

Real Time Data Streaming Simulator

Project Overview

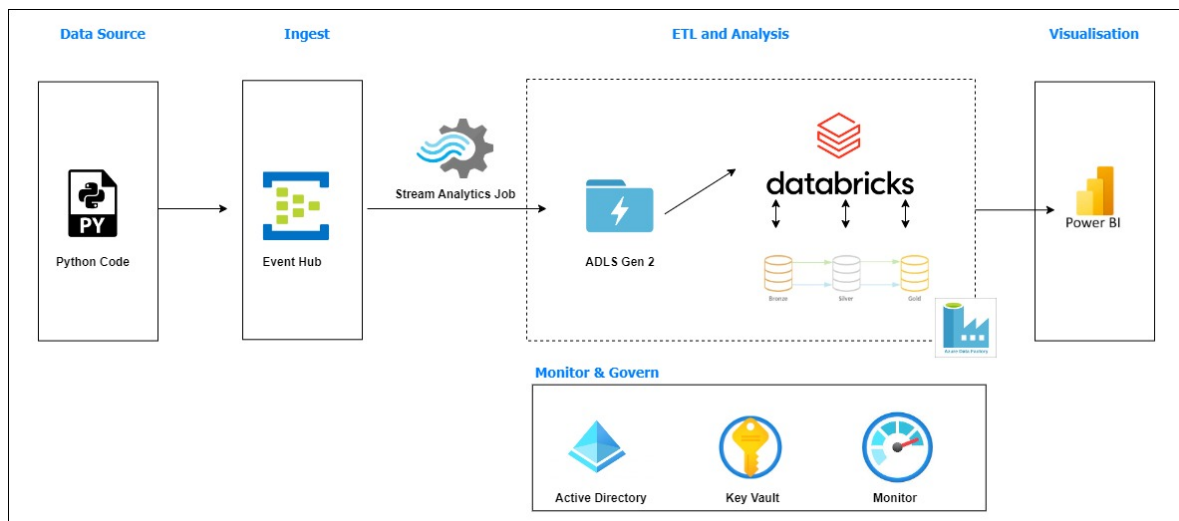
In this project we are going to build an End to End Data Engineering pipeline which simulates a real time data processing.

1. Problem statement

- Create a simulator code using python which generates the data. Using Azure services load the data, do transformations using Databricks and store it into a delta gold table.

2. High level description of the solution

- We have a device which generates the data at particular intervals(assume for every 5 minutes).
- Push this data into Kafka service (Azure EventHub)
- Load the data into ADLS using Stream Analytics job.
- When ever data is available in the ADLS, using ADF load the data from ADLS to Azure Databricks. Do the transformations.
- Connect the Power BI to Databricks for visualization. 📊



Data Collection and Ingestion

1. Data source

- Data source is a python code, where we create a Data frame using pandas by loading .csv file
 - **Details of the CSV file**
 - **Type of Data** : Amazon Product ratings
 - **Size** : 1GB
 - **No: columns** : 4 (UserId, ProductId, Rating, Timestamp)
- From the data frame, we are trying to push 1000 records every 3 minute into the Azure Eventhub.
- Below is the Python code

```
'''
Install the below libraries which are helpful to connect with azure eventhub
1. pip install azure-eventhub
```

```

2. pip install azure-identity
3. pip install aiohttp

'''
# import required modules/libraries
import pandas as pd

import asyncio
import pandas as pd
import json
import time

from azure.eventhub import EventData
from azure.eventhub.aio import EventHubProducerClient
from azure.identity import DefaultAzureCredential

connection_str = 'Endpoint=sb://datasimulatornamespace.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=
eventhub_name = 'datasimulatoreventhub'

#data file path
path = "C:\\Users\\RakeshAnkinapalli\\Downloads\\ratings_Beauty\\ratings_Beauty.csv"

#create dataframe using the given path
data = pd.read_csv(filepath_or_buffer=path)

#loop to get 1000 records every minute.
idx = 0
def get_record(idx):
    record = data.iloc[idx:(idx+1000)].to_dict('records')
    return record

async def main(idx):
    # Create a producer client to send messages to the event hub.
    # Specify a connection string to your event hubs namespace and
    # the event hub name.
    producer = EventHubProducerClient.from_connection_string(
        conn_str=connection_str, eventhub_name=eventhub_name
    )
    async with producer:
        # Create a batch.
        event_data_batch = await producer.create_batch()

        # Add events to the batch.
        event_data_batch.add(EventData(json.dumps(get_record(idx))))

        # Send the batch of events to the event hub.
        await producer.send_batch(event_data_batch)

#infinite loop runs to call the function which push data into eventhub
#And it sleeps for 1 minute
while True:
    asyncio.run(main(idx))
    time.sleep(60)
    idx += 1000



