Microsoft

Getting Started with Azure Stream Analytics

Contents

Overview

Summary

In this lab learn how to create an end-to-end solution for real-time fraud detection with Azure Stream Analytics. Bring events into an Azure event hub, write Stream Analytics queries for aggregation or alerting, and send the results to an output sink to gain insight over data with real-time processing.

Scenario

A telecommunications company has a large volume of data for incoming calls. The company needs the following from its data: * Pare this data down to a manageable amount and obtain insights about customer usage over time and geographical regions. * Detect SIM fraud (multiple calls coming from the same identity around the same time but in geographically different locations) in real-time so that they can easily respond by notifying customers or shutting down service.

Azure Stream Analytics Summary

Stream Analytics is a fully managed service providing low-latency, highly available, scalable complex event processing over streaming data in the cloud. 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.

With a few clicks in the Azure portal, you can author a Stream Analytics job specifying the input source of the streaming data, the output sink for the results of your job, and a data transformation expressed in a SQL-like language. You can monitor and adjust the scale/speed of your job in the Azure portal to scale from a few kilobytes to a gigabyte or more of events processed per second.

Stream Analytics leverages years of Microsoft Research work in developing highly tuned streaming engines for time-sensitive processing, as well as language integrations for intuitive specifications of such.

What can I use Stream Analytics for?

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.

Key capabilities and Benefits?

- **Ease of use:** Stream Analytics supports a simple, declarative query model for describing transformations. In order to optimize for ease of use, Stream Analytics uses a SQL variant, and removes the need for customers to deal with the technical complexities of stream processing systems. Using the Stream Analytics query language in the browser query editor, you get intellisense auto-complete to help you can quickly and easily implement time series queries, including temporal-based joins, windowed aggregates, temporal filters, and other common operations such as joins, aggregates, projections, and filters. In addition, in-browser query testing against a sample data file enables quick, iterative development.

- **Scalability:** Stream Analytics is capable of handling high event throughput of up to 1GB/second. Integration with Azure Event Hubs allows the solution to ingest millions of events per second coming from connected devices, clickstreams, and log files, to name a few. In order to achieve this, Stream Analytics leverages the partitioning capability of Event Hubs, which can yield 1MB/s per partition. Users are able to partition the computation into a number of logical steps within the query definition, each with the ability to be further partitioned to increase scalability.

- **Reliability, repeatability and quick recovery:** A managed service in the cloud, Stream Analytics helps prevent data loss and provides business continuity in the presence of failures through built-in recovery capabilities. With the ability to internally maintain state, the service provides repeatable results ensuring it is possible to archive events and reapply processing in the future, always getting the same results. This enables customers to go back in time and investigate computations when doing root-cause analysis, what-if analysis, etc.

- **Low cost:** As a cloud service, Stream Analytics is optimized to provide users a very low cost to get going and maintain real-time analytics solutions. The service is built for you to pay as you go based on Streaming Unit usage and the amount of data processed by the system. Usage is derived based on the volume of events processed and the amount of compute power provisioned within the cluster to handle the respective Stream Analytics jobs.

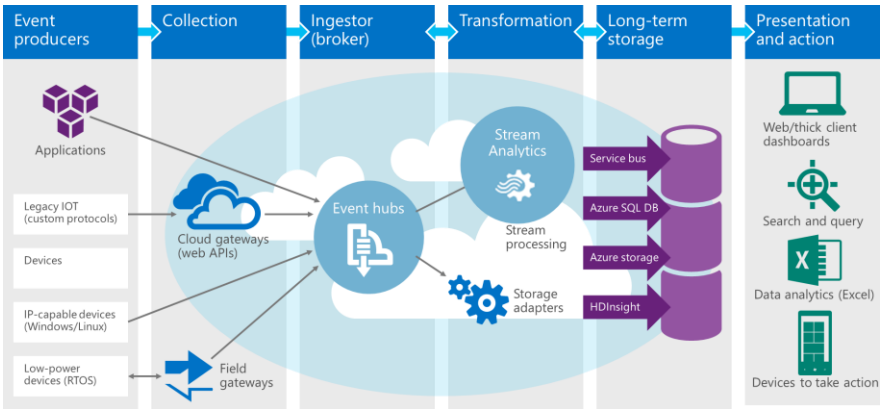
- **Reference data:** Stream Analytics provides users the ability to specify and use reference data. This could be historical data or simply non-streaming data that changes less frequently over time. The system simplifies the use of reference data to be treated like any other incoming event stream to join with other event streams ingested in real time to perform transformations.

- **Connectivity:** Stream Analytics connects directly to Azure Event Hubs for stream ingestion, and the Azure Blob service to ingest historical data. Results can be written from Stream Analytics to Azure Storage Blobs or Tables, Azure SQL DB, Event Hubs, Azure Service Bus Topics or Queues, and Power BI, where it can then be visualized, further processed by workflows, used in batch analytics via Azure HDInsight or processed again as a series of events. When using Event Hubs it is possible to compose multiple Stream Analytics together with other data sources and processing engines without losing the streaming nature of the computations.

End-to-end stream processing on Microsoft Azure

Applications or devices produce messages which are sent to an *Azure Event Hub*. Event Hubs acts as the "front door" for an event pipeline, and once data is collected into an event hub, it can be transformed and stored using any real-time analytics provider or batching/storage adapters. Event Hubs decouples the production of a stream of events from the consumption of those events, so that event consumers can access the events on their own schedule. Unlike Service Bus queues and topics, Event Hubs is focused on delivering messaging stream handling at scale. Event Hubs capabilities differ from topics in that they are strongly biased towards high throughput and event processing scenarios. As a result, Event Hubs do not implement some of the messaging capabilities that are available for topics. If you need those capabilities, topics remain the optimal choice.

Stream Analytics is the *consumer* of the event hub, which queries the data in the Event Hub (in real-time) using a SQL-like language and produces an output sink to a service bus, blob storage, Azure SQL DB or HDInsight



This allows the output of the Stream Analytics query to be presented in PowerBI, Web dashboards or in devices to take action.

