

Stream Processing Lab







Table of Contents

Stream Processing Lab	1
Lab Perquisites	3
Lab Overview: Simulating an Automated Toll Zone	5
Incoming data.....	6
Commercial Vehicle Registration Data.....	7
Creating a Storage Output	8
Provisioning an Azure Storage Account	8
Provisioning a Storage Container.....	9
Provisioning an Event Ingestor Input	11
Creating a Service Bus Namespace	12
Creating Event Hubs.....	13
Simulating Toll Data.....	15
Launching the App.....	15
Obtaining the Connection String	16
Specifying Event Hubs to Push Data Into.....	18
Provisioning Stream Processors	20
Enable Azure Stream Analytics feature	20

Lab Perquisites

- 1) Windows Operating System (or .NET emulator)
 - a) This lab will be simulating data into an event hub via a C# application, which requires .NET to execute
- 2) An Azure 30-day free trial account, or an Azure account with administrative access rights
 - a) A free trial account can be found here:
<http://azure.microsoft.com/en-us/pricing/free-trial/>
- 3) Text Editor (also called IDE; choose **ONE** of the following recommendations below)
 - a) Notepad++: <http://notepad-plus-plus.org/download/v6.7.5.html>
 - b) Sublime Text 2: <http://www.sublimetext.com/2>

File Descriptions

	SampleData
	SimulateTollsApp
	SimulateTollsSourceCode
	Analytics on Streaming Data with Azure Stream...
	Data Ingestion with Event Hub.pdf
	Tolls App Lab.pdf

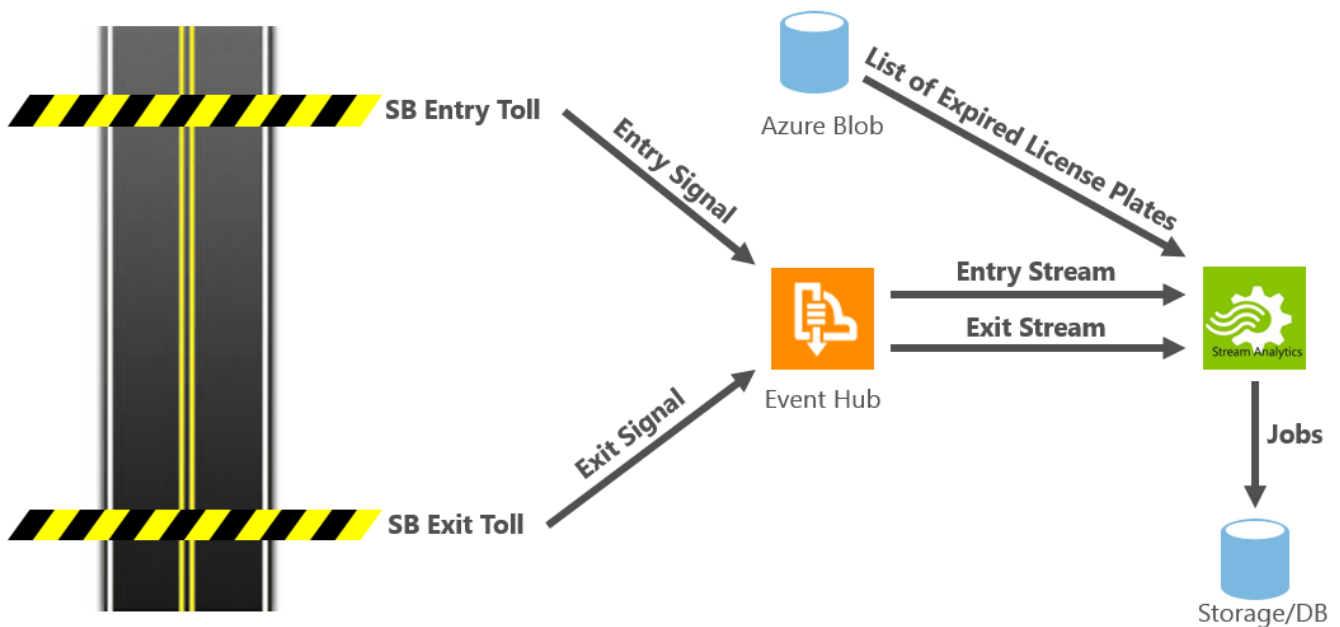
- 1) <folder>SampleData: contains sample files to test stream query logic against.
- 2) <folder>SimulateTollsApp: a .NET app that generates toll data (cars going through a toll zone).
- 3) <folder>SimulateTollSourceCode: source code for the SimulateTollsApp.
- 4) <pdf>Analytics on Streaming Data with Azure Stream Analytics: the PowerPoint presentation pdf on stream analytics with Azure Stream Analytics, and Azure stream query language.
- 5) <pdf>Data Ingestion with Event Hubs: the PowerPoint presentation pdf on event ingestion using Azure Event Hubs.

Lab Overview: Simulating an Automated Toll Zone

A tolling station is a common phenomenon – we encounter them in many expressways, bridges, and tunnels across the world. Each toll station has multiple toll booths, which may be manual – meaning that you stop to pay the toll to an attendant, or automated – where a sensor placed on top of the booth scans an RFID card affixed to the windshield of your vehicle as you pass the toll booth. It is easy to visualize the passage of vehicles through these toll stations as an event stream over which interesting operations can be performed.



This lab such a toll scenario. Data from these tolls will be generated by a .NET app and simulate cars going through a toll zone; a measurement when a car is going through an entry toll booth and another signal as a car goes through the exit toll booth and departs the toll zone. The measurements from both toll booths, called “signals”, will be pushed and consolidated to the cloud, to an Azure Service Bus. From there stream processors will be used to query, filter, and aggregate the data in real time.



Incoming data

We will work with two streams of data which are produced by sensors installed in the entrance and exit of the toll stations and a static look up dataset with vehicle registration data.

Entry Data Stream

Entry data stream contains information about cars entering toll stations.

Toll Id	EntryTime	License Plate	State	Make	Model	Vehicle Type	Vehicle Weight	Toll	Tag
1	2014-09-10 12:01:00.000	JNB 7001	NY	Honda	CRV	1	0	7	
1	2014-09-10 12:02:00.000	YXZ 1001	NY	Toyota	Camry	1	0	4	123456789
3	2014-09-10 12:02:00.000	ABC 1004	CT	Ford	Taurus	1	0	5	456789123
2	2014-09-10 12:03:00.000	XYZ 1003	CT	Toyota	Corolla	1	0	4	
1	2014-09-10 12:03:00.000	BNJ 1007	NY	Honda	CRV	1	0	5	789123456
2	2014-09-10 12:05:00.000	CDE 1007	NJ	Toyota	4x4	1	0	6	321987654
	...								

Here is a short description of the columns:

TollID	Toll booth ID uniquely identifying a toll booth
EntryTime	The date and time of entry of the vehicle to Toll Booth in UTC
LicensePlate	License Plate number of the vehicle
State	Is a State in United States
Make	The manufacturer of the automobile
Model	Model number of the automobile
VehicleType	1 for Passenger and 2 for Commercial vehicles
WeightType	Vehicle weight in tons; 0 for passenger vehicles
Toll	The toll value in USD
Tag	e-Tag on the automobile that automates payment, left blank where the payment was done manually

Exit Data Stream

Exit data stream contains information about cars leaving the toll station.

TollId	ExitTime	LicensePlate
1	2014-09-10T12:03:00.0000000Z	JNB 7001
1	2014-09-10T12:03:00.0000000Z	YXZ 1001
3	2014-09-10T12:04:00.0000000Z	ABC 1004
2	2014-09-10T12:07:00.0000000Z	XYZ 1003
1	2014-09-10T12:08:00.0000000Z	BNJ 1007
2	2014-09-10T12:07:00.0000000Z	CDE 1007
...		

