# Energy Demand Forecasting in Cortana Intelligence Suite

Contents

[Energy Demand Forecasting in Cortana Intelligence Suite 1](#_Toc454292363)

[Abstract 3](#_Toc454292364)

[Requirements 3](#_Toc454292365)

[Architecture 3](#_Toc454292366)

[Setup Steps 4](#_Toc454292367)

[1. Create a new Azure Resource Group 5](#_Toc454292368)

[2. Azure Storage Account 5](#_Toc454292369)

[Create the storage account 5](#_Toc454292370)

[Prepare the storage containers 6](#_Toc454292371)

[3. Azure Event Hub 6](#_Toc454292372)

[Check Event Hub 7](#_Toc454292373)

[4. Azure Stream Analytics Jobs 8](#_Toc454292374)

[Create Stream Analytics Jobs 8](#_Toc454292375)

[Configure Stream Analytics Jobs 8](#_Toc454292376)

[Configure Stream Analytics Jobs - Inputs 8](#_Toc454292377)

[Configure Stream Analytics Jobs - Query 9](#_Toc454292378)

[Configure Stream Analytics Jobs - Outputs 10](#_Toc454292379)

[Check Stream Jobs 11](#_Toc454292380)

[5. Configure Data Generator and test Event Hub / Stream Analytics 11](#_Toc454292381)

[Validating initial data generation 11](#_Toc454292382)

[6. Azure SQL Server and Database 12](#_Toc454292383)

[7. Create Azure Studio ML Workspace and Experiment 13](#_Toc454292384)

[8. Azure Data Factory 14](#_Toc454292385)

[Create Data Factory 15](#_Toc454292386)

[Create Linked Services 15](#_Toc454292387)

[Create Datasets 17](#_Toc454292388)

[Create Pipelines 17](#_Toc454292389)

[9. Get it all running 20](#_Toc454292390)

[Data Generator 20](#_Toc454292391)

[10. Power BI Dashboard 20](#_Toc454292392)

[11. Validation and Results 23](#_Toc454292393)

# Abstract

The recently published [Energy Demand Forecasting Solution Template](https://gallery.cortanaintelligence.com/SolutionTemplate/Demand-Forecasting-for-Energy-1) provides one-click deployment of an energy demand forecasting solution in Cortana Intelligence Suite. This [blog](https://blogs.technet.microsoft.com/machinelearning/2016/03/22/solution-template-for-energy-demand-forecasting/) gives a high level overview of the solution template. Advanced analytics solution implementers, i.e. Data Scientists and Data Engineers, usually need deeper understanding of the template components and architecture in order to use, maintain, and improve the solution.

This documentation provides more details of the solution template and step-by-step deployment instructions. Going through this manual deployment process will help implementers gain an inside view on how the solution is built and the function of each component.

# Requirements

You will need the following accounts and software to create this solution:

1. Source code and instructions from this GitHub repo
2. A Microsoft Azure subscription. (<https://azure.microsoft.com/>) **NOTE:** You need to have at least 20 cores quota under your Azure subscription for the on-demand HDInsight cluster. To see your quota, go to *portal.azure.com -> log in-> Subscriptions -> select your subscription -> All settings -> Usage +quotas.*
3. An Azure Machine Learning Studio account (<http://studio.azureml.net>)
4. A Microsoft Office 365 subscription for Power BI access.
5. A network connection
6. SQL Server Management Studio (<https://msdn.microsoft.com/en-us/library/mt238290.aspx>), Visual Studio (<https://www.visualstudio.com/en-us/visual-studio-homepage-vs.aspx>), or another similar tool to access a SQL server database.
7. Microsoft Azure Storage Explorer (<http://storageexplorer.com/>)
8. Power BI Desktop (<https://powerbi.microsoft.com/en-us/desktop>)

It will take about one day to implement this solution if you have all the required software/resources ready to use.

# Architecture

Figure 1 shows the overall architecture of the Energy Demand Forecasting Solution Template. Below is a brief introduction of each step.

**Data Generation:** We provided a data generator to simulate real time smart meter readings/electricity consumption data from multiple substations.

**Data Ingestion:** The simulated data are ingested by Azure Event Hub and Azure Stream Analytics jobs and saved to Azure Blob Storage.

**Data Preprocessing:** The raw data coming in every 10 seconds at substation level are aggregated into hour and region level by an on demand HDInsight cluster and used as the input of Azure Machine Learning experiment. The aggregated data are also copied to Azure SQL Database to be consumed by Power BI and other applications.

**Demand Forecasting:** An Azure Machine Learning web service is used to retrain the model and generate new demand forecasts for the next 24 hours. Results are saved to Azure Blob Storage and copied to Azure SQL Database.

**Data Visualization:** Power BI is used to visualize real time streaming data, demand forecasting results, and forecasting accuracy.

Azure Data Factory orchestrates the end-to-end data pipelines.

We will explain how to build this solution from the ground up in the “Setup Steps” section.

