**BIG DATA ENGINEERING PROJECT**

**IOT based solution for**

**Toll Traffic**

**using Stream Analytics**

**Team**

**Rutika Ranjolkar**

**Nehal Bhanushali**

**Jyoti Sharma**

**Amitha Murali**

**Table of Content**

1. Introduction
2. Research
3. Application architecture
4. Phase I: IOT simulation and Data preparation
   1. Installations and Setup
   2. Running the scripts and capturing output
5. Phase II: Stream Analytics
   1. Installations and Setup
   2. Create stream analytics job
   3. Event hub listener
   4. Executing SAJ job
6. Phase III: Power BI output Analysis
7. Use Cases
   1. Theft Detection
   2. Air pollution Analysis in major cities of USA
8. Challenges and Learnings
9. Future Scope
10. References

**Introduction**

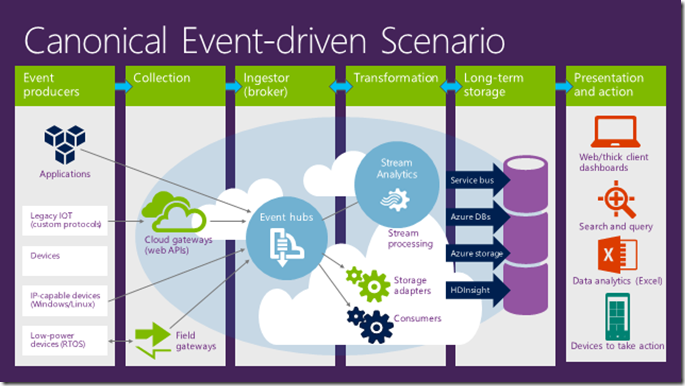
Vast amounts of data are flowing at high velocity over the wire today. Organizations that can process and act on this streaming data in real time can dramatically improve efficiencies and differentiate themselves in the market. Scenarios of real-time streaming analytics can be found across all industries: personalized, real-time stock-trading analysis and alerts offered by financial services companies; real-time fraud detection; data and identity protection services; reliable ingestion and analysis of data generated by sensors and actuators embedded in physical objects (Internet of Things, or IoT); web clickstream analytics; and customer relationship management (CRM) applications issuing alerts when customer experience within a time frame is degraded. Businesses are looking for the most flexible, reliable and cost-effective way to do such real-time event-stream data analysis to succeed in the highly competitive modern business world.

**Phase II: Stream Analytics**

Azure Stream Analytics is a fully managed, cost effective real-time event processing engine that helps to unlock deep insights from data. Stream Analytics makes it easy to set up real-time analytic computations on data streaming from devices, sensors, web sites, social media, applications, infrastructure systems, and more.

Developers can easily combine streams of data, such as click-streams, logs, and device-generated events, with historical records or reference data to derive business insights. As a fully managed, real-time stream computation service that's hosted in Microsoft Azure, Azure Stream Analytics provides built-in resiliency, low latency, and scalability to get you up and running in minutes.

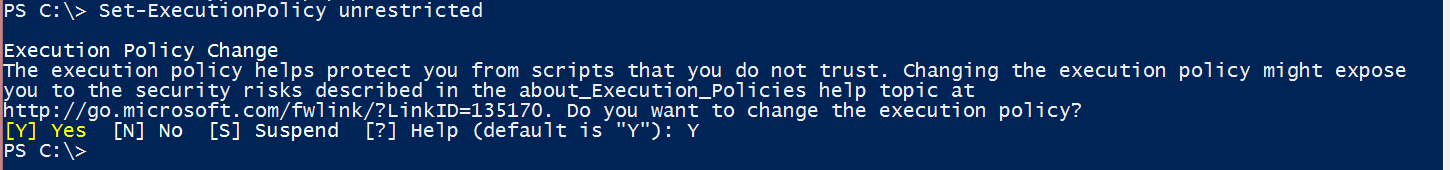
Here is an over view:



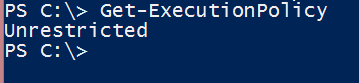
**Installations and Setup**

**Step1: Setting up environment**

To set up your Azure environment, open a **Microsoft Azure PowerShell** window as an administrator. Because Windows automatically blocks .ps1, .dll, and .exe files, you need to set the execution policy before you run the script. Make sure the Azure PowerShell window is running as an administrator. Run **Set-ExecutionPolicy unrestricted**. When prompted, type **Y**.



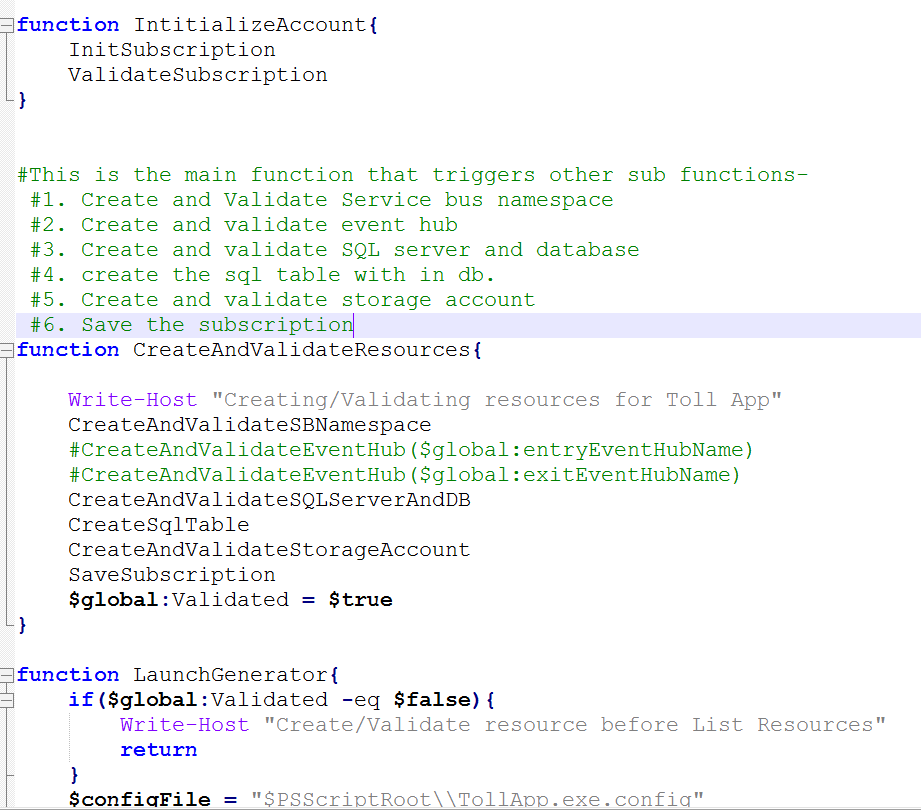
Run **Get-ExecutionPolicy** to make sure that the command worked.