Column descriptions:

TollId	Toll booth ID uniquely identifying a toll booth
ExitTime	The date and time of exit of the vehicle from Toll Booth in UTC
LicensePlate	License Plate number of the vehicle

Commercial Vehicle Registration Data

We will use a static snapshot of commercial vehicle registration database.

LicensePlate	RegistrationId	Expired
SVT 6023	285429838	1
XLZ 3463	362715656	0
BAC 1005	876133137	1
RIV 8632	992711956	0
SNY 7188	592133890	0
ELH 9896	678427724	1
...		

Column descriptions:

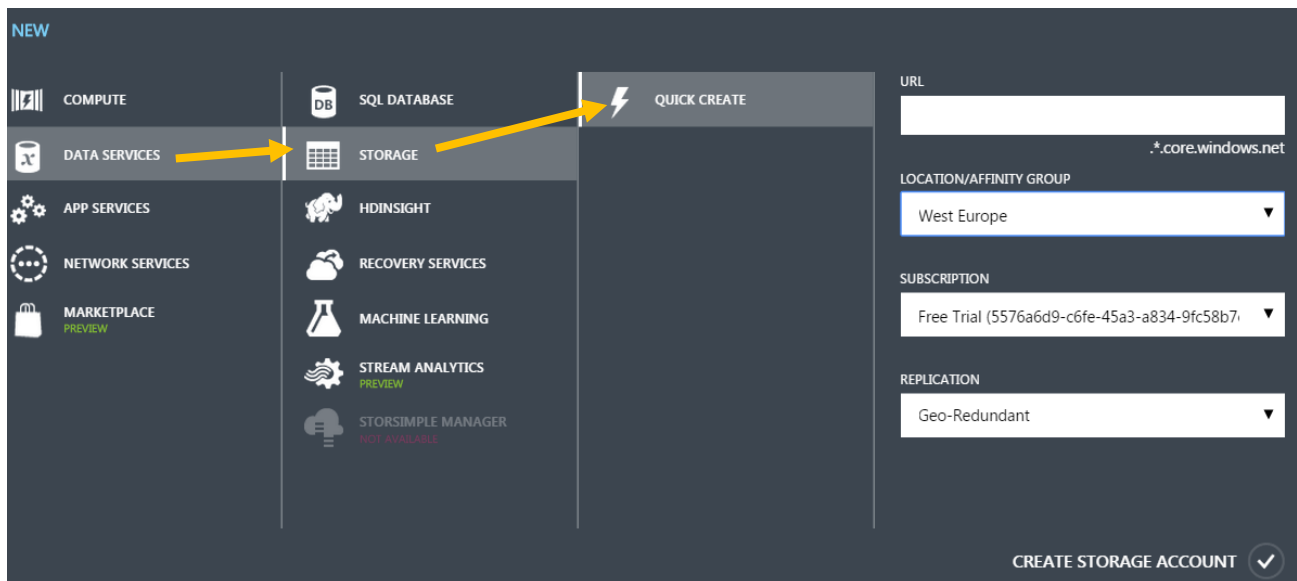
LicensePlate	License Plate number of the vehicle
RegistrationId	RegistrationId
Expired	0 if vehicle registration is active, 1 if registration is expired

Creating a Storage Output

A stream processor takes in stream(s) and outputs a processed stream. The output can go to a variety of outputs such as: Blob storage, Event Hub, Power BI dashboard, SQL Database or Table storage. For the purpose of this lab, a blob storage will be used as the stream's output.

Provisioning an Azure Storage Account

- 1) Once you are logged into your Azure portal (<https://manage.windowsazure.com/>), click **New>Data Services>Storage>Quick Create**



NEW

COMPUTE

DATA SERVICES

SQL DATABASE

STORAGE

QUICK CREATE

HDINSIGHT

RECOVERY SERVICES

MACHINE LEARNING

STREAM ANALYTICS

STORSIMPLE MANAGER

URL

LOCATION/AFFINITY GROUP

West Europe

SUBSCRIPTION

Free Trial (5576a6d9-c6fe-45a3-a834-9fc58b7)

REPLICATION

Geo-Redundant

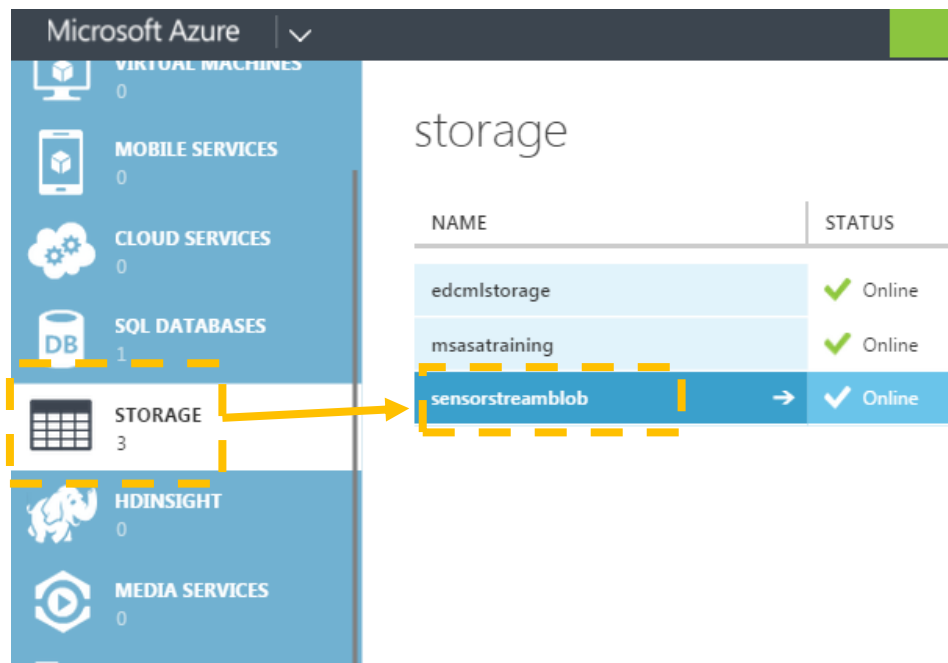
CREATE STORAGE ACCOUNT

- 2) URL: The URL will be the name of your storage and serve as a pseudo primary key for your storage name. Assign a unique name to it. It's called a URL because the storage account can be referenced directly via HTTP, ex: <https://mystorageaccount.blob.core.windows.net/>
- 3) Location: For the purpose of this lab, select **West Europe**.
- 4) Replication: Geo-Redundant or Locally Redundant will suffice.