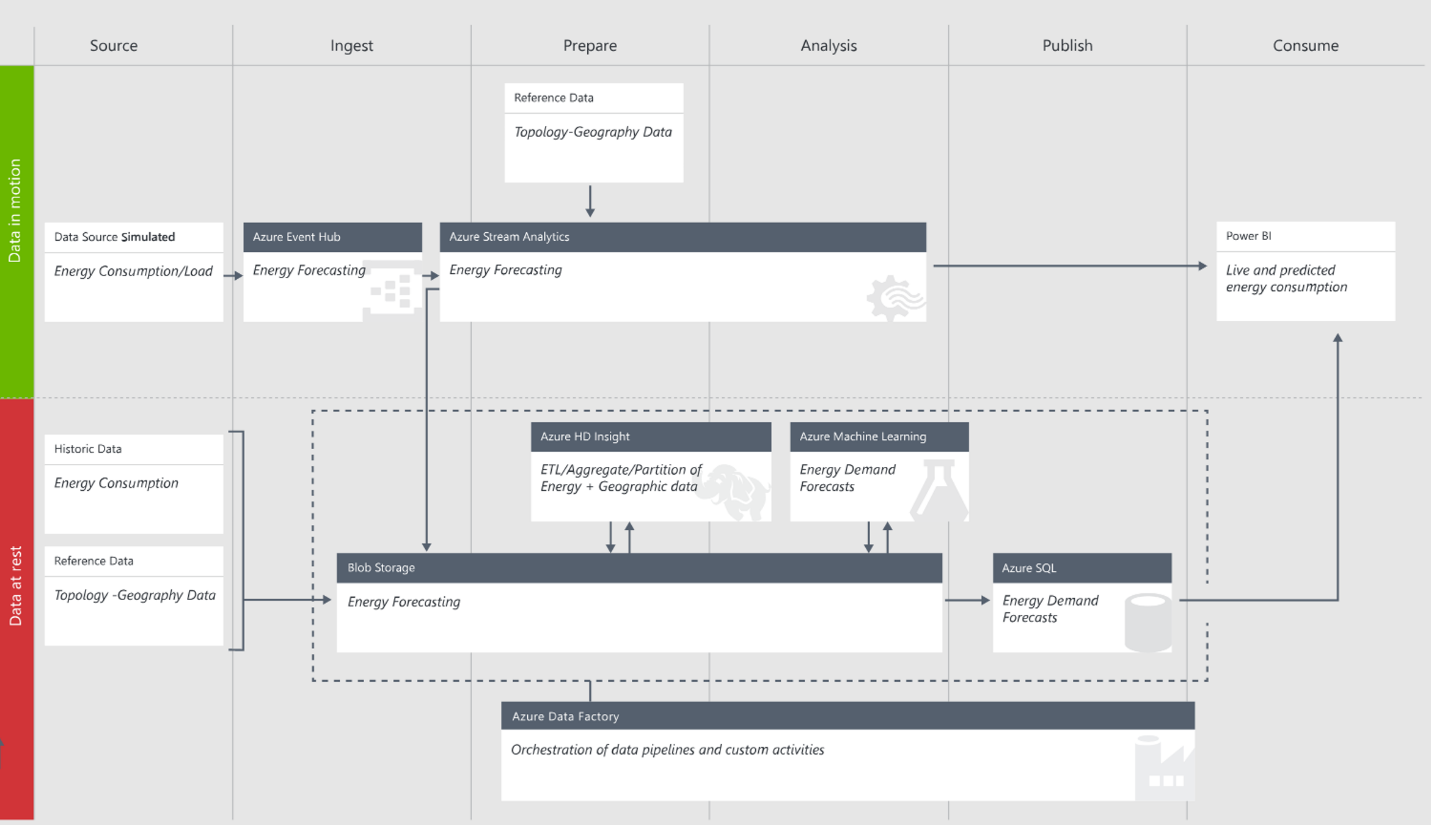


Figure 1. Energy Demand Forecasting Solution Template Architecture

# Setup Steps

This section walks the readers through the creation of each of the Cortana Intelligence Suite services in the architecture defined in Figure 1.

As there are usually many interdependent components in a solution, Azure Resource Manager enables you to group all Azure services in one solution into a [resource group](https://azure.microsoft.com/en-us/documentation/articles/resource-group-overview/#resource-groups). Each component in the resource group is called a resource.

Similarly, we want to use a common name for the different services we are creating. The remainder of this document will use the assumption that the base service name is:

energytemplate[UI][N]

Where [UI] is the users initials and N is a random integer that you choose. Characters must be entered in in lowercase. Several services, such as Azure Storage, require a unique name for the storage account across a region and hence this format should provide the user with a unique identifier.

So for example, Steven X. Smith might use a base service name of *energytemplatesxs01.*

**NOTE:** We create most resources in the *South Central US* region. The resource availability in different regions depends on your subscription. When deploying you own resources, make sure all data storage and compute resources are created in the same region to avoid inter-region data movement. Azure Resource Group and Azure Data Factory don’t have to be in the same region as the other resources. Azure Resource Group is a virtual group that groups all the resources in one solution. Azure Data Factory is a cloud-based data integration service that automates the movement and transformation of data. Data factory orchestrates the activities of the other services.

## Create a new Azure Resource Group

* Navigate to ***portal.azure.com*** and log in to your account.
* On the left tab click ***Resource Groups***
* In the resource groups page that appears, click ***Add***
* Provide a name ***energytemplate\_resourcegroup***
* Select a location.
* Click ***Create***

## Azure Storage Account

### Create the storage account

An Azure Storage account is used to store smart meter readings through Azure Event Hub and Azure Stream Analytics. The storage account is also used to hold Hive scripts that will be executed from Azure Data Factory when processing the electricity consumption data to pass into the Azure Machine Learning experiment.

* Navigate to ***portal.azure.com*** and log in to your account.
* On the left tab click **New>Data and Storage>Storage Account**
* Change the deployment model to *Classic*.
* Set the name to energytemplate[UI][N]
* Set the resource group to the resource group we created by selecting the link *Choose Existing*:
* Location set to *South Central US*
* Click ***Create***
* Wait for the storage account to be created.

Now that the storage account has been created we need to collect some information about it for other services like Azure Data Factory.

* Navigate to ***portal.azure.com*** and log in to your account.
* On the left tab click ***Resource Groups***
* Click on the resource group we created earlier ***energytemplate\_resourcegroup.*** If you don’t see the resource group, click ***Refresh.***
* Click on the storage account in Resources
* In the *Settings* tab on the right click *Keys*
* Copy the *PRIMARY CONNECTION STRING* and add it to the table below.

|  |  |
| --- | --- |
| **Azure Storage** | |
| Storage Account | energytemplate[UI][N] |
| Connection String |  |

### Prepare the storage containers

* Download and install the [Microsoft Azure Storage Explorer](http://storageexplorer.com/).
* Log in to your Microsoft account associated with your Azure Subscription. If you already logged in with an account, but want to use a different account, click the gear icon and then click the small triangle on the right side of the current account. Click “Add an account…” to log in with another account.
* Locate the storage account created in step 2 above and expand the nodes to see *Blob Containers*, etc.
* Create the two containers, *demandforecasting* and *energysadata*
  1. Right click on ***Blob Containers*** and choose ***Create Blob Container***
  2. Enter one of the container names.
  3. Repeat the previous two steps until both containers are created.
* Double click the *demandforecasting* container
* In the right panel, above the container listing, click the arrow on the ***Upload*** button and choose ***Upload Folder***
* Browse to the ***Data\referencedata*** folder in the solution package, select the folder and upload it. This will upload the reference geography data of the stations.
* Browse to the ***script*** folder in the solution package, select the folder and upload it. This will upload Hive and SQL queries.

## Azure Event Hub

Azure Event Hub is a highly scalable service that can ingest millions of records per second. This will be the ingestion point for the smart meter reading data.