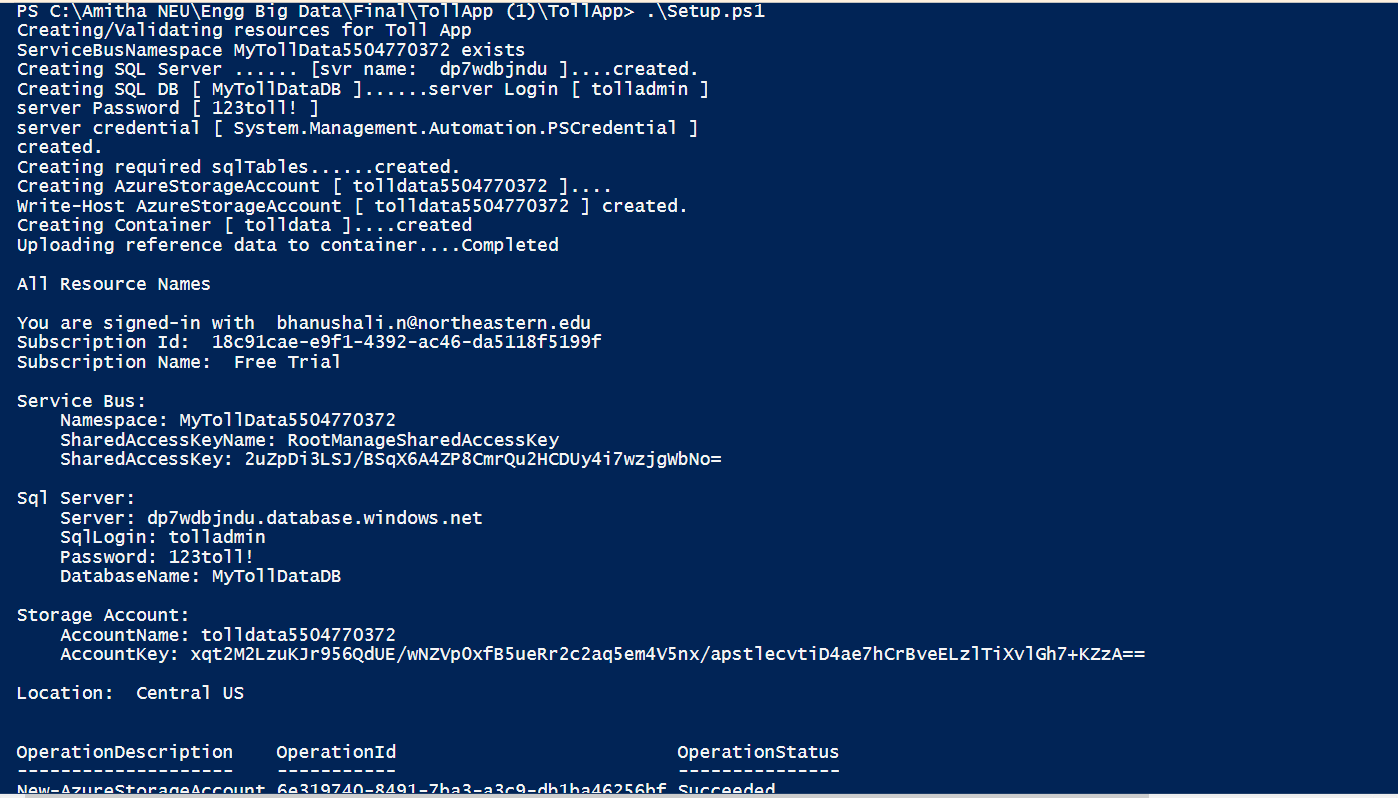
To set up your Azure account, create and configure all required resources, and start to generate events. The script randomly picks up a region to create your resources. To explicitly specify a region, you can pass the **-location** parameter as in the following example:

**.\Setup.ps1 -location “Central US”**

**else just type .\Setup.ps1**

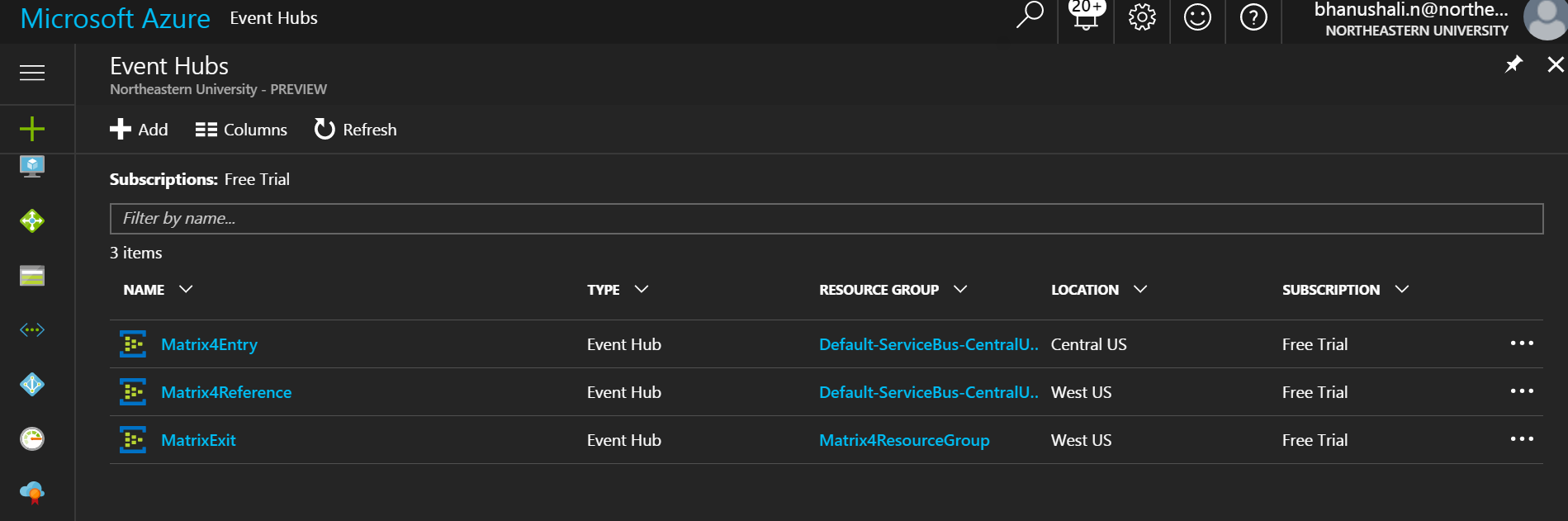


The script can take several minutes to run. After it finishes, the output should look like the following screenshot.



**Azure Event Hubs**

In the Azure portal, click More services on the bottom of the left management pane. Type Event hubs in the field provided and click Event hubs. This launches a new browser window to display the SERVICE BUS area in the classic portal. Here you can see the Event Hub created by the Setup.ps1 script.



Note: Matrix4Reference was later created manually.

