



Streaming Analytics using Power BI And Databricks

Summary: Creating real-time analytical reports using Power BI push dataset and Spark structured streaming in Azure for cycle time analysis for manufacturing industries.

Contents

1. Introduction	3
1.1 Overview	3
1.2 Problem Statement	3
1.3 Solution Approach	3
1.4 Storage	4
2 Streaming Analytics Architecture and Technical Components	5
2.1 Architecture Diagram and Flow	5
2.2 Technical Components	5
2.3 Sources	6
3 Power BI Streaming Datasets Overview	7
3.1 Streaming Capability in Power BI	7
3.2 Types of real-time Datasets in Power BI	7
3.3 Streaming Data Matrix	9
3.4 Inference	9
3 Creating Power BI Push Dataset	10
4 Create Spark Streaming Jobs with Databricks	15
5.1 Setup	15
5.2 Creating Spark Streaming Jobs	16
5 Connecting and Creating Insights with Power BI Streaming Dataset	20
6 Limitations and Consideration	26
7 Reference	27

1. Introduction

1.1 Overview

In this whitepaper we will address the issue of creating real-time KPIs using streaming data source. In manufacturing industries data would be generated via multiple sensors and making real-time analytics out of this data would add value to business and time critical insights would be delivered to the consumers much faster.

1.2 Problem Statement

Deriving insights from near real time streaming data can be very challenging specially when we have multiple streaming sources which require integration. Creating and refreshing KPIs from such disparate sources cannot be achieved traditional reporting methods.

1.3 Solution Approach

- We have used Azure Databricks and Spark structured streaming API to process the streaming data from Eventhub.
- There are two spark streaming jobs created
 - First job will store all the data in a delta table
 - Second job will filter the stream with condition Eventname='PartProcessed' and looks up the delta table for the corresponding "PartReceived" event of the unique part.
 - The cycle time is the difference in time between partreceived and partprocessed events for the unique part.
- Once cycle time is calculated, We will validate is the cycle time for the current unique part is above or below the standard cycle time of the part
- Once we join the master tables and consolidate all the KPIs such as cycletime and cycletimeAboveThreshold,we push the data to powerBI push dataset.

Can It done using Synapse or Stream Analytics ?

- The same can be achieved with Azure synapse analytics also but data bricks lets you integrate seamlessly with ADLS Gen2 and it supports delta tables which we can use to creating a lake house for reporting. Azure synapse uses open-source delta whereas Databricks Delta offers some additional optimizations.
- Even though Azure Stream analytics has similiar stream processing capabilities ,complex transformations cannot be handled with ease as it is with Databricks because of ADB's multiple language support(python,scala,sql etc)
Azure synapse can be used to process data streams but Databrciks has a proprietary data processing engine (Databricks Runtime) built on a highly optimized version of Apache Spark offering 50x performance

1.4 Storage

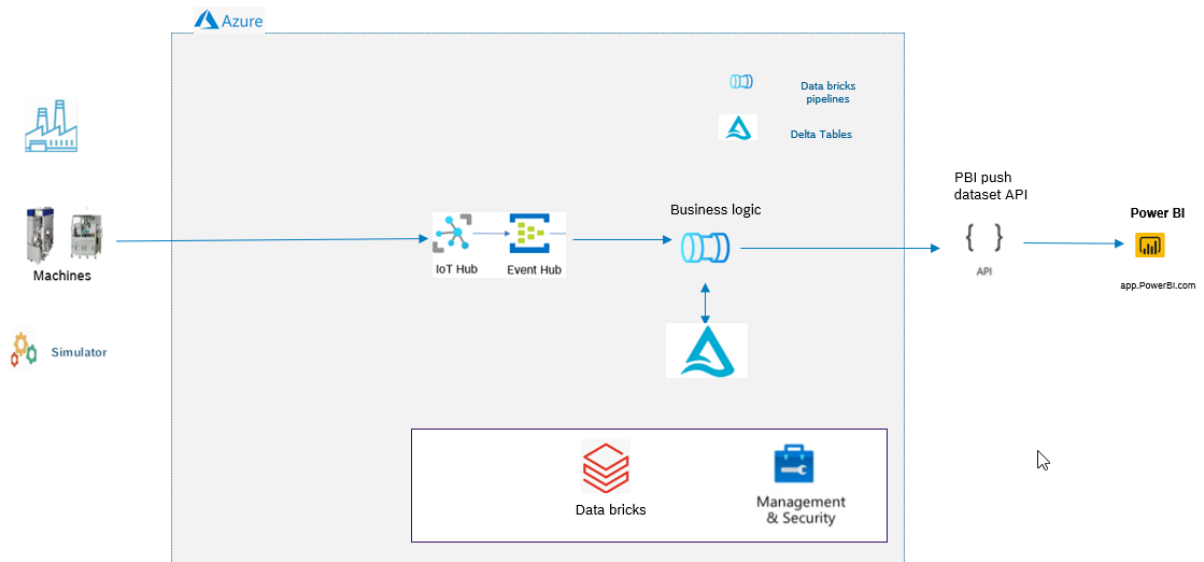
- Delta table is used to store the stream of data coming from eventhub.