* Navigate to ***portal.azure.com*** and log in to your account.
* On the left tab click ***Resource Groups***
* Click on the resource group we created earlier ***energytemplate\_resourcegroup***
* On the resource page click ***Add***
* On the page that appears on the right, type *Event Hub* in the search box.
* Choose ***Event Hub***
* Click ***Create*** on the page that comes up which will re-direct you to ***manage.windowsazure.com***.
* On the redirected page Choose ***Custom Create***
* Enter the name energytemplate[UI][N]
* Set region to *South Central US*
* In the ***NAMESPACE*** drop-down menu choose ***Create new namespace***
* The namespace will be created as energytemplate[UI][N]-ns.
* Click the next arrow, and enter *Partition Count* as 4 and *Retention Days* as 7
* Click the check button to complete the creation

This creates the Azure Event Hub we need to receive the smart meter readings. The Event Hub will be consumed by two Azure Stream Analytics jobs. To ensure processing of the hub is successful we need to create two [consumer groups](https://azure.microsoft.com/en-us/documentation/articles/event-hubs-programming-guide/#event-consumers) on the hub.

* Click the blue back arrow in the top left of the page ***twice*** to return to the ***SERVICE BUS*** page.
* In the list, choose the namespace we created above - energytemplate[UI][N]-ns
* Click ***EVENT HUBS*** at the top of the right pane
* The event hub we have created above (energytemplate[UI][N]) should be highlighted.If you don’t see the event hub, click ***Refresh.***
* Click the ***CREATE CONSUMER GROUP*** at the bottom of the right pane and add in “blobcg” as the consumer group name. Repeat this process and create another consumer group called “pbicg”.

Finally, we are going to need some information about this event hub for our event generation application that will feed the event hub. While still at ***manage.windowsazure.com***

* Click ***SERVICE BUS*** in the left panel
* Highlight the namespace we created above - energytemplate[UI][N]-ns by clicking on the row but not the actual namespace name.
* At the bottom of the page click ***CONNECTION INFORMATION***
* Copy the *CONNECTION STRING* information and paste it into the table below.

The connection string and event hub name information will be needed to configure the desktop data generation tool that simulates smart meter readings being sent to the event hub.

|  |  |
| --- | --- |
| **Azure Event Hub** | |
| Event Hub | energytemplate[UI][N] |
| Namespace | energytemplate[UI][N]-ns |
| Connection String |  |

### Check Event Hub

While running the demo you can validate the event hub created is receiving messages by following the steps below. This can be a useful debugging step to determine if the event hub is functioning as expected, but note that the event hub will only show activity when we start the data generation tool later in this deployment guide:

* Log in to ***manage.windowsazure.com***
* In the menu on the left side of the screen select ***SERVICE BUS***
* Choose the namespace created above
* Click on *EVENT HUBS* at the top of the right hand side of the page
* Choose the event hub created above
* The dashboard will show, with a 15 minutes latency, the activity in the event hub. You can also gain more information on the event hub by selecting ***Operation Logs*** on the dashboard page under *Management Services.*

## Azure Stream Analytics Jobs

### Create Stream Analytics Jobs

[Azure Stream Analytics](https://azure.microsoft.com/en-us/services/stream-analytics/) allows you to create near real-time insights from devices, sensors, infrastructure and applications. For this demo Azure Stream Analytics is used to create two jobs that read smart meter data from the Azure Event Hub.

The first job simply pipes all of the smart meter readings into our Azure Storage for later processing. The second job is used to populate Power BI datasets that will be used on the dashboard. Although we have not set up Power BI yet, we will create both stream jobs now.

The names of the two jobs are *energytemplateasablob* and *energytemplateasapbi*. For each of these two job names follow these steps:

* Navigate to ***portal.azure.com*** and log in to your account.
* On the left tab click ***Resource Groups***
* Click on the resource group we created earlier ***energytemplate\_resourcegroup***
* On the resource page click ***Add***
* On the page that appears on the right, type *Stream Analytics Job* in the search box.
* From the search results choose ***Stream Analytics Job***
* On the page that appears on the right click ***Create***
* Enter the job name
* Choose the resource group created earlier ***energytemplate\_resourcegroup***
* Choose the location as *South Central US*
* Click ***Create***

### Configure Stream Analytics Jobs

Once both jobs have been created, they can be configured. Both jobs will have the same input Event Hub, but will have different [stream queries](https://msdn.microsoft.com/en-us/library/azure/dn834998.aspx) and different outputs.

Some of the configuration functionality for the stream analytics jobs cannot be completed at ***portal.azure.com***, so the remaining configuration steps will be on the ***manage.windowsazure.com*** site.

### Configure Stream Analytics Jobs - Inputs

First we will create the stream inputs for both jobs. For each of the stream jobs we created at ***portal.azure.com***:

* Navigate to ***manage.windowsazure.com*** and log in to your account.
* On the left tab click ***STREAM ANALYTICS***
* Click on the one of the jobs that was created in the earlier steps.
* At the top of the right page, click ***INPUTS***
* Click on ***ADD AN INPUT***
* Through the set up wizard
  + Page 1: Choose Data Stream
  + Page 2: Choose Event Hub
  + Page 3:
    - Input Alias: EventHubSource
    - Choose the event hub we created earlier and the correct namespace.
    - Consumer Group
      * energytemplateasablob uses *blobcg*
      * energytemplateasapbi uses *pbicg*
  + Page 4: *EVENT SERIALIZATION FORMAT*: CSV
  + Click the check button to complete the creation of the input.

Then we will create three additional reference data inputs for the energytemplateasapbi job by repeating the following steps three times. These three inputs have different *Input Aliases* and *Path Patterns*, which are distinguished by 1), 2), and 3) in the instructions below.

* Click on the energytemplateasapbi job.
* At the top of the right page, click ***INPUTS***
* Click on ***ADD AN INPUT***
* Through the set up wizard
  + Page 1: Choose Reference Data
  + Page 2:
    - Input Alias:

1. RegionBlobSource
2. SubStationBlobSource
3. TopologyBlobSource
   * + Choose the storage account we created earlier
     + Storage container: demandforecasting
     + Path pattern:
4. referencedata/region/region.csv
5. referencedata/substation/substation.csv
6. referencedata/topologygeo/topologygeo.csv
   * Page 3: *EVENT SERIALIZATION FORMAT*: CSV
   * Click the check button to complete the creation of the input.

### Configure Stream Analytics Jobs - Query

Now we will create the queries for the jobs:

* Navigate to ***manage.windowsazure.com*** and log in to your account.
* On the left tab click ***STREAM ANALYTICS***
* Click on the one of the jobs that was created in the earlier steps.
* At the top of the right page click ***QUERY***
* In the query box, copy the content of one of the scripts from the package folder ***Stream Analytics Queries***. The query files are named identically to the job name.
* Click ***Save*** at the bottom of the page
* Repeat for the other job.

### Configure Stream Analytics Jobs - Outputs

Finally, we will create the outputs for the Stream Analytics jobs. Navigate to ***manage.windowsazure.com*** and log in to your account.

