Twitter Sentiment Analytics

By Microsoft Azure Analytics Platform

Provide By: Keivan Davari, Credexo

Twitter Sentiment Analytics

By Microsoft Azure Analytics Platform

In the following Document, we show the process of sentiment analytics of twitter statuses by Microsoft azure technologies like Machine learning, Stream Analytics, Service Bus and Power Query.

Table Of Content

Microsoft Azure Analytics Overview

Microsoft Azure Stream Analytics

Azure Stream Analytics is a fully managed service providing low-latency, highly available, scalable, complex event processing over streaming data in the cloud. It's a big data analytics service for the Internet of Things (IoT) that enables developers to combine streams of data with historic records or reference data to derive business insights easily and quickly.

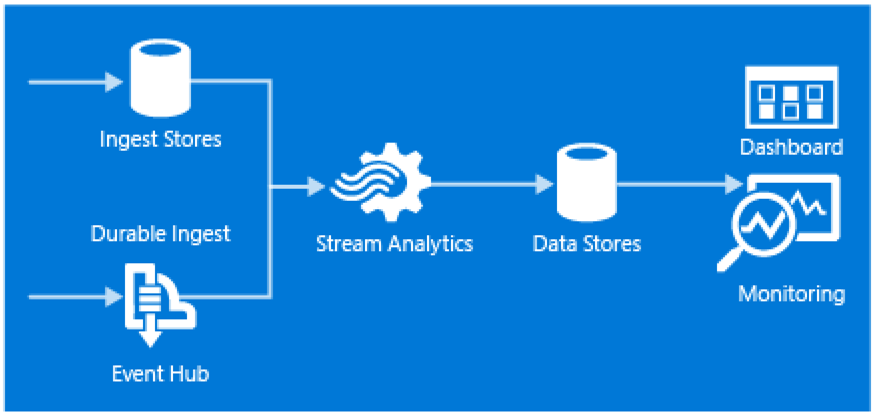


FIGURE 1, STREAM ANALYTICS ARCHITECTURE

Microsoft Azure Machine Learning

Machine learning is at work all around you. When you shop online, machine learning helps recommend other products based on what you've purchased. When your credit card is swiped, machine learning helps the bank do fraud detection and notify you if the transaction seems suspicious.
Machine learning uses predictive models that learn from existing data in order to forecast future behaviors, outcomes, and trends.

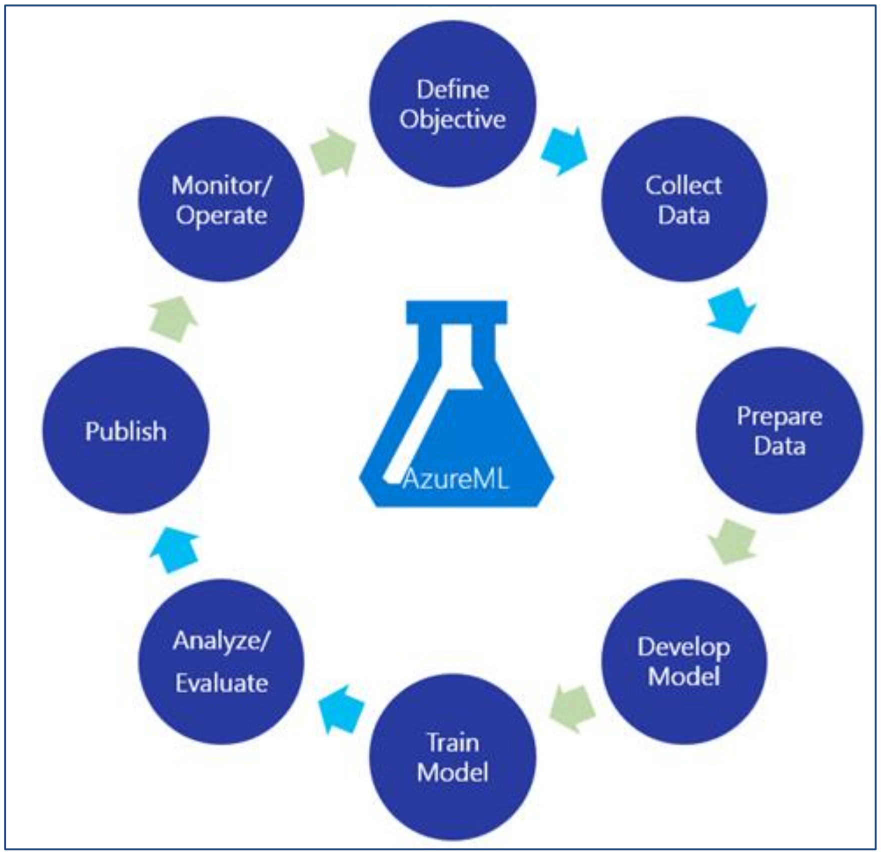


FIGURE 2, AZURE MACHINE LEARNING ARCHITECTURE

Microsoft Azure Service Bus

Service Bus provides a multi-tenant service for connecting applications through the cloud.
Within a namespace, you can use one or more instances of four different communication mechanisms, each of which connects applications in a different way. The choices are:
Queues: which allow one-directional communication. Each queue acts as an intermediary (sometimes called a broker) that stores sent messages until they are received. Each message is received by a single recipient.
Topics: which provide one-directional communication using subscriptions-a single topic can have multiple subscriptions. Like a queue, a topic acts as a broker, but each subscription can optionally use a filter to receive only messages that match specific criteria.
Relays: which provide bi-directional communication. Unlike queues and topics, a relay doesn't store in-flight messages, it's not a broker. Instead, it just passes them on to the destination application.
Event Hubs: which provide event and telemetry ingress to the cloud at massive scale, with low latency and high reliability**.**