This delta table is created in ADLS gen2 and accessed by Databricks using mount points

- Power BI Push Dataset is used for storing stream data to be consumed in Power BI report

2 Streaming Analytics Architecture and Technical Components

2.1 Architecture Diagram and Flow



2.2 Technical Components

#	Category	Technical Component	Purpose
1	Data Storage	Azure DataLake Gen 2	Azure Data Lake Storage Gen2 is a set of capabilities dedicated to big data analytics, built on Azure Blob Storage. Data Lake Storage Gen2 converges the capabilities of Azure Data Lake Storage Gen1 with Azure Blob Storage.
2	Data Processing	Azure Databricks	<i>Azure Databricks</i> is a fast, easy, and collaborative Apache Spark-based big data analytics service designed for data science and data engineering.
3	Data Ingestion	Azure IoT Hub	Azure IoT Hub is a managed service hosted in the cloud that acts as a central message hub for communication between an IoT application and its attached devices.
4	Data Format	Databricks Delta	Databricks Delta is a single data management tool that combines the scale of a data lake, the reliability and performance of a data warehouse, and the low latency of streaming in a single system for the first time.
5	Data Visualization	Power BI	Power BI is an interactive data visualization software product developed by Microsoft with primary focus on business intelligence. It is part of the Microsoft Power Platform
6	Scheduling & Orchestration	Azure Data Factory	Azure Data Factory is Azure's cloud ETL service for scale-out serverless data integration and data transformation.
7	Security	Key Vault	Azure Key Vault is a cloud service for securely storing and accessing secrets.

2.3 Sources

- **Event Data:**

This data is produced by sensors placed on the shopfloor at each station. These data will have information about

Plant

Line

Station

Eventname

- This data is routed through IoT Hub and eventhub.
- For the streaming jobs that we create, Eventhub will be the source

- **Master Data**

We need to look up the masterdata to get information such as

Station

Line

Shift

StandardCycleTime

The above data are stored in datalake as csv files.

```
{ "EnqueuedTimeUtc": "2022-02-11T09:56:56.21500",
  "EventId": "111252",
  "Identifier": "238",
  "EventName": "PARTPROCESSED",
  "timestamp": "2/11/2022 3:26:54 PM",
  "timezone": "IST",
  "plant": "ADG",
  "line_no": "12",
  "station_no": "15",
  "process_no": "238",
  "processname": "110",
  "batch_no": "BX22112",
  "part_no": "281013778",
  "product_family": "EDC16C39",
  "ManCycleTimeLoading": "61",
  "ManCycleTimeUnloading": "66",
  "DMC": "1000055625.5",
  "Result_State": "1"
}
```

3 Power BI Streaming Datasets Overview

3.1 Streaming Capability in Power BI

Power BI with real streaming data lets you create visual on dashboard which are updated in real-time. The source of these real-time data can be anything such as factory sensors, social media sources, service usage metrics, or many other time-sensitive data collectors or transmitters.

We will discuss how we can setup real time visual in Power BI using real-time datasets. These visuals can be a tile or a live report on a dashboard so that they will be refreshed automatically and in real-time.

3.2 Types of real-time Datasets in Power BI

Power BI offers 3 types of real-time datasets which used to design real-time visual on dashboards. There are following:

Push dataset

Streaming dataset

PubNub streaming dataset

Push Dataset: With a **push dataset**, data is pushed into the Power BI service. When the dataset is created, the Power BI service automatically creates a new database in the service to store the data. Since there is an underlying database that continues to store the data as it comes in, reports can be created with the data. These reports and their visuals are just like any other report visuals, which means you can use all of Power BI's report building features to create visuals, including Power BI visuals, data alerts, pinned dashboard tiles, and more.

Once a report is created using the push dataset, any of its visuals can be pinned to a dashboard. On that dashboard, visuals update in real-time whenever the data is updated. Within the service, the dashboard is triggering a tile refresh every time new data is received.

There are two considerations to note about pinned tiles from a push dataset:

- Pinning an entire report using the *pin live page* option will **not** result in the data automatically being updated.
- Once a visual is pinned to a dashboard, you can use **Q&A** to ask questions of the push dataset in natural language. Once you make a **Q&A** query, you can pin the resulting visual back to the dashboard, and that dashboard will *also* update in real-time.

Streaming dataset

With a **streaming dataset**, data is also pushed into the Power BI service, with an important difference: Power BI only stores the data into a temporary cache, which quickly expires. The temporary cache is only used to display visuals, which have some transient sense of history, such as a line chart that has a time window of one hour.

With a **streaming dataset**, there is *no* underlying database, so you *cannot* build report visuals using the data that flows in from the stream. As such, you cannot make use of report functionality such as filtering, Power BI visuals, and other report functions.

The only way to visualize a streaming dataset is to add a tile and use the streaming dataset as a **custom streaming data** source. The custom streaming tiles that are based on a **streaming dataset** are optimized for quickly displaying real-time data. There is little latency between when the data is pushed into the Power BI service and when the visual is updated, since there's no need for the data to be entered into or read from a database.

In practice, streaming datasets and their accompanying streaming visuals are best used in situations when it is critical to minimize the latency between when data is pushed and when it is visualized. In addition, it's best practice to have the data pushed in a format that can be visualized as-is, without any additional aggregations. Examples of data that's ready as-is include temperatures, and pre-calculated averages.

PubNub streaming dataset

With a **PubNub** streaming dataset, the Power BI web client uses the PubNub SDK to read an existing PubNub data stream. No data is stored by the Power BI service. Because this call is made from the web client directly, you would have to list traffic to PubNub as allowed.

As with the **streaming dataset**, with the **PubNub streaming dataset** there is no underlying database in Power BI, so you cannot build report visuals against the data that flows in, and cannot take advantage of report functionality such as filtering, Power BI visuals, and so on. As such, the **PubNub streaming dataset** can also only be visualized by adding a tile to the dashboard, and configuring a PubNub data stream as the source.

Tiles based on a **PubNub streaming dataset** are optimized for quickly displaying real-time data. Since Power BI is directly connected to the PubNub data stream, there is little latency between when the data is pushed into the Power BI service and when the visual is updated.