* On the left tab click ***STREAM ANALYTICS***
* Click on the one of the jobs that was created in the earlier steps.
* At the top of the right page, click ***OUTPUTS***
* Click ***ADD AN OUTPUT***

For **energytemplateasablob**

Page 1: Choose *Blob Storage*

Page 2:

* + OUTPUT ALIAS: RawDataBlobSink
  + Choose the storage account created earlier
  + Choose the storage container ***energysadata*** that was created earlier.
  + PATH PREFIX PATTERN demandongoing/date={date}/hour={time}
  + Setting the prefix pattern enables the ***DATE FORMAT*** combo box. Change the format from YYYY/MM/DD to YYYY-MM-DD. This defines the format of the path strings in the storage account and is required for the Hive scripts that will be executed as part of the larger data flow.

Page 3: EVENT SERIALIZATION FORMAT is CSV

For **energytemplateasapbi**

Page 1: Choose Power BI

Page 2: If you already have a Power BI account, click *Authorize Now* and sign in. Otherwise, click *Sign up* now.

Page 3:

Output Alias: PBIoutput

Dataset Name: EnergyStreamData

Table Name: EnergyStreamData

Workspace: Select the PowerBI workspace you want to save the data to. Leave it as “My Workspace” if you didn’t create other workspaces.

* Click the check button at the bottom of the wizard to add the output to the job.

### Check Stream Jobs

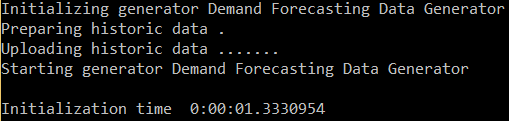
While running the demo you can validate the stream jobs are operating as expected by following the steps below. Please note that the Stream Analytics dashboard will only show activity when we start the data generation tool later in this deployment guide:

* Log in to ***manage.windowsazure.com***
* In the menu on the left side of the screen select ***STREAM ANALYTICS***
* Choose one of the stream jobs created above
* Click on *DASHBOARD* at the top of the right hand side of the page
* The dashboard will show, with a 15 minute latency, the activity in the Stream Analytics jobs. You can also gain more information on the Stream Analytics jobs by selecting ***Operation Logs*** on the dashboard page under *Management Services.*

## Configure Data Generator and test Event Hub / Stream Analytics

Now that we have the event hub and stream analytics configured we can configure the data generator and test that the flow to this point is working.

* Navigate to the hard disk location where the solution package was unzipped.
* Go into the ***Technical Deployment guide\Demand Forecasting Data Generator***directory and start the *Generator* application.
* In the prompt window, enter the event hub name, event hub connection string, and storage account connection string collected earlier.
* Type “0” to start data generation.
* The following message indicates that the data generation has started successfully.



### Validating initial data generation

Leaving the generator running for about 15 minutes, we can validate that the services thus far are operating as expected.

First validate event hub by following the steps in [Check Event Hub](#_Check_Event_Hub).

Next, start the Stream Analytics Jobs

* Navigate to ***portal.azure.com*** and login in to your account.
* On the left tab click ***Resource groups***
* Click the resource group created earlier ***energytemplate\_resourcegroup***
* Click on each stream analytics job and on the pane that appears, click the ***Start*** button at the top of the page.
* In the newly opened blade on the right, click the ***Start*** button at the bottom left corner.

Validate that the stream analytics job related to storage is working by following the steps in [Check Stream Jobs](#_Check_Stream_Jobs) for the energytemplateasablob job.

Finally, validate that the files are being created in the storage account by following these steps:

* Open the Microsoft Azure Storage Explorer
* Navigate to the storage account set up previously
* Open the blob container energysadata
* Note that a sub folder *demandHistory* has been created by the stream analytics job.

You may close the data generator as it is not required for the following steps. It will be needed later when the whole system is brought online after the remaining services have been configured.

## Azure SQL Server and Database

In this step, we will create an Azure SQL Database to store “actual” demand data generated by the data generator and forecasted demand data generated by Azure Machine Learning experiment. The data in Azure SQL Database are consumed by Power BI to visualize demand forecasting results and performance.

* Navigate to ***portal.azure.com*** and login in to your account.
* On the left tab click ***New>Data and Storage>SQL Database***
* Enter the name ***energytemplatedb*** for the database name
* Choose the resource group previously created ***energytemplate\_resourcegroup***
* Under Server click the arrow and choose ***Create new server***
  + Name : energytemplate[UI][N]
  + Enter in an administrator account name and password and save it to the table below.
  + Choose South Central US as the location to keep the SQL database in the same region as the rest of the services.
  + Click ***Select***
* Once returned to the SQL Database tab, click ***Create***
* Wait for the database and server to be created.
* From ***portal.azure.com,*** click on ***Resource Groups,*** then the group for this demo ***energytemplate\_resourcegroup.***
* In the list of resources, click on the SQL Server that was just created.
* Under ***Settings*** for the new server, click ***Firewall***and create a rule called ***open***with the IP range of 0.0.0.0 to 255.255.255.255. This will allow you to access the database from your desktop. Click ***Save.***

**Note:**This firewall rule is not recommended for production level systems but for this demo is acceptable. You will want to set this rule to the IP range of your secure system.

* Launch [SQL Server Management Studio](https://msdn.microsoft.com/en-us/library/mt238290.aspx)(SSMS), Visual Studio, or a similar tool, and connect to the database with the information you recorded in the table below.
  + NOTE: The server name in most tools will require the full name: energytemplate[UI][N].database.windows.net,1433
  + NOTE: Choose SQL Server Authentication
* ***THESE INSTRUCTIONS ARE FOR SSMS***
  + From the dropdown list of databases on the left side of ***!Execute***, select ***energytemplatedb*** that you created on the server
  + Click ***New Query*** at the tool bar.
  + Copy and execute the SQL script located in the package directory ***Query\SQL*** to create the necessary tables for the machine learning experiment and views and stored procedures that will be used by Azure Data Factory. We will more explain more details of the database objects in the Azure Data Factory section as they are closely related.

|  |  |
| --- | --- |
| **Azure SQL Database** | |
| Server Name | energytemplate[UI][N] |
| Database | energytemplatedb |
| User Name |  |
| Password |  |

## 

## Create Azure Studio ML Workspace and Experiment

This section assumes that you have not set up any workspaces in Azure Machine Learning Studio but that you do have an account at <https://studio.azureml.net/>.

The first thing we need to do is to create the workspace. A workspace is where Azure Machine Learning experiments are created. It is also tied to a storage account for intermediate results during experiment processing. You can invite others to your workspace by adding them as *USERS* to share and collaborate with people.

