Microsoft

Azure Business Data Warehouse for SAP

Published: Tuesday, November 12, 2019

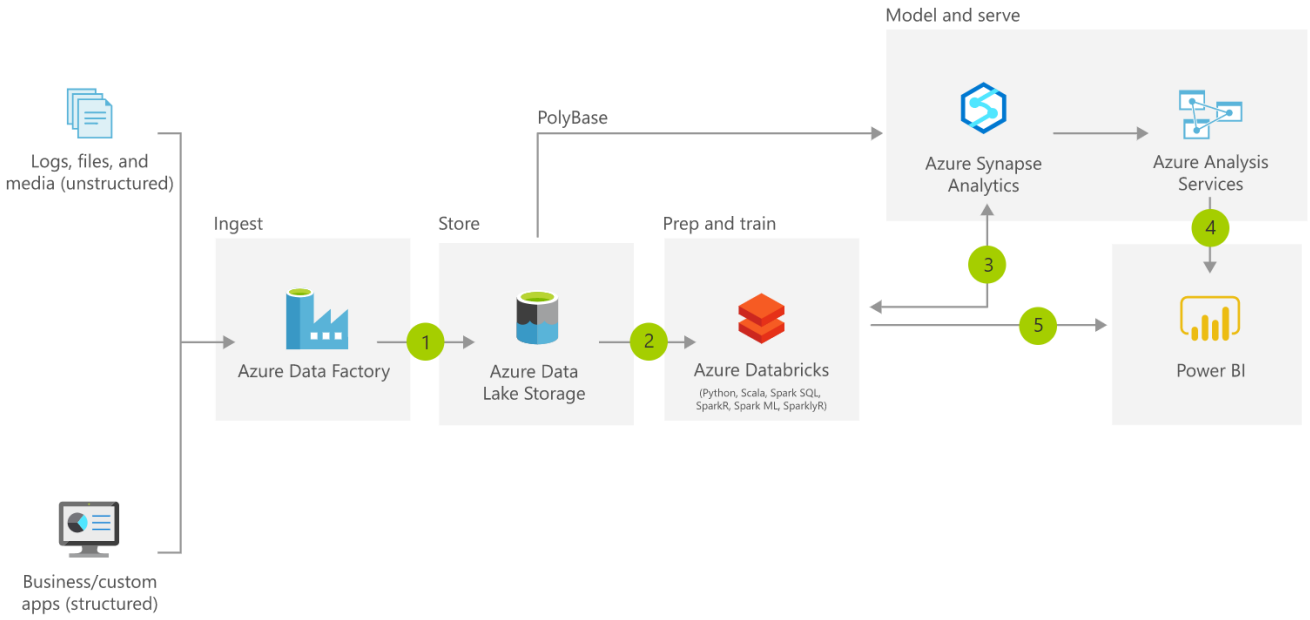
For the latest information, please see
https://github.com/msftargentina/azure-business-warehouse-for-sap

About

A modern data warehouse architecture enables bringing together data at any scale easily, and to get insights through analytical dashboards, operational reports, or advanced analytics. The following reference architecture explains how to implement a Business Data Warehouse for integration of SAP workloads in a centralized repository. This architecture becomes all the more compelling in light of the new Azure Synapse Analytics (formerly Azure SQL Data Warehouse) price-performance benchmarks released by GigaOM (reference: https://gigaom.com/report/data-warehouse-cloud-benchmark/)

Architecture overview

The following diagram shows the MDW architecture pattern recommended and broadly adopted. Some of the components are optional and can be deployed as needed, but we show the entire architecture for future reference and to include extensibility points on itself. See the description for each of the components bellow:



Components description

Required components

Azure Data Factory	Data Factory is a managed service that orchestrates and automates data movement and data transformation. In this architecture, it coordinates the various stages of the ELT process and loading procedure to SQL Data Warehouse using PolyBase technology which provides the best performance for loading. Orchestration is done using Pipelines which can be scheduled to run as frequently as needed.