3.3 Streaming Data Matrix

Capability	Push	Streaming	PubNub
Dashboard tiles update in real-time as data is pushed in	Yes. For visuals built via reports and then pinned to dashboard.	Yes. For custom streaming tiles added directly to the dashboard.	Yes. For custom streaming tiles added directly to the dashboard.
Dashboard tiles update with smooth animations	No.	Yes.	Yes.
Data stored permanently in Power BI for historic analysis	Yes.	No. Data is temporarily stored for one hour to render visuals.	No.
Build Power BI report atop data	Yes.	No.	No.
Max rate of data ingestion	1 requests/s 16 MB/request	5 requests/s 15 KB/request	N/A Data is not being pushed into Power BI
Limits on data throughput	1 M rows/hour	None.	N/A Data is not being pushed into Power BI

3.4 Inference

Now that we have understood the various options of real-time streaming in Power BI and real-time datasets available in Power BI, in the next section we will create a Push

dataset in Power BI which we will use to push data via spark streaming jobs in Databricks for the purpose of this whitepaper

How tableau does it?

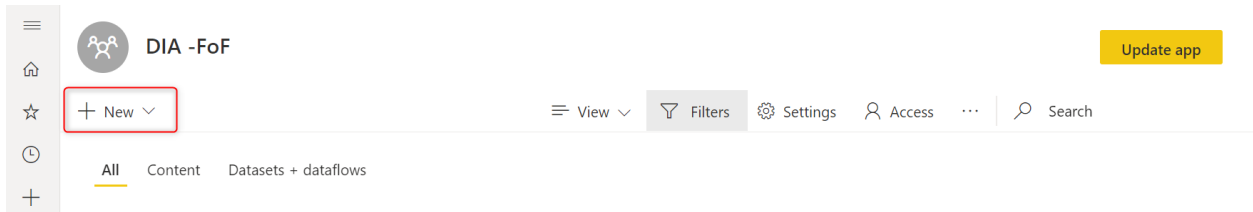
Tableau is not a natively live streaming tool. So, if you want to perform Tableau Real Time Data Streaming, you have to write a script or code that will perform the required operations for you. In real time streaming, the data is supposed to be refreshed without the involvement of the user.

3 Creating Power BI Push Dataset

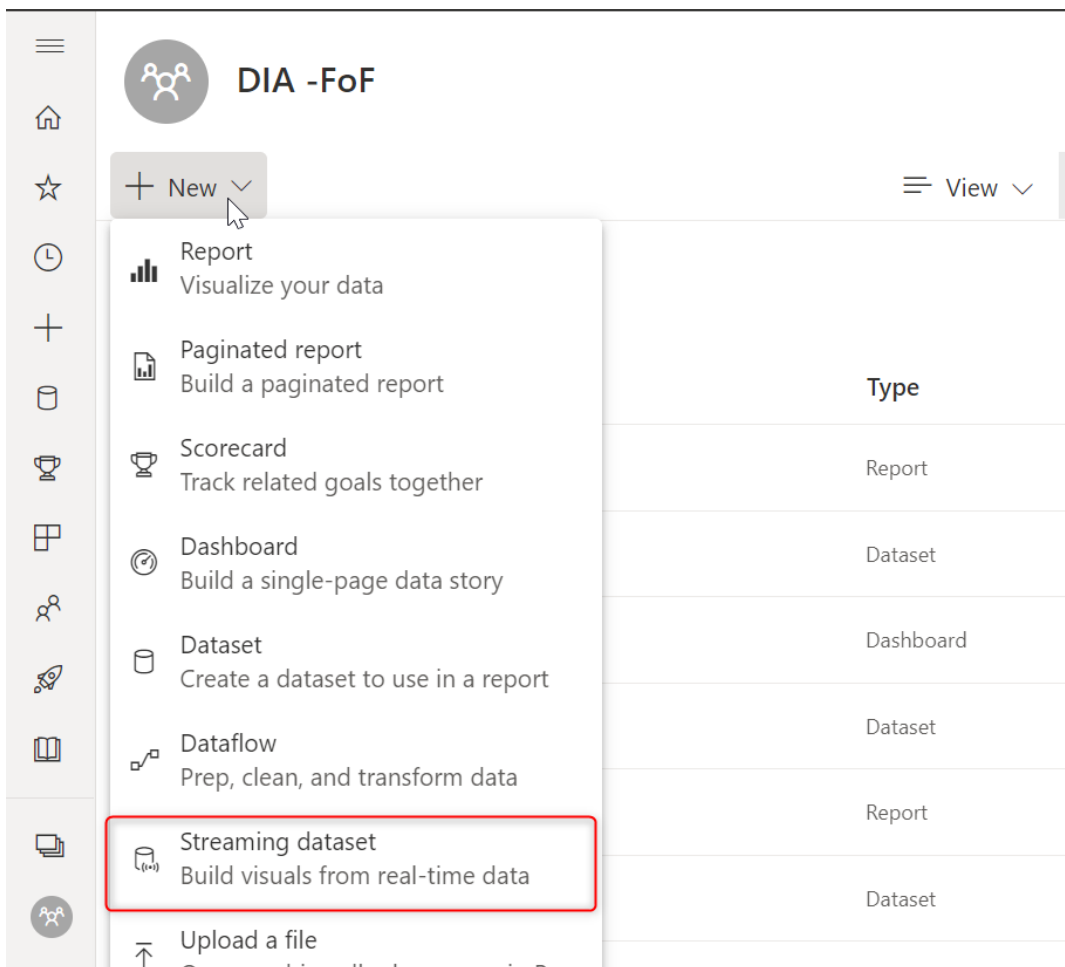
To Create a streaming push dataset in Power BI follow the given steps:

Step 1: Go to the Power BI workspace in which you want create you streaming push dataset

Step 2: Click on the New button



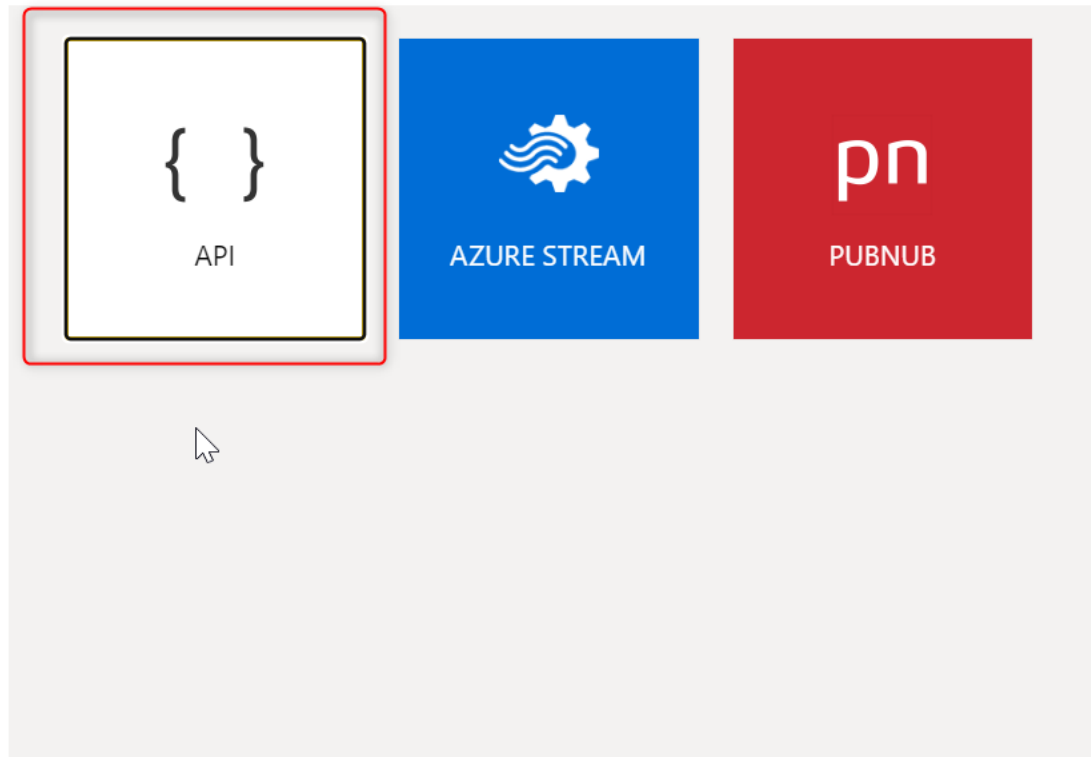
Step 3: Click on streaming datasets



Step 4: Select API and click next to create push dataset

New streaming dataset

Choose the source of your data



This will create a streaming dataset with an endpoint url which can be used to push data into the dataset.

Step 5: Define your datasets and switch on Historical data to store some history of data if not switched of dataset will store only the latest pushed data. Historically Push dataset by default stores max of 200000 rows with FIFO logic. Which can be changed to store up to 500000 rows.

New streaming dataset

Create a streaming dataset and integrate our API into your device or application to send data. [Learn more about the API.](#)

* Required

Dataset name *

Values from stream *

Historic data analysis

On

Back

Create

Cancel

Following is a sample dataset for cycle time analysis for a manufacturing industry.

```
[
{
  "DMC" : "AAAAA555555",
  "EventName" : "AAAAA555555",
  "Eventid" : "AAAAA555555",
  "ManCycleTimeLoading" : 98.6,
  "ManCycleTimeUnloading" : 98.6,
  "Result_State" : "AAAAA555555",
  "Identifier" : "AAAAA555555",
```

```

"batch_no" : "AAAAA555555",
"line_no" : "AAAAA555555",
"part_no" : "AAAAA555555",
"plant" : "AAAAA555555",
"process_no" : "AAAAA555555",
"processname" : "AAAAA555555",
"product_family" : "AAAAA555555",
"station_no" : "AAAAA555555",
"timestamp" : "2022-04-20T15:06:31.434Z",
"timezone" : "AAAAA555555",
"cycleTime" : 98.6,
"timeprocessed" : "2022-04-20T15:06:31.434Z",
"station_name" : "AAAAA555555",
"line_name" : "AAAAA555555",
"StandardCycleTime" : 98.6,
"ct_It_standard" : "AAAAA555555",
"delta_ct" : 98.6,
"Shift" : "AAAAA555555"
}
]

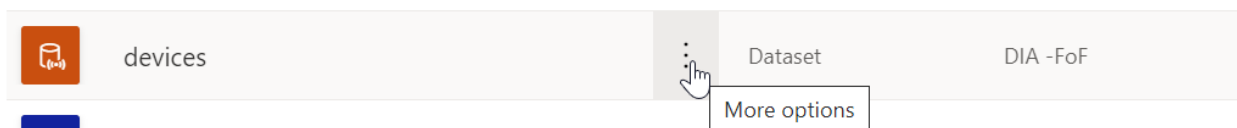
```

Step 6: Click on create to create the dataset

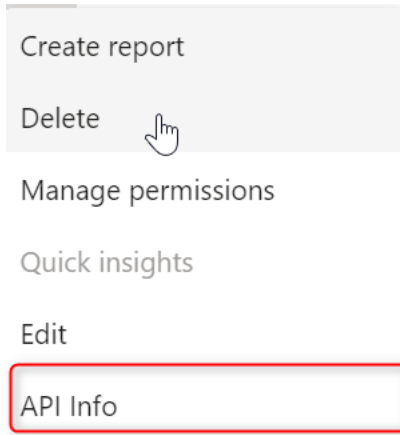
Now that we have created the streaming push dataset in Power BI, we go and find out end point URL for this dataset so that we can push data in the following section in Databricks.

