# Constructing On-Premise Data Pipeline into Azure for AML operationalization

Data driven decision making lies at the center of all fast growing companies. A vast majority of companies use SQL servers for data storage, aggregation and number crunching. This data can also be used to forecast machine breakdowns, detect anomaly, or to manage cost, etc. While data analytics can give very precise answers to static data questions but it lacks ability to predict trends, behaviors or anomalies in various processes, with continuously improving high degree of accuracy. Machine leaning fills this gap.

## Scope

This document helps you create an E2E deployment ready data pipeline for consuming an Azure Machine Learning (AML) solution in your on-premise SQL server.

This document will describe:

* Overall approach for operationalizing an on-premise to azure data pipeline
* An Ingress data pipeline that will bring in data for training and then subsequently for scoring (after a trained model is ready and published as a web service).
* An AML data pipeline to send incoming data (from on-premise data center in to Azure) for AML batch scoring and store the scores.
* An Egress data pipeline that will send the scores back to the on-premise data center (SQL server)
* Create sample data source in your SQL server wherever needed.
* A PowerBI dashboard to show results of scoring.

By the end of the document you should be able to create a fully operationalized data pipeline in Azure that will (1) connect to your on-premise SQL server data source, (2) call AML batch execution service endpoint for scoring, and (3) copy the scores back to your on-premise SQL server data source. This pipeline will be scheduled to run automatically on an hourly schedule for you (until you stop it).

## Assumptions

This document assumes user has an existing pre-published AML model and we will create operationalized machine learning model using the BES endpoint of that model. To create and publish a new AML model follow instructions [here](2_AMLScoringPipeline/CreateAMLWebServices.docx). Second requirement is that in order to follow the instructions to create an operational pipeline use this document, the reader *MUST* have access (read / write) to an on-premise SQL server.

## Big Picture

Building Azure Machine Learning model requires historical data, and perhaps lots of it based on the quality of the data. This is the reason why it is important to think about building data pipelines not just for building the model but also for getting scores from the model and integrating the results with your current business processes. For our current use case we have divided this pipeline into 3 discrete parts each serving a specific objective and can be tested separately if desired. We start by giving you an over view of the big picture below. And then we will briefly define various Azure platform services and tools we will use. Finally we will list step by step instructions you can follow to create an E2E data pipeline.