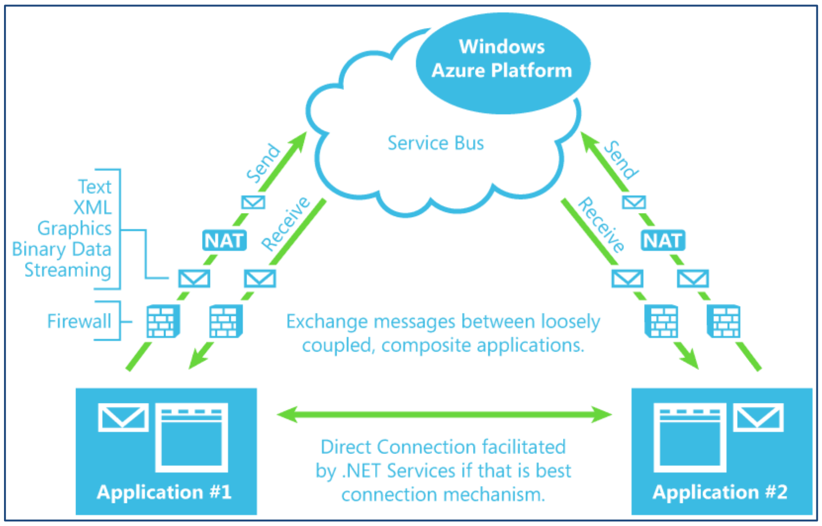
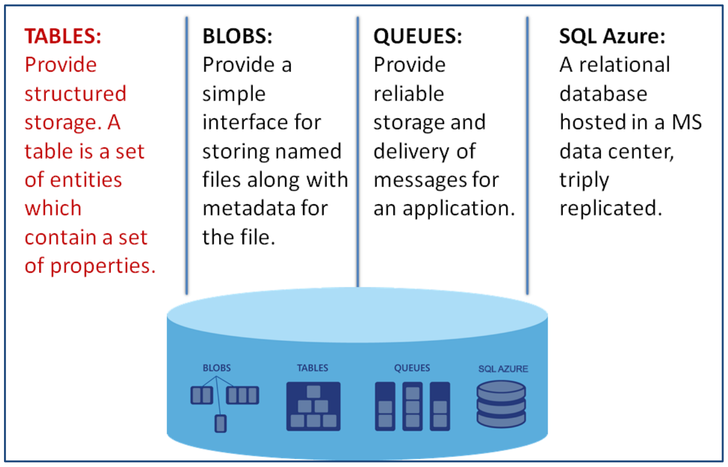

Windows Azure Platform
Service Bus
Send
Receive
Send
Receive
Text
XML
Graphics
Binary Data
Streaming
NAT
NAT
Firewall
Exchange messages between loosely coupled, composite applications.
Direct Connection facilitated by .NET Services if that is best connection mechanism.
Application #1
Application #2

FIGURE 3, AZURE SERVICE BUS SEND AND RECEIVE

When you create a queue, topic, relay, or Event Hub, you give it a name. Combined with whatever you called your namespace, this name creates a unique identifier for the object. Applications can provide this name to Service Bus, then use that queue, topic, relay, or Event Hub to communicate with one another.

Microsoft Azure Storage

Cloud computing enables new scenarios for applications requiring scalable, durable and highly available storage for their data – which is exactly why Microsoft developed Azure Storage. In addition to making it possible for developers to build large-scale applications to support new scenarios, Azure Storage also provides the storage foundation for Azure Virtual Machines, a further testament to its robustness.
Azure Storage is massively scalable, so you can store and process hundreds of terabytes of data to support the big data scenarios required by scientific, financial analysis, and media applications. Or you can store the small amounts of data required for a small business website. Wherever your needs fall, you pay only for the data you're storing. Azure Storage currently stores tens of trillions of unique customer objects, and handles millions of requests per second on average.
Azure Storage is elastic, so you can design applications for a large global audience, and scale those applications as needed - both in terms of the amount of data stored and the number of requests made against it. You pay only for what you use, and only when you use it.
Azure Storage uses an auto-partitioning system that automatically load-balances your data based on traffic. This means that as the demands on your application grow, Azure Storage automatically allocates the appropriate resources to meet them.
Azure Storage is accessible from anywhere in the world, from any type of application, whether it's running in the cloud, on the desktop, on an on-premises server, or on a mobile or tablet device. You can use Azure Storage in mobile scenarios where the application stores a subset of data on the device and synchronizes it with a full set of data stored in the cloud.
Azure Storage supports clients using a diverse set of operating systems (including Windows and Linux) and a variety of programming languages (including .NET, Java, and C++) for convenient development. Azure Storage also exposes data resources via simple REST APIs, which are available to any client capable of sending and receiving data via HTTP/HTTPS.


TABLES: Provide structured storage. A table is a set of entities which contain a set of properties.

BLOBS: Provide a simple interface for storing named files along with metadata for the file.

QUEUES: Provide reliable storage and delivery of messages for an application.

SQL Azure: A relational database hosted in a MS data center, triply replicated.

FIGURE 4, AZURE STORAGE OPTIONS

Microsoft Azure Data Factory

Traditionally, data integration projects have revolved around creating Extract-Transform-Load (ETL) processes that extract data from various data sources within an organization, transform the data to conform to the target schema of an Enterprise Data Warehouse (EDW), and load the data into an EDW. The EDW is then accessed as the single source of truth for BI analytics solutions.