Step 7: Got to workspace find the dataset which you have created

Step 8: Click on more option of that dataset.



Step 9: Click on API info



Step 10: Copy the push URL highlighted below which will be used for pushing the data from Databricks.

API info on devices

Use the API endpoint URL and one of the examples shown below to send data to your streaming dataset. For more information, [read our API documentation and integration guide](#).

Push URL

https://api.powerbi.com/beta/

Raw

cURL

PowerShell

```
[
  {
    "DMC" : "AAAAA55555",
    "EventName" : "AAAAA55555",
    "Eventid" : "AAAAA55555",
    "ManCycleTimeLoading" : 98.6,
    "ManCycleTimeUnloading" : 98.6,
    "Result_State" : "AAAAA55555",
    "Identifier" : "AAAAA55555",
    "batch no" : "AAAAA55555"
  }
]
```

Done

4 Create Spark Streaming Jobs with Databricks

5.1 Setup

1. Mounting ADLS Gen2 into databricks

Mounting the data lake in Azure Databricks

Steps involved:

- a. - Create a secret scope in azure databricks that is backed by an azure key vault instance.
- b. - Setup permissions (storage blob data contributor) for your service principal on the data lake account.
- c. - Store credentials necessary for your service principal in a key vault.
- d. - Build a function to mount your data lake.

```
adlsAccountName = '...'
adlsContainerName = "landing"
adlsFolderName = ""
#mountpoint is the alias for the ADLS container/folder
mountPoint = "/mnt/landing"
# Application (Client) ID
applicationId = dbutils.secrets.get(scope='...',key="ClientId")

# Application (Client) Secret Key
authenticationKey = dbutils.secrets.get(scope='...',key="ClientSecret")

# Directory (Tenant) ID
tenantId = dbutils.secrets.get(scope='...',key="TenantId")

endpoint = "https://login.microsoftonline.com/" + tenantId + "/oauth2/token"
source = "abfss://" + adlsContainerName + "@" + adlsAccountName + ".dfs.core.windows.net/" + adlsFolderName

# Connecting using Service Principal secrets and OAuth
configs = {"fs.azure.account.auth.type": "OAuth",
          "fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
          "fs.azure.account.oauth2.client.id": applicationId,
          "fs.azure.account.oauth2.client.secret": authenticationKey,
          "fs.azure.account.oauth2.client.endpoint": endpoint}

# Mounting ADLS Storage to DBFS
# Mount only if the directory is not already mounted
#if not any(mount.mountPoint == mountPoint for mount in dbutils.fs.mounts()):
dbutils.fs.mount(
source = source,
mount_point = mountPoint,
extra_configs = configs)
```

Why Mounting is necessary?

By mounting your data lake

1. you can seamlessly access data without requiring credentials.
2. Allows you to interact with object storage using directory and file semantics instead of storage URLs.

2. Creating delta tables

Steps involved

1. Import Required Libraries
2. Retrieve the source json definition
3. Create Delta table for storing the data stream

```
Create the delta table Python ▶ ▾ - ✕  
1 table_name='productionevents'  
2 save_path='/mnt/landing/productionevents'  
3 df.write.format("delta").mode("overwrite").save(save_path)  
4 spark.sql("CREATE TABLE " + table_name + " USING DELTA LOCATION '" + save_path + "'")  
5
```

Why delta tables

1. Delta Tables store the data as parquet, but it just has an additional layer over it with advanced features, providing history of events, and more flexibility on changing the content like, update, delete and merge capabilities.
2. Delta automatically versions the big data that you store in your data lake, and you can access any historical version of that data.
3. Merging as in databases
4. Schema evolution

5.2 Creating Spark Streaming Jobs

Creating Spark streaming jobs

1. Import Required Libraries
2. Retrieve the Schema from the model file

A model file which has the correct schema is stored in datalake. This file is used to retrieve the schema which will be passed to spark.readstream api.

3. Read master data

Master data such as Shift, line, station, Standardcycletime are read and stored as Dataframes

4. Create UDF for calculating shift

UDF:

UDF can be used to perform data transformation operations which are not already present in Pyspark built-in functionality

UDF can be thought of as an alternate to for-loops since they are much faster due to parallel processing unlike for-loops which performs step-by-step iteration.

Broadcasting

Broadcast variables allow the programmer to keep a read-only variable cached on each machine rather than shipping a copy of it with tasks. They can be used, for example, to give every node a copy of a large input dataset in an efficient manner. Spark also attempts to distribute broadcast variables using efficient broadcast algorithms to reduce communication cost.

5. Create a function for posting the processed data stream to Power BI's push data set

This function is used to push the processed the data to push dataset in powerBI. Data can be pushed to Power BI dataset's in realtime using REST APIs.

