Real-Time Event Processing with Azure Stream Analytics

Lab 4 - Getting Started with Azure Stream Analytics

By Amy Nicholson

Overview

In this lab, you will build a real-time event processing service using Azure Stream Analytics. The architecture will include adding different input and output data sources as well as the inclusion of integrating Azure Machine Learning algorithms.

What You'll Need

To complete the labs, you will need the following:

- A Microsoft Azure Subscription
- An Azure Machine Learning workspace
- Microsoft Visual Studio 2015 Community Edition
- Twitter Authentication details (same as used in Lab 2: Cognitive Services)
- Access to GitHub repository:
  https://github.com/MicrosoftLearning/BuildingIntelligentApplications/

NOTE: Please clone or download the repository rather than open raw files as the lab4_testdata.json file may include markup and not produce expected results during the lab.

Problem Domain

In this lab, we will use a Twitter sentiment analysis scenario, similar to that you have viewed in the videos content. You will be using data from the Twitter API in two different schemas that include data such as the Text of the tweet, CreatedAt, RetweetCount and FavouriteCount and Topic/keyword you asked the Twitter Client application to search for.

This is a common scenario used in many businesses when it comes to understanding the perception of a brand or how well the company performs when dealing with customers. With businesses and customers becoming more 24/7, there has been an increasing need for interaction when negative sentiment is expressed to combat against customer churn and improve brand to increase net new customer acquisition.

Build a Real-Time Event Processing Architecture

In this first section, we will explore building a full real-time event processing pipeline, including the Azure Stream Analytics service setup with chosen input and output services.

Build an Azure Event Hub

Log into the Microsoft Azure Portal using your Azure subscription and build an event hub by selecting New -> Internet of Things -> Event Hub
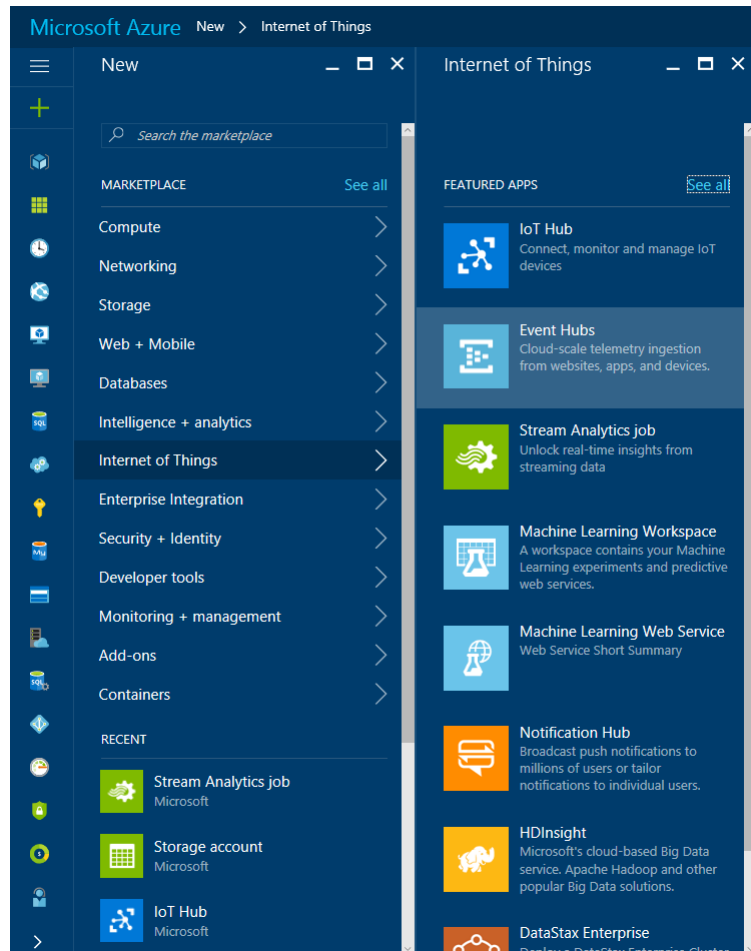
Now you need to create your Service Bus Namespace. Enter details for:

- Namespace name (example: lab4<your_initial>-ns)
- Choose the Basic pricing tier
- Select an Azure subscription
- Create a new resource group (example: lab4<your_initials>)
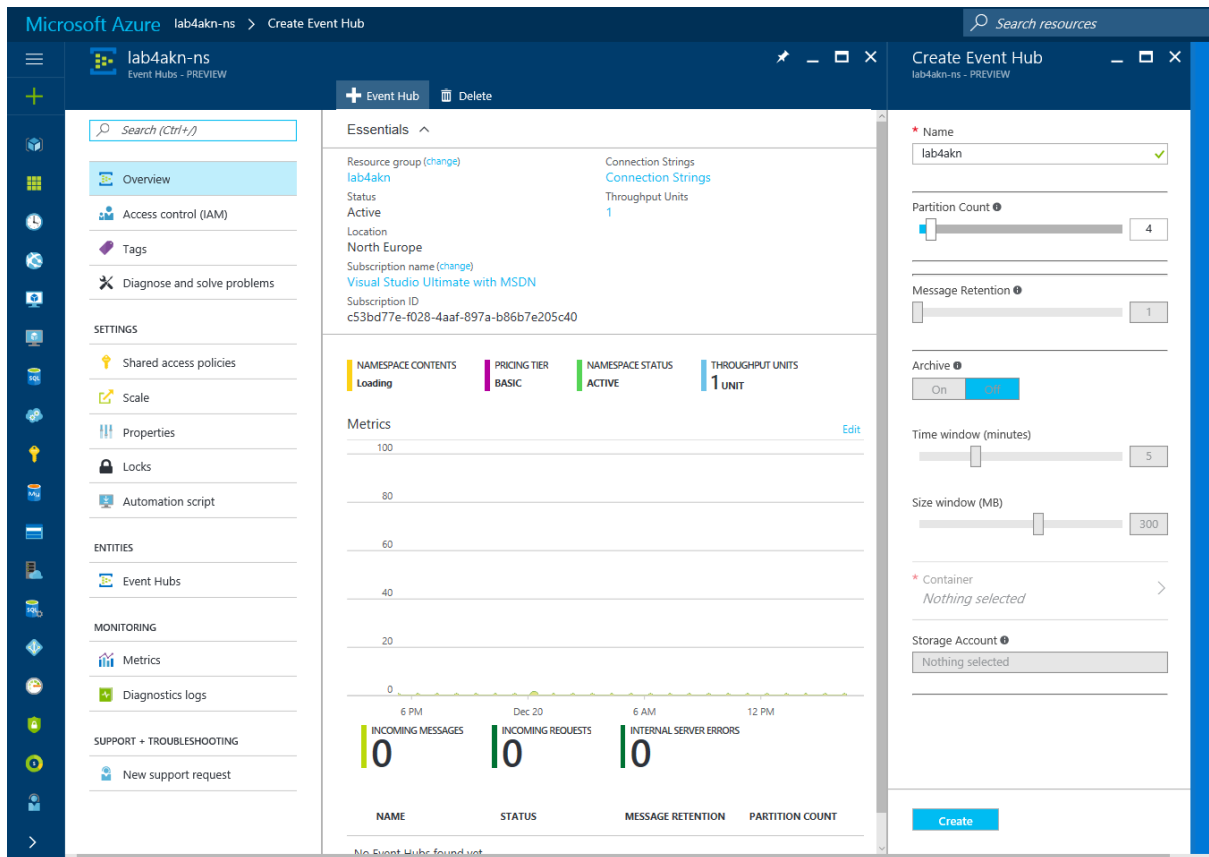- Select a location local to you (example: North Europe)