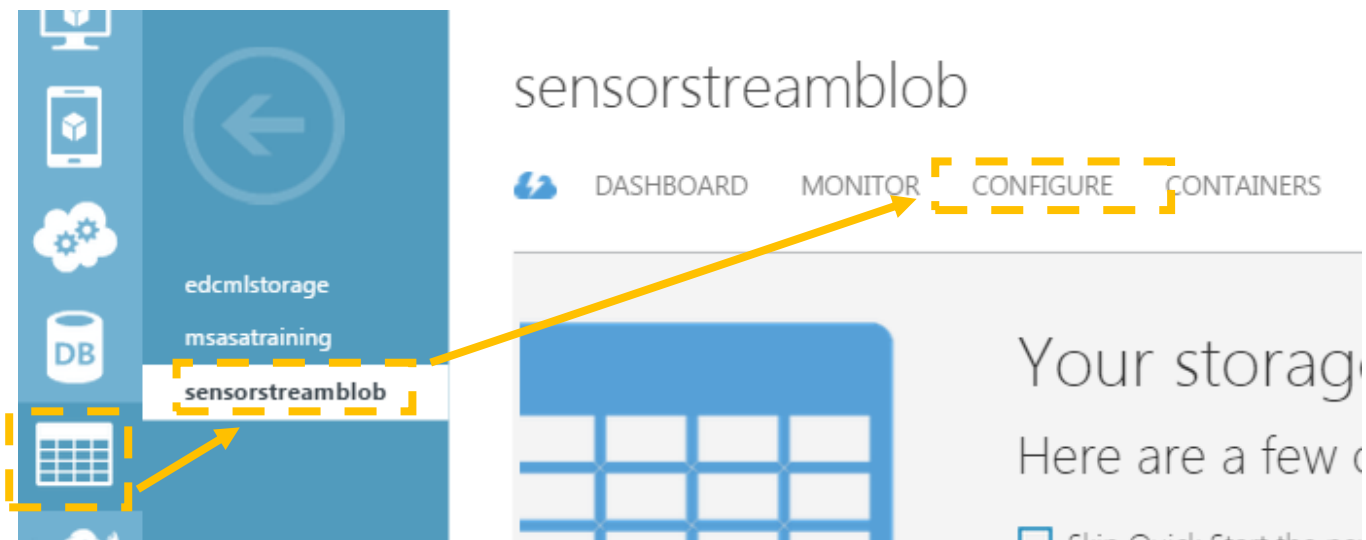
Provisioning a Storage Container

Files within an Azure Storage account are held within containers. Think of a container like a folder within a normal desktop environment. Create a container within the newly created Azure Storage account.

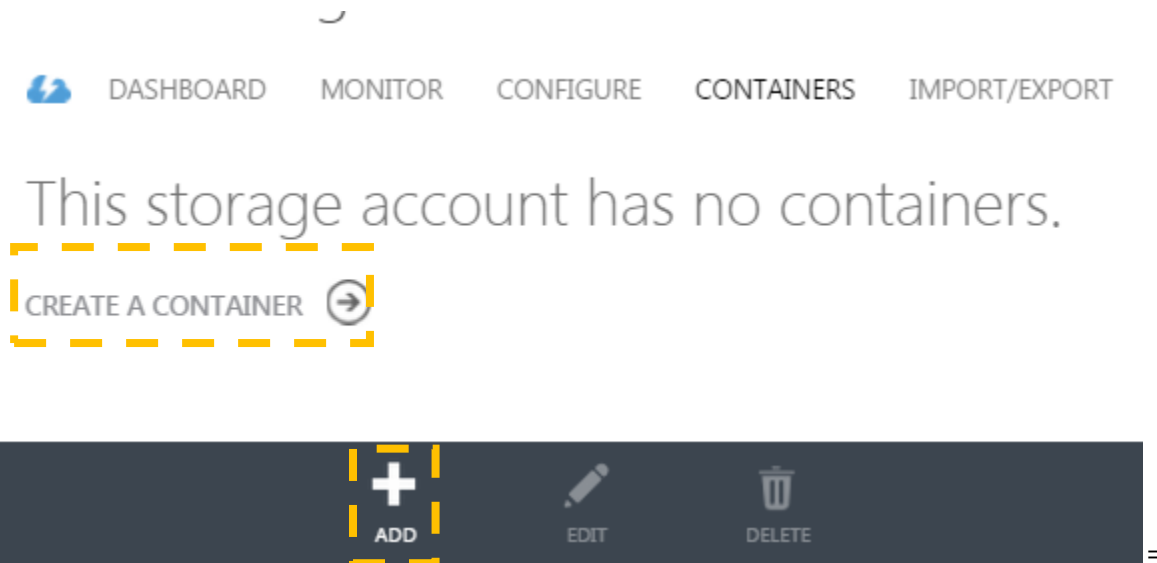
- 1) Within the Azure Management Portal (<https://manage.windowsazure.com/>)
 - a) **Storage > YourStorage**
 - b) Within the sample image below, the storage account is called "sensorstreamblob."



- c) Once inside the storage account's management dashboard, select **containers**.



d) Select either **Create a Container** or **Add**



e) Name the Container (in all lower case)

f) Access: any access right

New container

NAME

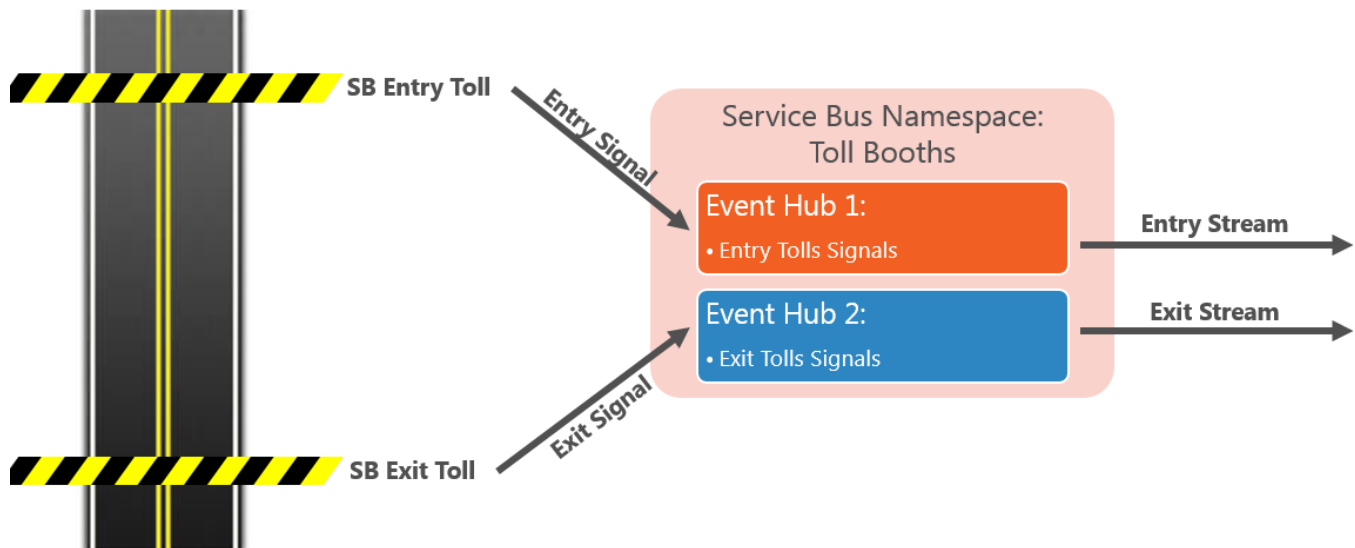
ACCESS ?



Provisioning an Event Ingestor Input

Signals from both the entry toll booths and the exit toll booths will be sent into the cloud. Specifically into an Azure Service Bus Namespace, an event ingestor which will consolidate the streaming data into a central location for replication, stream management, and short-term retention. This Service Bus Namespace will contain two Event Hubs, one to facilitate streams related to the entry booths and one related to exit booth streams. Think of the Service Bus Namespace as a server which will house all the Event Hubs.