Azure Data Lake	Azure Blob storage is a Massively scalable object storage for any type of unstructured data—images, videos, audio, documents, and more—easily and cost-effectively. This can be used for later analytics workloads or predictive modeling. Optionally, data can be loaded directly to SQL Synapse Analytics, however a landing zone is always recommended.
Azure Synapse Analytics	Azure Synapse Analytics is the fast, flexible and trusted cloud data warehouse that lets you scale, compute and store elastically and independently, with a massively parallel processing architecture.
Azure Analysis Services	Analysis Services is a fully managed service that provides data modeling capabilities. The semantic model is loaded into Analysis Services and it is where uses connect to discover the underlying data. Analysis services is also able to handle the concurrent connections and scalability of multiple users effectively in memory.

Optional components

Power BI	Power BI is a suite of business analytics tools to analyze data for business insights. In this architecture, it queries the semantic model stored in Analysis Services via Live Query which can run against the cache, even upon loading a report. This behavior improves load performance for the report.
Azure Databricks	Azure Databricks is a fast, easy, and collaborative Apache Spark-based analytics platform that can be integrated on top of Data Lake to support advance analytics, heavy data transformations and big data workloads.

Integration and monitoring

Azure Logic Apps	Logic Apps provides an easy way to implement workflows in the cloud. We will use Logic Apps to trigger updates in Azure Analysis Services using REST APIs.
Azure Log Analytics	Monitoring is important in each solution. We strongly recommend the configuration of Azure Log Analytics to provide a unified solution to monitor the platform.

Key points of the proposed architecture

Although they may be multiple ways to achieve the outcomes, this proposed architecture has the following key properties:

- **Fully automated pipeline orchestration provided by Data Factory:** Data Factory is a cloud base data orchestration solution, fully manageable using the browser. It also features connectors to a broad set of connectors (80+) to ingest data from both Microsoft and non-Microsoft technologies.
- **Parallel data loading to achieve high throughput**: Data Factory allows the ingestion of high volumes of data by using parallel data ingestion and partitioning the queries in the data source. **This enables scalability not just in**

the querying system but also in the queried backend. Multiple ingestion nodes can be configured for the same data source.

- **Cost-effective storage using Data Lake Gen2:** Access to a cost-effective storage using Azure Data Lake Gen2 which supports advance analytics workloads using HDFS-based solutions including Azure Databricks. Data Lake enables the integration of multiple data source in an unstructured data store.
- **Massive Parallel Processing using Azure Synapse Analytics:** Built based on Azure Data Warehouse, it enables the MPP architecture to serve and process petabytes of data. It enables efficient query and integration with Data Lake Gen2 by leveraging the PolyBase technology.
- **Scalability to thousands of users using Azure Analysis Services:** Analysis Services features in-memory processing allowing the solution to easily scale out to thousands of users without performance penalties. It also enables to provide a single semantic layer on top of your data model.