Once created the pane below will be visible. Now you need to select the '+Event Hub' button to create an event hub inside this Service Bus Namespace.
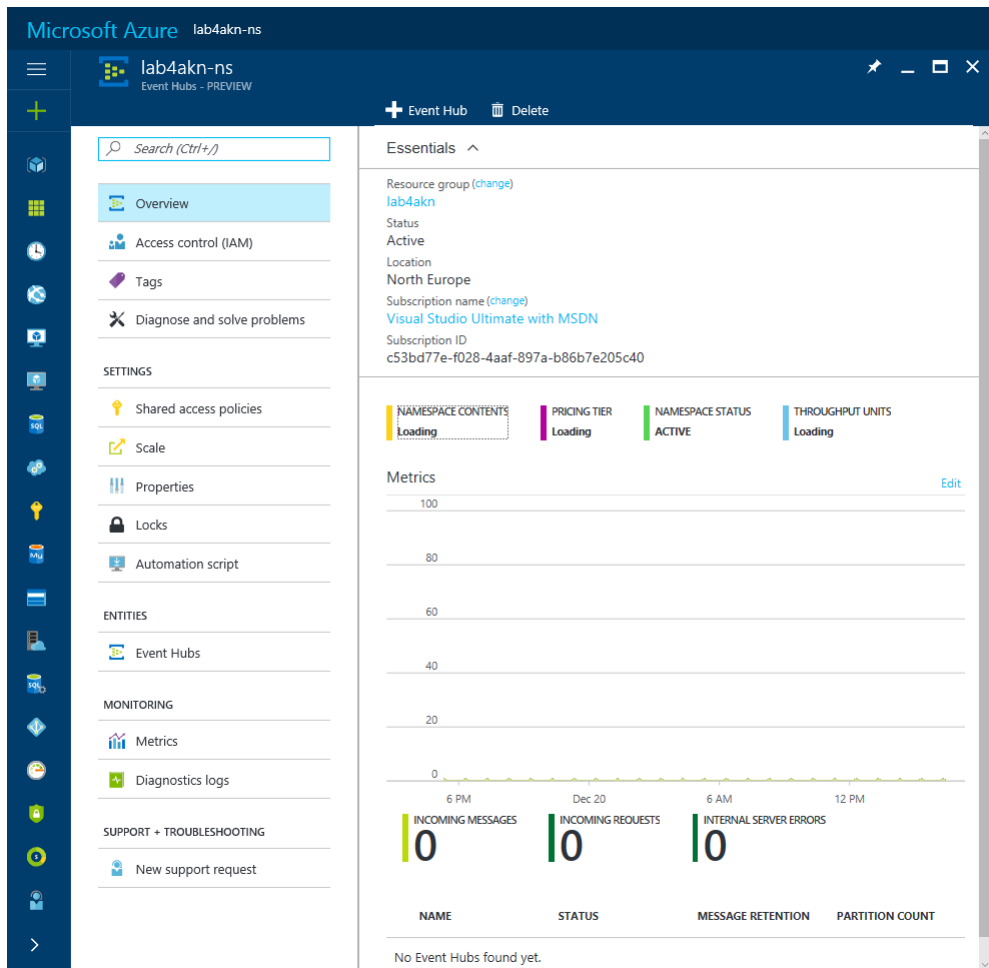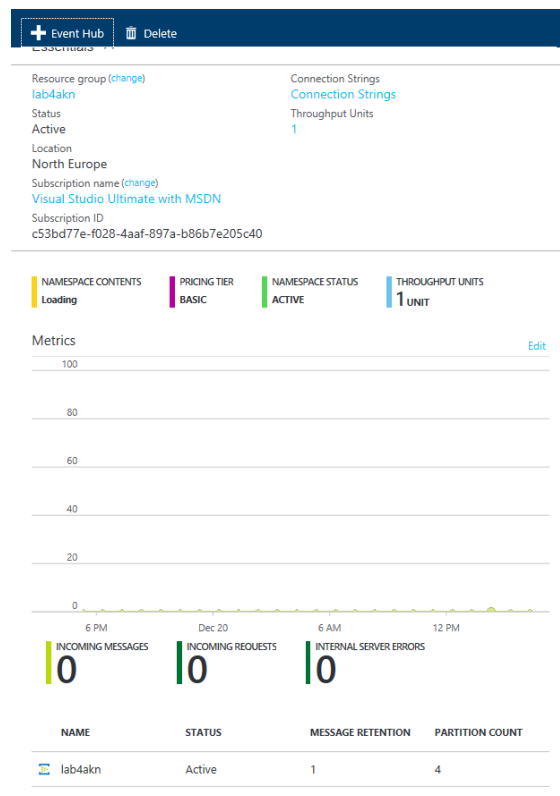
Add an event hub:

Give the event hub a name (example: lab4<your_initials>). Select the partition count as 4 and click create. See below for example screenshot.

Once created you should see the new event hub in the service bus namespace main page.

Choose the event hub and select 'Shared Access policies' from the menu.

NOTE: A Shared Access Signature (SAS) or Shared Access Policy provides delegates access to resources in your storage account type service. This means that you can grant a client limited permissions to objects in that service for a specified period of time and with a specified set of permissions, without having to share your account access keys. To find out more visit this link: https://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-shared-access-signature-part-1/

Under Shared Access Policies, create a new entry called 'manage' and give it permission to manage, send and listen. Then click Create once completed.
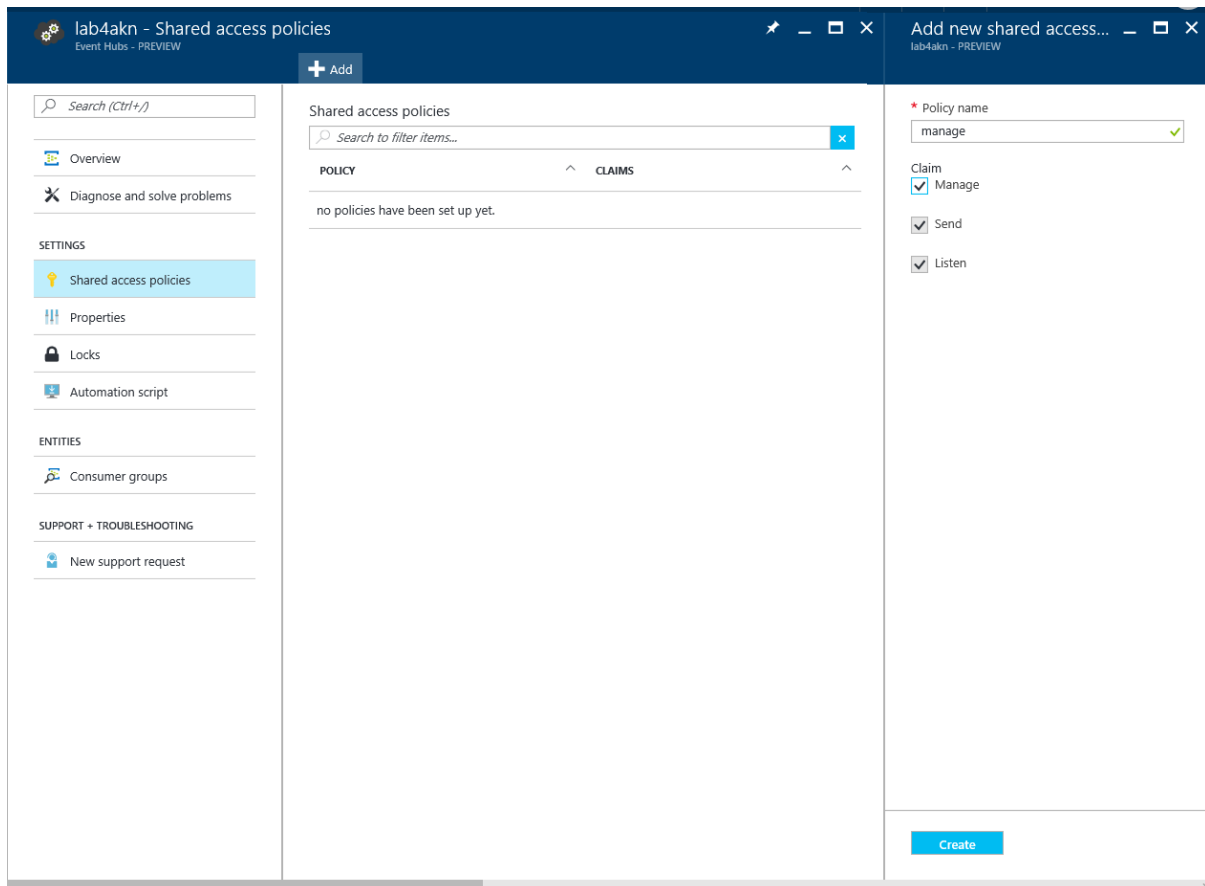

Note down the Event Hub Name and Connection string with primary key, by selecting the manage policy you just created. NOTE: your event hub name is at the top of the page (example: lab4<your_initials>). You will need these details later in the lab for the Twitter Client application.
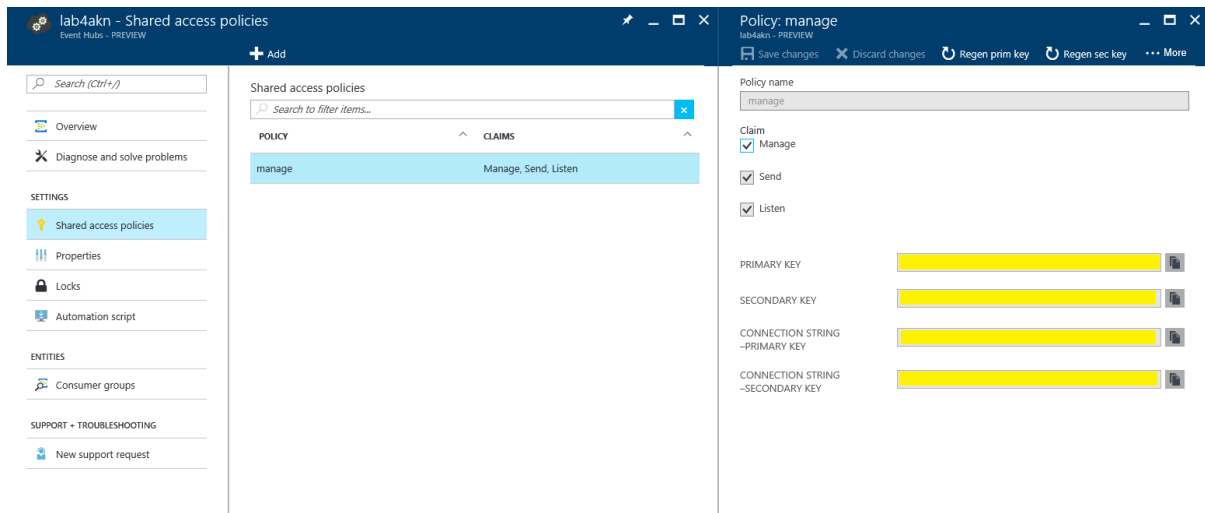
Build an Azure Blob Storage Account with Containers