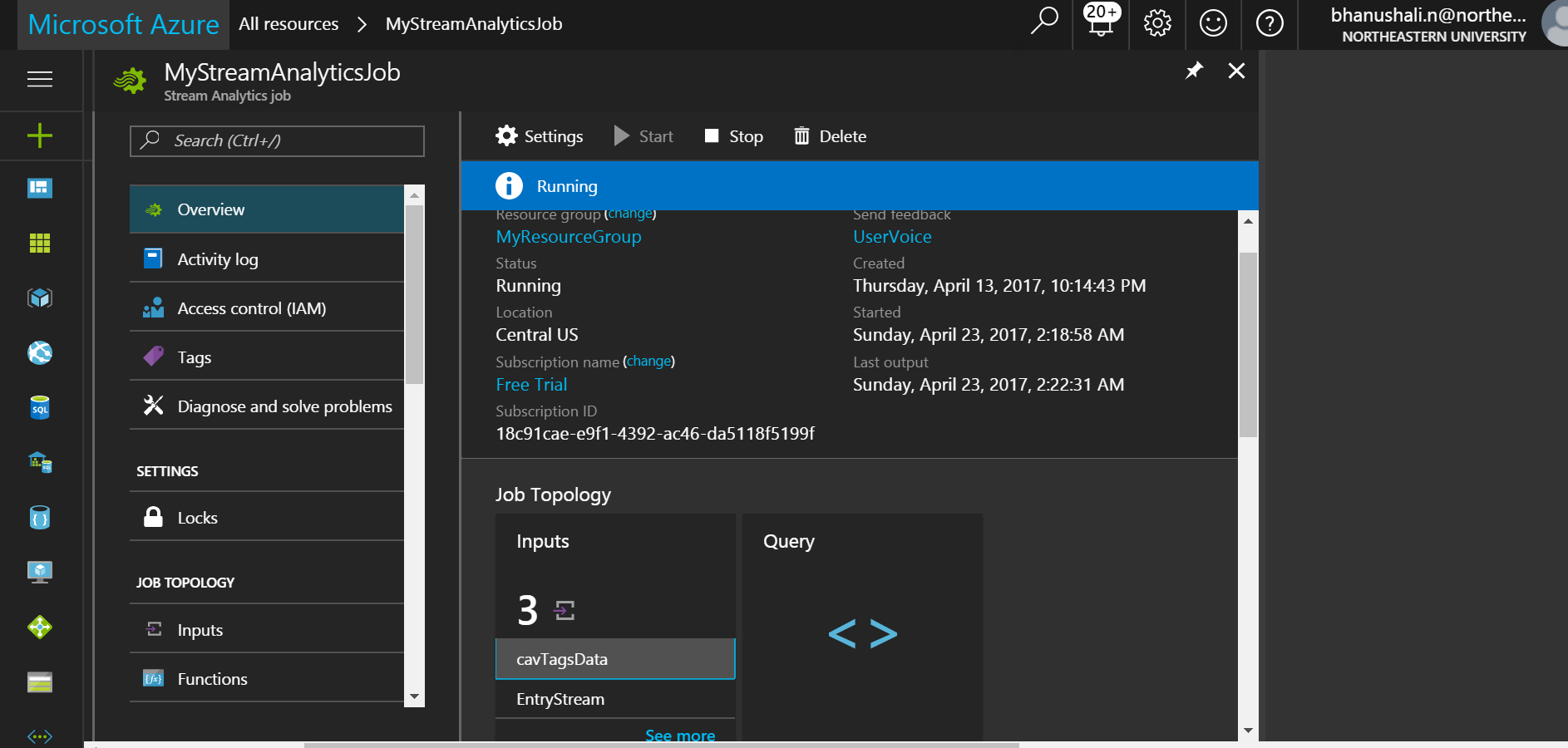
**Step2: Create a Stream Analytics job**

In the Azure portal, click the green plus sign in the top-left corner of the page to create a new Stream Analytics job. Select **Intelligence + Analytics** and then click **Stream Analytics job**.

Provide a job name, validate the subscription is correct and then create a new Resource group in the same region as the Event hub storage (default is South Central US for the script).Click Pin to dashboard and then CREATE at the bottom of the page.