Integration with SAP

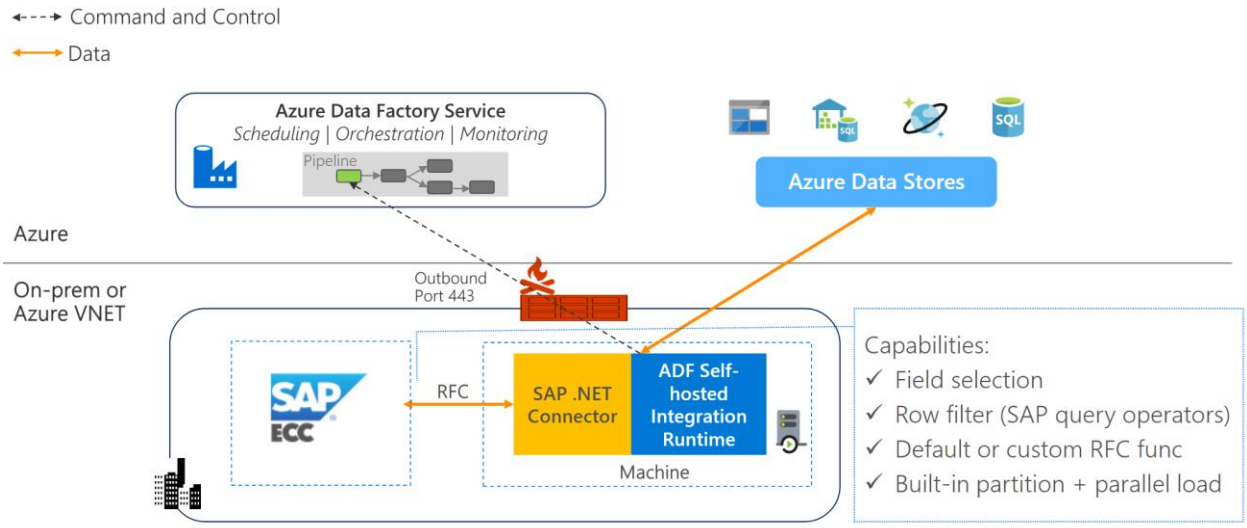
Overview

In the proposed architecture we use Azure Data Factory to extract and load data from SAP. We support 4 modes to achieve such based on your requirements. We are listing bellow all the supported modes **sorted by preference**. Although all supported, we strongly recommend you using SAP Table connector since it provides the best performance when combined with Azure Data Factory:

Connector	Supported objects to extract	Recommended for
SAP Table (Recommended)	Tables (transparent, pooled, cluster tables) and views	Large volumes. Fast with built-in parallel loading based on configurable partitioning
SAP BW Open Hub	DSO, InfoCube, MultiProvider, DataSource, etc	Well-thought-through workload for SAP BW. Built-in parallel loading based on OHD specific schema.
SAP BW via MDX	InfoCubes, QueryCubes	Exploratory workload. Small volume in SAP BW
SAP ECC	OData entities exposed via SAP Gateway (BAPI, ODP)	Small volume in S/4 HANA

Recommended configuration

We recommend using SAP Table each time possible as it provides the best performance for large volumes of data as well as support for both SAP BW and S/4 HANA. The following image shows how SAP Table connector works under the hood when integrated with Azure Data Factory and the supported configurations:



SAP Table supported capabilities

Supported versions	S/4 HANA, SAP ECC, BW or other applications version 7.01 and above, on-premises or in the cloud.
Supported server type	Connect to Application Server or Message Server
Supported SAP objects	SAP Transparent Table, Pooled Table, ClusterTable and View
Supported authentications	Basic (username & passwords) and SNC (Secure Network Communications)
Performance	Built-in parallel loading option based on configurable data partitioning
Mechanism and prerequisites	Built on top of SAP .NET Connector 3.0, pull data via NetWeaver RFC. Run on Self-hosted Integration Runtime via Azure Data Factory

Infrastructure deployment reference

Azure Data Factory

Azure Data Factory is the cloud-based ETL and data integration service that allows you to create data-driven workflows for orchestrating data movement and transforming data at scale. Data Factory is a full web-based platform. We will use Azure Data Factory to create and schedule data-driven workflows (called pipelines) that can ingest data from disparate data stores.

We can also build complex ETL processes that transform data visually with data flows or by using compute services such as Azure Databricks, thought they cannot be detailed in this guide.

Create the resource

Create an Azure Data Factory resource by using the portal in the following URL:
https://portal.azure.com/#create/Microsoft.DataFactory

Configuration

Name	Anyone
Version	V2
Subscription	Anyone
Resource group	We recommend using the same RG for all the resources in the solution
Location	Please consider placing the resources in the same location for best performance and latency and to minimize charges for data transfer.
Enable GIT	Data Factory offers full support for CI/CD of your data pipelines using Azure DevOps and GitHub. This allows you to incrementally develop and deliver your ETL processes before publishing the finished product. We recommend you use a source control solution to track changes. If checked, you will have to provide the repository URL and credentials. This can also be configured later.

Set up a self-hosted integration runtime (version 3.17 or later) for Azure Data Factory:

The integration runtime (IR) is the compute infrastructure that Azure Data Factory uses to provide data-integration capabilities across different network environments. 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. The installation of a self-hosted integration runtime needs on an on-premises machine or a virtual machine (VM) inside a private network.