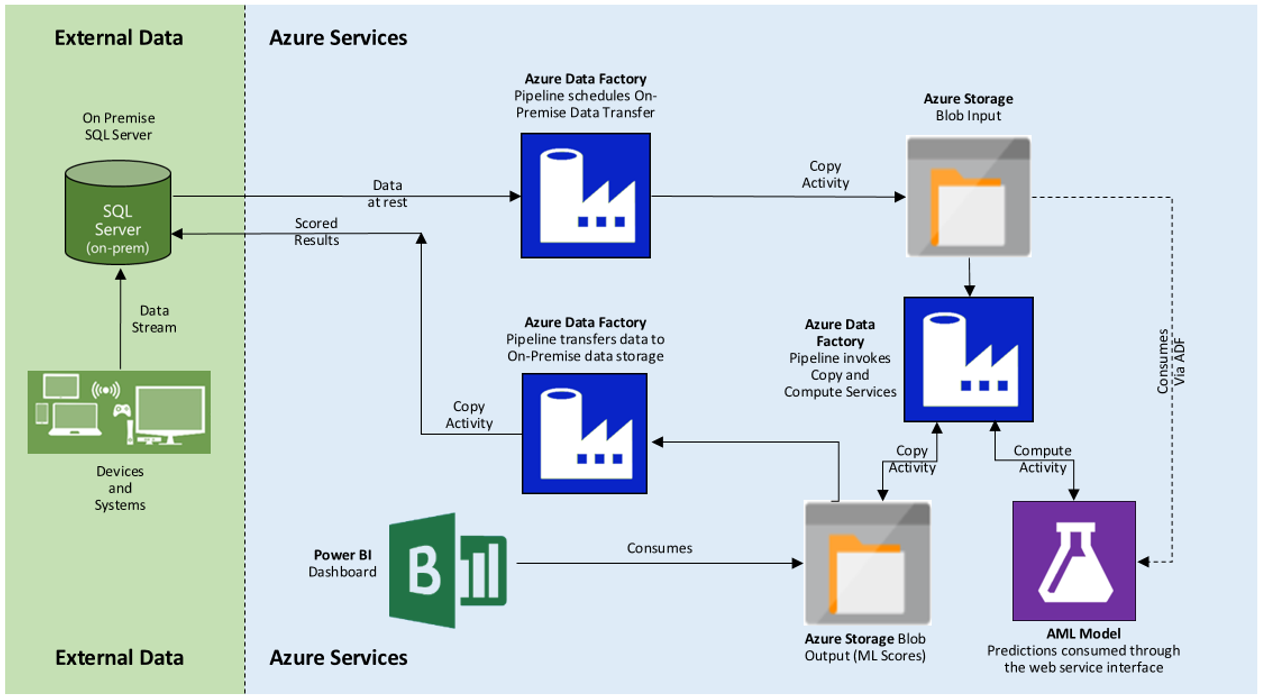


Figure 1: Overall Architecture of E2E data pipeline for On-Premise data source

In the architecture shown above, we have two key consumer and producer services – On-Premise SQL Server and Azure Machine Learning (AML) model. The on-premise database (hosted on SQL Server) servers as a data source and a data sink. The AML model consumes the data hosted by the on-premise database and produces prediction data that is copied back to the on-premise SQL database (consumer). We use Azure Data Factory (ADF) to move the data between on premise database and Azure as well as for calling AML web service endpoint for scoring batches of input data. An ADF can schedule execution of certain predefined activities (such as, copy or compute activity) and also allow for adding custom activities.

The data pipeline is divided in to three section as follows:

*Azure Data Factory is a service for orchestrating data storage, processing, and movement.*

1. We use an ADF to first used to copy data from an on-premise SQL Server to an Azure Blob Storage (we will refer this as *Ingress OnPrem Pipeline*).
2. The ADF then invokes ‘compute’ activity using AML model (using AML Batch Execution Service endpoint) for prediction and copies the predicted results to an output blob location (we will refer this as *AML Scoring Pipeline*).
3. The scored data from the output blob location is then copied to the on-premise SQL server for integration with the rest of the business process (we will refer this as *Egress OnPrem Pipeline*).

The scored data is also consumed by a dashboard that is published in Azure cloud using PowerBI. This dashboard shows hourly and daily ATM machine health scores. The dashboard is available anywhere from over the internet.

## Walkthrough

### Create ADF and Setup Data Management Gateway

First few steps in creating an E2E data pipeline to connect to an on-premise SQL server is to create an *Azure Data Factory* and install *Data Management Gateway.* If you are not familiar with the process you can find more detailed instructions on Azure Data Factory [getting started](http://azure.microsoft.com/en-us/documentation/articles/data-factory-get-started/) webpage. Follow steps to [create an Azure Data Factory](http://azure.microsoft.com/en-us/documentation/articles/data-factory-use-onpremises-datasources/#step-1-create-an-azure-data-factory) and [install the data management gateway](http://azure.microsoft.com/en-us/documentation/articles/data-factory-use-onpremises-datasources/#step-2-create-a-data-management-gateway) before returning to this document. As you follow instructions for creating ADF use the following values for ‘ResourceGroupName’ and ‘DataFactoryName’.

ResourceGroupName => *OnPremPipelineRG*

DataFactoryName => *OnPremPipeline*

### Create Sample Input Data

Next, on an on-prem SQL Server we create tables called ‘InputStatsData’ and ‘Scores’. Then, we insert some sample data into the *InputStatsData* table that acts as input data source for the data pipelines in the next section. The *Scores* table will hold the output from the Azure machine learning model. Use the [SQL script](SQLScripts/CreateTableAndInsertData.sql) provided with this package to create required input and output tables and sample insert data.

If the reader has an existing dataset or table in an on-prem SQL server that she wants to use for scoring, she can modify the "tableName” property in [InputOnPremSQLTable](file:///C:\Users\mithal\Documents\StarterKit\Source\Azure-MachineLearning-DataScience\Misc\Advanced%20Analytics\OnPremDataSource\1_IngressOnPremPipeline\InputOnPremSQLTable.json) definition to reflect the desired table name from the SQL database. Default value for this property is shown below.

*"tableName": "InputStatsData"*

The user must still create an output table called ‘Scores’ described earlier to store on-premise the results of AML predictions (a.k.a scoring).