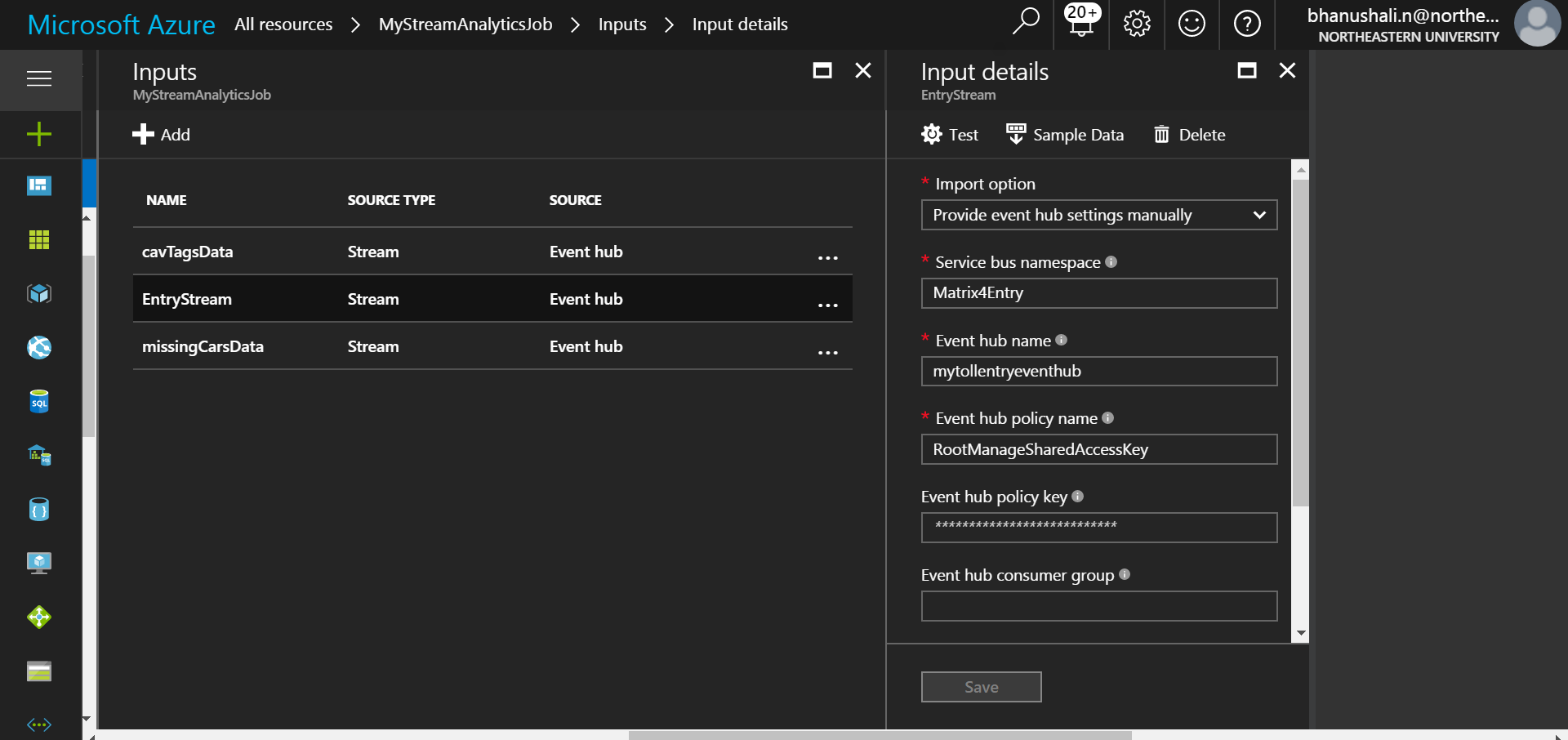
**Define Input Sources:**

The job will create and open the job page. Or you can click the created analytics job on the portal dashboard. Click the INPUTS tab to define the source data.

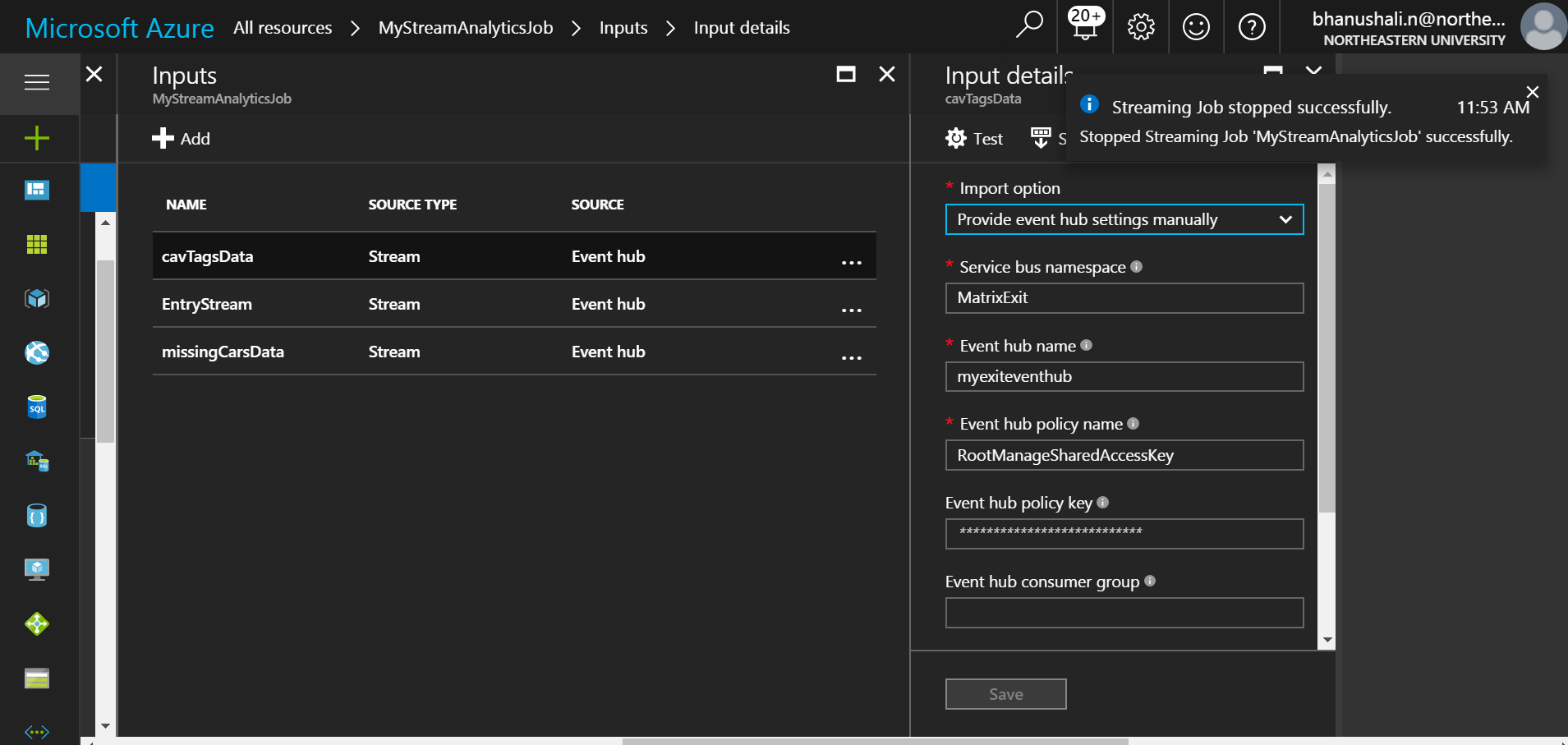


Now add new input (refer to the screen shots below):