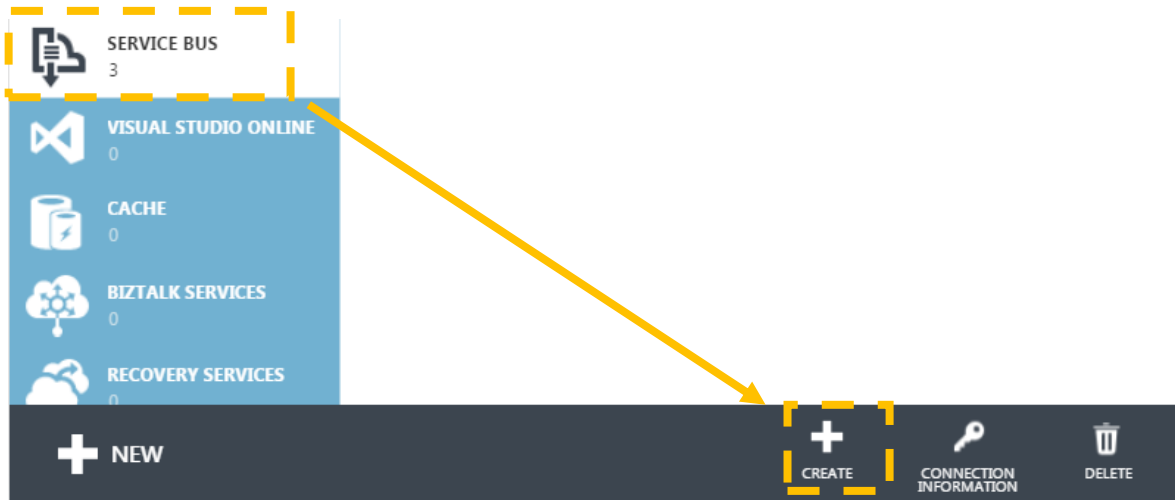
Inside the Event Hub



Creating a Service Bus Namespace

1) Within the Azure Management Portal (<https://manage.windowsazure.com/>)

a) **Service Bus > Create**




2) Name the Service Bus Namespace

- a) Namespace Name: globally unique URL name.
- b) Region: for the purpose of this lab, select **West Europe**.
- c) Type: Messaging
- d) Messaging Tier: Standard

CREATE A NAMESPACE

Add a new namespace

NAMESPACE NAME

brucewaynetollbooth 


.servicebus.windows.net

REGION


West Europe ▼

SUBSCRIPTION

Free Trial (5576a6d9-c6fe-45a3-a834-9fc58b7) ▼

TYPE 

MESSAGING NOTIFICATION HUB

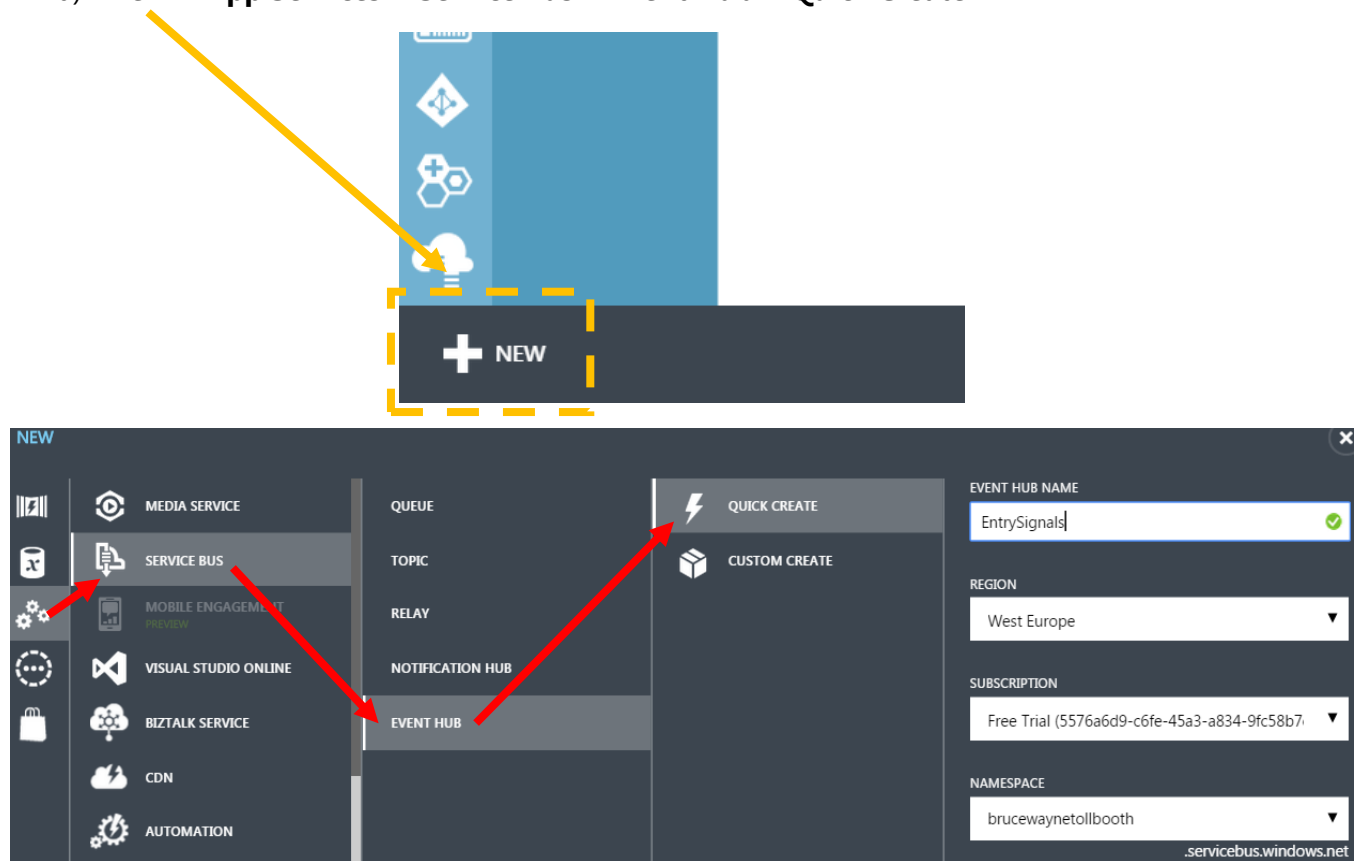
MESSAGING TIER 

BASIC STANDARD

Creating Event Hubs

1) Within the Azure Management Portal (<https://manage.windowsazure.com/>)

a) **New > App Services > Service Bus > Event Hub > Quick Create**



b) Create Event Hub #1, for entry signals.

- i) Event Hub Name: EntrySignals
- ii) Region: for the purpose of this lab, set **West Europe**.
- iii) Namespace: set the namespace that was created in the previous exercise.

- c) Create another event hub for exit signals.
- d) There should now be two event hubs within the same namespace, one for entry signals another for exit signals. Confirm their existing by selecting **Service Bus > My Service Bus > Event Hub**
 - i) The example below has a namespace service bus called "brucewaynetollbooth" with two event hubs inside called "entrysignals" and "exitsignals".

The screenshot shows the Azure portal interface for a Service Bus namespace named 'brucewaynetollbooth'. On the left, a sidebar contains navigation icons and a list of namespaces: 'TollTest', 'SensorTagHub-ns', and 'brucewaynetollb...'. A red arrow points from the 'brucewaynetollb...' namespace in the sidebar to the main content area. The main content area displays the 'brucewaynetollbooth' namespace with tabs for 'ALL', 'QUEUES', 'TOPICS', 'RELAYS', and 'EVENT HUBS'. The 'EVENT HUBS' tab is selected, showing a table with two event hubs: 'entrysignals' and 'exitsignals'. Both are in an 'Active' status. A red arrow points from the 'EVENT HUBS' tab to the table.

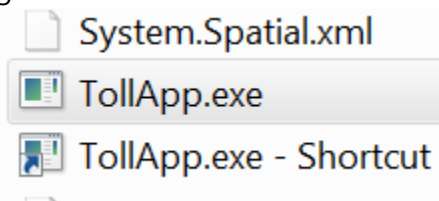
NAME	STATUS
entrysignals	✓ Active
exitsignals	✓ Active

Simulating Toll Data

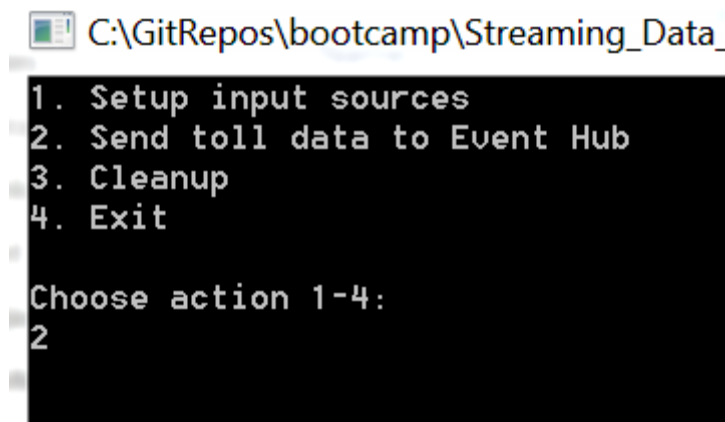
Now that the data pipelines have been set up, it is time to fill the pipeline with data. This lab will use a .NET app to simulate cars going through a toll booth and output it into the Event Hubs setup in the prior exercise.