### Create Data Pipelines

Finally, we will create 3 data pipelines (described earlier in this document) all in within the contest of the ADF we just created. Each data pipeline constitute of at least one input dataset, one on output dataset and an activity that transforms the input into the output. These activities could be simple (data) copy activity, compute activity (such as AML scoring) or some custom defined activity (such as ETL).

### The Ingress On-Premise Pipeline

The ingress pipeline is the first section of the overall data pipeline we will create in this document. Figure below highlights the ingress portion from the overall architecture.

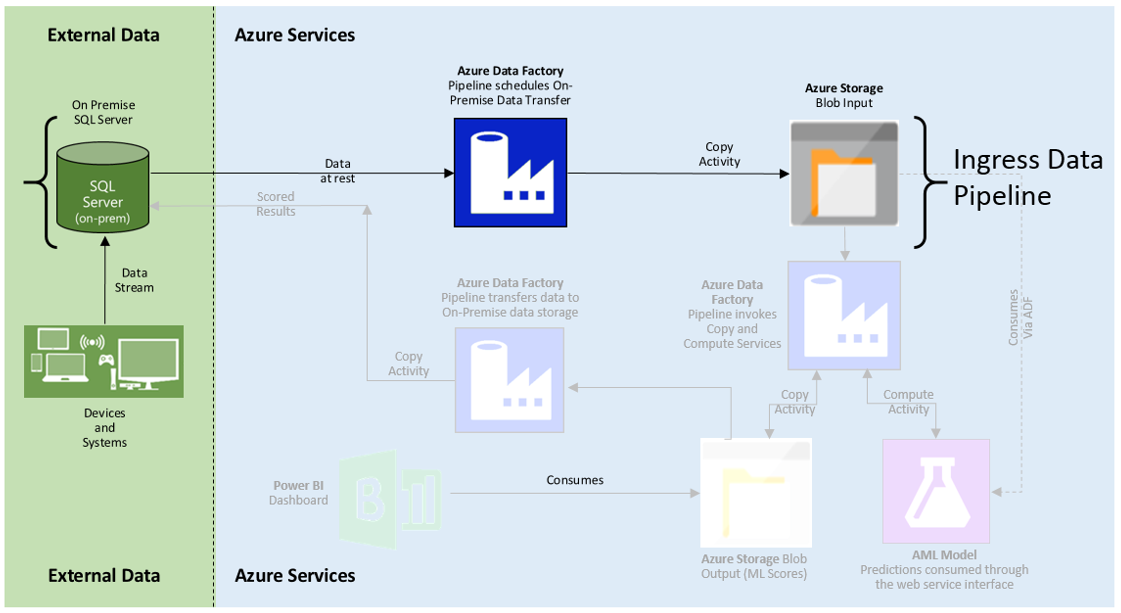


Figure 2: Ingress On-Premise Pipeline

This pipeline connects your company’s on-premise SQL Server (in your data center or lab) to Azure Cloud via the data management gateway installed in the one of the previous steps. The *ADF* [*pipeline*](http://azure.microsoft.com/en-us/documentation/articles/data-factory-introduction/#terminology) (shown in blue in the picture above) itself will run a *copy* activity to copy the data between SQL server database(s) and Azure Blob Storage. This is a good time to quickly take a break and review some of the ADF terminology [here](http://azure.microsoft.com/en-us/documentation/articles/data-factory-introduction/#terminology) and then install [Azure PowerShell](http://azure.microsoft.com/en-us/documentation/articles/powershell-install-configure/#Install) before continuing further.

Next we will use ADF [pipeline configuration scripts](1_IngressOnPremPipeline/IngressPipeline.json) that came with this document package to create the Ingress Data Pipeline.

Creating Input and Output tables and the pipeline:

1. Prepare Azure PowerShell to run ADF commands. Open PowerShell and run the following commands:

a) **Switch to Azure PowerShell** mode:

Switch-AzureMode AzureResourceManager

b) **Add your Azure Subscription** Account:

Add-AzureAccount

1. Create Linked Services: Each dataset (input or output) used in an ADF pipeline is backed by an underlying ***linked service*** that allows connection to the data store where the dataset resides. Each data store (storing input or output datasets for the ADF pipeline) must be uniquely defined as a new linked service. If both the input and the output datasets reside in the same data store (for example, same blob container, or same SQL database) then the same liked service can be used to connected to both input and output datasets . Next, we will define **two linked services** – one to connect to your **on-premise database** and another to an **azure blob storage** where your data is stored.

a) In the PowerShell run the following command and provide it the location of [OnPremSQLLinkedService](1_IngressOnPremPipeline/OnPremSQLLinkedService.json) definition on your computer to create an ADF *LinkedService* connecting to your on-premise SQL server.