Stream Analytics Query Language Overview

DML Statements

- SELECT
- FROM
- WHERE
- GROUP BY
- HAVING
- CASE
- JOIN
- UNION

Statistical Functions

- VAR
- VARP
- STDEV
- STDEVP

Date and Time Functions

- DATENAME
- DATEPART
- DAY
- MONTH
- YEAR
- DATETIMEFROMPARTS
- DATEDIFF
- DATADD

Windowing Extensions

- Tumbling Window
- Hopping Window
- Sliding Window
- Duration

Aggregate Functions

- SUM
- COUNT
- AVG
- MIN
- MAX

Scaling Functions

- WITH
- PARTITION BY

String Functions

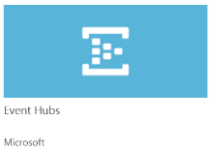
- LEN
- CONCAT
- CHARINDEX
- SUBSTRING
- PATINDEX

Create an Azure Namespace & Event Hub

The sample application will generate events and push them to an Event Hub instance for real-time processing. Service Bus Event Hubs are the preferred method of event ingestion for Stream Analytics and you can learn more about Event Hubs in Azure Service Bus documentation.

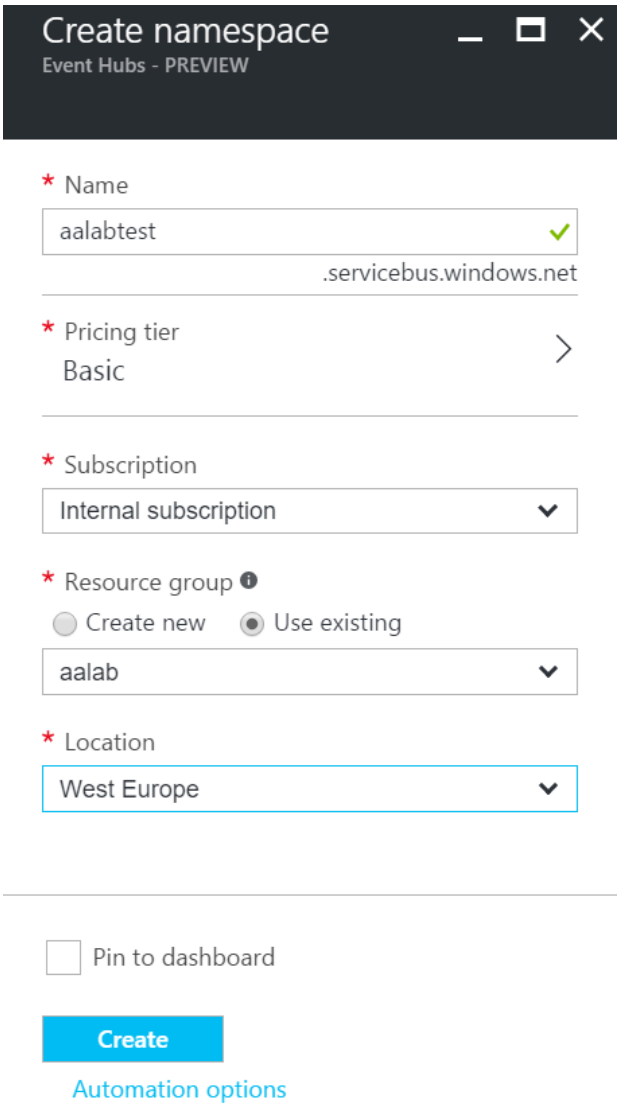
To create an Event Hub:

1. In the Azure portal click **New,** select **Internet of Things** category and then **Event Hubs**:

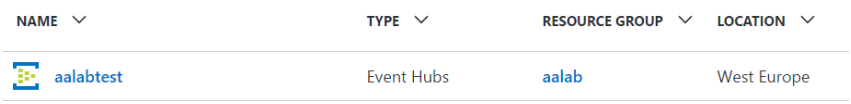


2. Click **Create** on service description blade.

3. Enter a valid **event hub namespace name** (it needs to be unique to the whole of Azure), select **Pricing Tier (Basic)** and select new or existing **Resource group**. Lastly, choose data center **Location** and click **Create**.