A self-hosted integration runtime can be created via scripting or via Azure Portal in the Data Factory component. Instructions for the creation can be found here: https://docs.microsoft.com/en-us/azure/data-factory/create-self-hosted-integration-runtime#create-a-self-hosted-ir-via-azure-data-factory-ui

> **Important:** When creating you IR, you will be presented with an authentication key used to secure the connection between the cloud and your data source. Please back-up this key as it can be used later for high-availability purposes (see next note).

Important: We highly recommend installing this IR in a Windows-based machine as close as possible to your data sources to reduce networking and latency accessing to the data. For highly availability purposes we recommend deployment the IR in at least 2 machines. These machines are called nodes for the IR. You can have up to four nodes associated with a self-hosted integration runtime. The benefits of having multiple nodes (on-premises machines with a gateway installed) for a logical gateway are

- Higher availability of the self-hosted integration runtime so that it's no longer the single point of failure in your big data solution
- Improved performance and throughput during data movement between on-premises and cloud data stores.

You can associate multiple nodes by installing the self-hosted integration runtime software. Then, register it by using either of the authentication keys. The following tutorial shows you how to do it: https://docs.microsoft.com/en-us/azure/data-factory/tutorial-hybrid-copy-powershell#install-the-integration-runtime

SAP Table Connector

To use this SAP table connector, you need to:

1. **Download the 64-bit SAP Connector for Microsoft .NET 3.0:** From SAP's website (https://support.sap.com/en/product/connectors/msnet.html), and install it on the self-hosted integration runtime machine. During installation, make sure you select the Install Assemblies to GAC option in the Optional setup steps window.

Azure Data Lake Gen2

We use Azure Data Lake as a landing zone for the data coming for all the sources. Data is later loaded to Azure Synapse Analytics using PolyBase technology. Although there are multiple ways to load data to data warehouse, PolyBase is the fastest way to achieve this goal with minimal impact.

Create the resource:

Create an storage account resource from Azure Portal using the following URL:
https://portal.azure.com/#create/Microsoft.StorageAccount

Configuration

Subscription	Anyone
Resource group	We recommend using the same RG for all the resources in the solution
Storage Account Name	Anyone
Location	Please consider placing the resources in the same location for best performance and latency and to minimize charges for data transfer.
Performance	Standard

Account type	StorageV2
Replication	As required
Access tier	Hot
Connectivity method	As required
Data Lake Storage Gen2	Enabled

Azure Synapse Analytics

Azure Synapse Analytics is the fast, flexible and trusted cloud data warehouse that lets you scale, compute and store elastically and independently, with a massively parallel processing architecture.

Create the resource:

Create an Azure Synapse Analytics resource from Azure Portal using the following URL:
https://portal.azure.com/#create/Microsoft.SQLDataWarehouse

Configuration

Data Warehouse Name	Anyone
Server	Create a new server under the same location that the rest of the resources. Please ensure your check the option “Allow Azure Services to access server”
Performance level	Gen2. The number of Data Warehouse Units (DWU) will depend of the size and requirements of your data. As a starting point you can put 1000 DWU considering that you are working with big data loads. You can always scale down or up later.
Subscription	Anyone
Resource group	We recommend using the same RG for all the resources in the solution

Azure Analysis Services

Analysis Services is a fully managed service that provides data modeling capabilities. The semantic model is loaded into Analysis Services and it is where uses connect to discover the underlying data. Analysis services is also able to handle the concurrent connections and scalability of multiple users effectively in memory.

Create the resource:

Create an Azure Analysis Services resource from Azure Portal using the following URL:
https://portal.azure.com/#create/Microsoft.AnalysisServices

Configuration

Server name	Anyone
Pricing tier	The pricing tier will depend on the size of the data and the processing requirements. You can start with S2 considering this is a big data solution, but you can scale up or down later.
Administrator	Select an Azure Active Directory account as the administrator of Azure Analysis Services. This account will be the owner of the service.
Backup storage settings	As needed
Storage key expiration	As needed
Subscription	Anyone
Resource group	We recommend using the same RG for all the resources in the solution

Azure Logic Apps

Logic Apps is a service in Azure that allows to implement task automation using a workflow like design. We use logic apps to trigger updates in Azure Analysis Services using its REST API.