Launching the App

- 1) Go to the bootcamp folder under:
- 2) `bootcamp\Streaming_Data_and_Real-time-Analytics\SimulateTollsApp`
- 3) Find a file called **TollApp.exe**
- 4) Right-click and run as administrator to launch the app.



- 5) Type '2' and hit enter to initiate input parameters.



- 6) The app will now ask for a "Service Bus Connection String to Namespace with RootManageSharedAccessKey". Essentially it is requesting the "god key" or the "master key" to access all the event hubs within the service bus namespace. In practice this is bad style as each user should only be given as much access as necessary to perform their task (separation of listen/send rights at the event hub level or consumer group level). However for the purpose of this lab, the "god key" will be used in order to minimize the amount of parameters that the user must enter (in this case there are 4 possible keys, two from each event hub). Continue to the next exercise to find the connection string.

Obtaining the Connection String

- 1) From the Azure management portal (<https://manage.windowsazure.com/>)
 - a) **Service Bus > Select Your Service Bus Namespace (light blue area) > Connection String**

The screenshot shows the Microsoft Azure management portal interface. On the left, a sidebar lists various services: MEDIA SERVICES, SERVICE BUS (highlighted in light blue), VISUAL STUDIO ONLINE, CACHE, BIZTALK SERVICES, and RECOVERY SERVICES. The main area displays the 'service bus' namespace selection screen. A table lists namespaces with columns for NAMESPACE NAME, STATUS, and TYPE. The namespace 'brucewaynetollbooth' is selected, and a red arrow points from the 'SERVICE BUS' sidebar item to this namespace. Another red arrow points from the 'brucewaynetollbooth' row to the 'CONNECTION INFORMATION' button in the bottom navigation bar. The bottom bar also includes 'NEW', 'CREATE', and 'DELETE' buttons.

NAMESPACE NAME	STATUS	TYPE
TollTest	✓ Active	Messaging
SensorTagHub-ns	✓ Active	Messaging
brucewaynetollbooth	✓ Active	Messaging

- 2) Copy the connection string of the RootManageSharedAccessKey by hitting the copy button the right of the connection string text box.

Access connection information

Use this connection information to manage namespace 'brucewaynetollbooth'. You can also use authorization policies configured here to connect to all entities in this namespace.

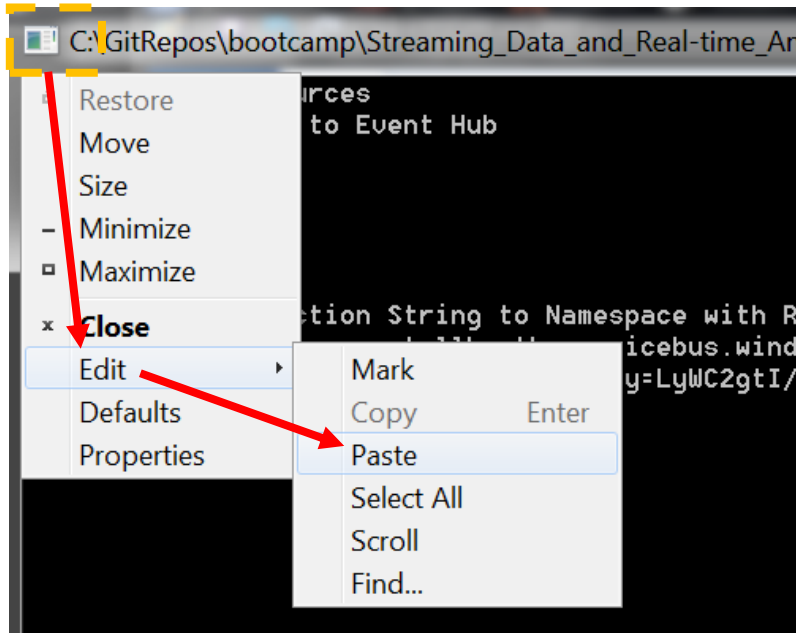
SAS ?

NAME	CONNECTION STRING
RootManageSharedAccessKey	Endpoint=sb://brucewaynetollbooth.servicebus.windows.net/

ACS

Looking for ACS connection information? Please see [here](#) for more information regarding using ACS with Service Bus.

- 3) Paste the connection string back into your app. Right-click the top left icon of the app to open up a menu. Go to Edit > Paste. Hit enter to be asked about the next parameter.



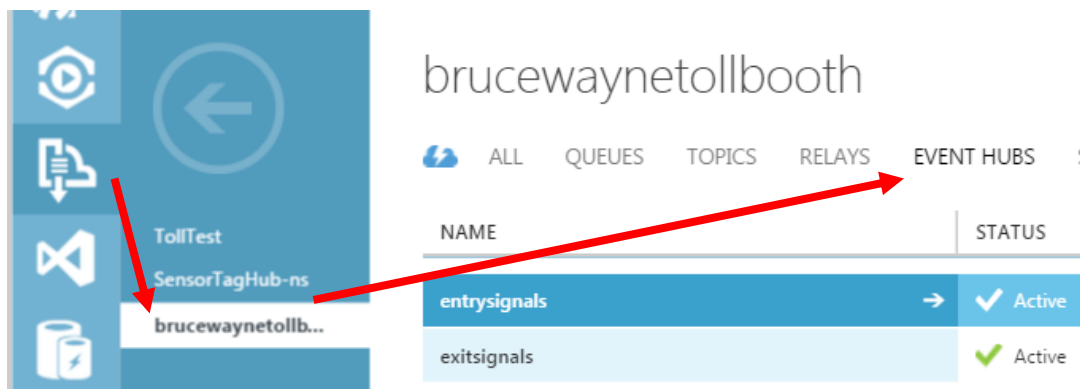
Specifying Event Hubs to Push Data Into

- 1) It will now ask for the name of the Event Hub in which to send the entry signals. Enter the name of the Event Hub that was created in the prior step.

```
Choose action 1-4:
2
Service Bus Connection String to Namespace with
Endpoint=sb://brucewaynetollbooth.servicebus.wi
otManageSharedAccessKey;SharedAccessKey=LyWC2gt
lak=
Name of the event hub to send entry signals.
```

a) **Service Bus > My Service Bus > Event Hub**

- i) The example below has a namespace service bus called "brucewaynetollbooth" with two event hubs inside called "entrysignals" and "exitsignals".



- 2) Enter in the name of the Entry Hub.
- 3) Enter in the name of the Exit Hub.
- 4) The app will now proceed to send streams of data into the event hubs.

```
otManageSharedAccessKey;SharedAccessKey=LyWC2gtI/Rui6M1R1tgN13BKVQRICaWqMeqo
lak=
Name of the event hub to send entry signals.
entrysignals
Name of the event hub to send exit signals.
exitsignals
Sending event hub data... Press Ctrl+c to stop.
Entry Time: 4/15/2015 1:37:40 AM | Entry Booth: 3 | LicensePlate: ALA 6638
Entry Time: 4/15/2015 1:37:40 AM | Entry Booth: 0 | LicensePlate: TOU 4081
Entry Time: 4/15/2015 1:37:40 AM | Entry Booth: 1 | LicensePlate: MQE 4119
Entry Time: 4/15/2015 1:37:40 AM | Entry Booth: 0 | LicensePlate: ERX 7056
Entry Time: 4/15/2015 1:37:40 AM | Entry Booth: 4 | LicensePlate: PGX 1137
```


Provisioning Stream Processors

Stream Processors can now be created that will query, filter, and aggregate the streams in real time as it goes through the event hub. This lab will utilize the Azure Stream Analytics stream processor.