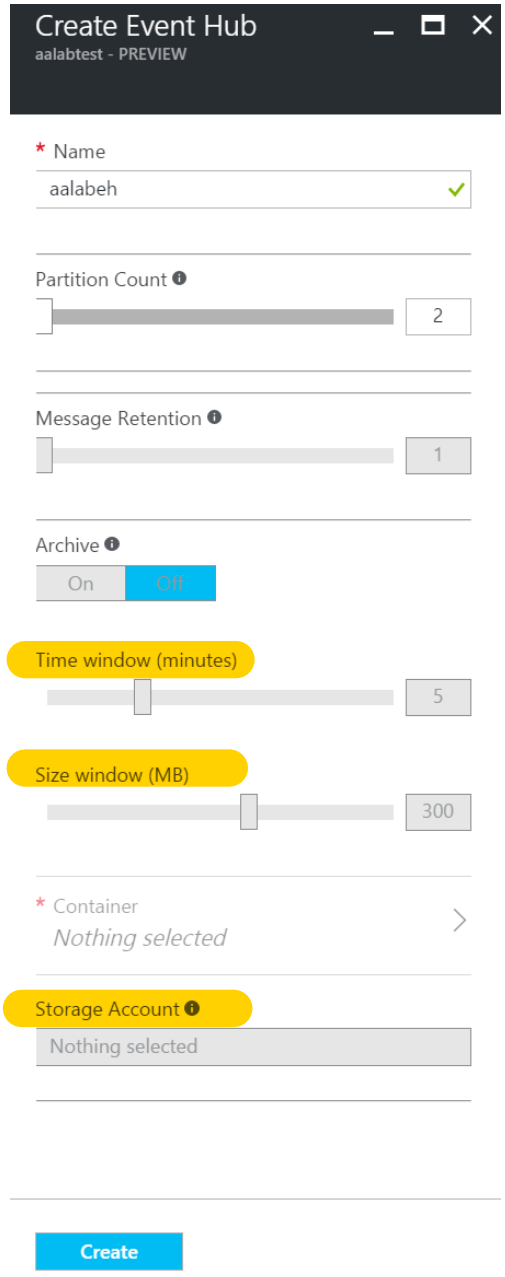


4. Once the Namespace is activated, click on the name i.e.

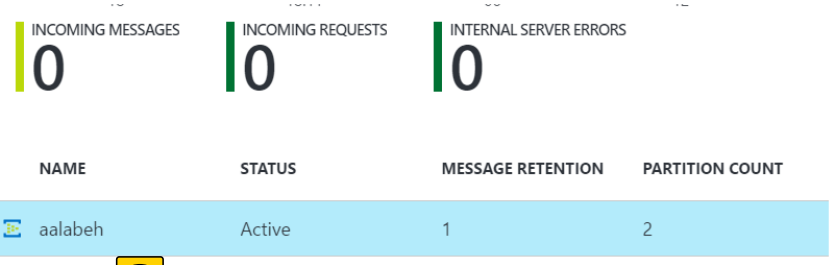


5. On the **Event Hub namespace** blade select **Add Event Hub** button:

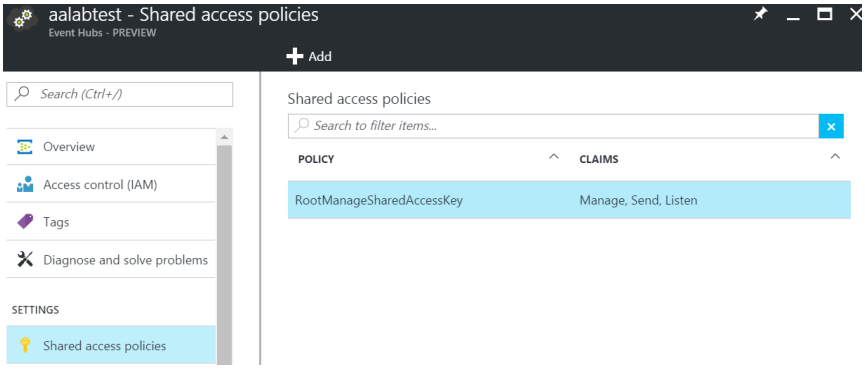
6. Enter valid event hub **Name** and click **Create** (the rest of the options are disabled as we're using basic tier of the service):



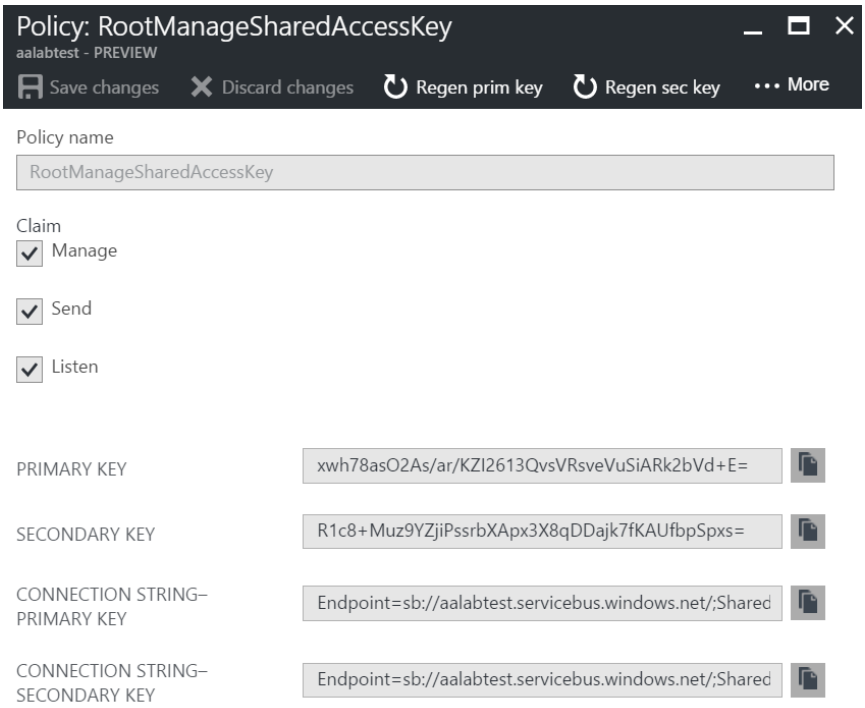
7. Once the event hub is successfully created you should see its its name on **Event Hub Namespace** blade:



8. On **Service Bus Dashboard** select **Settings > Shared Access Policies** to display list containing default access policy called **RootManageSharedAccessKey:**



Select it to see details of the policy. Make note that it contains connection string information (**Connection String – Primary Key**) required later on to establish communication with event hub – we will need it later:



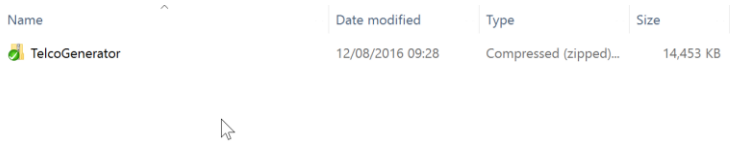
Start event generator application

We have provided a client application that will generate sample incoming call metadata and push it to Event Hub. In the folder that came with this document we have provided both the source code and application. In this section we will outline how to run the application

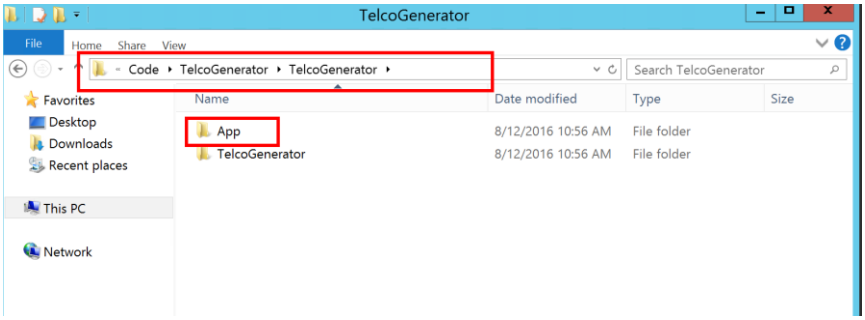
If you are an application developer, feel free to study the C# source code.