Now back in the Microsoft Azure Portal Main Page, let's create a Storage Account for this lab:

Select New -> Storage -> Storage Account
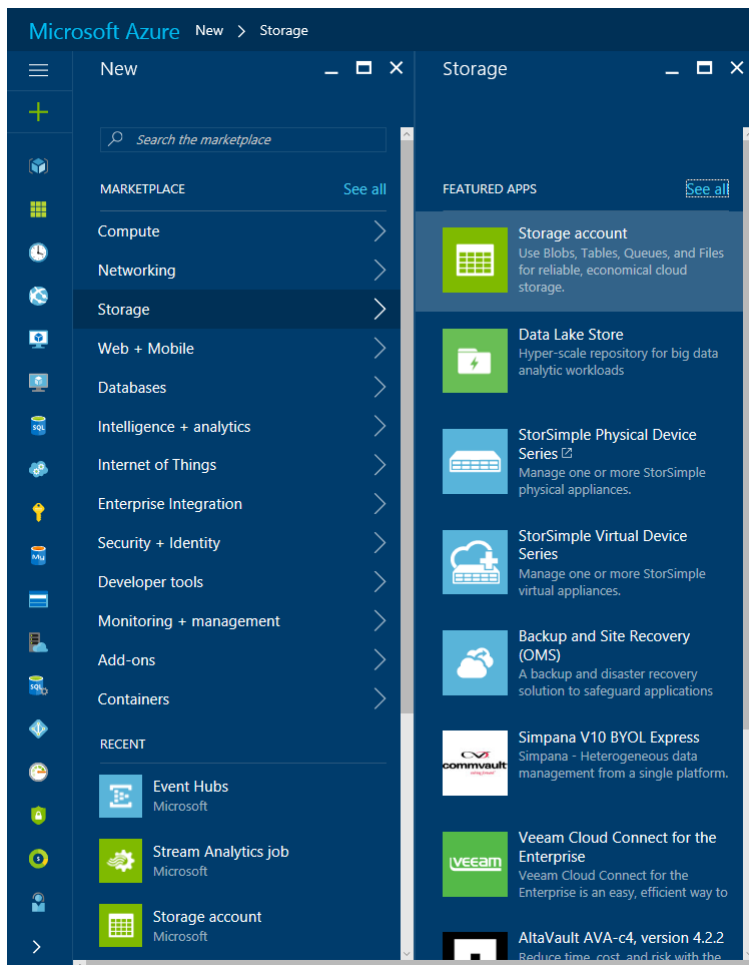
Next, enter storage account details below and click Create:

- **Name:** Choose a name for the service (example: lab4<your_initials>)
- **Deployment Model:** Resource Manager
- **Account Kind:** Blob Storage
- **Performance:** Standard
- **Replication:** Locally-redundant storage (LRS)
- **Access Tier:** Hot
- **Subscription:** Choose a subscription
- **Resource Group:** Choose the resource group you created with your event hub (example: lab4<your_initials>)
- **Location:** Choose a location
- **Pin to Dashboard:** Check this

NOTE: The Azure Storage Account may take a couple of minutes to provision on your dashboard and then new storage account blade will open automatically. If it does not, select the storage account created to open it from the dashboard.

Next, we need to add two containers inside this storage account. In the storage account blade select the '+ Container' button in the top toolbar and create one container named 'input' and one named 'output' both with the 'Private' access tier.
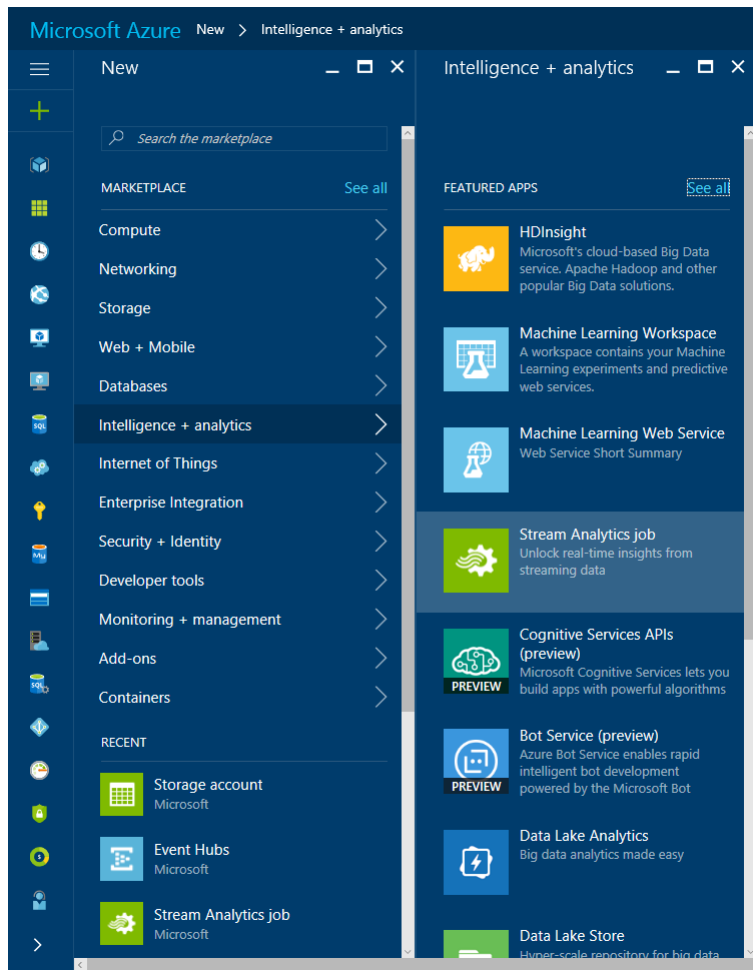
Build an Azure Stream Analytics Service

Now we have our input and output services created as part of the real-time event processing architecture we need to pull these services together with an Azure Stream Analytics instance.

New -> Intelligence + analytics -> Stream Analytics job

To create the Stream analytics service, provide the details below and click Create:

- **Job name:** Choose a job name (example: lab4<your_initials>)
- **Subscription:** Choose your Azure Subscription (same as the previous services)
- **Resource Group:** Use Existing and select the previously created resource group from the list (example: lab4<your_initials>)
- **Location:** Choose the same data center as previously created services.
- **Pin to Dashboard:** Check this.

Once created the Stream Analytics blade will open. Now we need to link the input, query and output services to build our real-time event processing pipeline.

Find the 'Job Topology' section and let's add an input by selecting 'Inputs' and then '+ Add'.

Select the details for the event hub just created as a Stream Analytics input and click Create:

- **Input Alias:** eventhub
- **Source Type:** Data Stream
- **Source:** Event Hub
- **Subscription:** Use event hub from current subscription
- **Service Bus namespace:** Choose the previously created namespace (example: lab4<yourinitials>-ns)
- **Event Hub Name:** Choose the previously created event hub (example: lab4<yourinitials>)
- **Event Hub Policy Name:** manage
- **Event Hub Consumer Group:** blank
- **Event Serialization format:** JSON
- **Encoding:** UTF-8
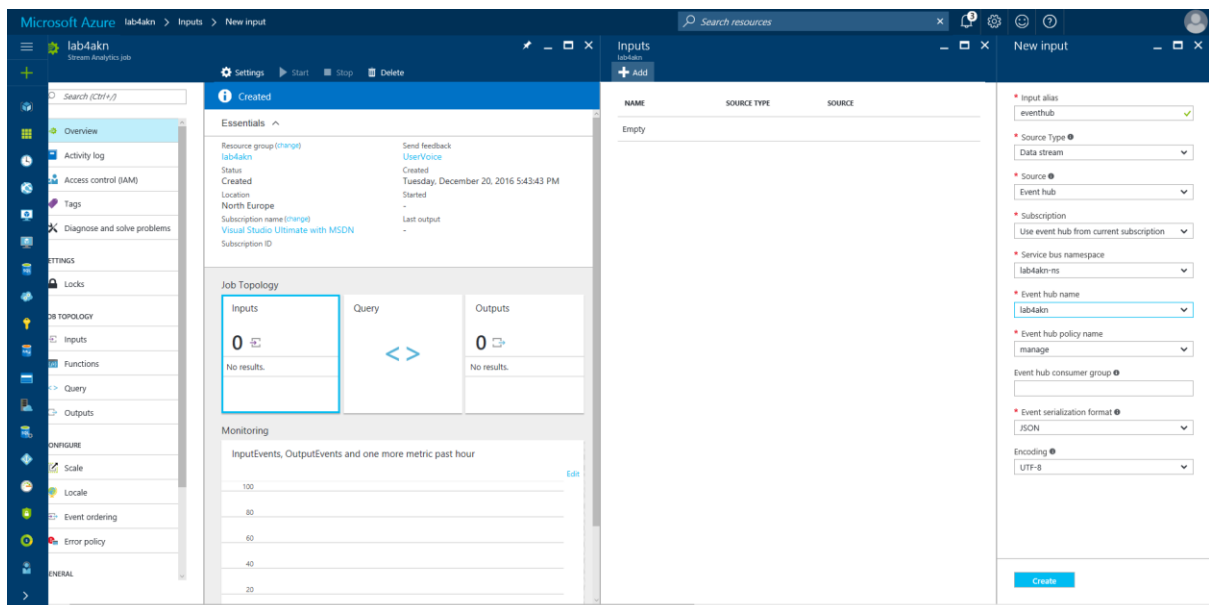
Next, we need to add an output in the same way. Select the 'Outputs' box in Job Topology -> '+ Add' and enter the details below and click Create:

- **Output Alias:** blobstor
- **Sink:** Blob Storage
- **Subscription:** Use blob storage from current subscription
- **Storage account:** Choose the previously created storage account (example: lab4data)
- **Storage account key:** *already populated*
- **Container:** output
- **Path pattern:** {date}
- **Date format:** YYYY/MM/DD
- **Time format:** *already populated* HH
- **Event Serialization format:** JSON
- **Encoding:** UTF-8
- **Format:** Line separated
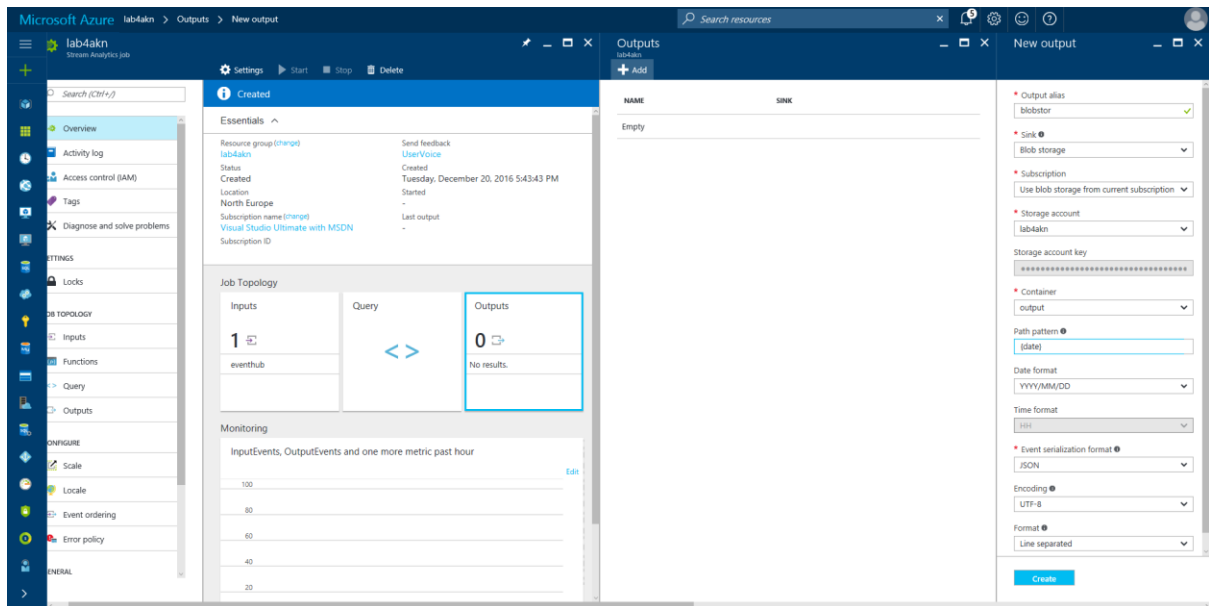
With the Stream Analytics input and output architecture complete, we are ready to create the logic that queries over the real time data. The setup should look like below:
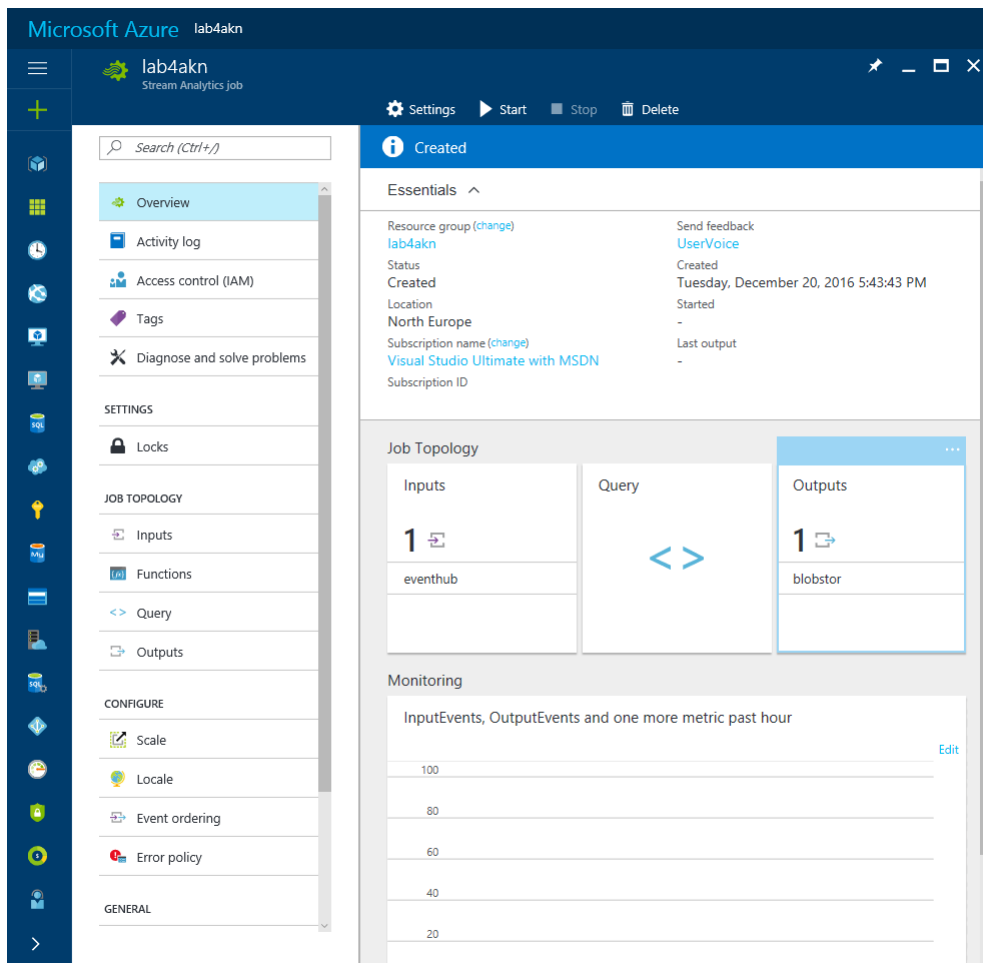
Create Real-Time Event Processing Queries

In this section we will start adding logic to the Stream Analytics service, starting with simple queries and adding more complexity. We will also see how to use test data to confirm the behaviour of your logic.

Create a Simple Query and Test

Open your Stream Analytics service and find the Query tab under 'Job Topology'.
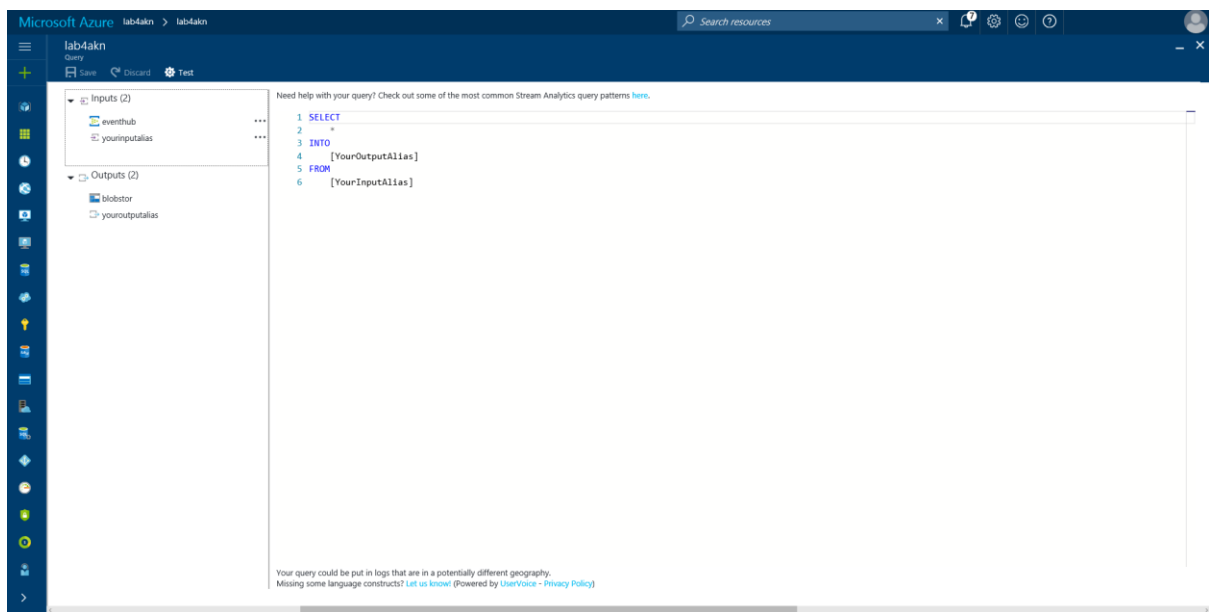
Click the Query box to edit the query so it selects all the data and sends it to your output storage account from your event hub.