New-AzureDataFactoryLinkedService -ResourceGroupName OnPremPipelineRG -DataFactoryName OnPremPipeline –File < location of SQL Linked Service definition >

b) In the PowerShell run the following command and provide it the location of [BlobLinkedService](1_IngressOnPremPipeline/BlobLinkedService.json) definition on your computer to create an ADF *LinkedService* connecting for your Azure Blob Storage Account.

New-AzureDataFactoryLinkedService -ResourceGroupName OnPremPipelineRG -DataFactoryName OnPremPipeline –File < location of SQL Linked Service definition >

1. Create input and output datasets: Now having linked services defined, we can use them to create ***dataset*** schema, partitioning and availability schedule. This allows the ADF pipeline to know where, how and when to read the input and write the results of ‘*activities’* defined in the pipeline. Next we will **define two datasets** – one that describes the schema and availability schedule of the table (containing input data) in your **on-premise database** and another that describes the partition, and schema and availability schedule for the **output blob store**.
2. Create SQL Table Schema: Open PowerShell and run the following command and provide it the location of [InputOnPremSQLTable](1_IngressOnPremPipeline/InputOnPremSQLTable.json) definition on your computer to create an ADF schema (table) mapping for your on-premise SQL table containing the input data for the machine learning model.

New-AzureDataFactoryTable -ResourceGroupName OnPremPipelineRG -DataFactoryName OnPremPipeline –File < location of [InputOnPremSQLTable](1_IngressOnPremPipeline/InputOnPremSQLTable.json) definition >

1. Next, in the PowerShell run the following command and provide it the location of [RawOutputBlobTable](1_IngressOnPremPipeline/RawOutputBlobTable.json) definition on your computer to create an ADF schema (table) mapping for Azure Blob Store that stores the data copied from your on-premise SQL table.

New-AzureDataFactoryTable -ResourceGroupName OnPremPipelineRG -DataFactoryName OnPremPipeline –File < location of [RawOutputBlobTable](1_IngressOnPremPipeline/RawOutputBlobTable.json) definition >

1. Finally, in the PowerShell run the following command and provide it the location of [IngressPipeline](1_IngressOnPremPipeline/IngressPipeline.json) definition on your computer to create an ADF pipeline that **connects on-premise SQL server table to the Azure blob storage** defined above and invokes ***‘copy’ activity*** to be run once every hour.
2. Create the Ingress data pipeline:

New-AzureDataFactoryPipeline -ResourceGroupName OnPremPipelineRG -DataFactoryName OnPremPipeline –File < location of [IngressPipeline](1_IngressOnPremPipeline/IngressPipeline.json) definition >

### The AML Scoring Pipeline

The AML scoring pipeline is show in Figure 3. This pipeline connects to the AML input blob location (defined as RawOutputBlobTable) and to the AML output blob location (defined as ScoredOutputBlobTable). Unlike the ingress pipeline, the AML scoring pipeline run a *compute* activity that calls in the AML web service batch execution endpoint, pass in the data from AML input blob location as a batch to the web service and writes the results back to the AML output blob location.

Next we will use ADF [pipeline configuration scripts](2_AMLScoringPipeline/AMLPipeline.json) that came with this document package to create the AML Scoring Pipeline.