Follow the steps below to set up this application.

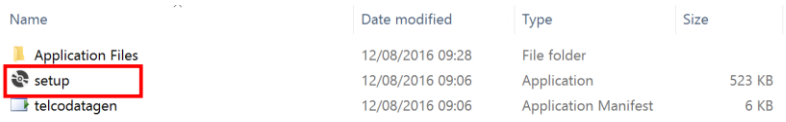
1. Open the folder **Code** and **unzip** the **TelcoGenerator** zip file.



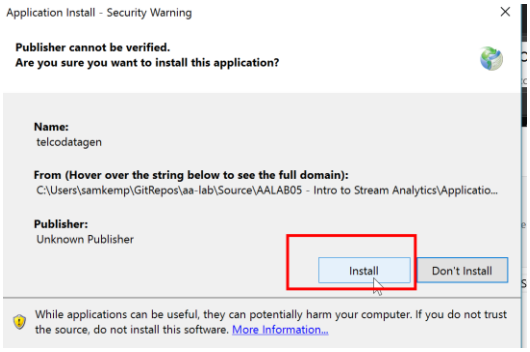
2. In file explorer open the **Code** Folder > **TelcoGenerator** > **TelcoGenerator** > **App**. If you would like to look at the C# code, it is stored in the TelcoGenerator folder.



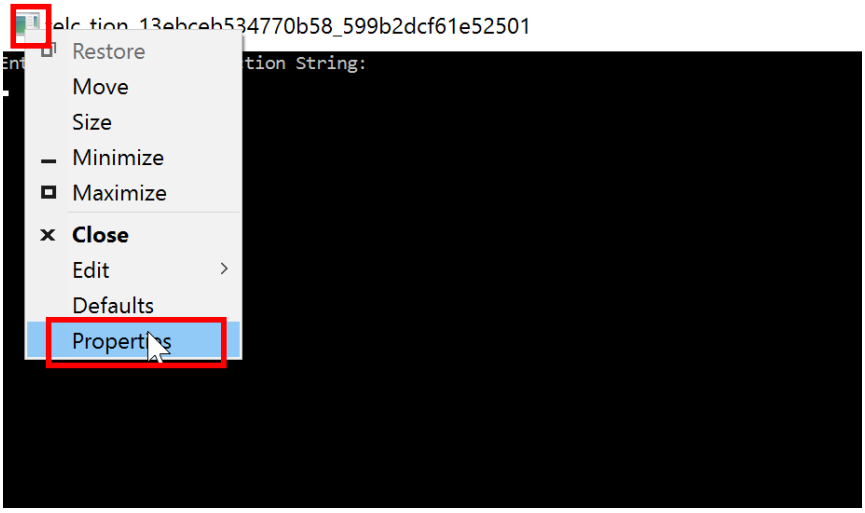
3. Run the setup program.



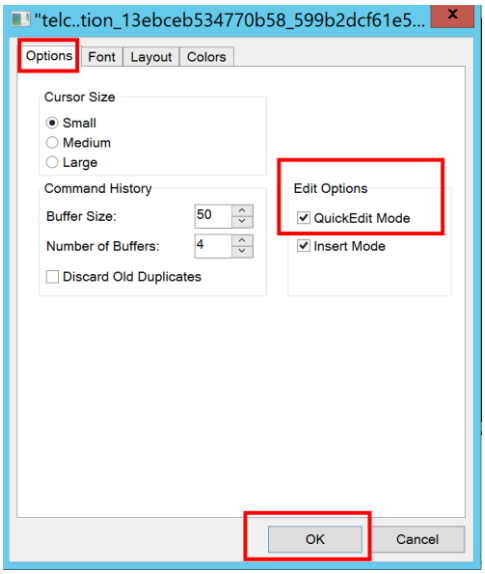
4. If happy to do so, Install the Program