With the query complete you need to test the output is as expected to return all the data from the event hub into the blob storage. To do this we need to upload some sample data to query against.

Under the 'Inputs' heading you will see your eventhub listed, select the three dots '…' and choose 'Upload sample data from file'
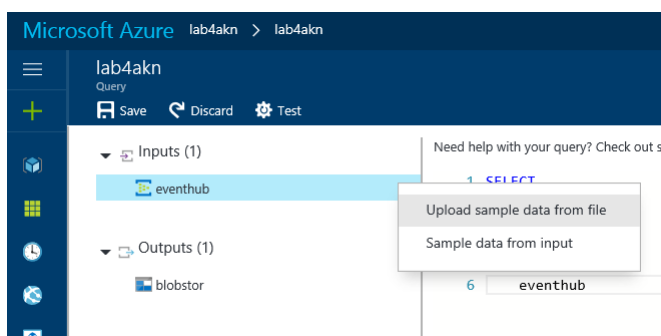
Choose to upload the **lab4_testdata.json** file provided in the GitHub repo here:
https://github.com/MicrosoftLearning/BuildingIntelligentApplications

Upload Test data

"lab4_testdata.json"
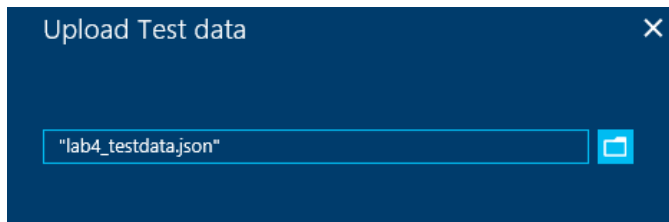
Once uploaded, click the Test button and the query will run on the test data and the results will be returned.

Assessment Question: Make a note of how many events the event hub processed? And how many columns there are in the test dataset passed to blobstor?

Well done! You now have a very simple query running. In the next sections, we will add further complexity to this query.

Add Filtering and Formatting

Now filter and format the results. Format the results by selecting only the columns needed as shown below:
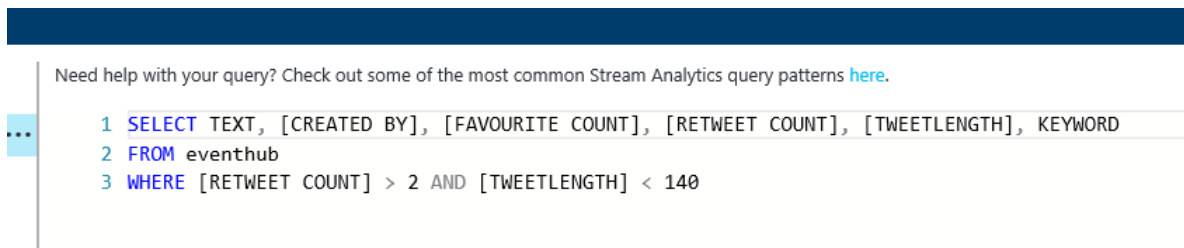
TEXT, [CREATED BY], [FAVOURITE COUNT], [RETWEET COUNT], [TWEETLENGTH], KEYWORD

And filter the results where:

- **[RETWEET COUNT]** is greater than 2
- **[TWEETLENGTH]** is less than 140

Need help with your query? Check out some of the most common Stream Analytics query patterns here.

```
1 SELECT TEXT, [CREATED BY], [FAVOURITE COUNT], [RETWEET COUNT], [TWEETLENGTH], KEYWORD
2 FROM eventhub
3 WHERE [RETWEET COUNT] > 2 AND [TWEETLENGTH] < 140
```

Test this new query against the same test data by clicking 'Test'.

Assessment Question: How many rows were output to blobstor once the filter was added to the query?

Insert a Timestamp

Next, let's view multiple different timestamps we can access with Stream Analytics and rename them in the query, so we understand what they are:

- **[TIMESTAMP]** AS [COLLECTED AT]
- **DATE.TIME** AS [CREATED AT]
- **System.Timestamp** AS [RECEIVED AT]

```
1 SELECT TEXT, [CREATED BY], [FAVOURITE COUNT], [RETWEET COUNT], [TWEETLENGTH], KEYWORD, [TIMESTAMP] AS [COLLECTED AT], DATE.TIME AS [CREATED AT]
2 , System.TimeStamp AS [RECIEVED AT]
3 FROM eventhub
4 WHERE [RETWEET COUNT] > 2 AND [TWEETLENGTH] < 140
```

Assessment Question: Which timestamps are part of the data payload and which timestamp is not?

Retest the query just created and view the extra columns added to the payload and the names they have in the blobstor output.

Add a Windowing Function

Next, we will add a windowing function to make the query aggregate the Twitter data over time based on KEYWORD and [CREATED AT].

Change the query in Stream Analytics to the T-SQL query below and replace the TumblingWindow parameters X and Y with the correct value for a TumblingWindow aggregating every 10 minutes.

HINTS

- **X** is time format
- **Y** is window time value

```
1 SELECT System.Timestamp AS WINDOWEND, KEYWORD,  COUNT(*) AS Count
2 FROM eventhub TIMESTAMP BY Date.Time
3 GROUP BY KEYWORD, TumblingWindow(X,Y)
```

After testing the output of this query, make sure you select the 'Save' icon in the bottom toolbar and confirm the Save.

Run the Real Time Event Processing Solution

In this section, we will run an Azure Stream Analytics pipeline against real-time data using a Twitter Client Application sending data to Azure and viewing the results of the Stream Analytics aggregation in blob storage.
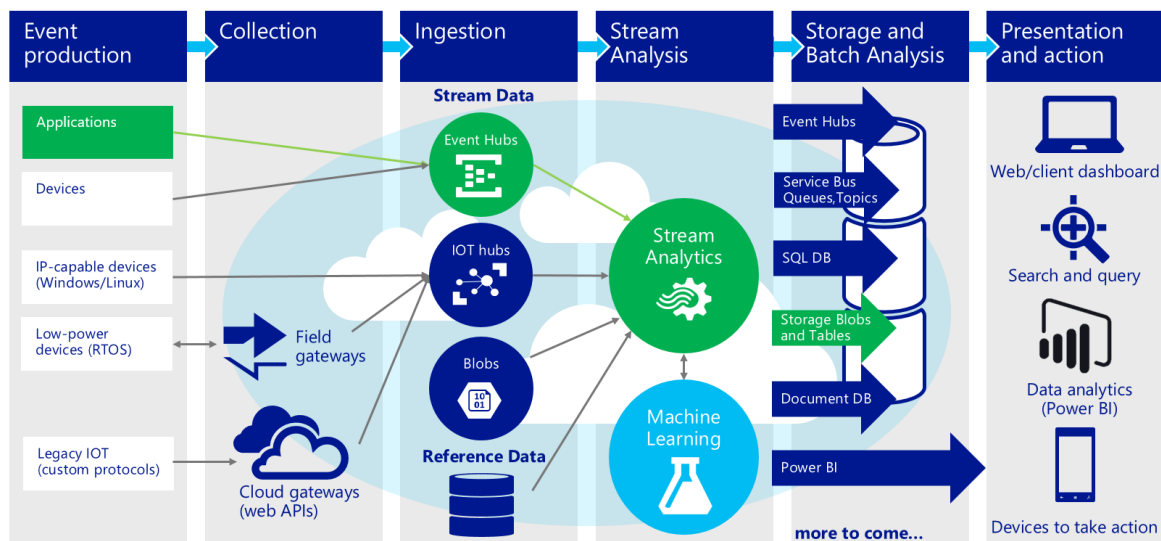
Start the Azure Stream Analytics Service

Enter a slightly different query as shown below and save in the editor:

```
1 SELECT System.Timestamp AS WINDOWEND,Topic, COUNT(*) AS Count
2 FROM eventhub TIMESTAMP BY createdat
3 GROUP BY Topic, TumblingWindow(second,10)
```

Start the Stream Analytics service by selecting the 'Start' button in the bottom toolbar and make sure it starts running from now (not a custom time).

NOTE: The streaming service may take a couple of minutes to start, but you will receive a notification in the Management Portal that the service is now running.

Modify the Twitter Client Application

Now we need to make sure the Twitter client application has all the correct keys to collect data from Twitter and send it to the Stream Analytics event hub service.

Download the Twitter Client solution from the lab GitHub repository: https://github.com/MicrosoftLearning/BuildingIntelligentApplications and open the solution in Visual Studio 2015 Community Edition.