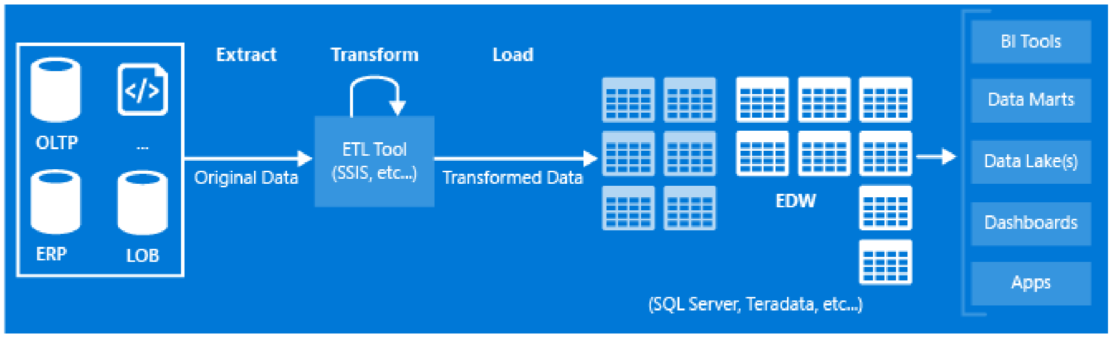


FIGURE 5, AZURE DATA FACTORY MODEL

Today's data landscape for enterprises continues to grow exponentially in volume, variety, and complexity. It is more diverse than ever with on-premises and cloud-born data of different forms and velocities. The data processing must happen across geographic locations, and includes a combination of open source software, commercial solutions and custom processing services and that are expensive, and hard to integrate and maintain. The agility needed to adapt to today's changing Big Data landscape is an opportunity to merge the traditional EDW with capabilities required for a modern information production system.

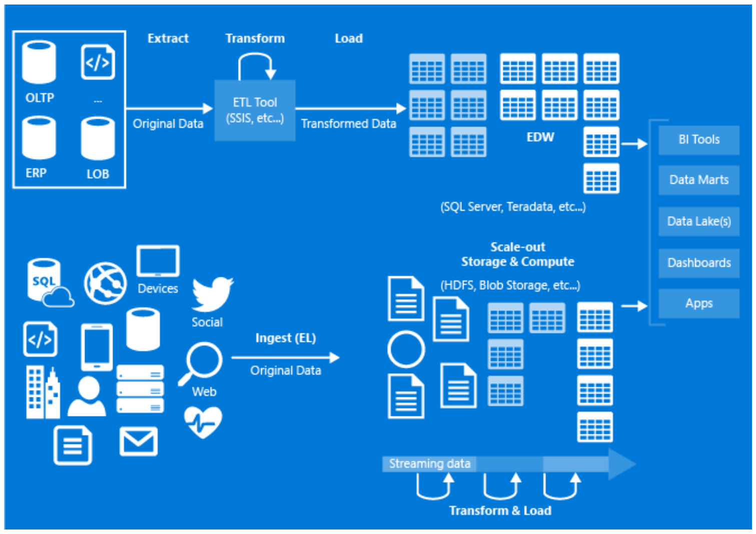


FIGURE 6, AZURE DATA FACTORY EXTENDED MODEL

The following diagram illustrates the application model supported by the Azure Data Factory service.

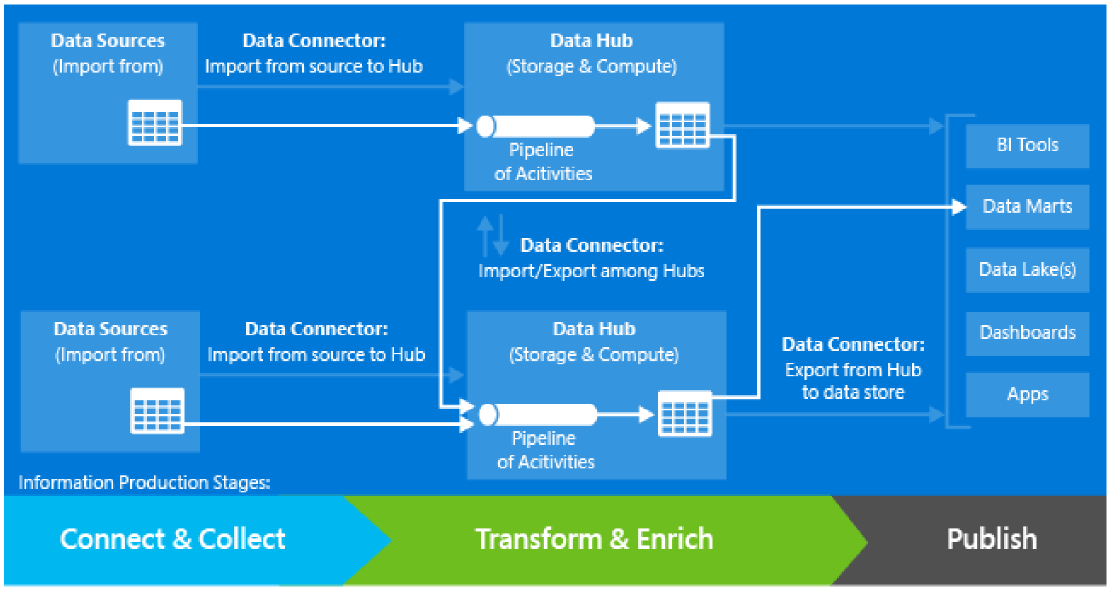


FIGURE 7, AZURE DATA FACTORY APPLICATION MODEL

There are three information production stages in an Azure data factory:

Connect & Collect: In this stage, data from various data sources is imported into data hubs. A pipeline in a data factory can have one or more activities. You use one or more Copy activities in a data pipeline to collect data from source data stores to a destination data store with in a data hub for further processing. An HDInsight cluster (compute) and its associated Azure blob storage (storage) together form a data hub, a HDInsight data hub. To use a HDInsight data hub, you copy all the source data into an Azure blob store associated with the HDInsight so that the data can be processed by HDInsight cluster. A pipeline runs on a compute resource in a data hub such as an HDInsight cluster.

Transform & Enrich: In this stage, the collected data is processed. For example, a HDInsight Activity in a pipeline can process data stored in the associated Azure blob store by performing transformations using Hive/Pig scripts to produce trusted information. Pipelines can be chained (as shown in the diagram) such that output data sets of a pipeline can be input data sets for another pipeline in the same data hub or another data hub.

Publish: In this stage, the data is published so that it can be consumed by BI tools, analytics tools, and other applications. For example, a Copy Activity in the pipeline can copy output data from the processing performed in the Transform & Enrich stage to a data store (for example: on-premises SQL Server) on top of which business intelligence solutions can be built.

Twitter Sentiment Analytics Process

Overview the process:

Here is a chart that show the architecture of the process:

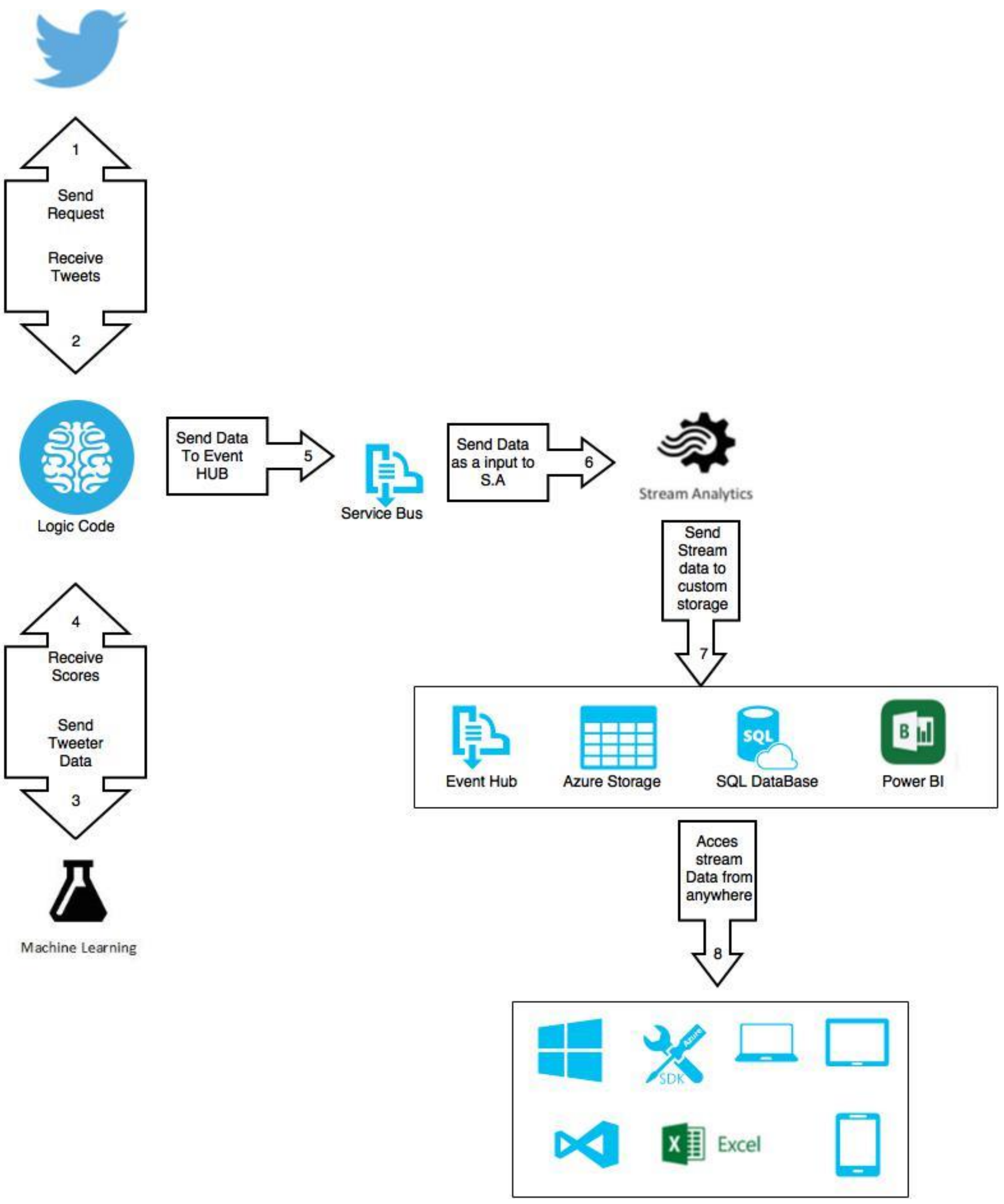
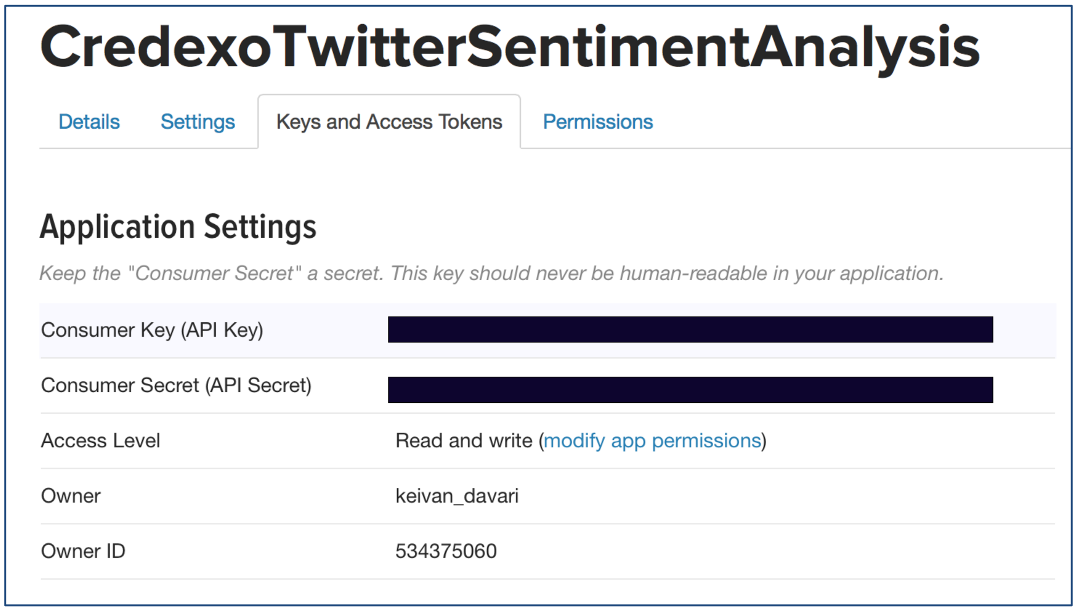


