Azure Stream Analytics real-time data processing utilization & use cases

Lab 7: Process event streams with Stream Analytics

Scenario: Imagine, You work for Contoso Inc. as a data engineer, and have been asked to build a traffic surveillance system for traffic police. This system must be able to analyse significant amounts of dynamically streamed data, captured from speed cameras and automatic number plate recognition (ANPR) devices, and then crosscheck the outputs against large volumes of reference data that holds vehicle, driver, and location information. Fixed roadside cameras, hand-held cameras (held by traffic police), and mobile cameras (in police patrol cars) are used to monitor traffic speeds and raise an alert if a vehicle is travelling too quickly for the local speed limit. The cameras also have built-in ANPR software that reads vehicle registration plates.

For this lab: we need to provide the followings:

* Create a Stream Analytics job to process Event Hub data
* Create azure data lake storage for storing Stored data
* Configure Stream Analytics job inputs
* Configure Stream Analytics job output
* Configure Stream Analytics job query
* Start ASA job
* Generate event hub data for processing with ASA
* View ASA job output in ADLS
* Stop the ASA job

**Task 1: Create a Data Lake Storage account**

1. Using the Microsoft account that is associated with your Azure Learning Pass subscription, use the Azure portal to create a Data Lake Store, with the following details:
   * **Name**: adls<*your name*><*date*>
   * **Resource group (Create new)**: CamerasRG
   * **Location**: Select your nearest location from the currently available Data Lake Store regions
   * Leave all other settings at their defaults
2. Wait until the storage has deployed before continuing with the lab.

**Task 2: Create an event hubs namespace and hub**

1. Create an event hub namespace blade, with the following details:
   * **Name**: camerafeeds<your name><date>
   * **Resource group (Use existing)**: CamerasRG
   * **Location**: Select the same location as you used for the Data Lake Store
   * Leave all other settings at their defaults
2. Wait until the namespace has deployed before continuing with the lab.
3. Add an event hub to your event hub namespace, with the following details:
   * **Name**: traffic
   * **Partition count**: 16
   * Leave all other settings at their defaults
4. Wait until the event hub has deployed before continuing with the lab.
5. Add a consumer group to the traffic event hub, with the following details:
   * **Name**: cameradatafeed
6. Add a second consumer group to the traffic event hub , with the following details:
   * **Name**: cameradatafeed2
7. Copy the **Primary Key** for your event hubs namespace **RootManageSharedAccessKey** policy, and save it as **Config\_details.txt** in your notepad.

**Task 3: Create a Stream Analytics job**

1. Create a Stream Analytics job, with the following details:
   * **Job name**: TrafficAnalytics
   * **Resource group (Use existing)**: CamerasRG
   * **Location**: Select the same location as you used for the Data Lake Store.
2. Wait until the Stream Analytics job has deployed before continuing with the lab.

**Task 4:Configure Stream Analytics job inputs**

1. Add an Event Hub input to the TrafficAnalytics Stream Analytics job, with the following details:
   * **Input alias**: CameraDataFeed
   * **Provide Event Hub settings manually** selected
   * **Service Bus namespace**: camerafeeds<*your name*><*date*> as you created earlier
   * **Event Hub name**: traffic
   * **Event Hub policy name**: RootManageSharedAccessKey
   * **Event Hub policy key**: Paste the key you copied into Config\_details.txt
   * **Event Hub consumer group**: cameradatafeed
2. Leave all other settings at their defaults.
3. Wait until the input has been successfully created before continuing with the lab.
4. Add a second Event Hub input to the TrafficAnalytics Stream Analytics job, with the following details:
   * **Input alias**: CameraDataFeed2
   * **Provide Event Hub settings manually**: selected
   * **Service Bus namespace**: camerafeeds<*your name*><*date*> as you created earlier
   * **Event Hub name**: traffic
   * **Event Hub policy name**: RootManageSharedAccessKey
   * **Event Hub policy key**: Paste the key you copied to Config\_details.txt
   * **Event Hub consumer group**: cameradatafeed2
5. Leave all other settings at their defaults.
6. Wait until the input has been successfully created before continuing with the lab.