Once the solution is open, view the solution explorer and find the App.config file and open it.
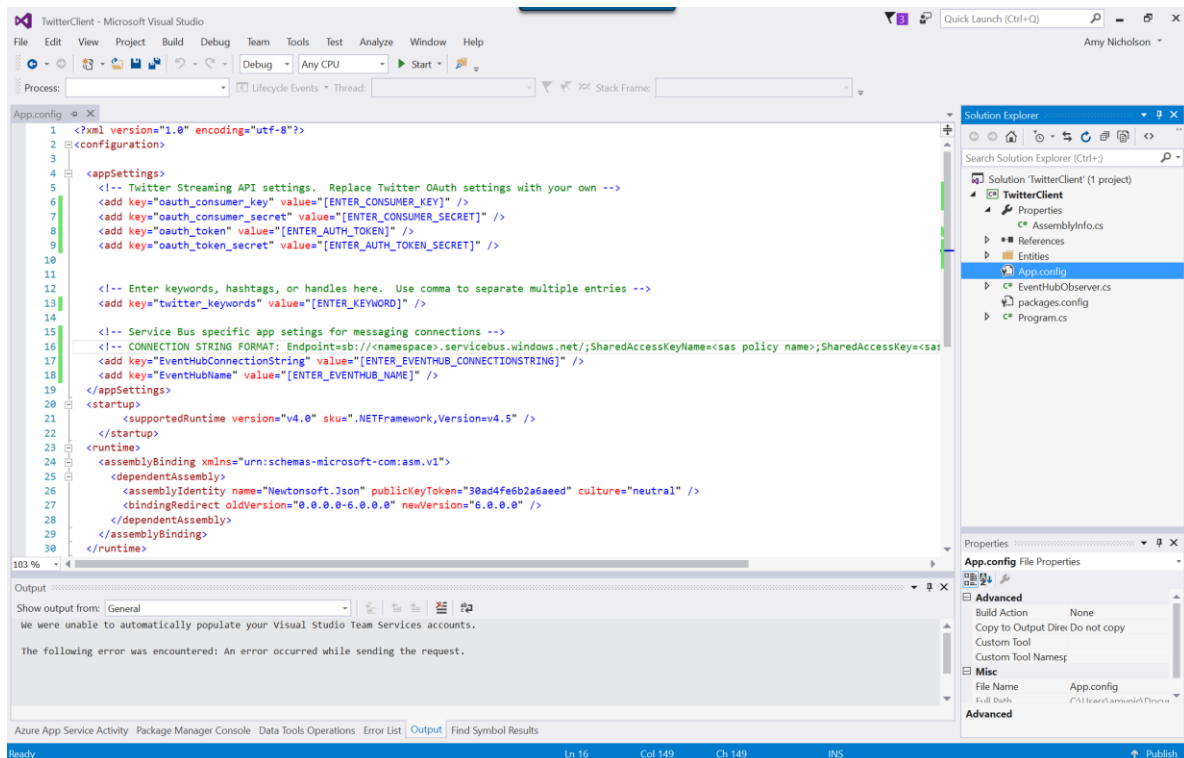
Now replace the values below with the relevant keys:

- **Twitter Consumer Key:** Use the credentials you created in Lab 1, for more information visit: https://apps.twitter.com/
- **Twitter Consumer Secret**
- **Twitter Auth Token**
- **Twitter Auth Token Secret**
- **Twitter Keyword:** Enter a keyword you wish to search for
- **Event Hub Connection String:** The event hub you have created in the first section of this lab
  - It should be in the format:
    ```
    Endpoint=sb://<namespace>.servicebus.windows.net/;SharedAccessKeyName=<s
    as policy name>;SharedAccessKey=<sas key>
    ```
- **Event Hub Name**

Save these changes and right-click the solution and choose 'Build Solution'.

If the build succeeds, then choose the Start button in the top toolbar in order to start collecting tweets and sending them to your event hub.

A console window will open, and you should start receiving tweets relating to the keyword you selected.

NOTE: If there are no tweets with that keyword being sent try something else (e.g. machine learning, Microsoft, etc.).
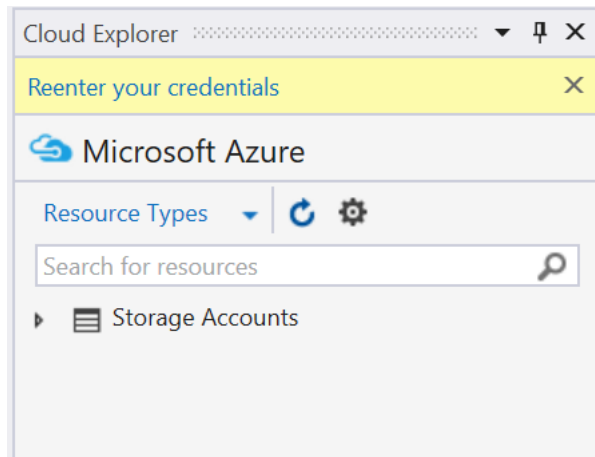
View the Results of the Running Pipeline

Leave the code running and let's view the output of the stream analytics pipeline.

Inside Visual Studio 2015, choose View -> Cloud Explorer to obtain the toolbar.
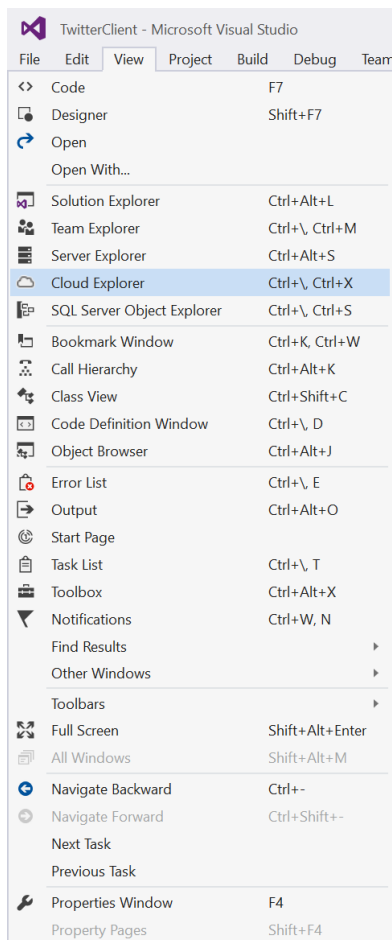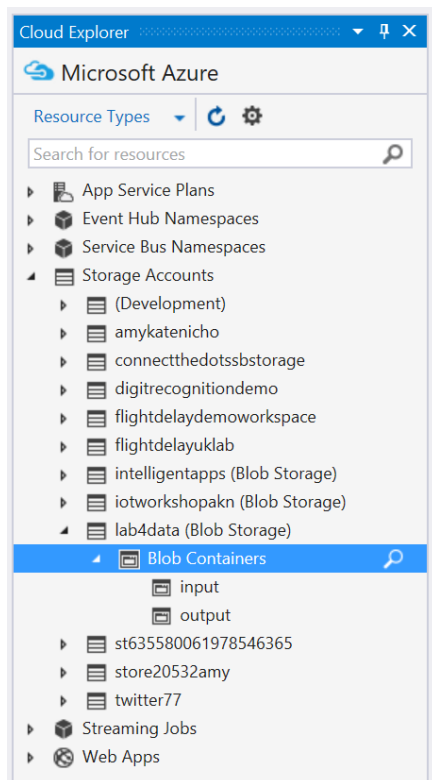
You are likely to be prompted to enter your Azure account credentials so the explorer can populate your resources.

Find the storage/container you created for the output of the Stream Analytics job. For example:
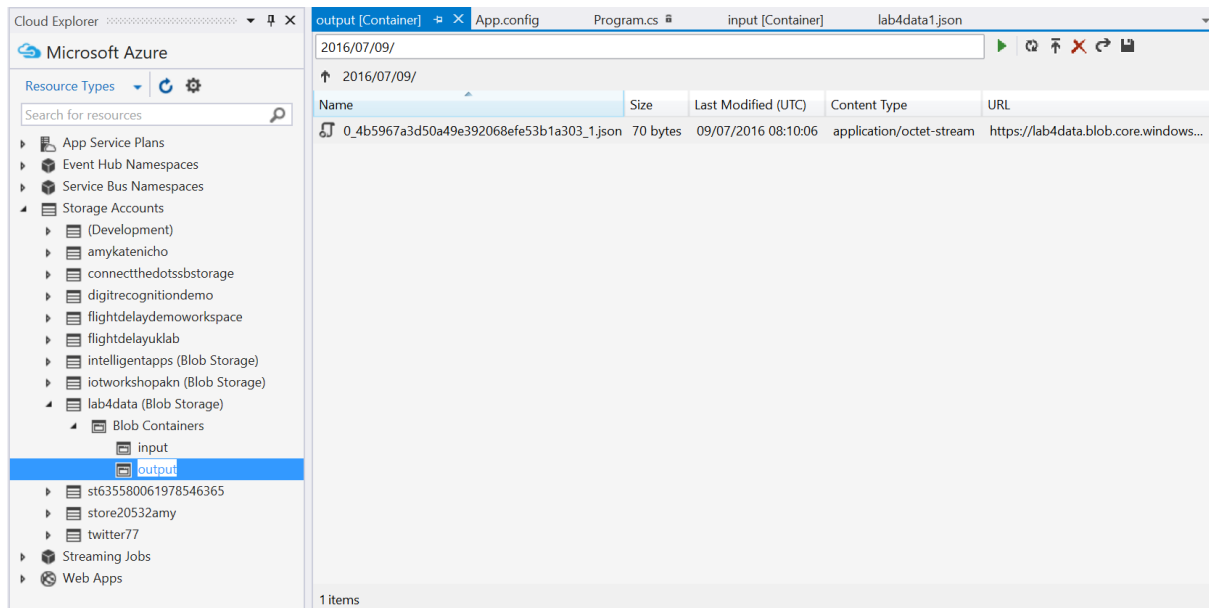
Storage Accounts -> lab4data -> Blob Containers -> output

Double click to open the 'output' container. Then you will see hierarchical folders for the date:

Example: 2016/07/09

Open these folders until you get to a JSON file.