Enable Azure Stream Analytics feature

Azure Stream Analytics is in the Preview program. Please open

<https://account.windowsazure.com/PreviewFeatures?fid=streamanalytics> and click on the OK button in the lower right corner of the window to enable Azure Stream Analytics in the Azure Portal.

Please note that you need to be Administrator of the subscription to be able to enable Preview features in the Azure Portal.

ADD PREVIEW FEATURE



STREAM ANALYTICS
CHARGES MAY BE INCLUDED

SUBSCRIPTIONS

Visual Studio Ultimate with MSDN



I understand this feature is in Preview and subject to reduced or different service terms, as set forth in the [service agreement](#), and I agree to the [preview supplemental terms](#).



Create a Stream Analytics Job

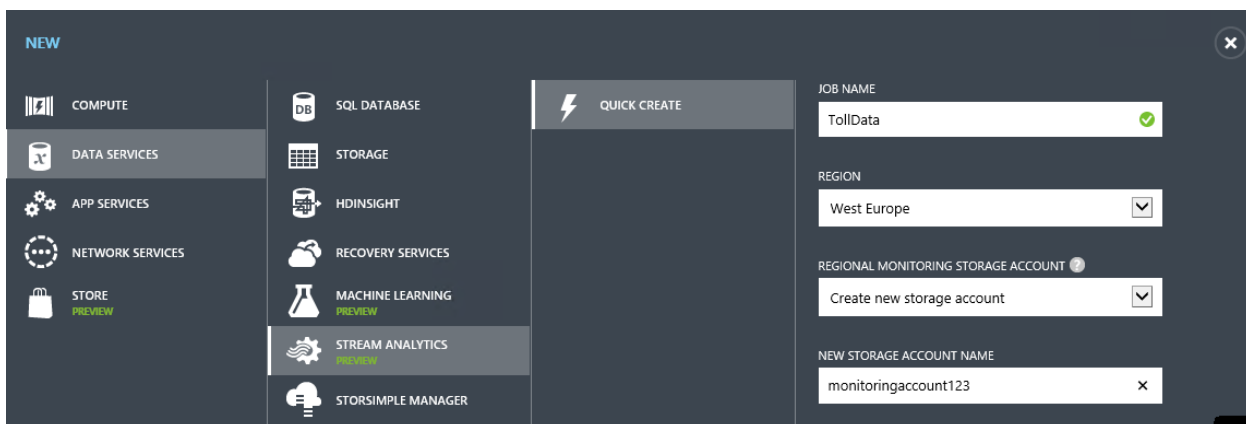
In Azure portal open Stream Analytics and click "New" in the bottom left hand corner of the page to create a new analytics job.



Click "Quick Create". Select either "West Europe" or "Central US" as the region.

For "Regional Monitoring Storage Account" setting, select "Create new storage account" and give it any unique name. Azure Stream Analytics will use this account to store monitoring information for all your future jobs.

Click "Create Stream Analytics Job" at the bottom of the page.



Define Input Sources

Click on the created analytics job in the portal.

stream analytics PREVIEW

NAME	STATUS	CREATED	SUBSCRIPTION	REGION	
TollData	✓ NOT STARTED	10/17/2014 7:22:30 PM	Prototypes	West Europe	

Open "Inputs" tab to define the source data.

tolldata PREVIEW

[DASHBOARD](#) [MONITOR](#) **INPUTS** [QUERY](#) [OUTPUT](#) [SCALE](#) [CONFIGURE](#)



Your Stream Analytics job has been created!
Here are a few options to help you get started.

☐ Skip Quick Start the next time I visit

1

Add inputs

[Create one or more new input sources](#) (if you don't have one already).

2

Develop a query

Download [code samples](#), and [tools](#), or explore our [documentation](#) and [tutorials](#) if you want more information.

Click "Add an Input"

You have no inputs. Add one to get started!

ADD AN INPUT

Select "Data Stream" on the first page

ADD AN INPUT

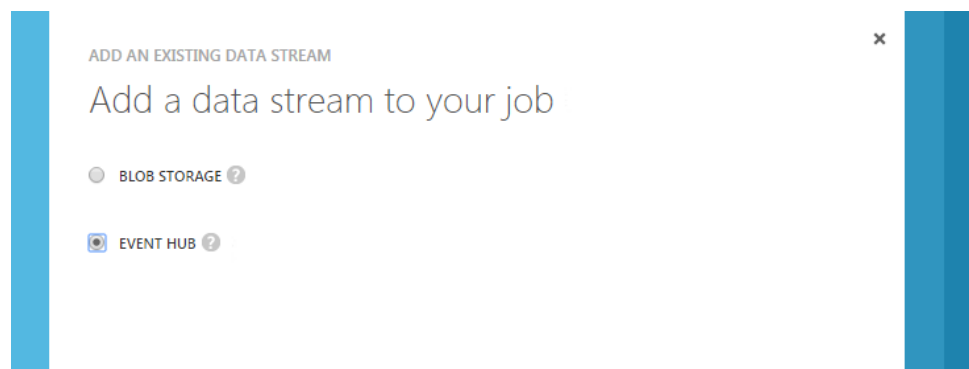
Add an input to your job

Each job requires at least one data stream input. Adding reference data input is optional.

DATA STREAM

A continuous sequence of data or events to be consumed and transformed by a Stream Analytics job.

Select "Event Hub" on the second page of the wizard.



Input Alias: "EntryStream"

Subscription: Use Event Hub from Current Subscription

Service Bus Namespace: Your Service Bus Namespace

Event Hub Name: Your Entry Signal Event Hub

Policy Name: RootManageSharedAccessKey

Consumer Group: Leave Blank

Event Hub settings

INPUT ALIAS	EntryStream
SUBSCRIPTION	Use Event Hub from Current Subscription ▼
SERVICE BUS NAMESPACE	brucewaynetollbooth ▼
EVENT HUB NAME	entrysignals ▼
EVENT HUB POLICY NAME	RootManageSharedAccessKey ▼
EVENT HUB CONSUMER GROUP	

Move to the next page. Leave default values (CSV, comma delimited, UTF8 encoding)

ADD AN EXISTING SERVICE BUS EVENT HUB

Serialization settings

EVENT SERIALIZATION FORMAT ?

CSV

DELIMITER ?

comma (,)

ENCODING ?

UTF8

Click OK at the bottom of the dialog to finish the wizard.

tolldata PREVIEW

[DASHBOARD](#) [MONITOR](#) [INPUTS](#) [QUERY](#) [OUTPUT](#) [SCALE](#) [CONFIGURE](#)

NAME	SOURCE TYPE	TYPE	CONNECTION STATUS	
EntryStream	Data stream	Event Hub	✓ Connected	

You will need to follow the same sequence of steps to create the second Event Hub input for the stream with Exit events. Make sure on the 3rd page you enter values as on the screenshot below.

tolldata PREVIEW

[DASHBOARD](#) [MONITOR](#) [INPUTS](#) [QUERY](#) [OUTPUTS](#) [SCALE](#) [CONFIGURE](#)

NAME	SOURCE TYPE	TYPE
EntryStream	Data stream	Event Hub

→

START **+** ADD INPUT TEST CONNECTION DELETE SAMPLE DATA

INPUT ALIAS

ExitStream ✓

SUBSCRIPTION

Use Event Hub from Current Subscription ▼

SERVICE BUS NAMESPACE

brucewaynetollbooth ▼

EVENT HUB NAME ?

exitsignals ▼

EVENT HUB POLICY NAME ?