* Navigate to ***manage.windowsazure.com*** and login in to your account.
* On the left tab click ***MACHINE LEARNING***
* In the bottom left hand corner click ***NEW***
* Choose *DATA SERVICES\MACHINE LEARNING\QUICK CREATE*
* For workspace name enter *energytemplate[UI][N]*
* Location South Central US
* Choose the storage account created earlier. **NOTE:** If the previously created storage account doesn’t show up, use an existing one or create a new one. This storage account is for storing intermediate results and log files of Azure Machine Learning.
* Click on ***Create an ML Workspace***

Now that we have a workspace to work within, we can copy over the required experiment from the Cortana Intelligence Gallery.

* Navigate to ***studio.azureml.net*** and log into your account
* In the upper right hand corner you will see a workspace name. If this does not match the name of the workspace we just created, use the arrow button to expose the workspace selection menu and choose the workspace *energytemplate[UI][N]*.
* Go to [energy forecast sample experiment](https://gallery.cortanaintelligence.com/Experiment/Energy-Demand-Forecast-Sample-Model-1) in Cortana Intelligence Gallery.
* On the experiment page click the ***Open In Studio*** button
* In the dialog ***Copy experiment from Gallery***, choose appropriate location ***South Central US*** and the workspace we created earlier that you would like to copy it into. Click the **√** button.
* This process may take a minute and the experiment will open in the requested workspace.
* Click ***RUN***at the bottom of the experiment page. This step will take less than a minute to finish and all objects in the graph will have a green check box on them to indicate they have finished running.
* Click ***DEPLOY WEB SERVICE*** at the bottom of the page to create the Azure Web Service associated with the experiment. When completed, the browser will redirect to the web service home page.
  + The web service home page can also be found by clicking the ***WEB SERVICES*** button on the left menu of the studio.azureml.net page once logged in.
* Copy the ***API key*** from the web service home page and add it to the table below as you will need this information later.
* Click the link ***BATCH EXECUTION*** under the ***API HELP PAGE*** section. On the BATCH EXECUTION help page, copy the ***Request URI*** under the ***Request*** section and add it to the table below as you will need this information later. Copy only the URI part https:… /jobs, ignoring the URI parameters starting with ? .

|  |  |
| --- | --- |
| **Web Service BES Details** | |
| API Key |  |
| Request URI\* |  |

## Azure Data Factory

We have now created all the necessary components for building data pipelines inAzure Data Factory. Data factory orchestrates all Azure data services to move and process data in pipelines.

Below is an overview of the major steps we will implement in the data factory of this solution. Feel free to skip this section if you want to jump into [building the data factory](#_Create_Data_Factory).

* Data aggregation on HDInsight cluster. We will create an on demand HDInsight cluster to read and process the raw data being streamed in through event hub and stream analytics jobs. Data collected every 10 seconds from substations are aggregated into every hour and 3 regions using Hive scripts. Three additional Hive scripts are used to concatenate historical and the latest demand data to be used as input of Azure Machine Learning experiment.
* Demand forecasting by calling Azure Machine Learning web service. We will call the Azure ML web service for each region and save the forecasting results to Azure blob.
* Data movement between Azure Blob and Azure SQL
* Copy region, substation, and topology data from blob storage to Azure SQL. Since these data are stationary, the copying data pipelines are only executed once.
* Copy historical demand data from blob storage to Azure SQL. This data pipeline is only executed once.
* Copy on-going demand data from blob storage to Azure SQL every hour.
* Copy demand forecasting results from blob storage to Azure SQL every hour.

In the following subsections, we will first create a Data Factory. Then in the data factory, we will create

* Linked services. Linked services contain the information needed to access Azure Services, e.g. storage account name and connection string for Azure storage accounts, API key and URI for Azure ML web services (a compute Linked service). Storage Linked services are used by Datasets and Compute Linked services are used by data pipeline activities.
* Datasets. Datasets are named references pointing to data in different Azure storage services, e.g. Azure blobs, Azure SQL tables.
* Data pipelines. Data pipelines are collections of activities, e.g. copy activity, Azure ML scoring activity, that are scheduled to run at certain frequency during certain period of time.

### Create Data Factory

* Navigate to ***portal.azure.com*** and login in to your account.
* On the left tab click ***New>Data and Analytics>Data Factory***
* Name: *energytemplate[UI][N]*
* Resource Group: Choose the resource group created previously ***energytemplate\_resourcegroup***
* Location: EAST US
* Click ***Create***

After the data factory is created successfully

* On the left tab in the portal page (portal.azure.com), click ***Resource groups***
* Search for the resource group created previously, ***energytemplate\_resourcegroup***
* Under Resources, click on the data factory we just created,*energytemplate[UI][N]*
* Locate the ***Actions*** panel and click on ***Author and deploy***.

In the ***Author and deploy*** blade, we will create all the components of the data factory. Note that Datasets are dependent on Linked services, and pipelines are dependent on Linked services and Datasets. So we will create Linked services first, then Datasets, and pipelines at last.

### Create Linked Services

We will create 5 Linked services in this solution. The scripts of the Linked services are located in the folder ***Data Factory\LinkedServices*** of the solution package***.***

First, we will create three data store Linked services for Azure storage services.

**StorageLinkedService.** This is the Linked Service for the Azure storage account.

* Open the file ***Data Factory\LinkedServices\StorageLinkedService.json***. Replace [Azure Storage Account Connection String] with the connection string from the [Azure Storage Account](#_Azure_Storage_Account_1) section.
* Go back to ***Author and deploy*** in the data factory on ***portal.azure.com.***
* Click ***New data store*** below the data factory name and select ***Azure storage*** as shown below.



* Overwrite the content in the editor window with the content of the modified StorageLinkedService.json.
* Click ***Deploy*** at the top of the editor window.

**HDInsightStorageLinkedService.**

* Repeat the steps of creating StorageLinkedServiceusing the file ***Data Factory\LinkedServices\HDInsightStorageLinkedService.json***

**AzureSqlLinkedService.** This is the Linked service for the Azure SQL database.

* Open the file ***Data Factory\LinkedServices\AzureSqlLinkedService.json***. Replace [Azure SQL server name], [Azure SQL database name], [SQL login user name], and [SQL login password] with the corresponding information from the [Azure SQL Server and Database](#_Azure_SQL_Server) section.
* Go back to ***Author and deploy*** in the data factory on ***portal.azure.com.***
* Click ***New data store*** and select ***Azure SQL***.
* Overwrite the content in the editor window with the content of the modified AzureSqlLinkedService.json.
* Click ***Deploy***.

Next, we will create two compute Linked services for Azure computing services.