Create the resource:

https://portal.azure.com/#create/Microsoft.EmptyWorkflow

Solution deployment reference

Based on the resources deployed before we are now presenting how to construct an Enterprise BI Data Warehouse solution. We will use Azure Data Factory as the orchestration for all the resources. The main element inside of Data Factory that will allow you to construct a solution is a Pipeline. Although there are several ways to orchestrate the solution, we will provide here a potential one.

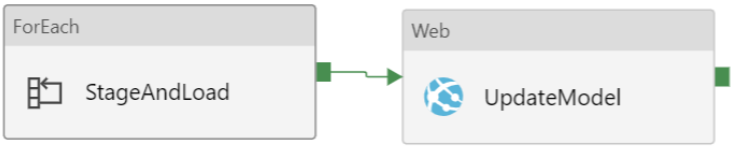
Data Factory

The main building blocks of Azure Data Factory are Pipelines and Activities:

Pipeline
A pipeline is a logical grouping of activities that performs a unit of work. A data factory might have one or more pipelines. Together, the activities in a pipeline perform a task. For example, a pipeline can contain a group of activities that ingests data from SAP and save it to Blob Storage, and then runs a Hive query on an Spark cluster to partition the data.
The benefit of this is that the pipeline allows you to manage the activities as a set instead of managing each one individually. The activities in a pipeline can be chained together to operate sequentially, or they can operate independently in parallel.

Activity
Activities represent a processing step in a pipeline. For example, you might use a copy activity to copy data from one data store to another data store. Similarly, you might use a Hive activity, which runs a Hive query on an Azure HDInsight cluster, to transform or analyze your data. Data Factory supports three types of activities: data movement activities, data transformation activities, and control activities.

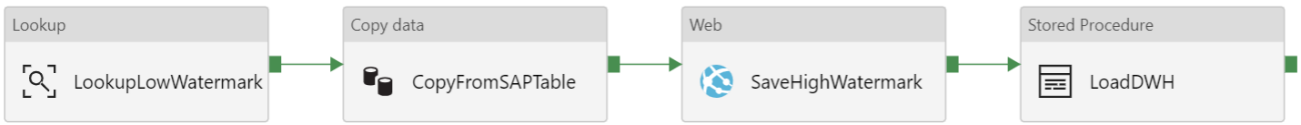
A loading activity can be orchestrated using a pipeline that iterates over all the data sources and execute the loading procedure and finally, once all the data is loaded, update the data model.



Activities:

- **StageAndLoad**: It is a Foreach activity which allows Data Factory to execute a set of activities repetitively. This set of activities are packaged in another pipeline. In this pipeline we stage the data from the data source into the cloud and then we load the data into our data warehouse solution. This activity is able to execute the loading procedure in parallel for each of the sources, without delaying the rest of them. This means that the execution time will be upper bounded by the time of the longest loading.
- **UpdateModel**: This activity executes an update in the model running in Azure Analysis Services. Since the pipeline won't move to this activity until all the previous loading activities success, the consistency of the model is guaranteed. The update is triggered using the Azure Analysis Services REST API implemented using an Azure LogicApp that encapsulates all the logic to update the model.

In the StageAndLoad activity, the loading procedure is implemented as follows and it is executed for each of the SAP tables you want to move to the cloud:

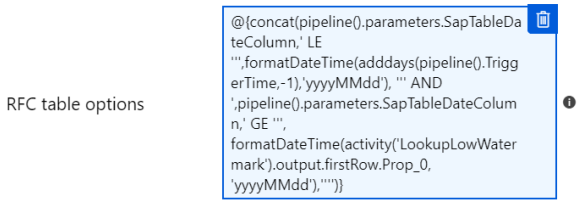


To efficiently load data in a cloud you may want to implement incremental loading for big tables. This is the case of the pipeline presented before which uses a column as a watermark to efficiently query the data source accordingly to the data generated. Tables that do not have a column to use for this purpose (always incremented ID, date time, etc) has to be loaded completed. This is usually the case of fact tables for instance.