RootManageSharedAccessKey ▼

EVENT HUB CONSUMER GROUP ?

You now have two input streams defined:

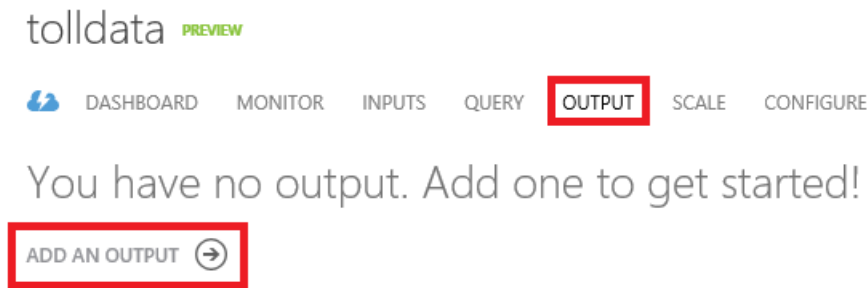
tolldata PREVIEW

 DASHBOARD MONITOR **INPUTS** QUERY OUTPUT SCALE CONFIGURE

NAME	SOURCE TYPE	TYPE	CONNECTION STATUS	
EntryStream	Data stream	Event Hub	 Unknown	
ExitStream	Data stream	Event Hub	✓ Connected	

Define Output

Go to "Output" tab and click "Add an output".



Select Blob Storage.

The screenshot shows the 'Add an output' form with 'Blob storage' selected. The form fields are: 'OUTPUT ALIAS' (blob), 'SUBSCRIPTION' (Use Storage Account from Current Subscription), 'STORAGE ACCOUNT' (brucewaynetollstorage), 'STORAGE ACCOUNT KEY' (masked with dots), 'CONTAINER' (streamoutput), and 'FILENAME PREFIX' (MyPrefix). The 'Blob storage' option is selected with a radio button and a question mark icon.

Output Alias: this is where the user gets to define what their output will be called for the purpose of the query.

Subscription: "Use Storage Account from Current Subscription"

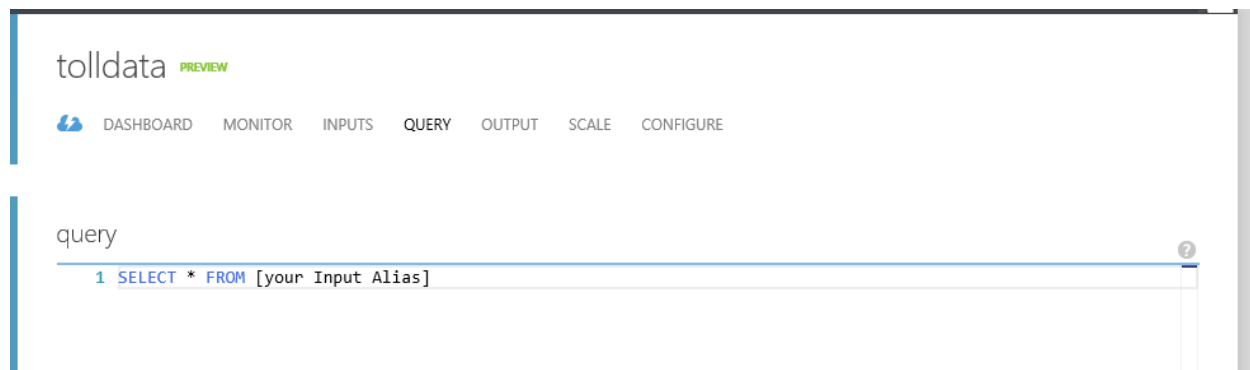
Storage Account: The storage account that was provisioned in the first exercise.

Container: The container within that storage account that was provisioned in the previous exercise.

FilenamePrefix: The output from this stream will have a unique identifier set by the user. As multiple stream processors can output to the same blob, it can get confusing as to which file within the blob is related to which stream processor. Think of this like customerID or employID but for the job output file.

Azure Stream Analytics Query

The Query tab contains a SQL query that performs the transformation over the incoming data.



Through this lab we will attempt to answer several business questions related to Toll data and construct Stream Analytics queries that can be used in Azure Stream Analytics to provide a relevant answer.

Before we start our first Azure Stream Analytics job, let's explore few scenarios and query syntax.

Let's say, we need to count the number of vehicles that enter a toll booth. Since this is a continuous stream of events, it is essential we define a "period of time". So we need to modify our question to be "Number of vehicles entering a toll booth every 3 minutes". This is commonly referred to as the Tumbling Count.

Let's look at the Azure Stream Analytics query answering this question:

```
SELECT TollId, System.Timestamp AS WindowEnd, COUNT(*) AS Count
FROM EntryStream TIMESTAMP BY EntryTime
GROUP BY TUMBLINGWINDOW(minute,3), TollId
```

As

you can see, Azure Stream Analytics is using a SQL-like query language with a few additional extensions to enable specifying time related aspects of the query.

The next sections will describe Timestamp and TumblingWindow constructs used in the query in detail.

Application time - TimeStamp

In any temporal system like Azure Stream Analytics, it's essential to understand the progress of time. Every event that flows through the system comes with a timestamp. In other words, every event in the system depicts a point in time.

This timestamp can be defined by the Application (e.g. specific field in the event). The user can specify it in the query using the *"TIMESTAMP BY"* clause (e.g. EntryTime column in our example).

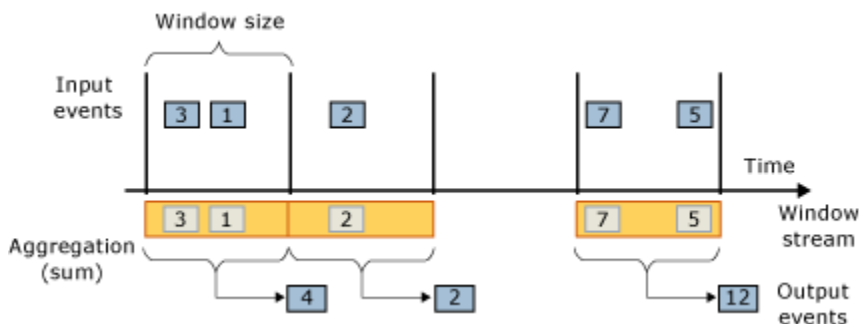
Alternatively, the system can assign the timestamp based on the event arrival time. Please note that in this case the value of the timestamp is potentially subject to certain factors such as network latencies. This can negatively affect accuracy of the further computations and analysis.

You can access the *timestamp* of any event using the *System.Timestamp* property which needs to be used with an alias. For any aggregate event like Sum(), Avg(), etc., produced as output from a window operation, it will also have a *timestamp* property like any other input event entering into the system. This timestamp of the output event aligns to the end time of the window.

Time windows – Tumbling Window

In applications that process real-time events, a common requirement is to perform some set-based computation (aggregation) or other operations over subsets of events that fall within some period of time. In Azure Stream Analytics, these subsets of events are defined through windows.

A window contains event data along a timeline and enables you to perform various operations against the events within that window. For example, you may want to sum the values of payload fields in a given window as shown in the following illustration.



Azure Stream Analytics supports several functions to define window aggregates (Tumbling, Hopping, Sliding windows). Please refer to the Query Language Reference Guide in MSDN for more details.

Our sample query uses TumblingWindow function to specify the size of the window

GROUP BY TUMBLINGWINDOW(minute,3)

Testing Queries

Within the folder: **bootcamp\Streaming_Data_and_Real-time_Analytics\SampleData** are the following 3 files which can be used to test the stream query logic before the job is fired.

1. Entry.json
2. Exit.json
3. Registration.json

Question 1 - Number of vehicles entering a toll booth

Open the Azure Management portal and navigate to your created Azure Stream Analytic job. Open the Query tab and copy paste Query from the previous section.