**AzureMLEndpoint.** This is the Linked service for the Azure Machine Learning web service.

* Open the file ***Data Factory\LinkedServices\AzureMLEndpoint.json***. Replace [Azure Machine Learning web service URI] and [Azure Machine Learning web service API key] with the corresponding information from the [Create Azure Studio ML Workspace and Experiment](#_Create_Azure_Studio) section.
* Go back to ***Author and deploy*** in the data factory on ***portal.azure.com.***
* Click ***New compute*** and select ***Azure ML***.
* Overwrite the content in the editor window with the content of the modified AzureMLEndpoint.json.
* Click ***Deploy***.

**HDInsightLinkedService.** This is the Linked service for the HDInsight cluster.

* In ***Author and deploy***, click ***New compute*** and select ***On Demand HDInsight cluster***.
* Overwrite the content in the editor window with the content of the file ***Data Factory\LinkedServices\*** ***HDInsightLinkedService.json***
* Click ***Deploy***.

### Create Datasets

We will create 10 datasets pointing to Azure SQL tables and 16 datasets pointing to Azure blobs. We will use the JSON files located at ***Data Factory\Datasets.*** No modification is needed on the JSON files.

On ***portal.azure.com*** navigate to your data factory and click the ***Author and Deploy*** button.

For each JSON file under ***Data Factory\Datasets\BlobDatasets***,

* At the top of the left tab, click ***New dataset*** and select ***Azure blob storage***
* Copy the content of the file into the editor.
* Click ***Deploy***

For each JSON file under ***Data Factory\Datasets\SQLDatasets***,

* At the top of the left tab, click ***New dataset*** and select ***Azure SQL***
* Copy the content of the file into the editor.
* Click ***Deploy***

The excel file ***\Data Factory\Datasets\DatasetDescription.xlsx*** describes the content of each dataset and how it gets updated. Re-visiting this file after creating the data pipelines may help you better understand the datasets.

### Create Pipelines

We will create 12 pipelines as shown in the figure below.



We will use the JSON files located at ***Data Factory\Pipelines.*** At the bottom of each JSON file, the “start” and “end” fields identify when the pipeline should be active and are in UTC time. We will modify the start and end time of each file to customize the schedule. For more information on scheduling in Data Factory, see [Create Data Factory](https://azure.microsoft.com/en-us/documentation/articles/data-factory-create-pipelines/) and [Scheduling and Execution with Data Factory](https://azure.microsoft.com/en-us/documentation/articles/data-factory-scheduling-and-execution/).

Four pipelines are only executed once to copy stationary data from Azure Blob storage to Azure SQL. The other eight pipelines are scheduled to run at the end of every hour to aggregate and prepare the ongoing data on the HDInsight cluster, call the Azure ML web service to generate new forecasts, and copy the new forecasts from Azure Blob to Azure SQL.

#### One-time pipelines

First we will create the one-time data pipelines. For the JSON files in the **OneTimePipelines** folder,note that the activities are scheduled to run every 1(“interval”) Day(“frequency”) as specified in the “scheduler” field. Therefore, for each JSON files in **OneTimePipelines,**

* Specify an active period of 1 day, so that these pipelines will only be executed once. This period needs to be in the past, so that they will be executed once the pipelines are deployed and upstream data are ready. For example, if today is May 11th 2016, then set the start and end time as follows:

"start": "2016-05-10T00:00:00Z",

"end": "2016-05-11T00:00:00Z",

* On ***portal.azure.com*** navigate to your data factory and click the ***Author*** ***and Deploy*** button.
* At the top of the tab, click ***More commands*** and then ***New pipeline***
* Copy the content of the modified JSON file into the editor.
* Click ***Deploy***

#### Hourly data aggregation pipeline

Next we will create a pipeline called **AggregateDemandDataTo1HrPipeline**. This pipeline contains a single activity - an HDInsightHive activity using the HDInsightLinkedService that runs a Hive script to aggregate every 10 seconds streamed in consumption data in substation level to hourly region level and save the aggregated data to Azure Blob storage. The activity in this pipeline is scheduled to run every hour. For the JSON file ***Data Factory\Pipelines\AggregateDemandDataTo1HrPipeline.json***,

* Specify an active period that you want the pipeline to run. For example, if you want to test the template for 1 day, then set the start and end time as something like

"start": "2016-05-11T00:00:00Z",

"end": "2016-05-12T00:00:00Z",

**NOTE:** Please limit the active period to the amount of time you need to test the pipeline to limit the cost incurred by data movement and processing.

* On ***portal.azure.com*** navigate to your data factory and click the ***Author*** ***and Deploy*** button.
* At the top of the tab, click ***More commands*** and then ***New pipeline***
* Copy the content of the modified JSON file into the editor.
* Click ***Deploy***

#### Hourly scoring pipelines

Next we will create an Azure Machine Learning scoring pipeline for each of the three regions after aggregating data from substations. The JSON files are located in the **HourlyScoringPipelines** folder.

Each scoring pipeline has the following three activities:

1. SchemaPrepForML activity. This custom HDInsightHive activity concatenates the latest demand data and the historical data to be used by Azure ML model training. Note that only data of the past two years are selected.
2. FileExtensionPrepforMLActivity. This copy activity moves the results from the HDInsightHive activity to a single Azure Storage blob that can be consumed by the AzureMLBatchScoring activity.
3. AzureMLBatchScoring activity. This activity calls the Azure Machine Learning web service to generate new demand forecasts for the next 24 hours and save the results to Azure Blob storage.

All the activities above are scheduled to run every hour. For each JSON file in the ***HourlyScoringPipelines***folder, set the active period to be the same as **AggregateDemandDataTo1HrPipeline** and deploy the pipelines as described previously.

#### Hourly copying pipelines

Finally, we create four pipelines that copy data from Azure Blob storage to Azure SQL, so that the forecast results can be consumed by Power BI. The JSON files are located in the ***HourlyCopyingPipelines*** folder.

The three pipelines named **CopyScoredResultRegion[N]Pipeline** contain a Copy activity that moves the results of the Azure Machine Learning experiment to Azure SQL Database.

The pipeline named **CopyAggDemandPipeline** contains a single Copy activity that moves the aggregated ongoing demand data from Azure Blob to Azure SQL.

For each JSON file in the ***HourlyCopyingPipelines***folder, set the active period to be the same as **AggregateDemandDataTo1HrPipeline** and deploy the pipelines as described previously.

## Get it all running

Now we have created all the components and services needed for this solution except for Power BI dashboard. It is time to start up the system to have the data flow through the services.