**Task 5: Configure Stream Analytics job outputs**

1. Add a Power BI output to the TrafficAnalytics Stream Analytics job, with the following details:
   * **Output alias**: VisualData
   * Click **Sign up** to create a new Power BI account, then click **Authorize**, and sign in using your Power BI credentials. Click **Sign up** and create a new PowerBI account if you don't already have one.
   * **Dataset Name**: TrafficData
   * **Table Name**: TrafficData
2. Wait until the output has been successfully created before continuing with the lab.
3. Add a second output to the TrafficAnalytics Stream Analytics job, with the following details:
   * **Output alias**: StoredData
   * **Sink**: Data Lake Store
   * Click **Authorize**, and sign in using your Power BI credentials (you might not be prompted this time).
   * **Path prefix pattern**: SpeedData/{date}/{time}
   * Leave all other settings at their defaults
4. Wait until the output has been successfully created before continuing with the lab.

**Task 6: Configure a Stream Analytics job query**

* Add the following query to the TrafficAnalytics Stream Analytics job:
* SELECT
* CameraID,VehicleRegistration,Speed,SpeedLimit,LocationLatitude,LocationLongitude,Time
* INTO
* StoredData
* FROM
* CameraDataFeed
* SELECT
* CameraID, AVG(Speed) AS AvgSpeed
* INTO
* VisualData
* FROM
* CameraDataFeed2
* TIMESTAMP BY
* Time
* GROUP BY

CameraID, TumblingWindow(second, 30)

**Task 7: Start the Stream Analytics job**

1. Start the TrafficAnalytics Stream Analytics job.
2. Wait until the job has successfully started before continuing with the lab.

### Task 8: Generate event hub data for processing with Stream Analytics

1. Start Visual Studio, and open the  **SpeedCameraDevice.sln** project by downloading from [this OneDrive](https://1drv.ms/f/s!Ai6oG1oOWxh_h-1I20x5FBy-qBd6Tg) link.
2. Edit the **appSettings** section of **App.config** and replace the text **YourNamespace** in the Endpoint value with **camerafeeds<your name><date>**.
3. Also in the Endpoint value, replace the text **YourPrimaryKey** with the primary key you copied to your notepad.
4. Set **SpeedCameraDriver** as the startup project.
5. Build the solution, and verify that the app compiles successfully.
6. Start the app; it opens a console window displaying generated speed camera data that is being sent to the event hub.

### Task 9: Visualize Stream Analytics output using Azure Cosmos DB

### Task 10: View Stream Analytics output in Data Lake Store

1. Use the Data Explorer for your Data Lake Store to view the subfolders under **SpeedData**.
2. Open the log file in File Preview; verify that the data includes the fields you specified in your Azure Stream Analytics query.

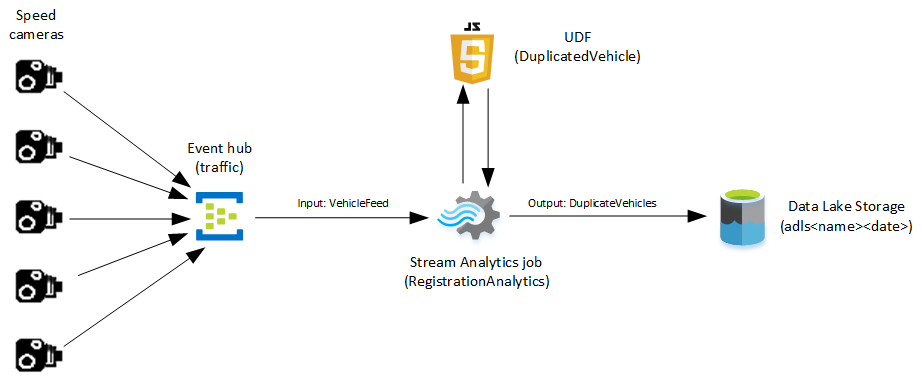
### Task 11: Stop the TrafficAnalytics job

1. Stop the TrafficAnalytics Stream Analytics job.
2. In the Visual Studio app window (where the data is being generated), press Enter to stop the app.