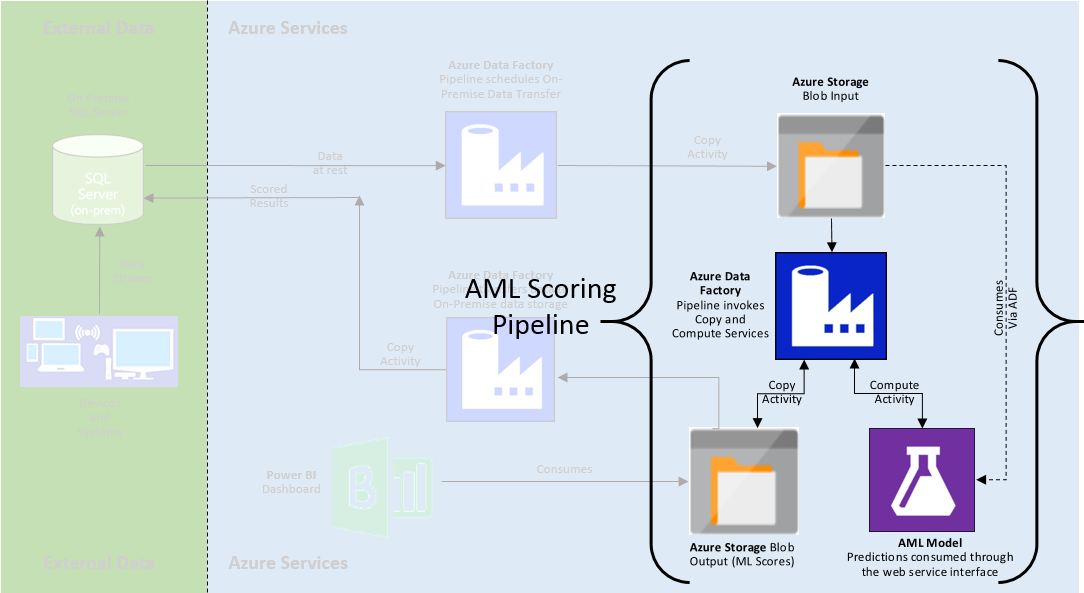
**

Figure 3: AML Scoring Pipeline

Creating AML Linked Service and the pipeline:

In this pipeline we use the output blob dataset from the *Ingress Pipeline* as an input and define a new blob store dataset as the output of this pipeline and an activity that invokes AML web service to transforms the input into the output. In order to invoke AML web service as an *activity* we will to create a new linked service that connects the AML web service batch scoring endpoint to the ADF pipeline.

1. In the PowerShell run the following command and provide it the location of [AML Linked Service](2_AMLScoringPipeline/AMLLinkedService.json) definition on your computer to **link your AML batch execution endpoint** to this ADF pipeline.
2. **Create Linked Service**:

New-AzureDataFactoryPipeline -ResourceGroupName OnPremPipelineRG -DataFactoryName OnPremPipeline –File < location of [AML Linked Service](file:///C:\Users\mithal\Documents\StarterKit\Source\Azure-MachineLearning-DataScience\Misc\Advanced%20Analytics\OnPremDataSource\2_AMLScoringPipeline\AMLLinkedService.json) definition >

1. **Create Blob Storage** dataset: Just like in data ‘Ingress Pipeline’ In the PowerShell run the following command and provide it the location of ScoredOutputBlobTable definition on your computer to create an ADF schema (table) mapping for Azure Blob Store that stores the AML scores.
2. New-AzureDataFactoryTable -ResourceGroupName OnPremPipelineRG -DataFactoryName OnPremPipeline –File < location of ScoredOutputBlobTable definition >
3. Finally, in the PowerShell run the following command and provide it the location of [AMLScoringPipeline](2_AMLScoringPipeline/AMLPipeline.json) definition on your computer to create an ADF pipeline that **connects input and output blob storages** and invokes *“****AMLBatchScoringActivity”*** to be run every hour.
4. Create the AML scoring pipeline:

New-AzureDataFactoryPipeline -ResourceGroupName OnPremPipelineRG -DataFactoryName OnPremPipeline –File < location of [AMLScoringPipeline](file:///C:\Users\mithal\Documents\StarterKit\Source\Azure-MachineLearning-DataScience\Misc\Advanced%20Analytics\OnPremDataSource\2_AMLScoringPipeline\AMLPipeline.json) definition >

### The Egress Pipeline

The two pipelines created so far together bring the data into Azure for AML model and produce AML scores on an ongoing bases (stored in a temporally partitioned azure blob storage). Next pipeline in this process will allow us to copy the scored results back to your on-premise SQL Server. Figure 4 shows the Egress Data Pipeline.

Next we will use ADF [pipeline configuration scripts](3_EgressOnPremPipeline/EgressPipeline.json) that came with this document package to create the Egress Data Pipeline.