5. A console application will pop up > **right click** in the **left-hand** corner > click **Properties**



6. On the **Options** tab ensure that the **QuickEdit Mode** is checked.



7. When prompted enter:

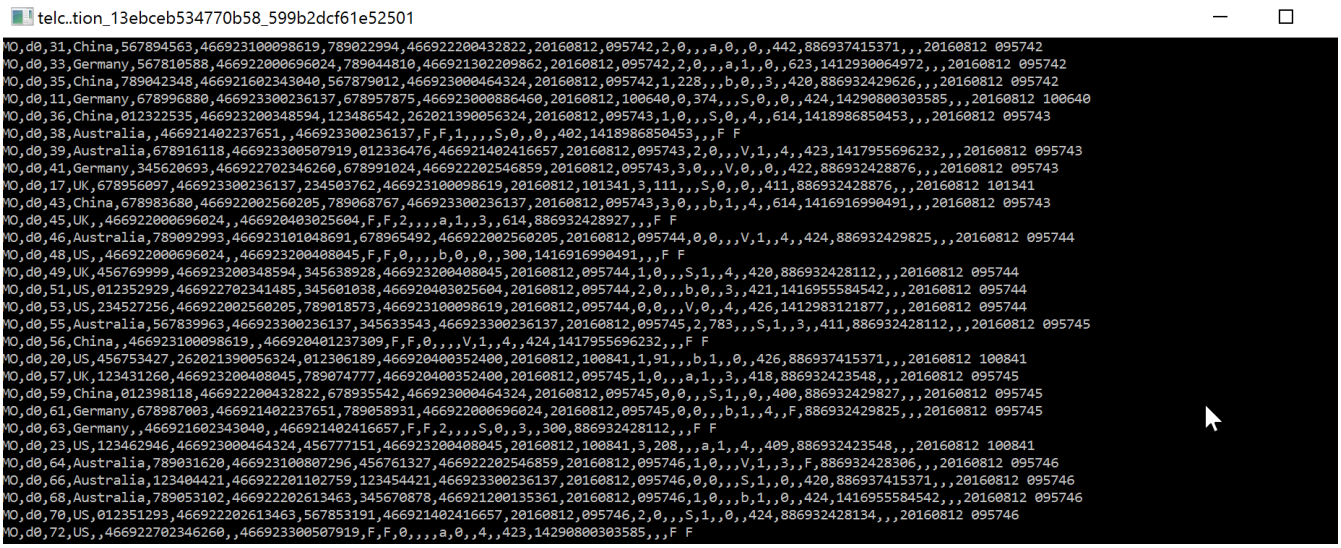
   - **Connection String** – follow the last step in the previous section to get the connection string. When you have copied the connection string you can paste it into the console with a simple right click.
   - **Event Hub Name** (**beware it is NOT the name of event hub namespace, but individual hub you created**)

We will be generating 1000 events with a 20% chance of fraud over the course of 2 hours.

You will see records being sent to your Event Hub. **Keep the console app running for the duration of the lab so you see real-time events.**

Some key fields that we will be using in this real-time fraud detection application are defined here:

RECORD	DEFINITION
CallrecTime	Timestamp for the call start time.
SwitchNum	Telephone switch used to connect the call.
CallingNum	Phone number of the caller.
CallingIMSI	International Mobile Subscriber Identity (IMSI). Unique identifier of the caller.
CalledNum	Phone number of the call recipient.
CalledIMSI	International Mobile Subscriber Identity (IMSI). Unique identifier of the call recipient.



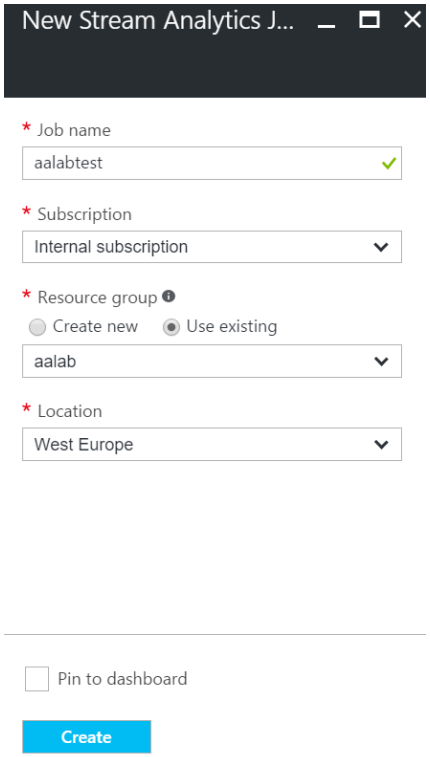
Create Stream Analytics job

Now that we have a stream of telecommunications events, we can set up a Stream Analytics job to analyse these events in real-time.

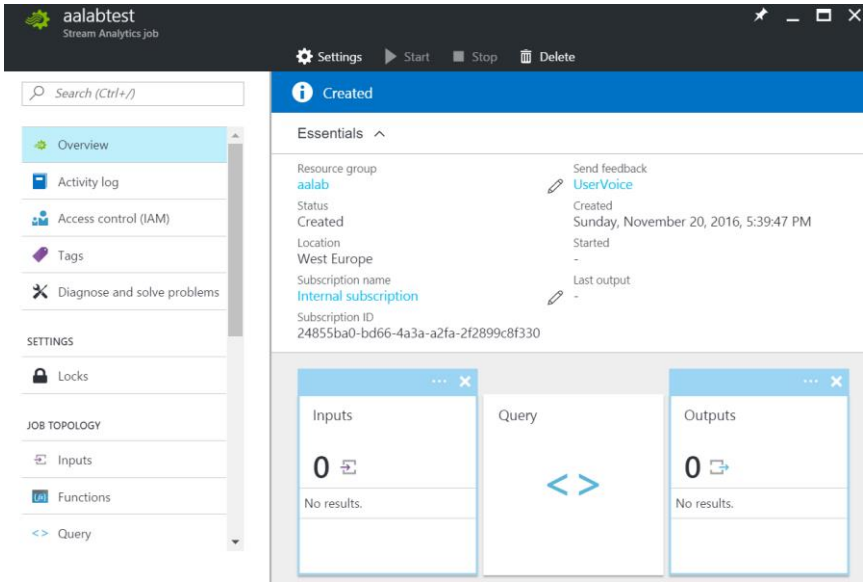
1. In the Azure portal, click **New** > **Intelligence + analytics** > **Stream Analytics job**.

2. Specify the following values, and then click **Create**:

   - **Job Name**: Enter a job name.
   - **Location**: Select the region where you want to run the job. Consider placing the job and the event hub in the same region to ensure better performance and to ensure that you will not be paying to transfer data between regions.
   - **Resource group**: choose either new or existing resource group.