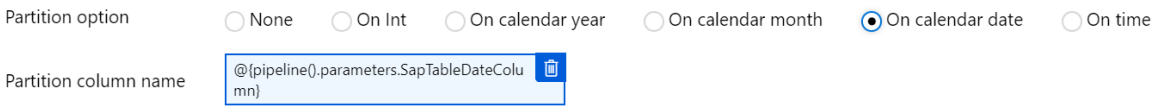
Activities:

- **LookupLowWatermark**: This activity looks for the last copied key value for the SAP Table. This is the LowWatermark. The storage of this values can be anyone you are familiar with including: A SQL Database, Azure Table, Azure Blob (file), etc. In this case we are using a text file in a blob storage. A lot of users may be comfortable creating a SQL Database to store these values.

- **CopyFromSAPTable:** This activity executes the copy of the data from the data source on-premises to the landing zone using Azure Data Lake Gen2. The query executed against the data source should include filtering for the specific watermark so incremental loading can be implemented.



The activity also supports parallel loading against the data source using partitioning. We recommend the use of this feature if the data source is big.



The destination of the copy activity will be an Azure Data Lake Gen2. We recommend the use of directory structures that will be used later to efficiently query the data.

- **SaveHighWatermark**: this activity will save the high watermark for the copy activity executed before. This is the counterpart of the Lookup activity executed at the very beginning and therefore can be implemented in different ways. In this case, we are using the REST API for Azure Blob Storage to update the values.

- **LoadDWH**: This activity will update and load the data inside the data warehouse using the staging data loaded in the Data Lake. There are multiple ways to implement this, but one of the most efficient ways to load the data is to stage the data as quickly as possible into the Data Warehouse in an staging table then switch the staging table for the productive table and finally drop the staging table. All this logic can be encapsulated in a store procedure, which is the one that this activity executes.

  The advantages of this method is that it loads the data in staging as quickly as possible and since the loading is done in a different table rather than the production, it reduces the effect of dirty reads, integrity and downtime perceived by the users.

  To query data stored in a data lake we can use External tables which allows Azure Synapse Analytics to leverage the PolyBase technology to query data efficiently. External tables can be created as follows:

```
CREATE EXTERNAL TABLE [dbo].[AUFK]
(
     ERDAT VARCHAR(8),
     KTEXT VARCHAR(64),
     (…)
)
WITH
```

```
(
    LOCATION = '/SAPS4HANA18TBL/AUFK/',
    DATA_SOURCE = saps4hana18dlg2,
    FILE_FORMAT = TextFileCsvFormat
)
```

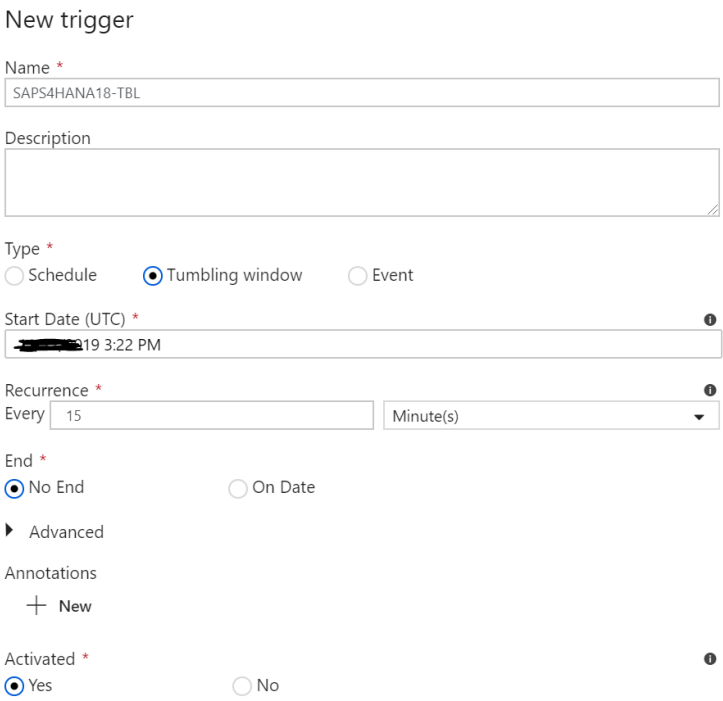
This procedure must be executed for each of the data sources so they can be accessed from the data warehouse using SQL. Notice that external tables do not load data into the data warehouse.