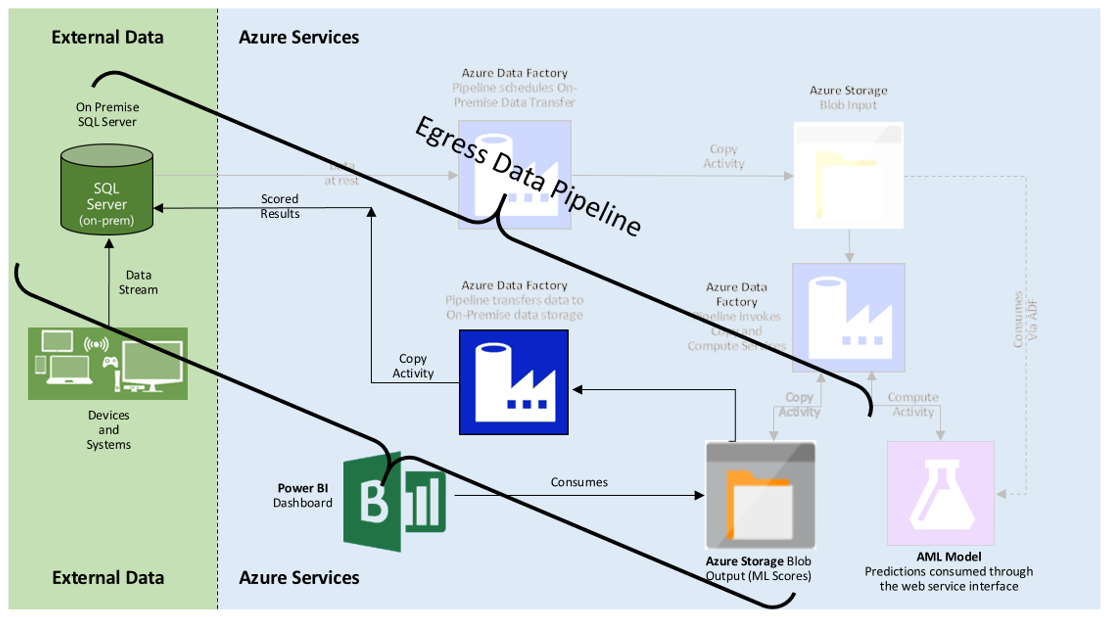


Figure 4: Egress On-Premise Pipeline

Creating Input and Output tables and the pipeline: In this pipeline we use the output blob dataset from the *AML Scoring Pipeline* (containing predicted results) as an input, define an on-premise SQL table dataset as the output of this pipeline and an activity that invokes ***‘copy’ activity*** to be run once every hour.

1. On-premise SQL Table definition: In PowerShell and run the following command and provide it the location of [OutputOnPremSQLTable](3_EgressOnPremPipeline/OutputOnPremSQLTable.json) definition on your computer to create an ADF schema (table) mapping for your **on-premise SQL table where the scores from the machine learning model will be stored**. It is worth noting that you can a chose different SQL database to write the score results back. In that case you will need to define a new *Linked Service*, similar to the one we created in *Ingress Pipeline* connecting to the database storing the output dataset table.

a) Create SQL Table Schema:

New-AzureDataFactoryTable -ResourceGroupName OnPremPipelineRG -DataFactoryName OnPremPipeline –File < location of [OutputOnPremSQLTable](3_EgressOnPremPipeline/OutputOnPremSQLTable.json) definition >

1. Finally, in the PowerShell run the following command and provide it the location of [EgressPipeline](3_EgressOnPremPipeline/EgressPipeline.json) definition on your computer to create an ADF pipeline that **connects your on-premise SQL server table to the Azure blob storage** to copy the scores back to your on-premise server.
2. Create the Egress data pipeline:

New-AzureDataFactoryPipeline -ResourceGroupName OnPremPipelineRG -DataFactoryName OnPremPipeline –File < location of [EgressPipeline](file:///C:\Users\mithal\Documents\StarterKit\Source\Azure-MachineLearning-DataScience\Misc\Advanced%20Analytics\OnPremDataSource\3_EgressOnPremPipeline\EgressPipeline.json) definition >

### Putting it all together - *Activating* the Pipeline

With all the ADF pipelines in place we are ready to **activate the ADF end to end flow**. In the power shell run the following commands to activate the E2E *OnPremPipeline* ADF pipeline.

1. Set-AzureDataFactoryPipelineActivePeriod -ResourceGroupName OnPremPipelineRG-DataFactoryName OnPremPipeline-StartDateTime 2015-03-24T21:30:00 –EndDateTime 2015-03-24T21:45:00 –Name IngressPipeline
2. Set-AzureDataFactoryPipelineActivePeriod -ResourceGroupName OnPremPipelineRG-DataFactoryName OnPremPipelineRG-StartDateTime 2015-03-24T21:30:00 –EndDateTime 2015-03-24T21:45:00 –Name AMLPipeline
3. Set-AzureDataFactoryPipelineActivePeriod -ResourceGroupName OnPremPipelineRG-DataFactoryName OnPremPipelineRG-StartDateTime 2015-03-24T21:30:00 –EndDateTime 2015-03-24T21:45:00 –Name EgressPipeline

After you activate the pipelines you will see input data copied to Azure blob, scored by AML and finally AML scores available in your on-premise SQL server every hour. The pipelines are activated sequentially so any failure in one if the pipelines will stop the pipelines down the stream from running and producing garbled results.