### Data Generator

Navigate to the folder ***Demand Forecasting Data Generator*** in the solution package and start the Generator.exe application. From previous steps in this document, the application should already be configured to send messages to the event hub. Enter 0 in the command window to start feeding events to the event hub.

|  |
| --- |
| **TIP** |
| The data generator will only run when your laptop is on and has a network connection. It is possible when your computer goes into “sleep” mode that the generator will stop running. To run this generator over a longer period of time it, should be moved to an Azure Virtual Machine or a computer that doesn’t go to sleep and executed there. |

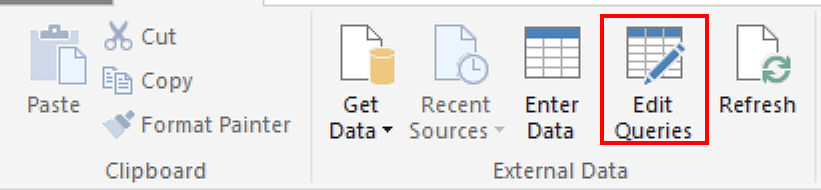
## Power BI Dashboard

It will take some time to generate the first batch of demand forecast results. In the meanwhile, we are going to create the Power BI Dashboard

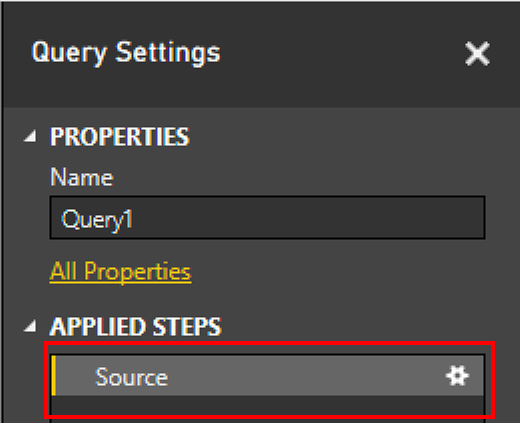
#### Demand forecast visualization (Cold Path)

Power BI is used to visualize the demand forecast results and the forecasting accuracy. We will first use Power BI Desktop to prepare the report. The report can also be viewed via Power BI online.

* Install the Power BI Desktop application (<https://powerbi.microsoft.com/en-us/desktop>)
* With the application, open the template PBIX file in the package directory ***Power BI Template***
* On the application ribbon menu, choose *Edit Queries* as shown below



* Go to the *Query Settings* on the right side of the newly opened Query Editor window,and click on the gear icon next to *Source*

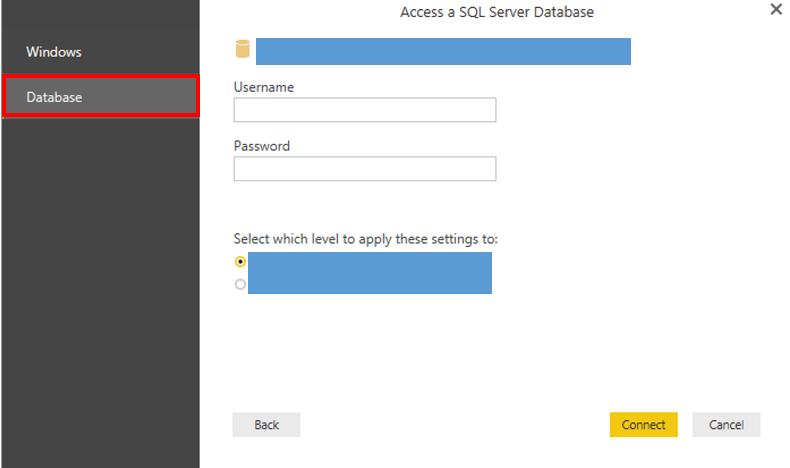


* In the SQL Server Database dialog that appears enter in the names of the SQL server and database created earlier. For example

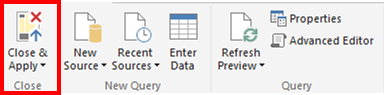
*Server: energytemplatehl1.database.windows.net,1433*

*Database: energytemplatedb*

* Click OK.
* When prompted for credentials choose ***Database*** tab on the left and enter the user name and password for the SQL server created earlier. Click ***Connect.***



* On the application ribbon menu click *Close and Apply* which will return you to the main application window.

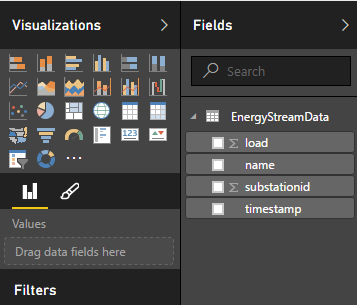


* On the application ribbon menu click *Publish*. If prompted “Do you want to save your changes?”, click ***Save***. Then you will be prompted for your credentials.
* Navigate to msit.powerbi.com site, open the menu on the top left corner of the screen, navigate to Reports to see the report (DemandForecastPowerBI) that has been published.
* Click the ***Pin Visual*** icon on the top right corner of one of the charts, a *Pin to Dashboard* window will pop up for you to choose a dashboard. Create a new dashboard *Energy Demand Forecast*.
* Pin the other charts to the *Energy Demand Forecast* dashboard.

