SALES	SALES	2016-03-05	SELECT * FROM HIST_SALES WHERE TIMESTAMP >= 2016...	90001	2018-08-31 13:31:02.233	2018-08-31 13:32:45.000	90001	2018-08-31 13:
6	8	1	HIST_SALES	SALES	2016-03-10	SELECT * FROM HIST_SALES WHERE TIMESTAMP >= 2016...	88754	2018-08-31 13:31:03.000	2018-08-31 13:32:14.500	88754	2018-08-31 13:
7	9	1	HIST_SALES	SALES	2016-03-17	SELECT * FROM HIST_SALES WHERE TIMESTAMP >= 2016...	22345	2018-08-31 13:31:30.010	2018-08-31 13:32:33.600	22345	2018-08-31 13:
8	10	1	HIST_SALES	SALES	2016-04-01	SELECT * FROM HIST_SALES WHERE TIMESTAMP >= 2016...	12000	2018-08-31 13:31:56.633	2018-08-31 13:33:00.633	12000	2018-08-31 13:
9	16	1	HIST_SALES	SALES	2016-04-02	SELECT * FROM HIST_SALES WHERE TIMESTAMP >= 2016...	44560	2018-08-31 13:33:01.200	2018-08-31 13:33:58.140	44560	2018-08-31 13:
10	17	1	HIST_SALES	SALES	2016-04-03	SELECT * FROM HIST_SALES WHERE TIMESTAMP >= 2016...	98700	2018-08-31 13:33:25.017	2018-08-31 13:34:12.007	98700	2018-08-31 13:
11	18	1	HIST_SALES	SALES	2016-04-04	SELECT * FROM HIST_SALES WHERE TIMESTAMP >= 2016...	33345	2018-08-31 13:33:01.600	2018-08-31 13:34:57.633	33345	2018-08-31 13:

One row in the control table implies one partition, and the columns for each row describe all the metadata for that partition. The descriptions for the columns are as followings:

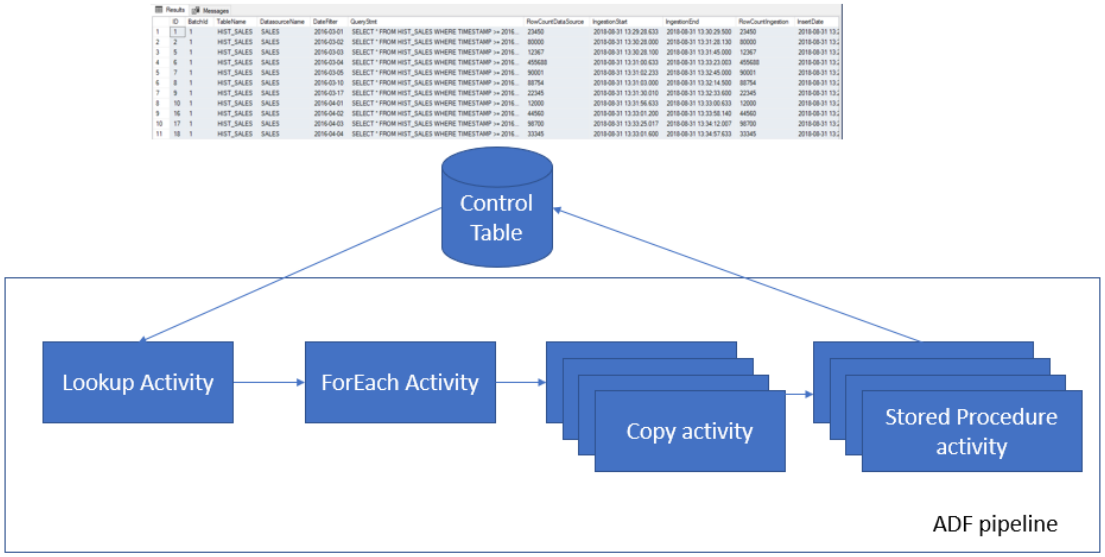
- The Column "TableName" is specifying the table name of data source store hosting this partitioned data. If there are several partitions in one table, you will see several rows in the control table sharing the same table name for this column.
- The Column "QueryStmt" is the query to get this partitioned data from on-premise data warehouse. E.g. "SELECT * FROM Table WHERE TIMESTAMP >= '2016-04-22 00:00:00' and TIMESTAMP <= '2016-04-22 23:59:59.999999'"
- The Column "RowCountDataSource" implies how many rows of this partition in your on-premise data warehouse.
- The Column "IngestionStart", "IngestionEnd" and "RowCountIngestion" are the statistical data for each data copying job. The value for these three columns will be NULL at the beginning, and only be populated after the copy job completing loading this partition to Azure Data Lake Storage. If you see the rows are NULL in these 3 columns, it means these partitions have not been loaded to Azure yet.