FIGURE 8, TWITTER SENTIMENT ANALYTICS ARCHITECTURE

Step 1: Communicate to Twitter

The first thing we should do is getting API keys from twitter account to connect to twitter from the logic code. The keys can be found in developer page in twitter account.



CredexoTwitterSentimentAnalysis

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	[redacted]
Consumer Secret (API Secret)	[redacted]
Access Level	Read and write (modify app permissions)
Owner	keivan_davari
Owner ID	534375060

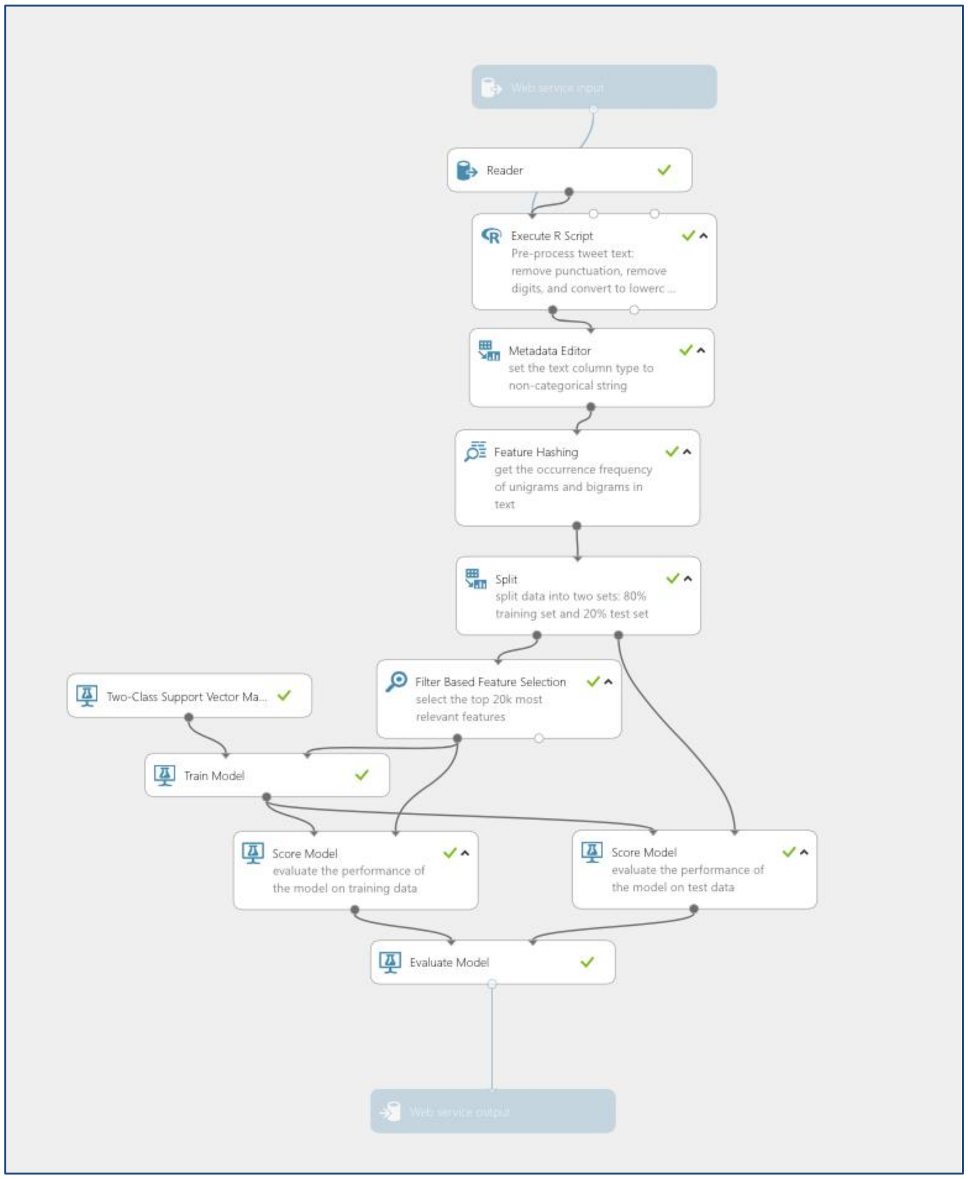
FIGURE 9, TWITTER API KEY PAGE

Now, we can connect to the twitter stream service from the logic code and get data from twitter:

```
public static IEnumerable<Tweet> StreamStatuses(TwitterConfig config)
{
    DataContractJsonSerializer jsonSerializer = new DataContractJsonSerializer(typeof(Tweet));

    var streamReader = ReadTweets(config);

    while (true)
    {
        string line = null;
        try { line = streamReader.ReadLine(); }
        catch (Exception) { }

        if (!string.IsNullOrWhiteSpace(line) && !line.StartsWith("{\"delete\""))
        {
            var result = (Tweet)jsonSerializer.ReadObject(new MemoryStream(Encoding.UTF8.GetBytes(line)));
            result.RawJson = line;
            yield return result;
        }
```

Step 2: Analyzing Tweets with Machine Learning