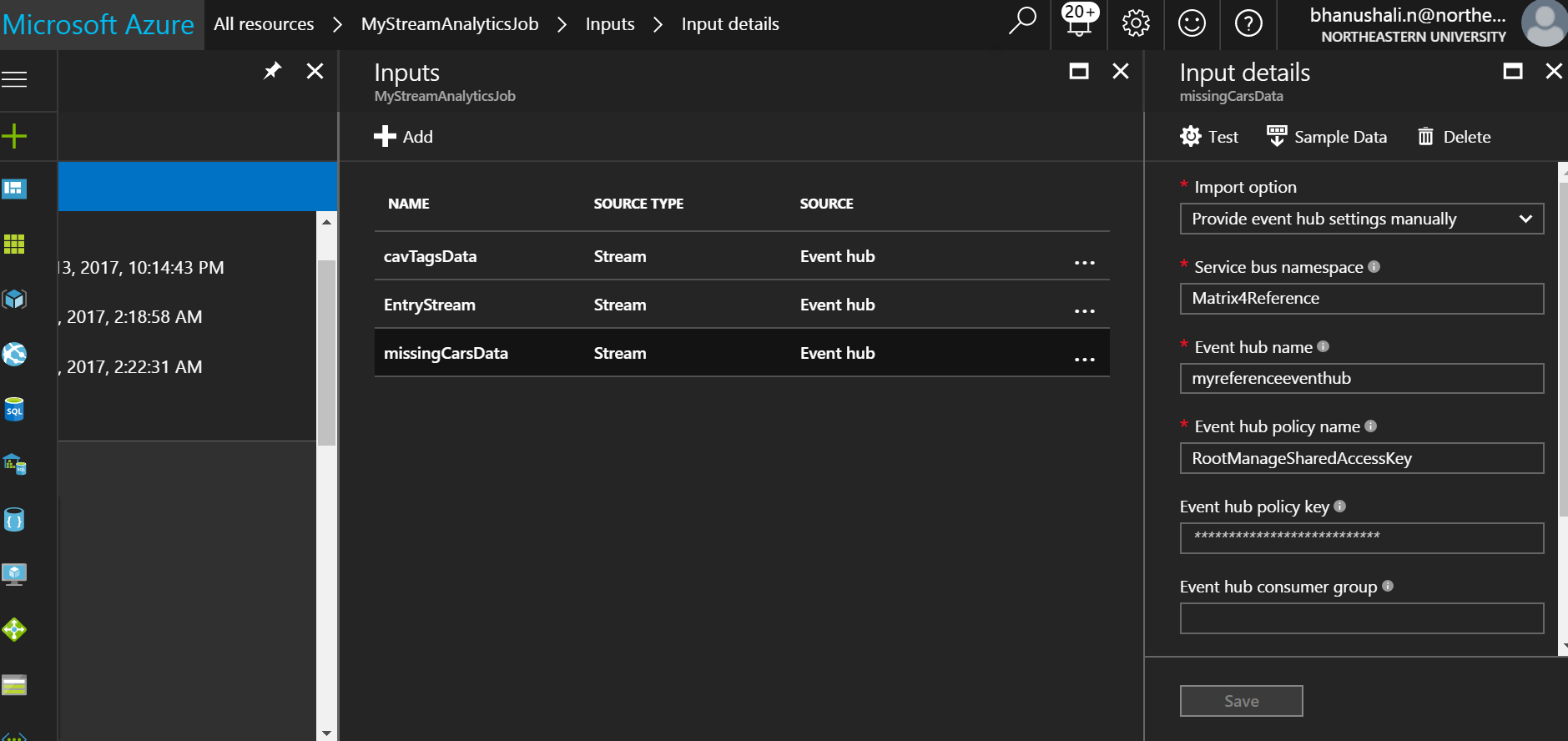
**Entry Stream**



**CAV tags input:**



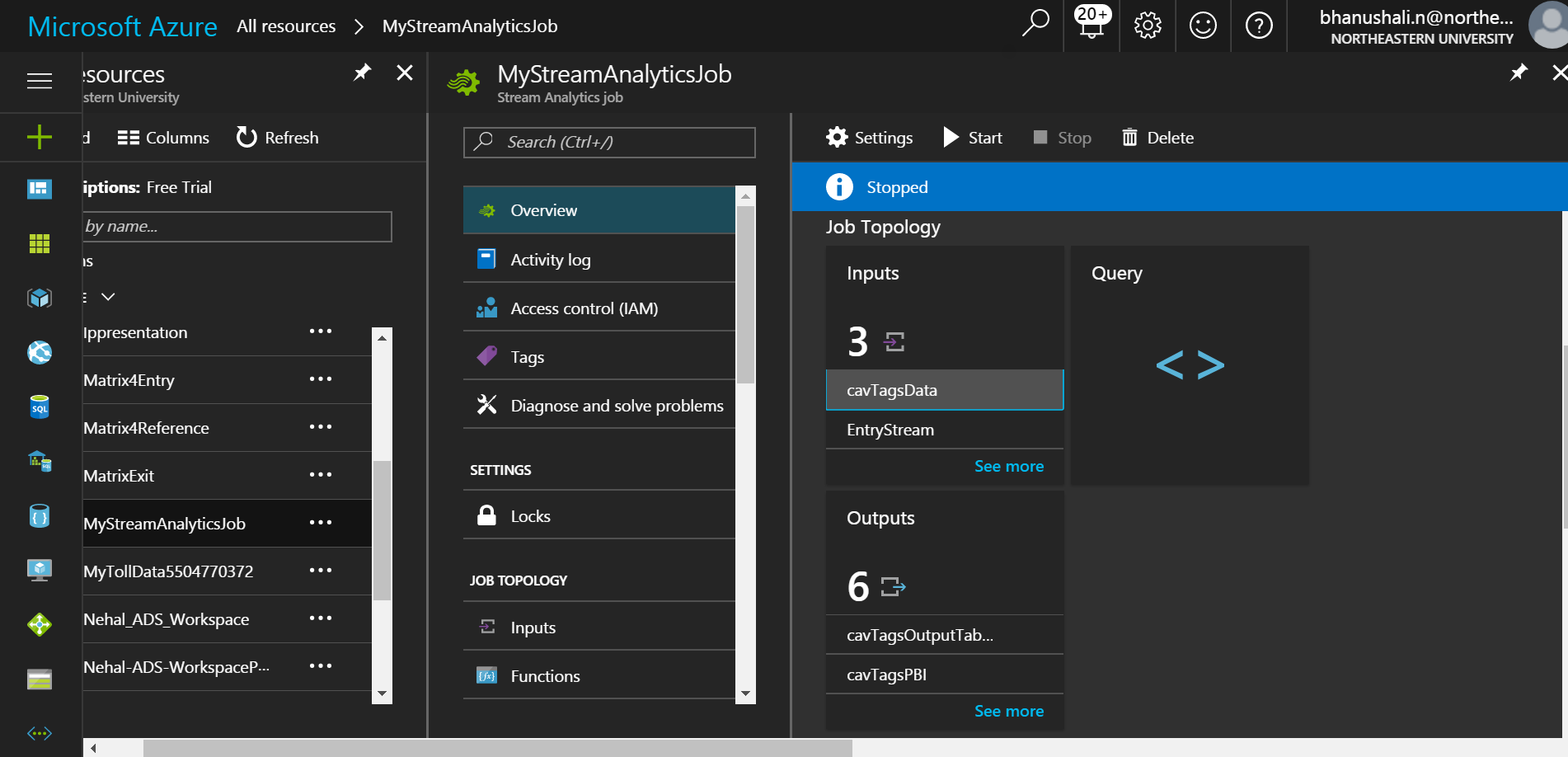
**Missing Cars input:**



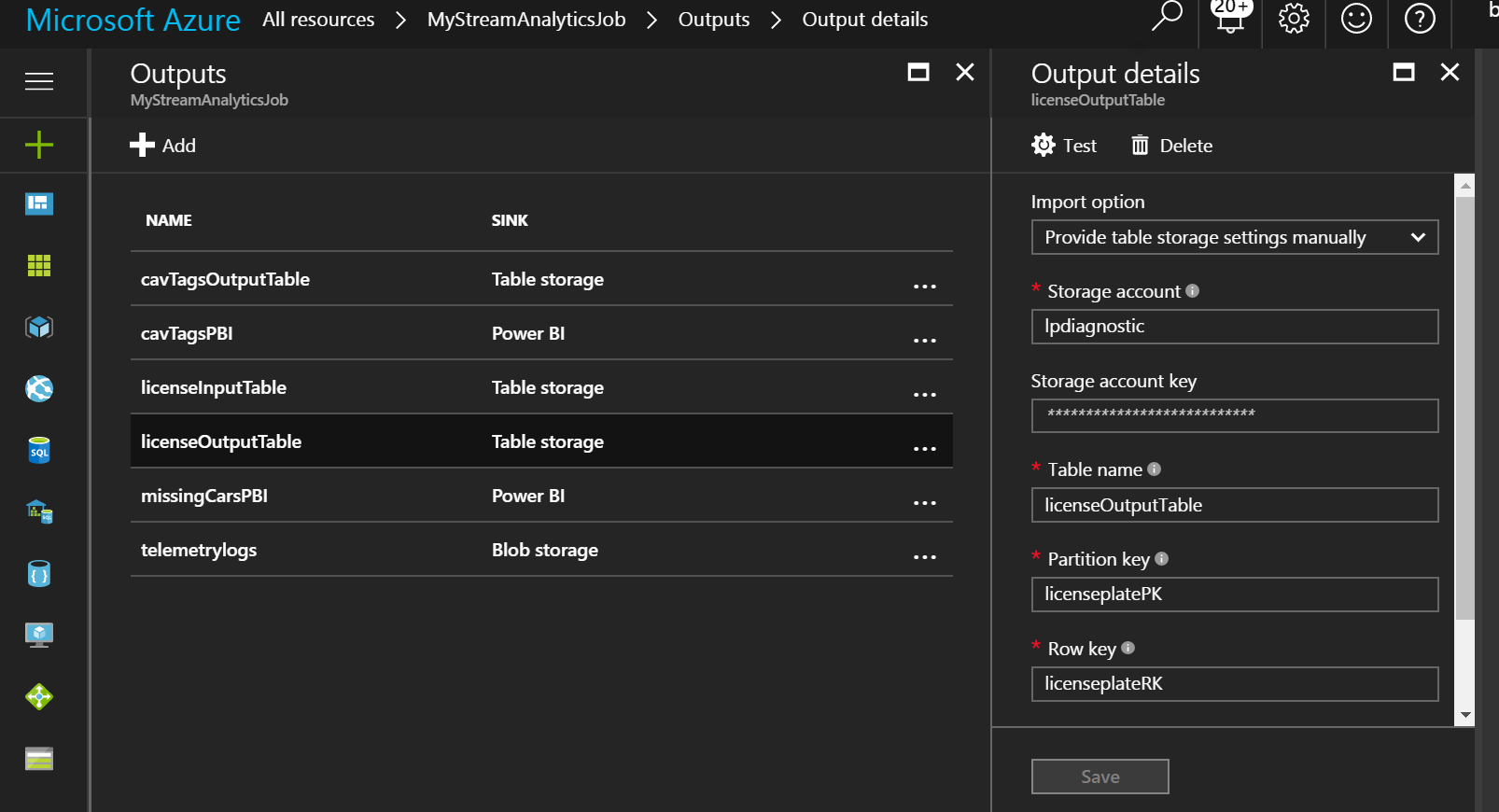
Note: Please make sure your service bus namespace and event hub names are unique for each of these streams. We would require them for connecting through event hub listener.

**Define Output:**

On the Stream Analytics job overview pane, select **OUTPUTS**.