The last step is to decide how many partitions you will load to Azure within one working day. One of the practices is to copy 48 partitions in one working day, so that it will take less than 9 hours (by math of 1.42 * 48 / 8) to complete the data migration in every working day. Given you have 12 hours per day, you can have some buffer (about 2 hours) to handle unexpected issues. So altogether, you will need 21.3 working days (by math of 1024 / 48) to complete the whole data migration from on-premise data warehouse to Azure Data Lake Storage. You can add the "BatchId" for each partition in the control table, so that you can select one batch of partitions to copy to Azure in each day.

5 Do the data migration

Now you can update the pipeline in ADF to copy the entire data from on-premise data warehouse to Azure. There are 2 big updates. One is to add a lookup activity at the beginning of the pipeline to read the partitions from the control table. Another is to add a stored procedure activity at the end of pipeline to write the statistical data back to the control table, so that people can monitor the progress of the data migration.

The logical diagram will be as following:

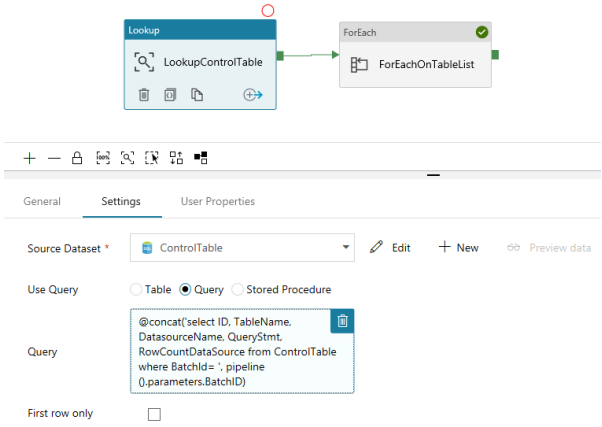


1. Add the lookup activity in the pipeline.

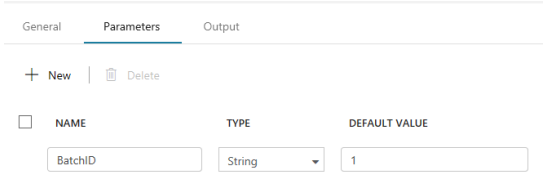
    a. When creating the dataset and linked service for the lookup activity, connect to the database server hosting your control table.

    b. Write the query in the lookup activity to get the partition list from the control table. E.g. *@concat('select ID, TableName, DatasourceName, QueryStmt, RowCountDataSource from ControlTable where BatchId= ', pipeline().parameters.BatchID)*

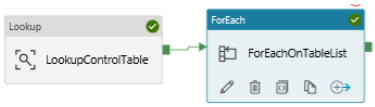
    Note: the "BatchId" can be the pipeline parameter in your query, so you can choose different batch of partitions to be copied to Azure without updating the pipeline.

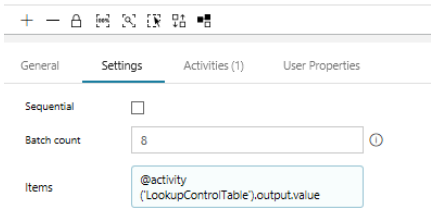


Pipeline parameter:

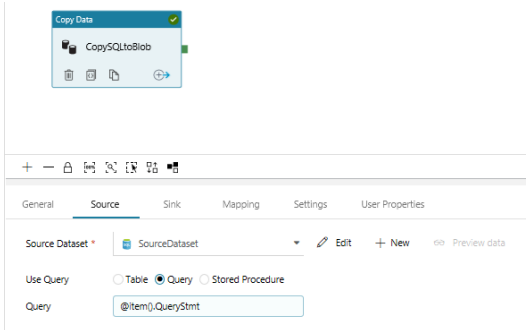


2. Update the foreach activity to consume the output from lookup activity.





3. Update the copy activity to consume the query expression from the control table to read the partitioned data and copy to Azure.