6-Read and process the data from streaming source

Intialize the connection string and other necessary configurations

```
1 #should ideally come from key vault
2 connectionString="
3 #connectionString = "Endpoint=sb://streaminp.servicebus.windows.net/;SharedAccessKeyName=PreviewDataPolicy;
4 SharedAccessKey=SuWdG4u794ENfSg0CeY9qUr4YwKuKwJA17JKTRco10o;EntityPath=deviceData"
5 ehConf = {
6     "eventhubs.connectionString" : connectionString,
7     "ehName":"testeventhub"
8 }
9 #the connection string should be encrypted
10 ehConf['eventhubs.connectionString'] = sc._jvm.org.apache.spark.eventhubs.EventHubsUtils.encrypt(connectionString)
11 ehConf['eventhubs.consumerGroup'] = "$Default"
```

7-Write the data stream to delta table sink

use "readstream" to read the data from event hub

```
1 #format is "eventhubs"
2 #pass the configuration dictionary to options
3 #add a new column by invoking the udf created earlier and pass plant and time as input to it
4 df_stream_in = spark \
5     .readStream \
6     .format("eventhubs") \
7     .options(**ehConf) \
8     .schema(dataset_schema) \
9     .load().withColumn('Body',f.from_json(f.col('Body').cast('string'),body_schema))\
10     .withColumn('timestamp2',f.from_unixtime(f.unix_timestamp(f.col('Body.timestamp')), "M/d/yyyy hh:mm:ss a"), "yyyy-MM-dd'T'HH:mm:ss"))\
11     .withColumn('timeonly',f.col('timestamp2').substr(12,8)) \
12     .withColumn('Shift',udf(f.col('timeonly'),f.col('Body.plant'))))
13
14
```

8-Write the data stream to delta table sink

Write the stream to delta table,created during setup

Python ▶ ▼ - ✕

```
1 #output mode is append,which will insert the data into existng delta table
2 #check point location is needed for tracking the progress of the job
3 out_1=df_stream_in_1.writeStream\
4   .format("delta")\
5   .outputMode("append")\
6   .option("checkpointLocation", "/mnt/landing/checkpoints/checkpoint_deltatable/")\
7   .table('productionevents')
```

9-Calculate the cycle time of the current processed part

When the eventname='part processed' we will lookup its corresponding 'part received' using DMC

```
1
2 out_2=df_stream_in_1.alias('a').withColumn('DMC',f.when(f.col('DMC').contains('.'),f.expr("substring(DMC,0,length(DMC)-2)").otherwise(f.col('DMC'))
   ).filter("a.EventNames='PARTPROCESSED'").)
3 join(deltaTable.withColumn('DMC',f.when(f.col('DMC').contains('.'),f.expr("substring(DMC,0,length(DMC)-2)").otherwise(f.col('DMC'))).alias('b'),
   ['DMC','line_no','station_no','process_no','batch_no','plant'],'inner')\
4   .select('a.*','DMC',f.col('b.timestamp').alias('timestarted'))\
5   .withColumn('starttime_unix',f.unix_timestamp(f.col('timestarted'), "M/d/yyyy hh:mm:ss a")+f.rand()*10)\
6   .withColumn('processtime_unix',f.unix_timestamp(f.col('timestamp'), "M/d/yyyy hh:mm:ss a"))\
7   .withColumn('cycleTime',f.abs(f.col("processtime_unix") - f.col("starttime_unix")))\
8   .withColumnRenamed('timestamp','timeprocessed')\
9   .withColumnRenamed('timestarted','timestamp')\
10  .drop('starttime_unix')\
11  .drop('processtime_unix')
12
```

▶ out_2: pyspark.sql.dataframe.DataFrame = [line_no: string, station_no: string ... 20 more fields]

10-Check if the cycle time is above the standard cycle time of the station

Merge the output with standard cycle time data

```
1
2 out3=out_2.alias('c1').join(df_standard_cyclotime.alias('c2'),['line_no','station_no','process_no','part_no','plant']).select('c1.*','c2.StandardCycleTime')\
3   .withColumn('StandardCycleTime',f.col('c2.StandardCycleTime').cast('int'))\
4   .withColumn('ct_lt_standard',f.when(f.col('StandardCycleTime')>f.col('c1.cycletime'),'Y').otherwise('N'))\
5   .withColumn('delta_ct',f.col('c1.cycletime')-f.col('StandardCycleTime'))
6
```

11-Push the processed data to power Bi

LMG =>

```
1 out_4=out3.writeStream\
2   .outputMode("append")\
3   .option("checkpointLocation", "/mnt/landing/checkpoints/checkpoint_pbidatset/")\
4   .foreachBatch(fn_post)\
5   .start()
6
7
8
```

```
1
2 def fn_post(ds,id):
3     print('no of rows is',ds.count())
4     print(ds.head(10))
5     df=ds.toPandas()
6     #print(df)
7     df1=df.to_dict(orient='index')
8     print('dict is',df1)
9     ls_of_dct=list(df1.values())
10    # copy "Push URL" from "API Info" in Power BI
11    url =
12    %28%28I:
13
14    # timestamps should be formatted like this
15    time = datetime.strftime(
16        datetime.now(),
17        "%Y-%m-%dT%H:%M:%S.%f"
18    )
19
20    # data dict must be contained in a list
21
22    # post/push data to the streaming API
23    headers = {
24        "Content-Type": "application/json"
25    }
26
27    try:
28        response = requests.request(
29            method="POST",
30            url=url,
31            headers=headers,
32            data=json.dumps(ls_of_dct)
33        )
34    except Exception as e:
35        print('error is ',e)
36    else:
37        print('succeeded')
38        #print('processed',ls_of_dct)
39        print(response)
```