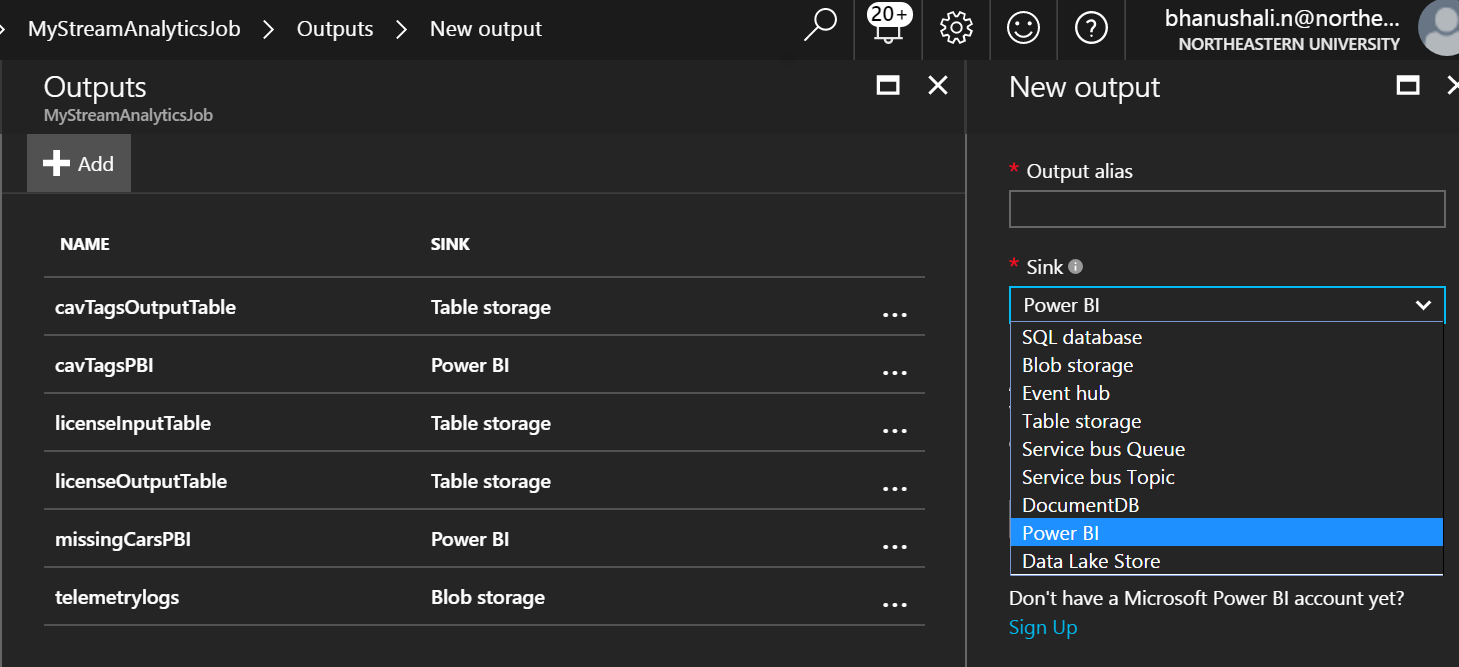


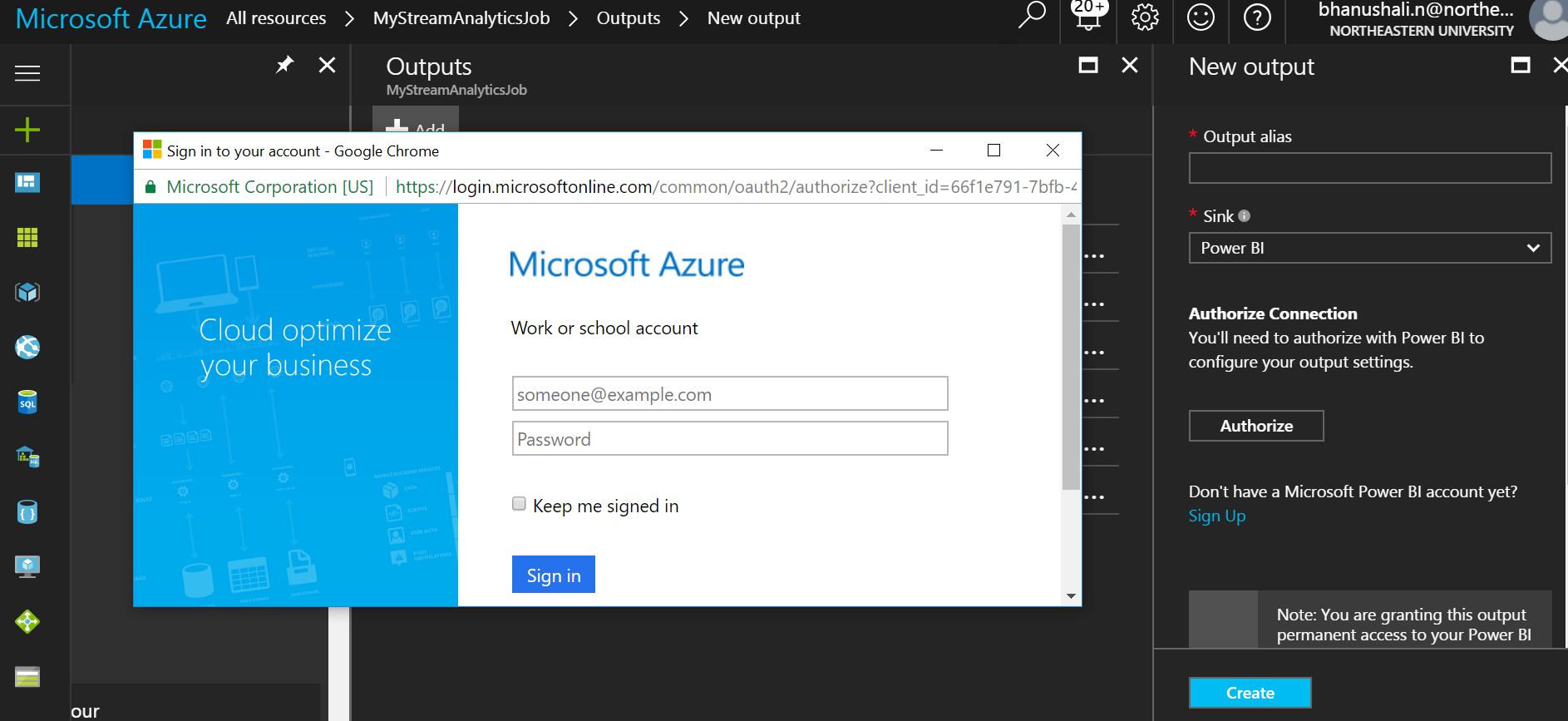
Output tables are table storage. We would need three of them for the three input streams.



**Power BI output:**

For power BI output, select ‘power BI’ as the option in ‘sink’. Now, authorize the power BI account to connect to Azure so that the data can flow directly.

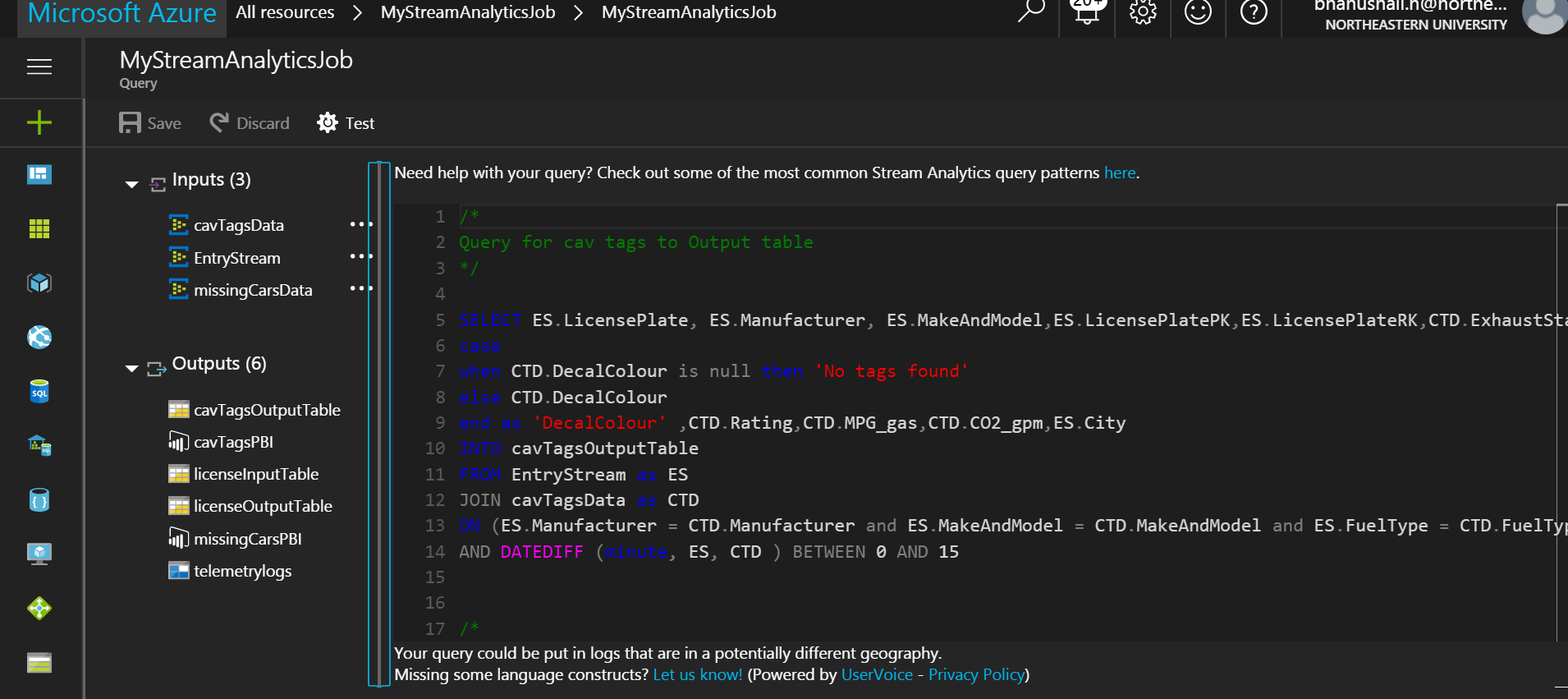




Note: Enter **tolladmin** in the **USERNAME** field, **123toll!** in the **PASSWORD** field, and **TollDataRefJoin** in the **TABLE** field.