### Visualizing the Results

The last and most important step in any solution operationalization is to visualize and integrate the results back into your business process. The egress pipeline discussed in the last step enables you to integrate the machine learning predictions back into your on-premise SQL server. Next we are going to create a dashboard in PowerBI to visualize the results from the machine learning model.

#### Acquiring the data

Open up a new excel workbook and click on the *Power Query* tab. If you do not have a Power Query tab then install the plugin [here](http://www.microsoft.com/en-us/download/details.aspx?id=39379). Following the next few steps to add Azure blob as a data model which would then be uploaded to the PowerBI.

* Click on from other sources and select [*From Microsoft Azure Blob Storage*](Images/DataModel/1PowerBI-AzureBlobStore.png)*.*
* Next you will need to enter your blob storage [account name](Images/DataModel/3BlobStoreSetting.png) and [keys](Images/DataModel/4BlobStoreSetting2.png) (‘starterkit’ in our example here). If you do not know these you can get these from the Azure portal [here](https://manage.windowsazure.com/).
* If the credentials are entered correctly you will [see your blob container](Images/DataModel/5PowerQueryNavigator.png) (‘*onprem’* in our example here containing the AML predictions). Double click on the container to open it in Query Editor.
* In the editor window, examine that the data looks like what you expected and then click on [‘*Close & Load To’*](Images/DataModel/6LoadDataModel.png)*.*
* On the [screen dialog box that follows](Images/DataModel/6xLoadDataModel.PNG) select “Only Create Connection” and click on the checkbox “Add this data to the Data Model”. This should [load your data](Images/DataModel/7DataModel.png) to the excel workbook.
* Save the excel workbook by proving an appropriate name.

#### Creating a Dashboard

Next we logon to the PowerBI webpage, create a new dashboard and upload excel workbook we created in the last section containing the data from the data model.

* Visit [Power BI webpage](https://msit.powerbi.com/) and click on the [new dash board button](Images/NewDashboard/8NewDash.PNG) (+ sign) and give it an appropriate name of your choice.
* Click on [Get Data](Images/NewDashboard/8NewPBIDashboard.png). Select [Excel Workbook](Images/NewDashboard/9LoadExcellSource.png) and navigate to the storage location where you save the workbook created in the last step.
* Double click on the new dataset that gets added at the [bottom of the home screen](Images/NewDashboard/9NewDataset.PNG) (*ScoredDataForOnPren* in our example here). This opens up a report that you can add different visualizations to.
* You can [simply drag and drop various fields](Images/NewDashboard/10CreateVisualizations.png) on the left of the report to the center of the canvas. And then select the axis on which you want to display each of these values.
* When you are done clink on  icon to pin the visualization to your dash board. You can share this dash board with others by clicking on the share icon.

#### Auto Refreshing Dashboards

In the last 2 sections we created a new dashboard and added visualizations around the AML predictions data that is produced every hour by the E2E pipeline we created earlier in this document. In this section we will add a refresh schedule to the dashboard so that as the new predictions become available from the pipeline, the dashboard visualizations are updated automatically.

* Right click on the newly add dataset [at the bottom of the home screen](Images/AutoRefreshingDashboard/11-1RefreshScedule.png) (*ScoredDataForOnPren* in our example here) and select “*SCHEDULE REFRESH*”.
* Under [“Edit Credentials”](Images/AutoRefreshingDashboard/11-2RefreshScedule.png) enter your blob [storage account key](Images/AutoRefreshingDashboard/11-3RefreshScedule.png) (key for ‘starterkit’ azure blob storage container in our example here).
* Now under “*Refresh Schedule*” flick the [‘Keep you data up to date’](Images/AutoRefreshingDashboard/11-4RefreshScedule.png) button to Yes. And provide refresh schedule (‘Daily’ in our example here). Hit apply.
* Visualization you pinned on your dashboard should be daily refreshed now.