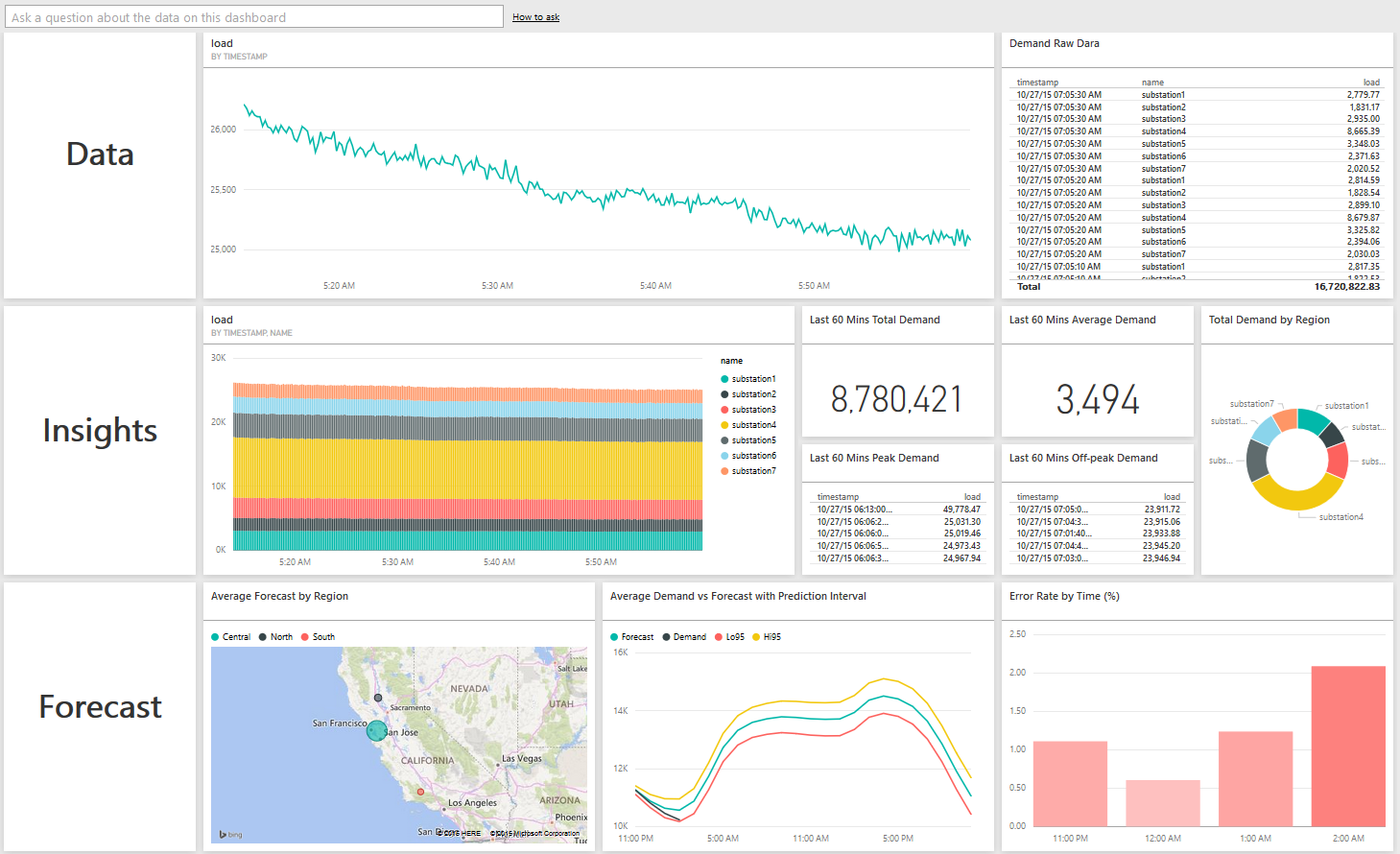
#### Real time streaming data visualization (Hot Path, optional)

PowerBI can also visualize real time data streamed through Stream Analytics.

* Log in to [Power BI online](https://powerbi.microsoft.com/en-us/)
* On the left panel Datasets section in My Workspace, you should be able to see the dataset **EnergyStreamData**. This is the streaming data you pushed using the [Azure Stream Analytics Job](#_Azure_Stream_Analytics) **energytemplateasapbi.**
* Make sure the *Visualizations* and *Fields* panes are open and shown on the right side of the screen as shown below. If not, click “Edit report” on top of the report.



* Click dataset ‘EnergyStreamData’ on the left panel *Datasets* section.
* Click the *Line Chart* icon https://acom.azurecomcdn.net/80C57D/cdn/mediahandler/docarticles/dpsmedia-prod/azure.microsoft.com/en-us/documentation/articles/cortana-analytics-technical-guide-demand-forecast/20160415071221/powerbipic8.png in the Visualization pane.
* Click ‘EnergyStreamData’ in the *Fields* panel if the list of fields are not displayed.
* Click “Timestamp” and drag it to the "Axis" field in the Visualization pane.
* Click “Load” and drag it to the "Values" field in the Visualization pane.
* Click SAVE on the top and name the report as “EnergyStreamDataReport”. The report named “EnergyStreamDataReport” will be shown in Reports section in the Navigator pane on left.
* Pin this new chart to the *Energy Demand Forecast* dashboard.
* Rename the visualization in the Power BI dashboard. Hover the mouse over the tile on the dashboard, click "…” (open menu) icon on the top right corner. In the prompt window, click the pencil icon (Tile details) and enter “Demand by Timestamp” as the title.
* Create other customized visualizations using the datasets available in Power BI. Below is an example of the final dashboard.



## Validation and Results

It will take about 2~3 hours to generate the first batch demand forecast results. If you specified the “start” time of the data factory pipelines to be a time in the future, you need to wait for 2~3 hours after that time and keep the data generator running between the “start” time and the “end” time. If you specified the “start” time to be in the past, you should start seeing data in your dashboard 2~3 hours after data factory deployment. Use the ***Refresh*** button in Power BI to get the latest data visualization.

To ensure that the system is functioning as expected, you can check the database tables DemandRealHourly and DemandForecastHourly to verify that data are being added to these tables hourly. If the DemandForecastHourly table is not receiving results in approximately 3 hours after starting the services you can take the following steps to see where the issue might be, but also consider that the dashboards for the services update about every 15 minutes.

#### Check data streaming

1. Log into ***manage.windowsazure.com***
2. Navigate to the event hub that was created and verify that it is receiving messages by viewing the dashboard.
3. Navigate to the stream analytics jobs that were created and verify that they are processing messages by viewing the dashboard.

You can look into the operations logs for these services to determine what errors, if any, have occurred.

#### Check data factory

1. Log into ***portal.azure.com***
2. Navigate to the data factory created.
3. Open the Diagram and examine the components.

You can double click on a dataset and check the status of data slices of different hours. Double click on a slice to see more details. In the *Monitoring* pane, double click on a run to see the error message if any.

You can right click on a data pipeline and click “Open pipeline” to see a more detailed diagram of that pipeline.

* First, check PartitionedDemandDataTable to see for which hours data slices are ready. If no slices are ready, go back and check data streaming in more details.
* Second, check the data ready tables: SQLHistoricalDataReadyTable and SQLOnGoingDataReadyTable.

If both datasets are not ready, check the pipeline LoadHistoryDemandDataPipeline by opening it and examining each dataset in it.

If SQLOnGoingDataReadyTable is not ready for the hours where PartitionedDemandDataTable is ready, check the components of the pipeline AggregateDemandDataTo1HrPipeline.

The data ready tables are set to re-try 10 times with a 15 minutes interval. This means, if the data slices are not ready within 150 minutes of the time they are scheduled to be ready, the data validation will fail and the following pipelines will not be executed. For slices of the hours/days before deploying the pipelines, data validation will fail if the slices are not ready within 150 minutes after pipeline deployment. If you run into issue in previous steps and spend extra time to fixing them, manually re-run the data ready table slices by double clicking a particular slice and clicking the “Run” button above the slice “Summary”.

* Check the rest of the datasets/pipelines in the order they appear in the diagram.

Alternatively, you can use the ***Monitor & Manage*** function under ***Actions*** to monitor the status of the data pipeline and datasets.