3. Once the job provisioning has been finished you may see it on the list on the list of available stream analytic jobs. Click on its name to see the details:



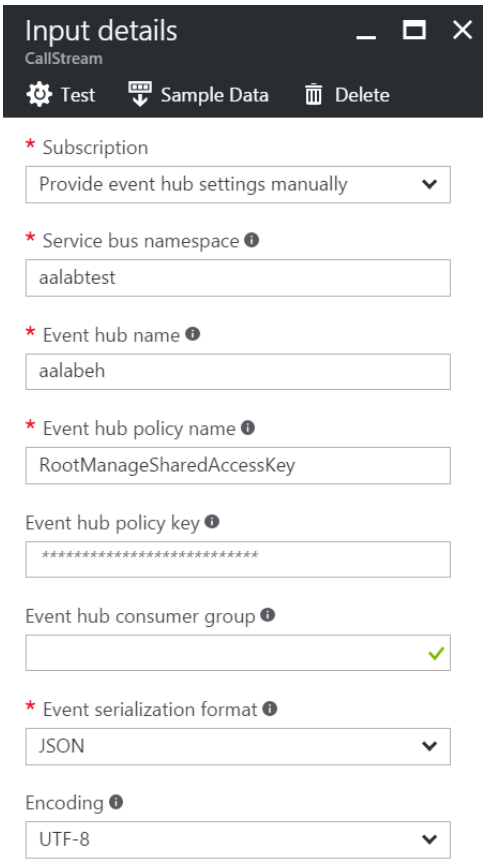
4. The new job will be shown with a status of **Created**. Notice that the **Start** button on the top of the page is disabled. You must configure the job input, output, and query before you can start the job.

5. In your Stream Analytics job click **Inputs** from the top of the page, and then click **+ Add**. The blade that opens will walk you through several options to set up your input:

    - **Input Alias**: Enter a friendly name for this job input such as **CallStream**. Note that you will be using this name in the query later.
    - Select **Data Stream** as **Source Type.**
    - Select **Event hub** as **Source**.
    - Select **Use event hub from current subscription** as **Subscription**.
    - In **Service bus namespace** select the namespace you created before.
    - In **Event hub name** select event hub that collect information from TelcoGenerator application.
    - Select **RootManageSharedAccessKey** as **Event hub policy name**.
    - Leave **Event hub consumer group** field empty.
    - Select **JSON** as **Event serialization format**.
    - Select **UTF-8** as **Encoding**.

    Click **Create**.

6. To verify whether your input is working properly, select it on the list and click on **Test** button. You should see “Successful connection test” message:



Querying the stream

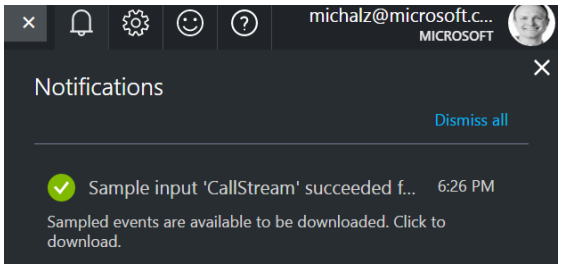
Stream Analytics supports a simple, declarative query model for describing transformations for real-time processing. To learn more about the language, see the Azure Stream Analytics Query Language Reference. This tutorial will help you author and test several queries over your real-time stream of call data.

To validate your query against actual job data, you can use the **Sample Data** feature to extract events from your stream and create a .JSON file of the events for testing. The following steps show how to do this and we have also provided a sample Telco.json file for testing purposes:

1. Select your Event Hub input and click **Sample Data** at the top of the blade.

2. Specify a **Start Time** to start collecting data from and a **Duration** for how much additional data to consume.

3. Click **OK** to start sampling data from the input. It can take a minute or two for the data file to be produced. When the process is completed, you will receive notification message and data will be available for download:



4. If you want to archive every event, you can use a **passthrough query** to read all the fields in the payload of the event or message. To start with, do a simple passthrough query that projects all the fields in an event.

5. Click **Query** on your Stream Analytics blade.

6. Add the following to the code editor:

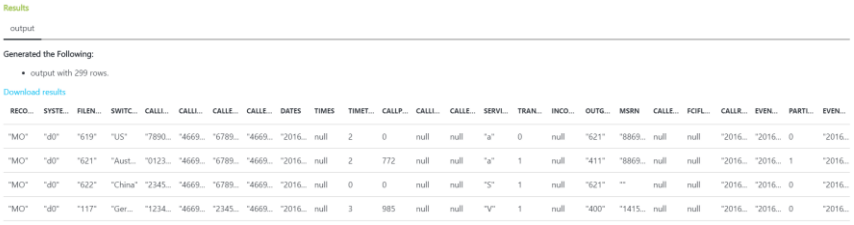
```
SELECT * FROM CallStream
```

Make sure that the name of the input source matches the name of the input you specified earlier.

7. Right click on the name of the **CallStream** input and select **"Upload sample from file"** to upload the sample file that you created previously. Alternatively you may select "**Sample data from input**" to start sampling again.

8. Click **Test** at the top of the editor.

9. See the results displayed below the query definition:

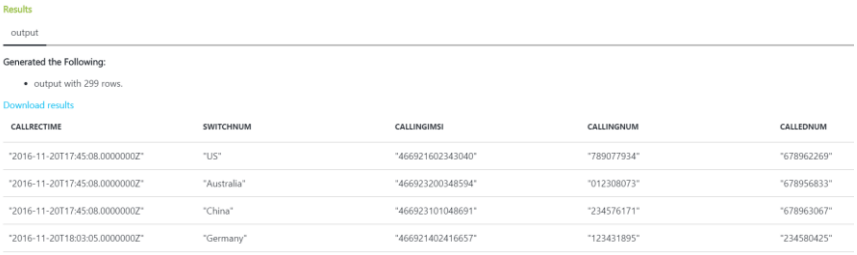


We'll now pare down the returned fields to a smaller set.

10. Change the query in the code editor to:

```
SELECT CallRecTime, SwitchNum, CallingIMSI, CallingNum, CalledNum
FROM CallStream
```

11. Click **Test** in the query editor again to see the results of the query:



To compare the amount that incoming calls per region we'll leverage a TumblingWindow to get the count of incoming calls grouped by SwitchNum every 5 seconds.