5 Connecting and Creating Insights with Power BI Streaming Dataset

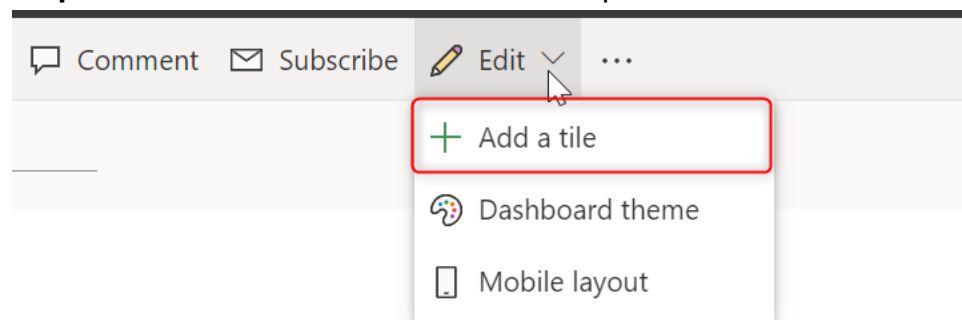
We have created and pushed data into Power BI streaming push dataset, now we will see what are the different ways to consume and create real-time insight out of this Dataset.

There are two ways to create insight using push dataset, they are as following:

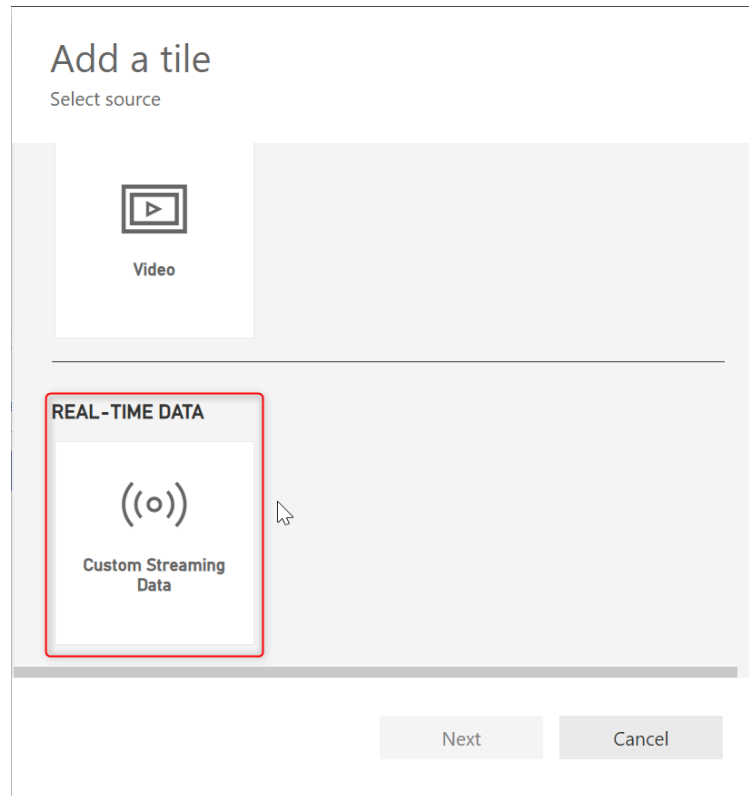
1. **Dashboard Tile:** If we have any streaming dataset then we use that create an insight using a dashboard tile by following steps.

Step 1: Go to dashboard on which want to create a live streaming tile.

Step 2: Click on edit and select Add Tile option



Step 3: Choose real-time data as source and click next.



Step 4: Select your dataset and click next

Add a custom streaming data tile

Choose a streaming dataset

[+ Add streaming dataset](#)

YOUR DATASETS

RealTimePubNubSample

devices

[Manage datasets](#)

[Back](#) [Next](#) [Cancel](#)



Step 5: Select visualization type and add value or axis, click next and then apply.

Add a custom streaming data tile

Choose a streaming dataset > Visualization design

Visualization Type

Card

Fields

+ Add value

[Manage datasets](#)

[Back](#) [Next](#) [Cancel](#)

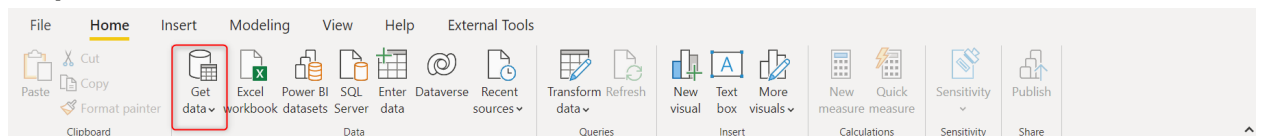
2. **Creating a report using push dataset:** Now we will see how we can import the streaming dataset in Power BI desktop and create a report out of it.

Please note that this report has to be pinned live on any dashboard to make it real-time and auto refresh.