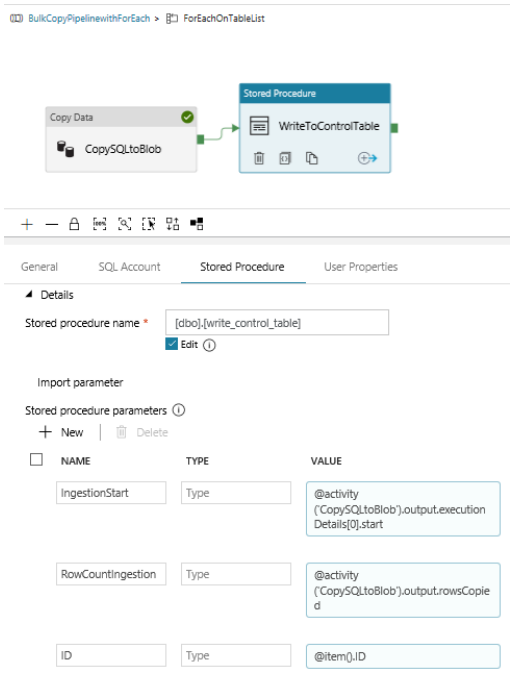
4. Add stored procedure activity in the pipeline.

   You will firstly need to create a stored procedure in the same database server hosting the control table. The logic of the stored procedure can be something like checking if the row count being copied is the same as the actual row count in the data source store; writing the statistical data to control table for monitoring purposes; or any additional logic after one partition has been copied to Azure Data Lake Storage.

   And then add a stored procedure activity in the pipeline.

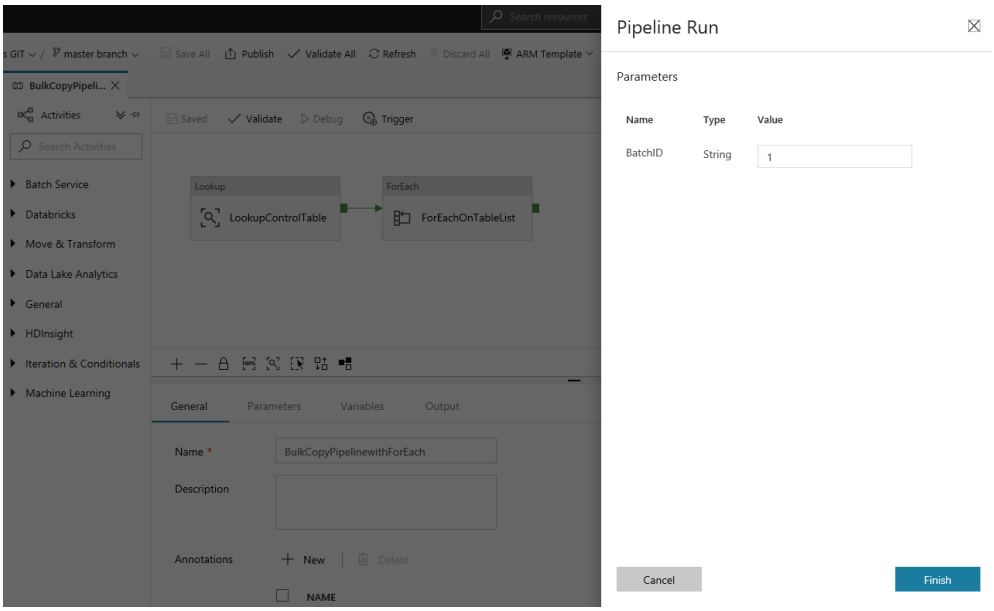
   a. Input the SQL account of your stored procedure.

   b. Select the stored procedure with the parameter value. You can see from the picture below that the statistical report from the copy activity will be consumed by the stored procedure as parameters.