Double click to download and open the file to see the JSON output from the Stream Analytics query. The count of tweets by topic over a tumbling window will be shown.

NOTE: feel free to shorten the tumbling window to 10 seconds to see a faster result of multiple lines of JSON data
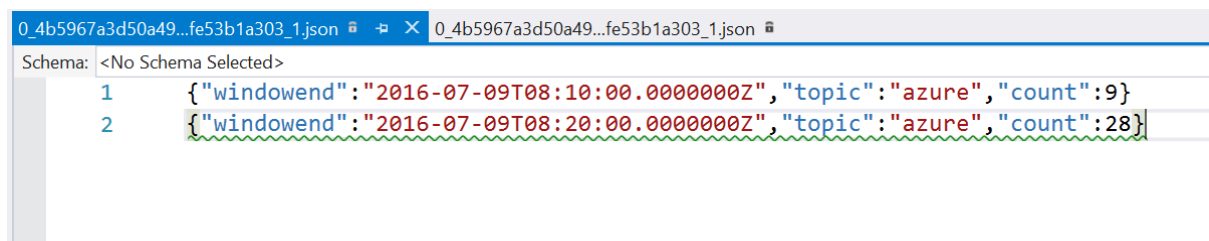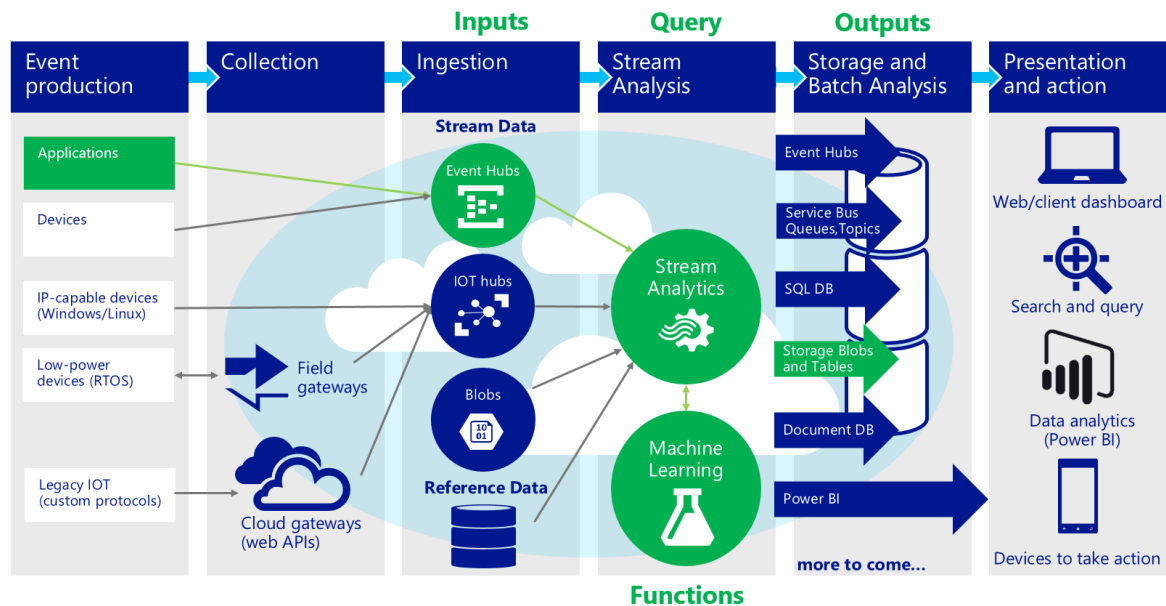

Stop the Azure Stream Analytics Service

Once you have finished investigating your end-to-end Stream Analytics Pipeline, go back to the portal and stop the service from running.
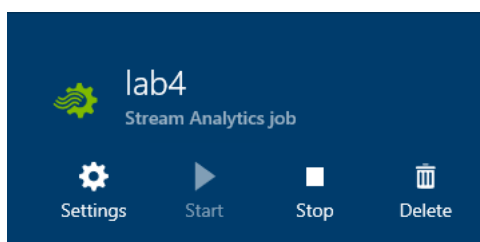
Integrate an Azure Machine Learning Algorithm

In this section, we will add additional complexity to our T-SQL query by including a call to an Azure Machine Learning web service.
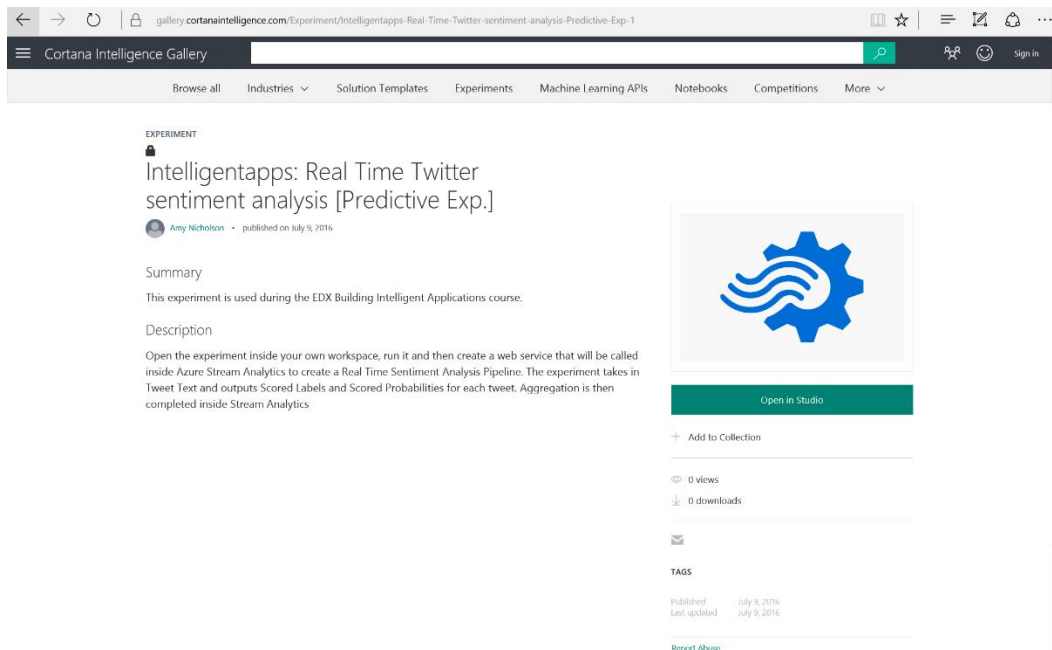

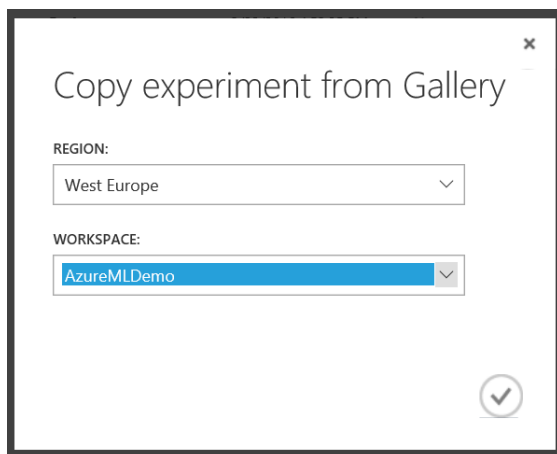## Create an Azure Machine Learning Web Service

First, we need to create a Sentiment Analysis Azure Machine Learning web service to call. Sign into Azure Machine Learning: http://studio.azureml.net
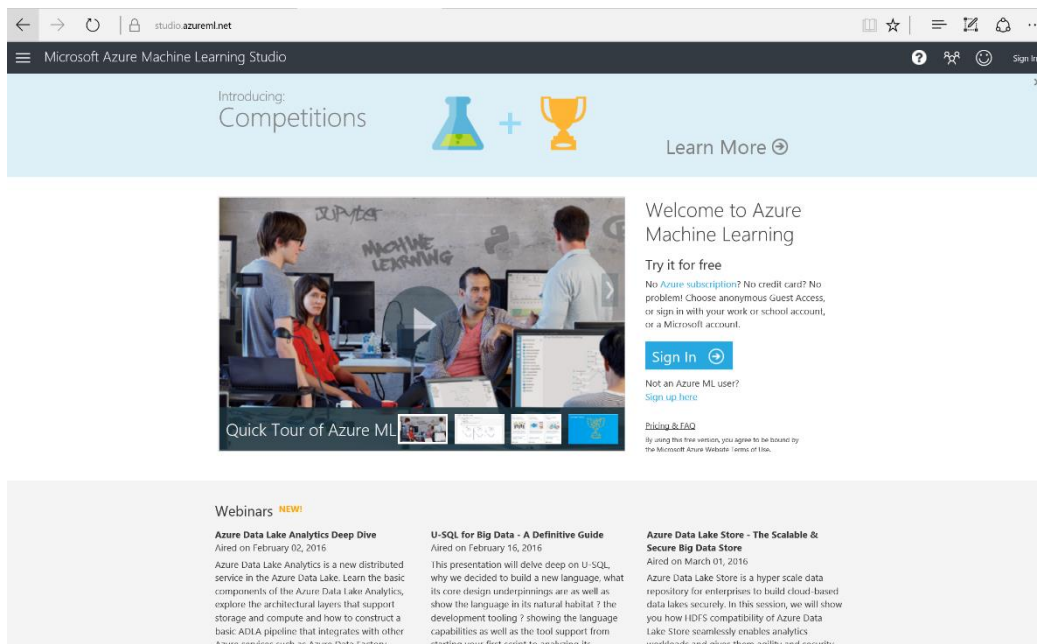
then open a new browser tab and go to https://aka.ms/lab4experiment