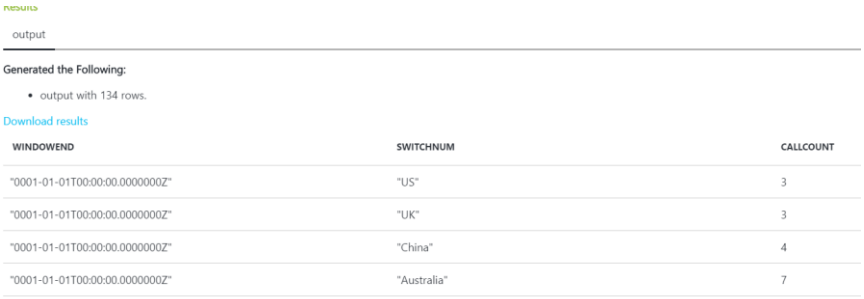
12. Change the query in the code editor to:

```
SELECT System.Timestamp as WindowEnd, SwitchNum, COUNT(*) as CallCount
FROM CallStream TIMESTAMP BY CallRecTime
GROUP BY TUMBLINGWINDOW(s, 5), SwitchNum
```

This query uses the **TIMESTAMP BY** keyword to specify a timestamp field in the payload to be used in the temporal computation. If this field wasn't specified, the windowing operation would be performed using the time each event arrived at Event Hub. See "Arrival Time Vs Application Time" in the Stream Analytics Query Language Reference.

Note that you can access a timestamp for the end of each window by using the **System.Timestamp** property.

13. Click **Test** in query editor to see the results of the query.



output

Generated the Following:

- output with 134 rows.

Download results

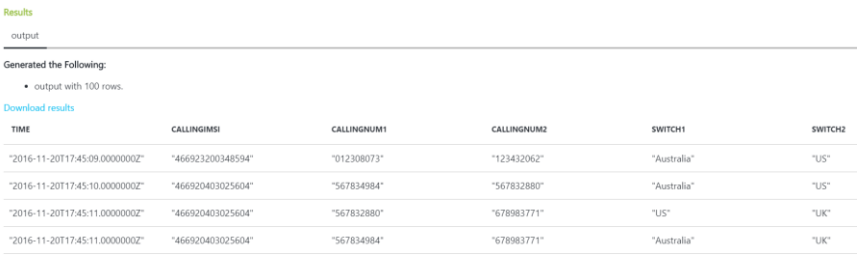
WINDOWEND	SWITCHNUM	CALLCOUNT
"0001-01-01T00:00:00.0000000Z"	"US"	3
"0001-01-01T00:00:00.0000000Z"	"UK"	3
"0001-01-01T00:00:00.0000000Z"	"China"	4
"0001-01-01T00:00:00.0000000Z"	"Australia"	7

To identify potentially fraudulent usage, we'll look for calls originating from the same user but in different locations in less than 5 seconds. We join the stream of call events with itself (self join) to check for these cases.

14. Change the query in the code editor to:

```
SELECT System.Timestamp as Time, CS1.CallingIMSI, CS1.CallingNum
as CallingNum1, CS2.CallingNum as CallingNum2, CS1.SwitchNum as
Switch1, CS2.SwitchNum as Switch2
FROM CallStream CS1 TIMESTAMP BY CallRecTime
JOIN CallStream CS2 TIMESTAMP BY CallRecTime
ON CS1.CallingIMSI = CS2.CallingIMSI
AND DATEDIFF(ss, CS1, CS2) BETWEEN 1 AND 5
WHERE CS1.SwitchNum != CS2.SwitchNum
```

15. Click **Test** in the query editor to see the results of the query.



Results

output

Generated the Following:

- output with 100 rows.

Download results

TIME	CALLINGIMSI	CALLINGNUM1	CALLINGNUM2	SWITCH1	SWITCH2
"2016-11-20T17:45:09.0000000Z"	"466923200348594"	"012308073"	"123432062"	"Australia"	"US"
"2016-11-20T17:45:10.0000000Z"	"466920403025604"	"567834984"	"567832880"	"Australia"	"US"
"2016-11-20T17:45:11.0000000Z"	"466920403025604"	"567832880"	"678983771"	"US"	"UK"
"2016-11-20T17:45:11.0000000Z"	"466920403025604"	"567834984"	"678983771"	"Australia"	"UK"

Create output sink & Job output

Now that we have defined an event stream, an Event Hub input to ingest events, and a query to perform a transformation over the stream, the last step is to define an output sink for the job. We'll write events for fraudulent behavior to Blob storage.

Follow the steps below to create a container for Blob storage if you don't already have one.

1. Use an existing storage account or create a new storage account by clicking **NEW** > **Storage** > **Storage** and following the instructions.

2. Select the storage account, click **Blobs** to see list of containers and then click **+ Container** to create new container. 

3. Specify a **NAME** for your container and set its **ACCESS** to **Private**.

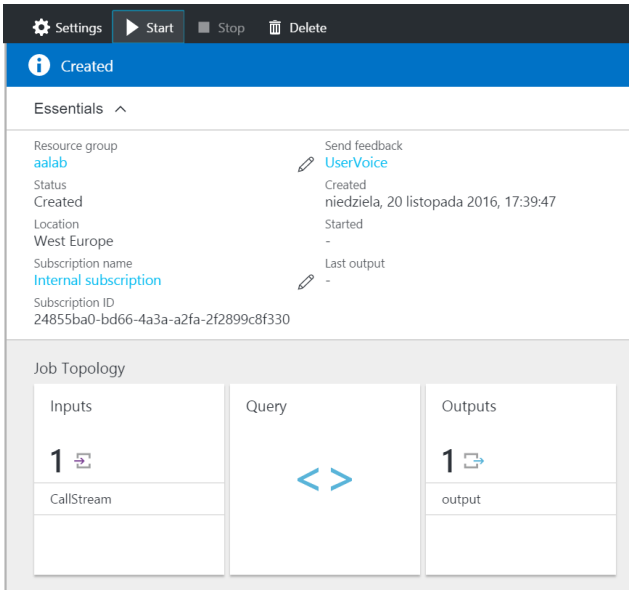
4. In your Stream Analytics job click **OUTPUT** and then click **+Add**. The blade that opens will walk you through a number of options to set up your output:

    - **OUTPUT ALIAS**: Enter a friendly name for this job output.
    - Select **Blob Storage** as **Sink**.
    - Select **Use blob storage from current subscription** as **Subscription.**
    - Select name of storage account you just created as **Storage Account.**
    - Put name of your container in **Container** field**.**
    - Leave **Path pattern** field empty
    - Select **JSON** as **Event serialization format**.
    - Select **UTF-8** as **Encoding**.
    - Select **Line separated** as **Format**.