Now, we should create our model to analyze our tweets. In this step we need to work in machine learning studio. As we know, the machine learning studio create models based on some real data. For providing real data we use a database that has a 1,600,000 tweets with their sentiment scores.



Execute R Script
Pre-process tweet text: remove punctuation, remove digits, and convert to lowerc ...

Metadata Editor
set the text column type to non-categorical string

Feature Hashing
get the occurrence frequency of unigrams and bigrams in text

Split
split data into two sets: 80% training set and 20% test set

Two-Class Support Vector Ma...

Filter Based Feature Selection
select the top 20k most relevant features

Train Model

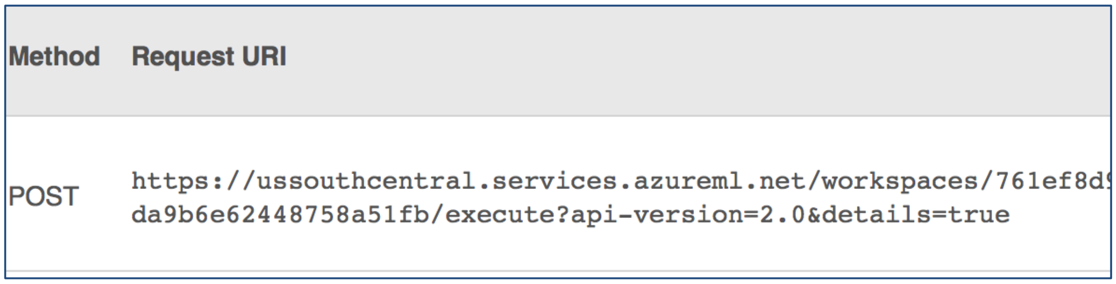
Score Model
evaluate the performance of the model on training data

Score Model
evaluate the performance of the model on test data

Evaluate Model

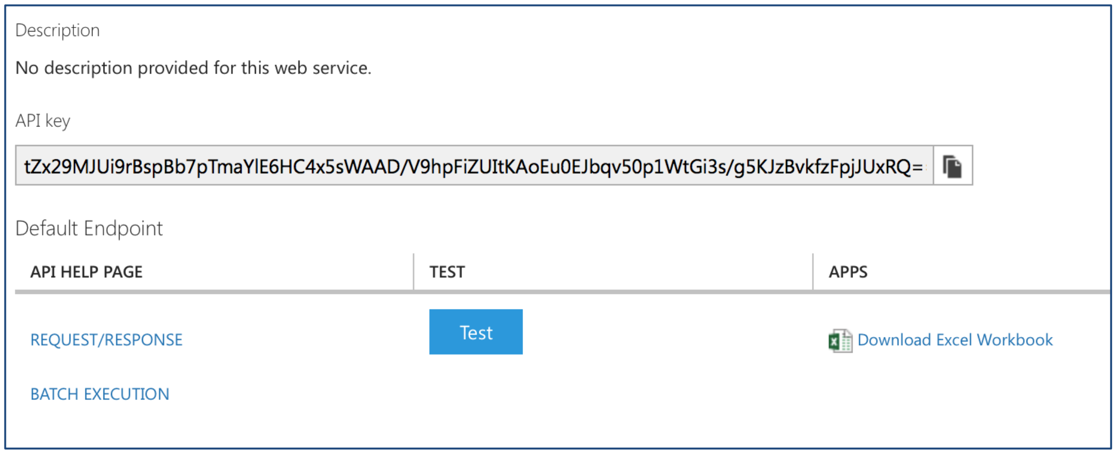
Figure 10, Machine Learning Studio

After creating the model we need to make a web service form the model to sending the data from our logic code. After creating the web service, azure give e API Key and Request URI to connect to the model from the Logic code:



Method	Request URI
POST	https://ussouthcentral.services.azureml.net/workspaces/761ef8d9 da9b6e62448758a51fb/execute?api-version=2.0&details=true

FIGURE 11, MACHINE LEARNING WEB SERVICE REQUEST URI



Description

No description provided for this web service.