## Exercise 2: Use a Stream Analytics UDF to identify duplicate data records

### Scenario

For example. As a Data Architect for Fabrikum Inc, you will build logic to identify vehicles that appear to be using the same registration number(VIN). This could be indicative of a stolen vehicle running on false plates. In this exercise, you will use a JavaScript function to determine whether a vehicle with the same registration (not necessarily speeding) has been spotted at two locations that are an impossible distance apart within a given timeframe. You will store the details (times, registrations, locations).



The main tasks for this exercise are as follows:

1. Update the event hub and add a consumer group
2. Create and configure a new Stream Analytics job
3. Start the Speed Camera app
4. Examine the generated data
5. Close jobs and apps

**Task 1: Update the event hub and add a consumer group**

* Add a new consumer group called **vehiclefeed** to the **traffic** event hub.

**Task 2: Create and configure a new Stream Analytics job**

1. Create a new Stream Analytics job with the following details:
   * **Job name**: RegistrationAnalytics
   * **Resource group**: Use existing, and select CamerasRG.
   * **Location**: select the same location as you used for the Data Lake Store
   * **Streaming units**: 3
2. Wait until the Stream Analytics job has deployed before continuing with the lab.
3. Add a Data Lake Store output to the RegistrationAnalytics Stream Analytics job, with the following details:
   * **Output alias**: DuplicateVehicles
   * **Select Data Lake Store from your subscriptions**: selected
   * **Account name**: adls<*your name*><*date*>
   * **Path prefix pattern**: Duplicates/{date}/{time}
   * Click **Authorize**
   * Leave all other settings at their defaults
4. Wait until the output has been successfully created before continuing with the lab.
5. Add an input to the RegistrationAnalytics Stream Analytics job, with the following details:
   * **Input alias**: VehicleFeed
   * **Select Event Hub from your subscriptions**: selected
   * **Event Hub namespace**: camerafeeds<*your name*><*date*>
   * **Event Hub name**: Use existing, and select traffic
   * **Event Hub consumer group**: vehiclefeed
   * Leave all other settings at their defaults
6. Wait until the input has been successfully created before continuing with the lab.
7. Add a JavaScript UDF function with the alias **DuplicatedVehicle** to the RegistrationAnalytics Stream Analytics job,
8. Replace the template function code text with the code from the file **UDF2 .txt** downloading from [this OneDrive link](https://1drv.ms/t/s!Ai6oG1oOWxh_h-1LLOrH87bfc0cR7g). This UDF iterates through the list of vehicles captured during the latest time window and examines the data to determine whether a vehicle with the same registration number has been spotted at two or more locations. If so, the UDF determines the distance between the two locations and calculates whether it is possible for the vehicle to have travelled between these points in the time available. If not, the registration number is flagged as a possible duplicate.
9. Add the following query to the RegistrationAnalytics Stream Analytics job to send information about suspicious vehicles to the Data Lake Store:
10. SELECT
11. UDF.DuplicatedVehicle(Collect())
12. INTO
13. DuplicateVehicles
14. FROM
15. VehicleFeed
16. GROUP BY

TumblingWindow(minute, 2)

1. Start the RegistrationAnalytics Stream Analytics job. Wait until the job has been successfully started before continuing with the lab.

**Task 3: Start the Speed Camera app**

1. Start Visual Studio, and open the **SpeedCameraDevice** project from this [Github link](https://github.com/imcuteani/PacktAzureTraining/tree/master/Day1/HOL/Lab7) by extracting the project SpeedCaperaDevice(1).zip folder.
2. In **App.config**, in the **appSettings** section, in the **ServiceBusConnectionString** key, replace **YourNamespace** with **camerafeeds<*your name*><*date*>**, and replace **YourPrimaryKey** with the event hub primary key you copied to **notepad** in Exercise 1.
3. Set **SpeedCameraDriver** as the **Startup Project**.
4. Build the solution, verify that the app compiles successfully, and then start the app. The app opens a console window displaying generated speed camera data that is being sent to the event hub.

**Task 4: Examine the generated data**

1. Use the Data Explorer in the Data Lake blade of the Azure portal to view the contents of the JSON file in the Data Lake Duplicates folder.
2. Verify that the data includes duplicate registration numbers, and the locations at which they have been detected.

**Task 5: Close jobs and apps**

1. Close the **RegistrationAnalytics** job.
2. Stop the **SpeedCameraDevice2** app.
3. Close Visual Studio.