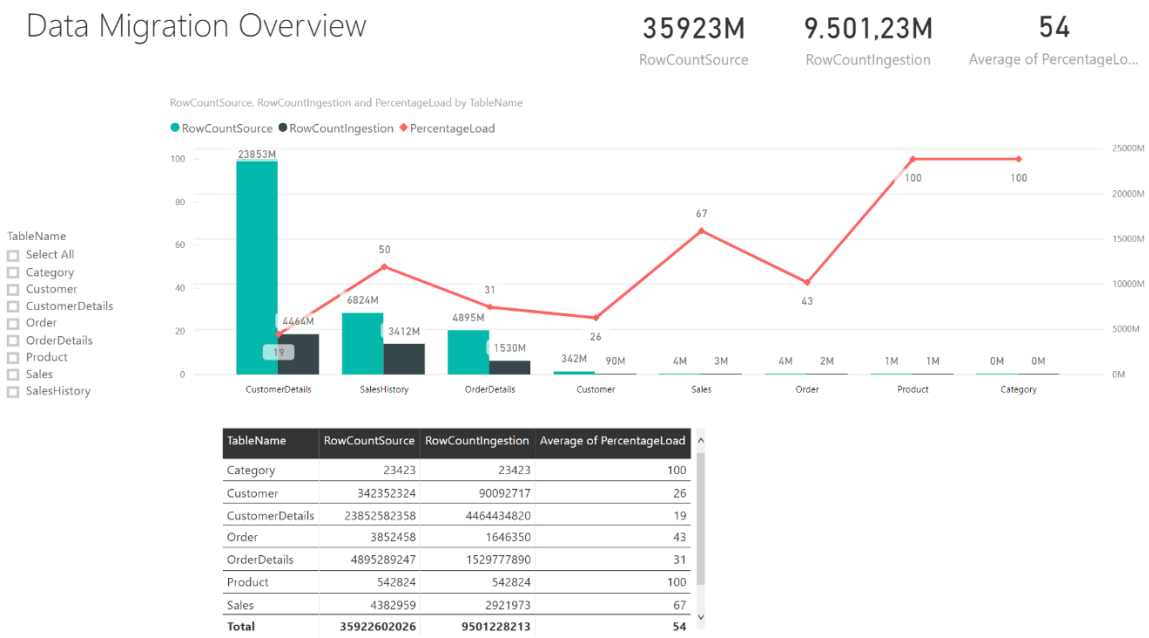


6 Monitor the data migration

After you completed the steps to update your pipeline in ADF, you can trigger the pipeline within the migration window to copy data from on-premise data warehouse to Azure Data Lake Storage based on your plan. You will be asked to input the parameter (e.g. BatchID) to the pipeline before triggering it, so that you can select one batch you defined in the control table to load the data.



During the data migration, you can keep monitoring the pipeline status from the ADF UI or your control table to get the latest status. Through the control table, you can also build Power BI dashboard against its data to get more customized and fancy reporting as following.



TableName	RowCountSource	RowCountIngestion	Average of PercentageLoad
Category	23423	23423	100
Customer	342352324	90092717	26
CustomerDetails	23852582358	4464434820	19
Order	3852458	1646350	43
OrderDetails	4895289247	1529777890	31
Product	542824	542824	100
Sales	4382959	2921973	67
Total	**35922602026**	**9501228213**	**54**

If there is any partition that failed to be copied to Azure due to whatever reason, you can reload that particular partition. One of the practices to reload is to change the value of “BatchID” for that partition in control table and rerun the pipeline with new “BatchID”.

7 Conclusion

Now you have gone through the major steps of data migration from an on-premise data warehouse (e.g. Oracle Server, Netezza Server, Teradata Server or SQL Server) to Azure (e.g. Azure Data Lake Storage). The following list summarizes the main points of the article:

1. Technical aspects from the solution design on data migration from on-premise data warehouse to Azure.
    - Define a reasonable parallelism and throughputs for data loading from on-premise data warehouse to Azure. The key point is to make ADF fully leverage bandwidth of on-premise data warehouse (multiple sessions) and network channel by configuring concurrent setting in ADF as well as capacity planning for the ADF self-hosted IR machine.
    - Create a control table to gain the flexibility of managing the data loading process so that any on-demand request or exceptions can be smoothly handled (for example, you may be asked for changing the migration windows, re-running the specific jobs, etc.). The control table can also be used to monitor the latest progress of data migration.
    - Separate initial load and incremental load into different workflows. Initial load is a one-off job which is more focusing on loading big volume of data, parallelism/throughputs, partitions, etc. Incremental load is the periodical job which is more focused on identifying the new or updated data from the data source, where data volume is relatively small.

2. Data migration requires careful planning with resolving all of the dependencies. Doing a POC test first is strongly recommended so that you can remove all the dependencies and issues (for example with drivers, available time window for migration, etc.), resolve the throughput bottleneck and obtain a high-confidential timeline to complete the data migration.

3. Monitor the migration progress. Given you are doing hybrid data loading from on-premise to Azure, any exception could be occurred during the data migration, like network transient issue, etc. It is therefore important that you monitor the progress and are ready to remediate any issues and re-run the failed copy jobs by adjusting the control table timely.

Happy migrating!