Once the external table is created, the procedure can query the data and loaded as described at the beginning. A simple version of the store procedure is included here, but the actual implementation may vary depending on your data.

```
ALTER PROC [dbo].[LoadFromExternalTablesToStage]
@TableName nvarchar(128),
@Watermark nvarchar(128),
@HighWatermark date,
@LowWatermark date
AS
BEGIN
      DECLARE @query NVARCHAR(MAX);
      DECLARE @stagingTable NVARCHAR(MAX);
      DECLARE @prodTable NVARCHAR(MAX);
      DECLARE @tempTable NVARCHAR(MAX);

      SET @stagingTable = N'staging_' + @TableName
      SET @prodTable = N'prod_' + @TableName
      SET @tempTable = N'temp_' + @TableName

      SET @query =
      N'CREATE TABLE ' + @stagingTable + '
      WITH
      (
            CLUSTERED COLUMNSTORE INDEX,
            DISTRIBUTION = ROUND_ROBIN
      )
      AS
      SELECT *
      FROM ' + @TableName + '
      WHERE ' + @Watermark + ' > ' + @LowWatermark + ' AND ' + @Watermark + ' <= ' +
@HighWatermark
      EXEC executesql @query

      SET @query = 'RENAME OBJECT ' + @prodTable + ' TO ' + @tempTable
      EXECUTE sp_executesql @query

      SET @query = 'RENAME OBJECT ' + @stagingTable + ' TO ' + @prodTable
      EXECUTE sp_executesql @query

      SET @query = 'DROP TABLE ' + @tempTable
      EXECUTE sp_executesql @query
END
```

Scheduling

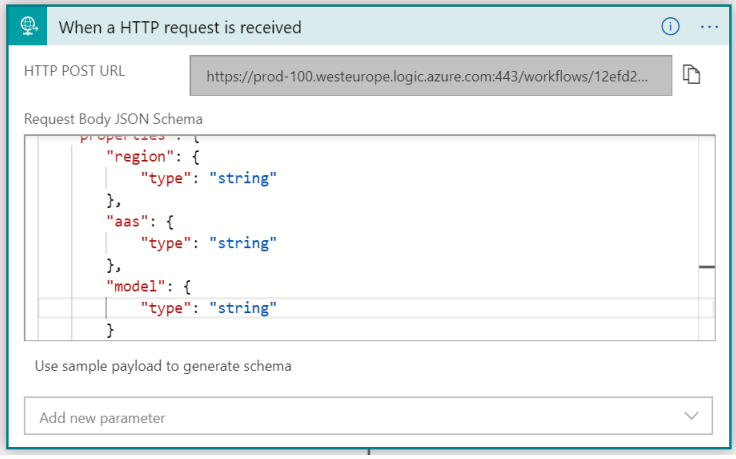
Once the pipeline is created and tested, we need to schedule its execution. This should be configured depending on the size of the data, the business requirements and the capabilities of the resources you are deploying. In the following example, we configure the execution each 15 minutes:



Azure Logic Apps

We use Logic Apps to trigger the updates in Azure Analysis Services. This task can be implemented using its REST API. A generic way to implement the refresh of any model stored in Azure Analysis Services can be done as follows using a trigger and an action:

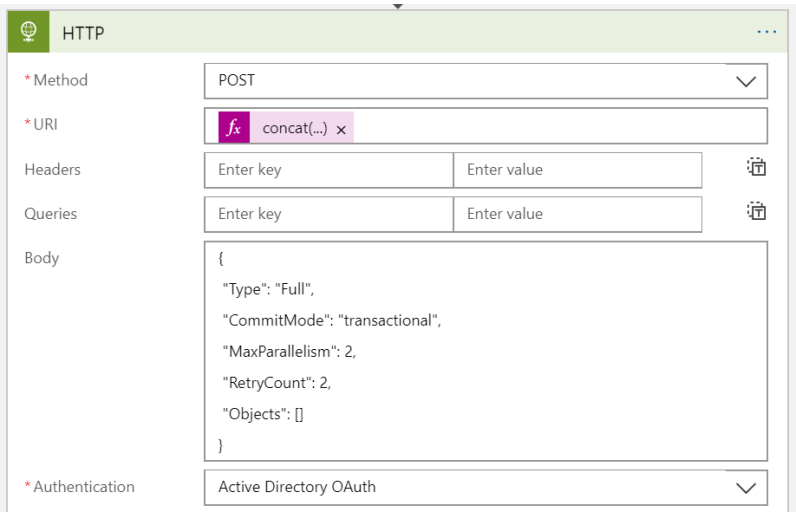
Trigger:



This trigger makes our workflow to fire each time it receives an HTTP post. This post will be invoked from Azure Data Factory. The post will include 3 parameters to identify which model to refresh:

- Region: the region of the Azure Analysis Services that will be updated
- AAS: The name of the service
- Model: The name of the model to update inside of the service.

Action:



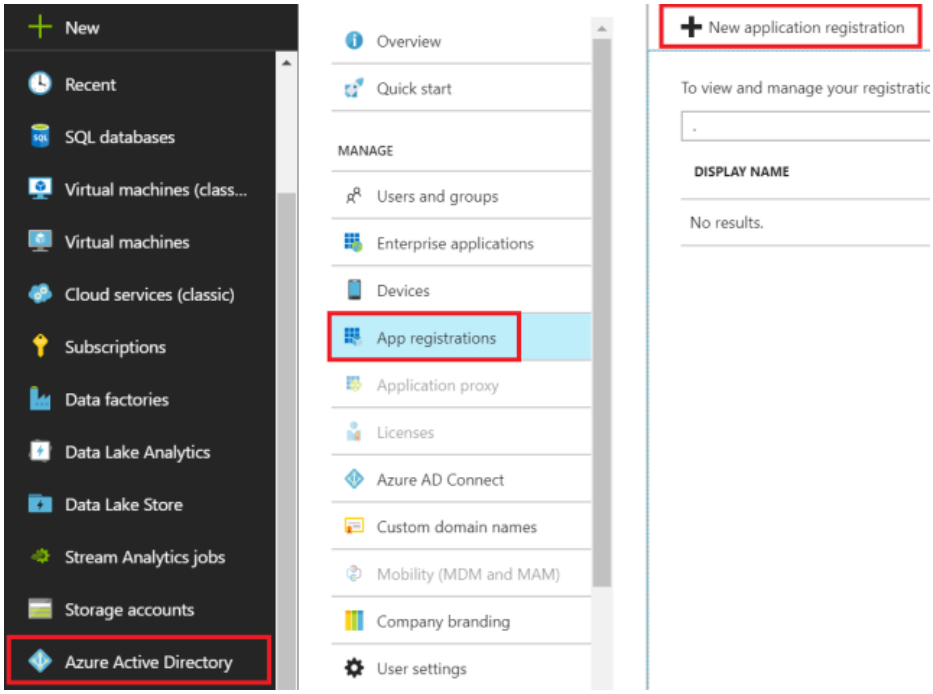
The action will trigger another HTTP call to the Azure Analysis Services. The URI that is invoked is dynamically created using a function as follows:

```
concat('https://',triggerBody()?['region'],'.asazure.windows.net/servers/',triggerBody()?['aas'],'/models/',triggerBody()?['model'],'/refreshes')
```

In the body of the query we can specify how we want the update to be executed. Note that we are using Active Directory OAuth authentication mechanism which is orchestrated using a specific identity created in Azure Active Directory (the app id and secrets are not included in the screenshot).

Security considerations

To create External Tables in Azure Synapse Analytics and to interact with several services using REST API you will have to choose a way to securely connect with the services. We recommend you create an Active Directory Application to use for all these interactions, so it is easy to track and audit the data flow. Create an Azure Active Directory Application in the Azure Portal using the following URL:



Once registered, you can use that Application as the identity to assign permissions for your different components when interacting with each other. Although this is not the only way to implement access, it is the preferred way.

This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for informational purposes only. Microsoft makes no warranties, express or implied, in this document.

The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.