Here you will find a completed Sentiment Analysis Predictive Experiment, which takes in Tweet Text and returns Scored Labels and Scored Probabilities. Choose 'Open in Studio' and select a workspace to copy the experiment to:

Next, use the 'Run' button in the Azure Machine Learning Studio to run the experiment until all modules have green ticks next to them. You now need to publish this as a web service to gain the web service endpoint and key to use in Azure Stream Analytics. Choose 'Deploy as a Web Service (Classic)' on the bottom toolbar:
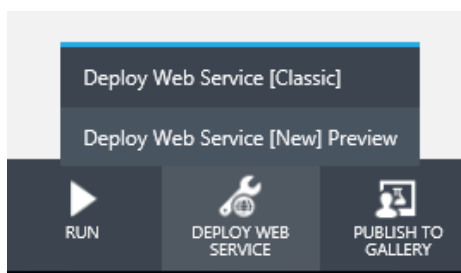
This will create a Web Service Page like below:
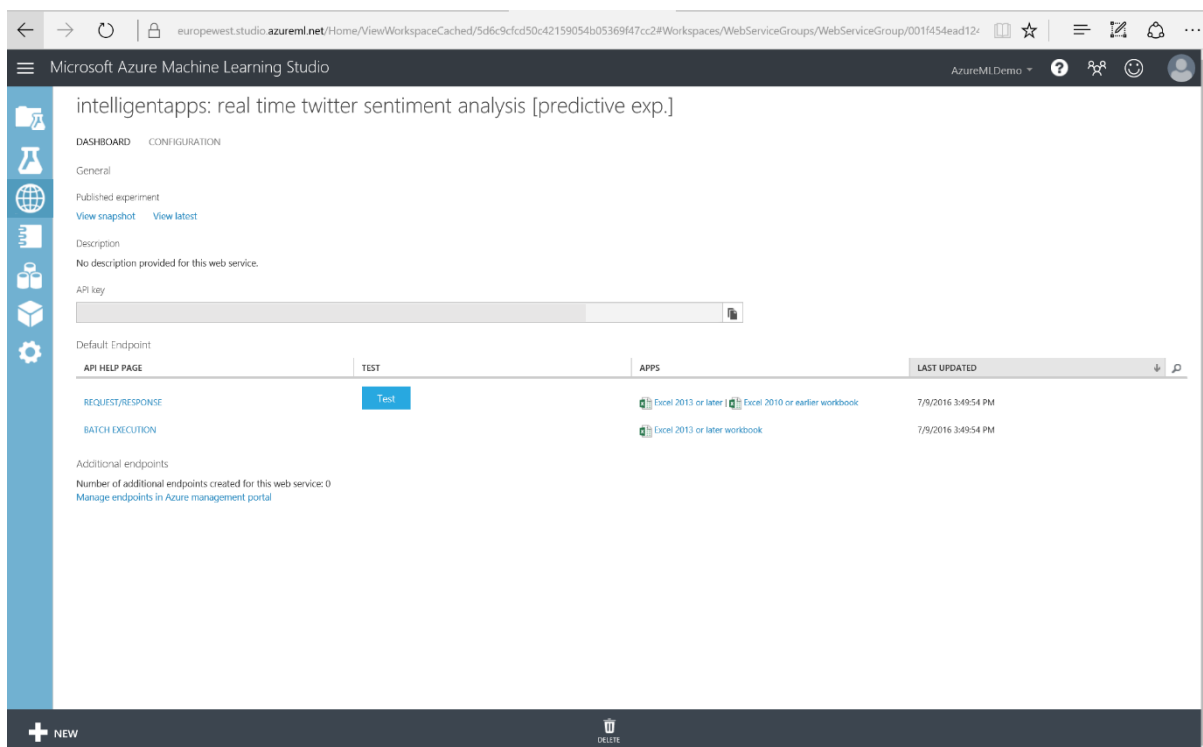
To configure this web service in Stream Analytics we need two pieces of information:

- **Web Endpoint URL:** Open the REQUEST/RESPONSE API help page, find the Heading Request and take note of the post request endpoint value. You need the part of the URL in bold and green below. Do not include the part that is not bold in red.
  - **https://<datacenterregion>.services.azureml.net/workspaces/<workspaceid>/services/<serviceid>/**execute?api-version=2.0&details=true
- **API Key:** The API key can be found on the front web service page, take a note of this.

Edit the Azure Stream Analytics Query

You have published a Machine Learning web service. Now we need to add this change inside the Azure Stream Analytics job. Open the job created previously inside the Azure Preview Portal.
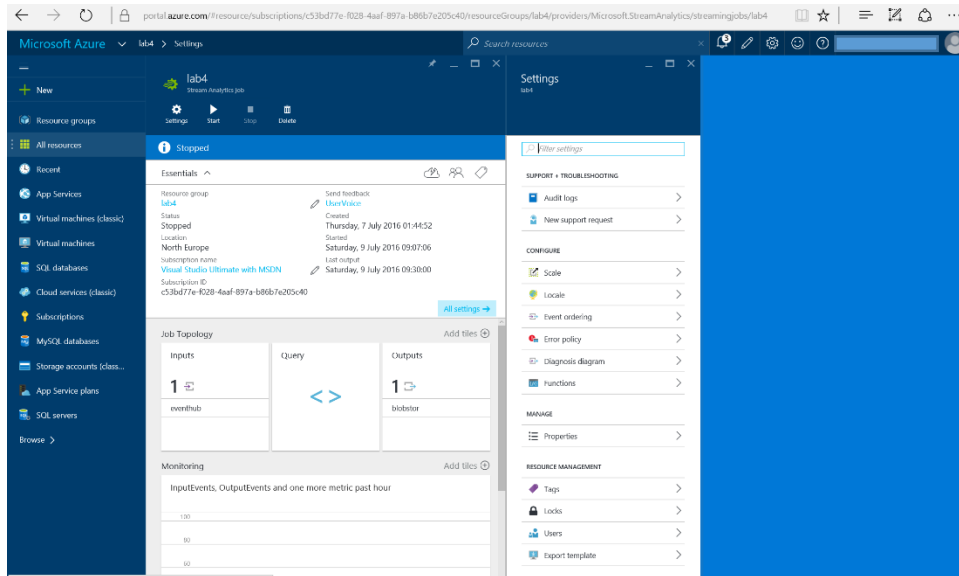
The input will stay the same, and the tweets will still come from your Twitter Client Application in JSON format.

The output will stay the same, and you will view the averaged sentiment of tweets in real time in the Cloud Explorer Storage Account UI.

To configure the Azure Machine Learning function, go to 'Settings' -> 'Functions' -> 'Add' and enter the details below and click 'Create':

- **Function Alias:** sentiment
- **Function Type:** Azure ML
- **Subscription:** Import from a different subscription
- **URL:** Your web service endpoint URL noted down previously
- **Key:** Your web service key, noted down previously

This is now added as an Azure Machine Learning function that you can call in your Stream Analytics query.

Next, let's edit the query to take this new functionality into account as shown below and 'Save' the query changes.

```
1  WITH step1 AS (
2      SELECT Text, sentiment(Text) as Result, Topic, CreatedAt
3      FROM eventhub
4  )
5
6  SELECT avg(Result.[Scored Probabilities]) as AvgSentiment
7  INTO blobstor
8  FROM step1
9  GROUP BY Topic, TumblingWindow(second,10)
```

Run and View the Results

To view the results of the real-time sentiment analysis, start the Stream Analytics job

Once this is running, start the previous Twitter Client Application in Visual Studio 2015 to start collecting new tweets.

Once everything has been running for a minute go to the Cloud Explorer in Visual Studio and find the output container for your Storage account.
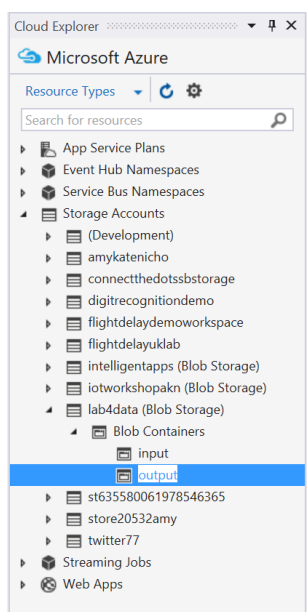
You should now see a JSON file that is filling up with AvgSentiment values every 10 seconds due to the tumbling window syntax.

Summary

Well done for completing the lab exercise! Make sure you answer the Lab Assessment Questions relating to the lab material.

In this lab you have:

- Created an Azure Stream Analytics Architecture
- Written and tested Stream Analytics T-SQL queries
- Ran the Stream Analytics service and viewed the results in Blob Storage
- Published an Azure ML experiment as a web service
- Integrated the Azure ML web service into the Stream Analytics query