API key

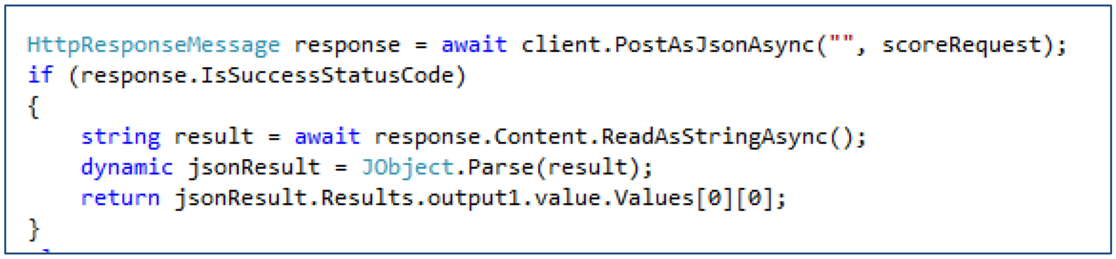
tZx29MJUi9rBspBb7pTmaYlE6HC4x5sWAAD/V9hpFiZUItKAoEu0EJbqv50p1WtGi3s/g5KJzBvkfzFpjJUxRQ=

Default Endpoint

API HELP PAGE	TEST	APPS
REQUEST/RESPONSE	Test	Download Excel Workbook
BATCH EXECUTION		

FIGURE 12, MACHINE LEARNING WEB SERVICE API KEY

Now we can send the tweets form our logic code to Azure Machine Learning and receive the sentiment scores from the Azure machine learning:

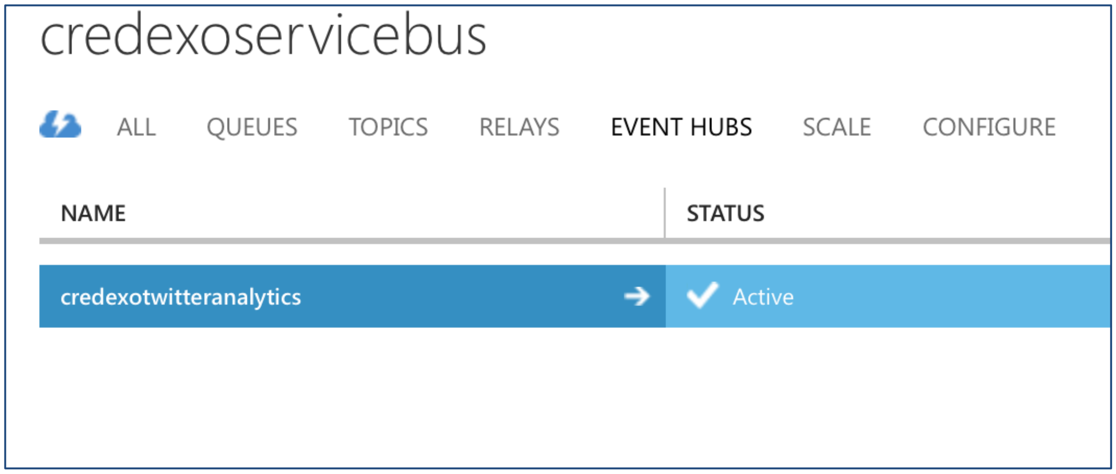


```
HttpResponseMessage response = await client.PostAsJsonAsync("", scoreRequest);
if (response.IsSuccessStatusCode)
{
    string result = await response.Content.ReadAsStringAsync();
    dynamic jsonResult = JObject.Parse(result);
    return jsonResult.Results.output1.value.Values[0][0];
}
```

FIGURE 13, COMMUNICATE TO AZURE MACHINE LEARNING FROM LOGIC CODE

Step3: Streaming Analyzed Data by Azure Stream Analytics

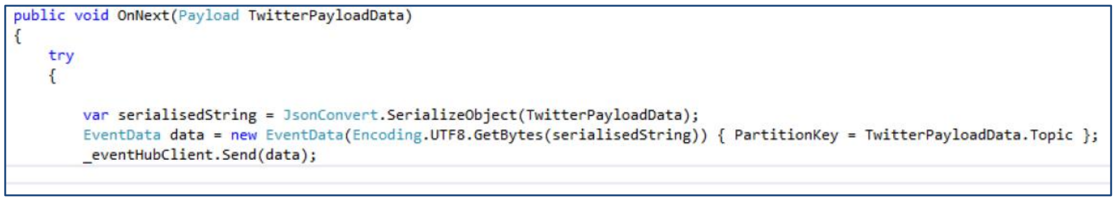
In this step, we should send our analyzed data to Azure Stream Analytics. For this Achievement we need to create the Service Bus as our data gate and then receive them from the Azure Stream Analytics.