**Azure Stream analytics query**

The stream-processing logic in ASA is expressed in a SQL-like query language with some added extensions such as windowing for performing temporal calculations.

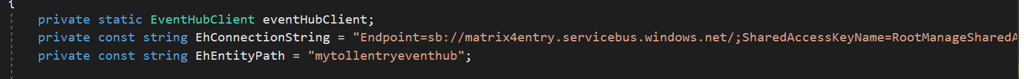


**Step 3: Event Hub Client Listener**

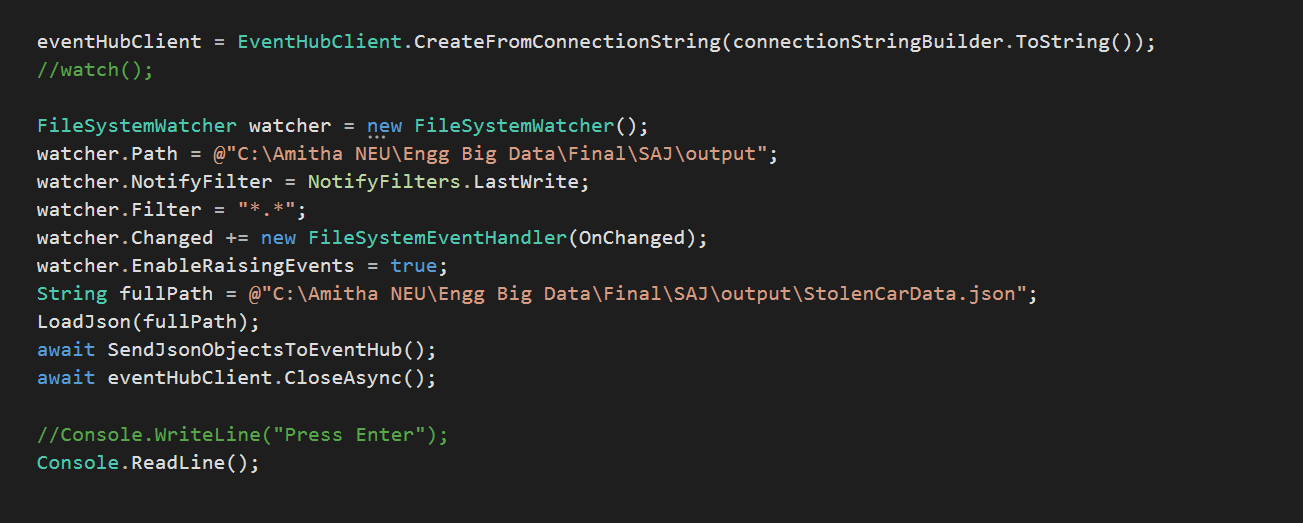
The primary class for interacting with Event Hubs is Microsoft.ServiceBus.Messaging.EventHubClient. This class provides both sender and receiver capabilities.

var client = EventHubClient.Create(description.Path);

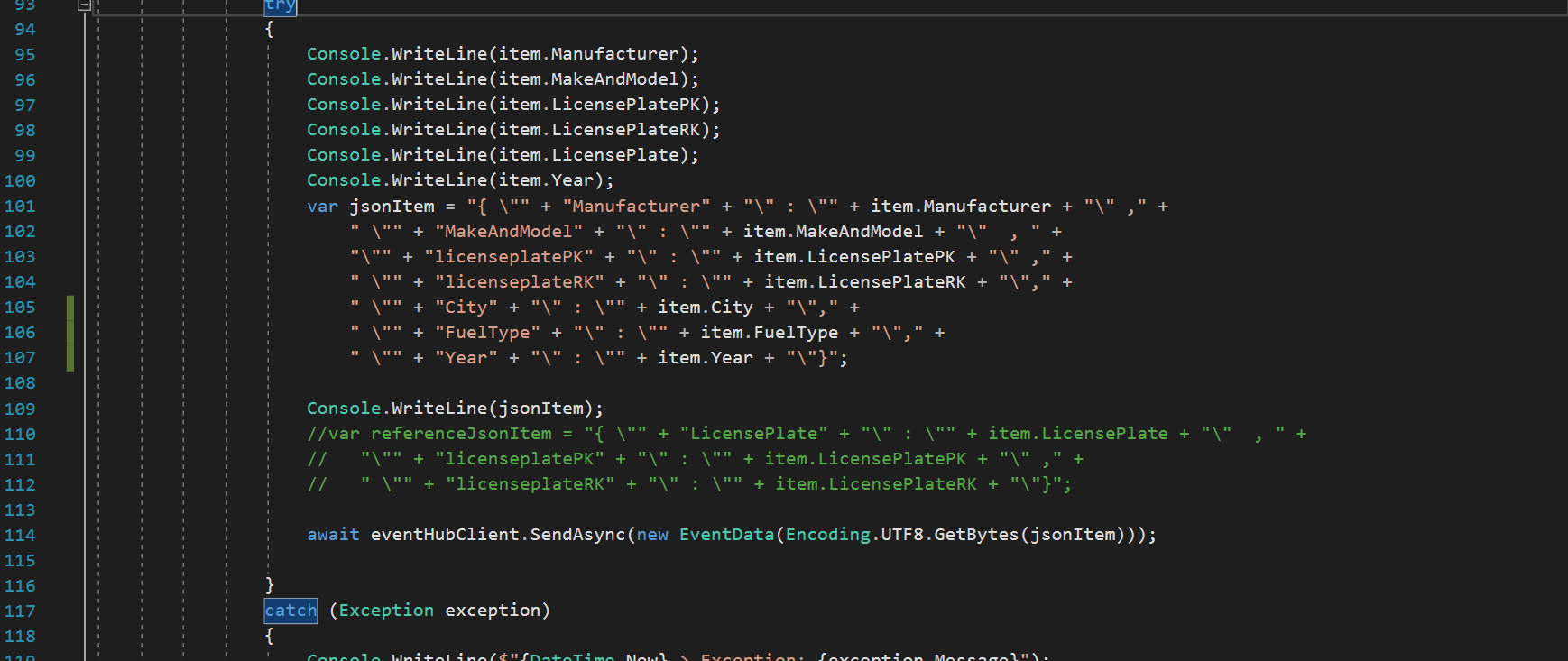
* Create a console application and add Microsoft.Azure.EventHubs package.
* Use await eventHubClient.SendAsync method to send the output.json file created in phase I.
* Add the ‘connection string’ from the event hub you are trying to connect to and then add the event hub name in ‘entity path’ as shown below in the screen shot:



We have used ‘windows file watcher’ to watch the folder. Every time phase I emits a new output file, it triggers an event and the file watcher send the json file path to onchange event.



Parse json and send them to event hub on Azure.



**Step 4: Executing SAJ job**

**Reference**

1. Stream analytics job setup:

<https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-build-an-iot-solution-using-stream-analytics#introduction>

1. Event hub listener:

<https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-dotnet-standard-getstarted-send>