 DASHBOARD MONITOR INPUTS QUERY OUTPUT SCALE CONFIGURE

query

```
1 SELECT TollId, System.Timestamp AS WindowEnd, COUNT(*)AS Count
2 FROM EntryStream TIMESTAMP BY EntryTime
3 GROUP BY TUMBLINGWINDOW(minute,3), TollId
4
```

Test

Rerun

To validate this query against sample data, click the Test button. In the dialog that opens, navigate to Entry.json, a file with sample data from the EntryTime event stream.

TEST DATA ?

Input

Sample File

entrystream



Entry.json

Validate that the output of the query is as expected:

output

15 rows were generated by processing:

- 23 events from entrystream

TOLLID	WINDOWEND	COUNT
1	2014-09-10T12:03:00.000Z	3
2	2014-09-10T12:03:00.000Z	1
3	2014-09-10T12:03:00.000Z	1
2	2014-09-10T12:06:00.000Z	2
1	2014-09-10T12:09:00.000Z	1
2	2014-09-10T12:09:00.000Z	3
3	2014-09-10T12:09:00.000Z	2
1	2014-09-10T12:12:00.000Z	2
3	2014-09-10T12:12:00.000Z	2



SAVE



DISCARD

Question 2 - Report Total time for each car to pass through the toll booth

We want to find average time required for the car to pass the toll to assess efficiency and customer experience.

For this we need to join the stream containing EntryTime with the stream containing ExitTime. We will join the streams on TollId and LicencePlate columns. JOIN operator requires specifying a temporal wiggle room describing acceptable time difference between the joined events. We will use DATEDIFF function to specify that events should be no more than 15 minutes from each other. We will also apply DATEDIFF function to Exit and Entry times to compute actual time a car spends in the toll. Note the difference of the use of DATEDIFF when used in a SELECT statement compared to a JOIN condition.

```
SELECT EntryStream.TollId, EntryStream.EntryTime,
ExitStream.ExitTime, EntryStream.LicensePlate, DATEDIFF(minute,
EntryStream.EntryTime, ExitStream.ExitTime) AS
DurationInMinutes
FROM EntryStream TIMESTAMP BY EntryTime
JOIN ExitStream TIMESTAMP BY ExitTime
ON (EntryStream.TollId= ExitStream.TollId AND
EntryStream.LicensePlate = ExitStream.LicensePlate)
AND DATEDIFF(minute, EntryStream, ExitStream ) BETWEEN 0 AND 15
```

To

test this query, update the query on the Query tab of your job:

tolldata PREVIEW

 DASHBOARD  MONITOR  INPUTS  **QUERY**  OUTPUT  SCALE  CONFIGURE

query

```
1 SELECT EntryStream.TollId, EntryStream.EntryTime, ExitStream.ExitTime, EntryStream.LicensePlate, DATEDIFF(
2 FROM EntryStream TIMESTAMP BY EntryTime
3 JOIN ExitStream TIMESTAMP BY ExitTime
4 ON (ExitStream.TollId= ExitStream.TollId AND EntryStream.LicensePlate = ExitStream.LicensePlate)
5 AND DATEDIFF(minute, EntryStream, ExitStream ) BETWEEN 0 AND 15
6
7
```


Click test and specify sample input files for EntryTime and ExitTime.

TEST DATA ?

Input	Sample File
entrystream	<div><div></div>Entry.json</div>
exitstream	<div><div></div>Exit.json</div>

Click the checkbox to test the query and view output:

output

23 rows were generated by processing:

- 23 events from entrystream
- 23 events from exitstream

TOLLID	ENTRYTIME	EXITTIME	LICENSEPLATE	DURATIONINMINUTES
1	2014-09-10T12:01:00.000Z	2014-09-10T12:03:00.000Z	JNB 7001	2
3	2014-09-10T12:02:00.000Z	2014-09-10T12:04:00.000Z	ABC 1004	2
1	2014-09-10T12:02:00.000Z	2014-09-10T12:03:00.000Z	YXZ 1001	1
2	2014-09-10T12:03:00.000Z	2014-09-10T12:07:00.000Z	XYZ 1003	4
1	2014-09-10T12:03:00.000Z	2014-09-10T12:08:00.000Z	BNJ 1007	5
2	2014-09-10T12:05:00.000Z	2014-09-10T12:07:00.000Z	CDE 1007	2
2	2014-09-10T12:06:00.000Z	2014-09-10T12:09:00.000Z	BAC 1005	3
1	2014-09-10T12:07:00.000Z	2014-09-10T12:10:00.000Z	ZYX 1002	3

Question 3 – Report all commercial vehicles with expired registration

Azure Stream Analytics can use static snapshots of data to join with temporal data streams. To demonstrate this capability we will use the following sample question.



If a commercial vehicle is registered with the Toll Company, they can pass through the toll booth without being stopped for inspection. We will use Commercial Vehicle Registration lookup table to identify all commercial vehicles with expired registration.

```
SELECT EntryStream.EntryTime, EntryStream.LicensePlate,  
EntryStream.TollId, Registration.RegistrationId  
FROM EntryStream TIMESTAMP BY EntryTime  
JOIN Registration  
ON EntryStream.LicensePlate = Registration.LicensePlate  
WHERE Registration.Expired = '1'
```

Note that testing a query with Reference Data requires that an input source for the Reference Data is defined, which we have done in Step 5.

To test this query, paste the query into the Query tab, click Test, and specify the 2 input sources:

TEST DATA ?

Input	Sample File
entrystream	 Entry.json
registration	 Registration.json

View the output of the query:

output

2 rows were generated by processing:

- 23 events from entrystream
- 10000 events from registration

ENTRYTIME	LICENSEPLATE	TOLLID	REGISTRATIONID	
2014-09-10T12:06:00.000Z	BAC 1005	2	876133137	
2014-09-10T12:15:00.000Z	BAC 1005	2	876133137	

Start the Stream Analytics Job

Now as we have written our first Azure Stream Analytics query, it is time to finish the configuration and start the job. Save the query from Question 3, which will produce output that matches the schema of our output table TollDataRefJoin.

Navigate to the job Dashboard and click Start.

tolldata PREVIEW

DASHBOARD MONITOR INPUTS QUERY OUTPUT SCALE CONFIGURE

ERRORS INPUT EVENTS OUTPUT EVENTS REL

6:00 6:06 6:12 6:18 6:24 6:30 6:36 6:42 6:48

START DELETE

In the dialog that appears, change the Start Output time to Custom Time. Edit the Hour and set it to an hour back. This will ensure that we process all events from the Event Hub since the moment we started generating the events in the beginning of the lab. Now click the check mark to start the job.

START OUTPUT ?

JOB START TIME **CUSTOM TIME**

2014-10-17 04 : 29 : 12 PM

LOCAL TIME (UTC-08:00)

Starting the job can take a few minutes. You will be able to see the status on the top-level page for Stream Analytics.

The screenshot shows the Microsoft Azure portal interface. On the left is a navigation pane with icons and labels for SCHEDULER, API MANAGEMENT, MACHINE LEARNING, INTELLIGENT SYSTEMS, STREAM ANALYTICS, and NETWORKS. The main area is titled 'stream analytics' with a 'PREVIEW' label. Below the title is a table with columns: NAME, STATUS, CREATED, SUBSCRIPTION, and REGION. A single row is visible with the name 'TollData1', status 'STARTING', and creation time '10/8/2014 6:22:02 AM'. At the bottom of the main area are three buttons: START, STOP, and DELETE. The bottom of the portal has a dark bar with a '+ NEW' button on the left and a user profile icon on the right.

NAME	STATUS	CREATED	SUBSCRIPTION	REGION
TollData1	STARTING	10/8/2014 6:22:02 AM	auxtm020Current1998	West US