NAME	STATUS
credexotwitteranalytics	✓ Active

FIGURE 14, CREATED AZURE SERVICE BUS

Now, we can send the data to Azure Service Bus from the logic code.



```
public void OnNext(Payload TwitterPayloadData)
{
    try
    {

        var serialisedString = JsonConvert.SerializeObject(TwitterPayloadData);
        EventData data = new EventData(Encoding.UTF8.GetBytes(serialisedString)) { PartitionKey = TwitterPayloadData.Topic };
        _eventHubClient.Send(data);
```

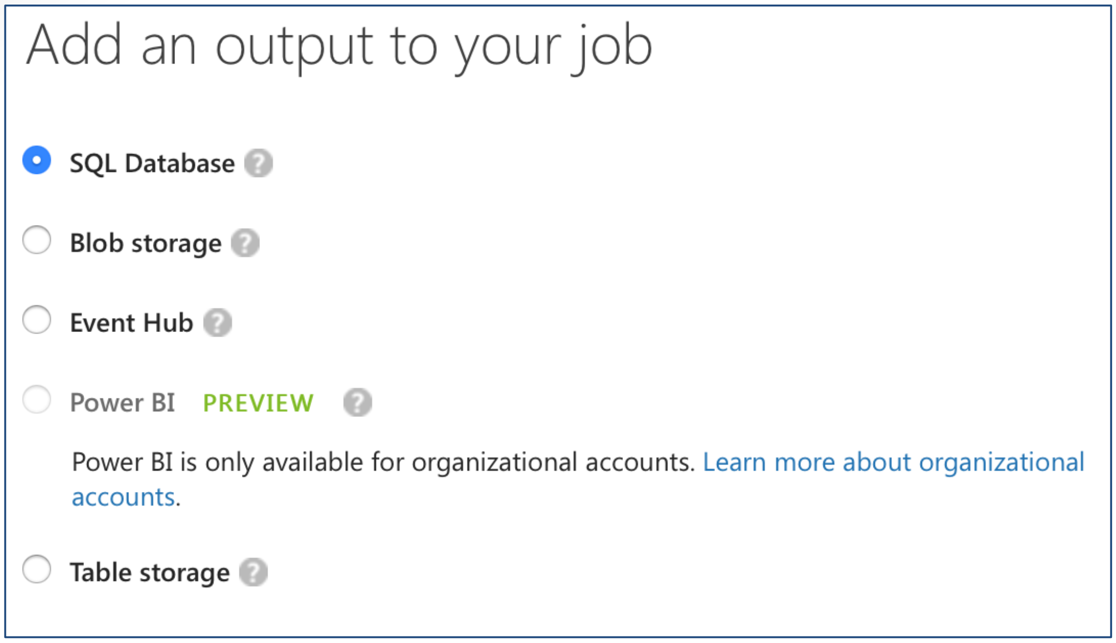
FIGURE 15, SENDING DATA TO AZURE SERVICE BUS IN LOGIC CODE

Stream Analytics has a 3 main Part:
Input: the data we want to stream, in this scenario the input is our twitter data.
Query: Choose data from the stream to save in the storage.

```
SELECT System.Timestamp as Time, Topic, COUNT(*), AVG(SentimentScore), MIN(SentimentScore),
Max(SentimentScore), STDEV(SentimentScore)
FROM TwitterStream TIMESTAMP BY CreatedAt
GROUP BY TUMBLINGWINDOW(s, 5), Topic
```

FIGURE 16, AZURE STREAM ANALYTICS SAMPLE QUERY

Output: Choose one of the options as a storage to save the data:



Add an output to your job

[✓ SQL Database]

Blob storage

Event Hub

Power BI PREVIEW

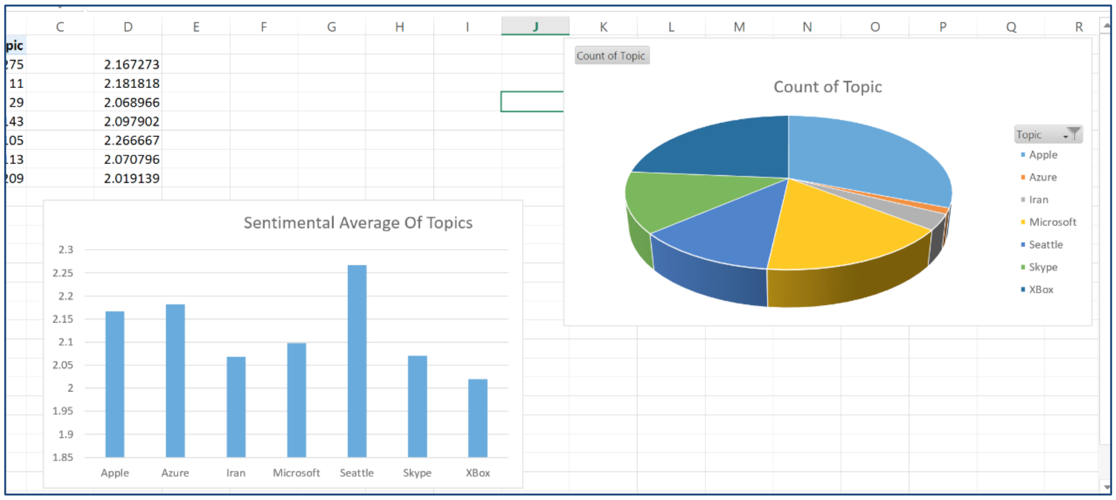
Power BI is only available for organizational accounts. Learn more about organizational accounts.

Table storage

FIGURE 17, AZURE STREAM ANALYTICS OUTPUT OPTIONS

Step 4: Visualize Data by tools

Now, we can visualize our stream data. There are many tools we can use Like Excel, Power BI or from the logic code and customized controls, etc.



pic		D
75		2.167273
11		2.181818
29		2.068966
43		2.097902
05		2.266667
13		2.070796
09		2.019139

FIGURE 18, VISUALIZE SAMPLE DATA BY POWER QUERY IN EXCEL