    Finally click **Create**.

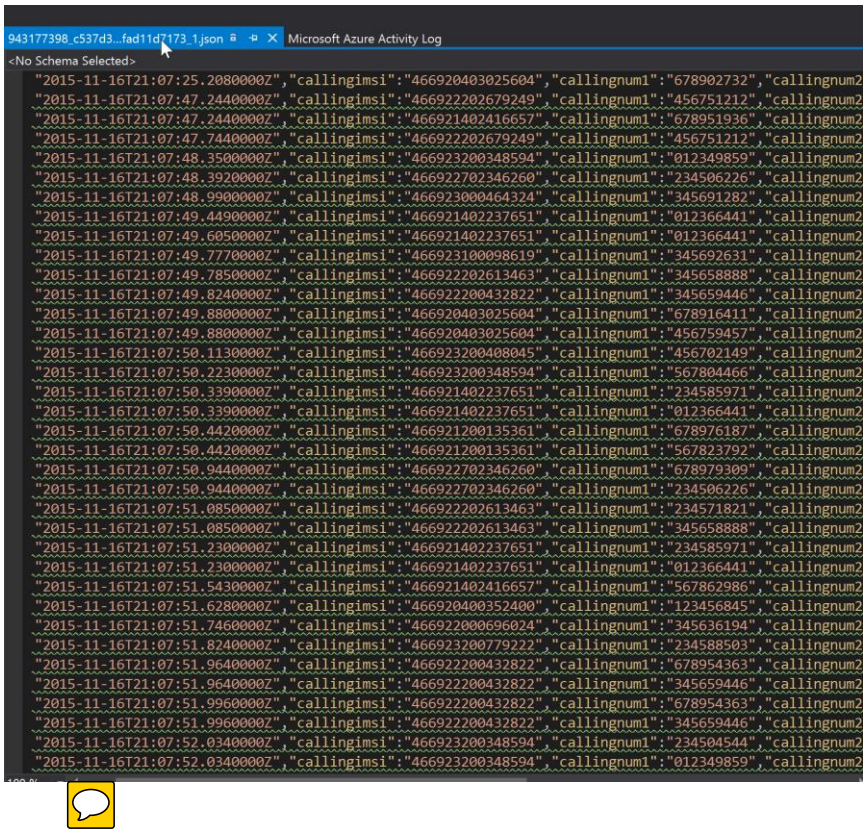
    Since a job input, query, and output have all been specified, we are ready to start the Stream Analytics job for real-time fraud detection.

1. From the Stream Analytics job blade, click **START** at the top of the page.



2. In the dialog box that appears select **Now** as **Job output start time** and then click **Start** button on the bottom. The job status will change to **Starting** and will shortly move to **Running**.

3. Use a tool like Visual Studio Cloud Explorer or Azure Storage Explorer to view fraudulent events as they are written to your output in real-time.



Terms of Use

By using this Hands-on Lab, you agree to the following terms:

The technology/functionality described in this Hands-on Lab is provided by Microsoft Corporation in a “sandbox” testing environment for purposes of obtaining your feedback and to provide you with a learning experience. You may only use the Hands-on Lab to evaluate such technology features and functionality and provide feedback to Microsoft. You may not use it for any other purpose. You may not modify copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, or sell this Hands-on Lab or any portion thereof.

COPYING OR REPRODUCTION OF THE HANDS-ON LAB (OR ANY PORTION OF IT) TO ANY OTHER SERVER OR LOCATION FOR FURTHER REPRODUCTION OR REDISTRIBUTION IS EXPRESSLY PROHIBITED.

THIS HANDS-ON LAB PROVIDES CERTAIN SOFTWARE TECHNOLOGY/PRODUCT FEATURES AND FUNCTIONALITY, INCLUDING POTENTIAL NEW FEATURES AND CONCEPTS, IN A SIMULATED ENVIRONMENT WITHOUT COMPLEX SET-UP OR INSTALLATION FOR THE PURPOSE DESCRIBED ABOVE. THE TECHNOLOGY/CONCEPTS REPRESENTED IN THIS HANDS-ON LAB MAY NOT REPRESENT FULL FEATURE FUNCTIONALITY AND MAY NOT WORK THE WAY A FINAL VERSION MAY WORK. WE ALSO MAY NOT RELEASE A FINAL VERSION OF SUCH FEATURES OR CONCEPTS. YOUR EXPERIENCE WITH USING SUCH FEATURES AND FUNCITONALITY IN A PHYSICAL ENVIRONMENT MAY ALSO BE DIFFERENT.

FEEDBACK. If you give feedback about the technology features, functionality and/or concepts described in this Hands-on Lab to Microsoft, you give to Microsoft, without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft software or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement.

MICROSOFT CORPORATION HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE HANDS-ON LAB, INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. MICROSOFT DOES NOT MAKE ANY ASSURANCES OR REPRESENTATIONS WITH REGARD TO THE ACCURACY OF THE RESULTS, OUTPUT THAT DERIVES FROM USE OF THE VIRTUAL LAB, OR SUITABILITY OF THE INFORMATION CONTAINED IN THE VIRTUAL LAB FOR ANY PURPOSE.

DISCLAIMER

This lab contains only a portion of the features and enhancements in Microsoft Azure. Some of the features might change in future releases of the product.