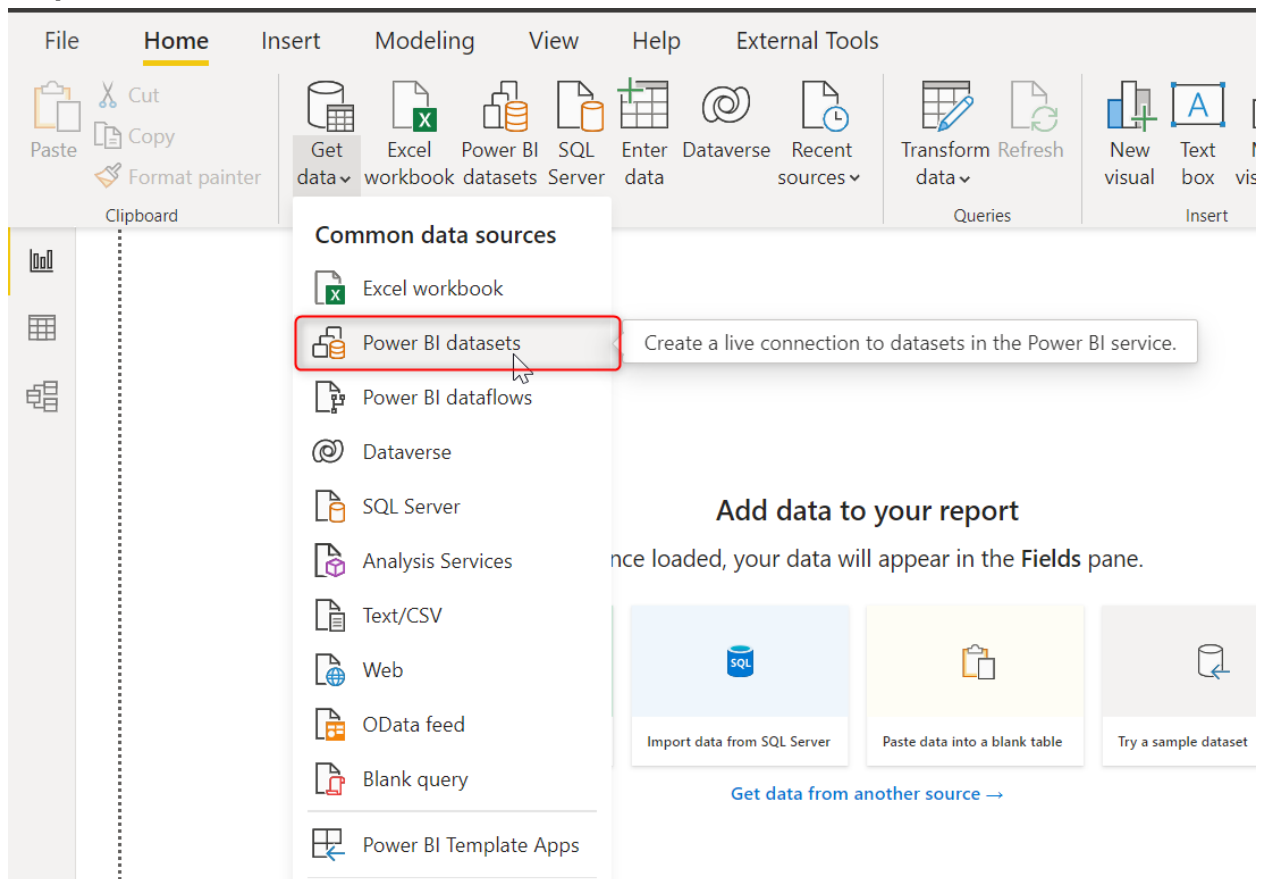
To connect to real-time dataset in power BI desktop follow the given steps:

Step 1: Make sure you are signed in with the Power BI service account which have access to the streaming dataset workspace.

Step 2: Click on Get Data



Step 3: Click on Power BI Dataset



Step 4: Select the dataset (Devices) and click on create.

Select a dataset to create a report

All Recent My datasets

Name	Endorsement	Owner	Workspace	Refreshed	Sensitivity
devices	-	Ignacio...	Power BI...	3/16/22, 12:43:11 PM	-
...
...
...
...
...
...

Step 5: This will import the data in Power BI desktop with live connection, you can use this to create insights and upload your report back to Power BI Service for the users.

File Home Insert Modeling View Help External Tools

Clipboard Data Queries Insert Calculations Sensitivity Share

Build visuals with your data
Select or drag fields from the Fields pane onto the report canvas.

Visualizations Fields

Build visual

Filters

Search
RealTimeData

Values
Add data fields here

Drill through
Cross-report Off
Keep all filters On
Add drill-through fields here

Page 1

Connected live to the Power BI dataset: devices in

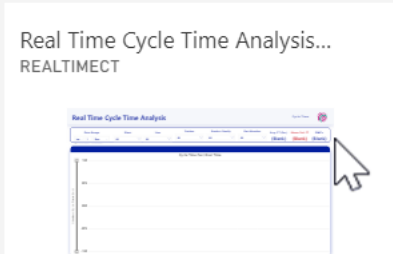
Below is an example of report prepared for real-time cycle time Analysis



Step 6: Pin live page to dashboard

Step 7: Click on Pin live, the report will be added to dashboard

Preview: Last saved state



Pin to dashboard

Select an existing dashboard or create a new one.

Where would you like to pin to?

- Existing dashboard
- New dashboard

Select existing dashboard

Cycle Time Analysis.pbix

i Pin live page enables changes to reports to appear in the dashboard tile when the page is refreshed.

Pin live Cancel

6 Limitations and Consideration

1. This API call only supports push datasets.
2. 75 max columns
3. 75 max tables
4. 10,000 max rows per single POST rows request
5. 1,000,000 rows added per hour per dataset
6. 5 max pending POST rows requests per dataset
7. 120 POST rows requests per minute per dataset
8. If table has 250,000 or more rows, 120 POST rows requests per hour per dataset
9. 200,000 max rows stored per table in FIFO dataset
10. 5,000,000 max rows stored per table in 'none retention policy' dataset
11. 4,000 characters per value for string column in POST rows operation

7 Reference

<https://docs.microsoft.com/en-us/power-bi/connect-data/service-real-time-streaming>

<https://docs.microsoft.com/en-us/rest/api/power-bi/push-datasets>

<https://docs.microsoft.com/en-us/rest/api/power-bi/>

<https://docs.databricks.com/data/data-sources/azure/adls-gen2/azure-datalake-gen2-sp-access.html>

<https://docs.databricks.com/spark/latest/structured-streaming/index.html>