```

2. Create Resource Group, Storage Account




- While creating storage account enable hierarchical namespace for ADLS

Resources

Recent Favorite

Name	Type
 endtoendprojectssa	Storage account
 projects	Resource group

- Create a container called input data





 **endtoendprojectssa** | Containers   ...

Storage account

<< [+ Container](#) [Change access level](#) [Restore containers](#)

Name
<input type="checkbox"/> \$logs
<input type="checkbox"/> inputdata

Data storage

-  Containers
-  File shares
-  Queues
-  Tables

3. Create Namespace and EventHub

- If you know the fundamentals of Kafka,
 - namespace → Broker
 - eventhub → Topic
- Try to select the location which is near to you.

- We will confront an option called capture while creating eventhub, which is used to load data into blob or ADLS directly from the eventhub.
- To use capture we have to upgrade our plan to standard or premium.

Create Namespace ...

Event Hubs

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Azure for Students ▼

Resource group * projects ▼
[Create new](#)

Instance Details

Enter required settings for this namespace, including a price tier and configuring the number of units (capacity).

Namespace name * datasimulatornamespace ✓
.servicebus.windows.net

Location * Central India ▼
i The region selected supports Availability zones. Your namespace will have Availability Zones enabled. [Learn more.](#)

Pricing tier * Basic (~\$11 USD per TU per Month) ▼
[Browse the available plans and their features](#)

Throughput Units * 1

- Create an eventhub.
- We can have multiple event hubs.
- To access the eventhub from our python code. We need create a shared access policy for the eventhub.
- Click on eventhub. Go to shared access policy and create a new policy by ticking manage check box.
- Here we can get the connection string.

datasimulatoreventhub (datasimulatornamespace/datasimulatoreventhub) | Shared access policies ☆ ...

Event Hubs Instance

Search « + Add

[Diagnose and solve problems](#)

Settings

- Shared access policies**
- Properties
- Locks

Entities

- Consumer groups

Search to filter items...

Policy	Claims
↑ eventhubspolicy	Manage, Send, Listen

4. Create Stream analytics job

- While creating new analytics job, make use you choose the same region as your namespace.
- If not job will fail to pick up the streaming data.

Input source configuration

- Keep authentication mode as Connection string.
- For event hub policy use the one which we created initially.

Input details

datasimulatoreinputsource

[Test](#) [Delete](#) [Open Event Hub](#)

Input alias

datasimulatoreinputsource

☐ Provide Event Hub settings manually

☒ Select Event Hub from your subscriptions

Subscription

Azure for Students

Event Hub namespace * ⓘ

datasimulatorenamespace

Event Hub name * ⓘ

☐ Create new ☒ Use existing

datasimulatoreventhub

Event Hub consumer group * ⓘ

☐ Create new ☒ Use existing

\$Default

Authentication mode
Connection string

Event Hub policy name ⓘ
☐ Create new ☒ Use existing

eventhubsaspolicy

Event Hub policy key
.....

Partition key ⓘ

Event serialization format * ⓘ
JSON

Encoding ⓘ
UTF-8


Event compression type ⓘ


Output source configuration


-

Output details

datasimulatoroutputsource

 Test

 Delete

 Open Blob storage/ADLS Gen2

Output alias

datasimulatoroutputsource

☐ Provide Blob storage/ADLS Gen2 settings manually

☒ Select Blob storage/ADLS Gen2 from your subscriptions

Subscription

Azure for Students

Storage account *

endtoendprojectssa

Container * ⓘ

☐ Create new

☒ Use existing

inputdata

Authentication mode

Connection string

Storage account key

.....

Event serialization format * ⓘ

JSON

Format ⓘ

Line separated

Encoding ⓘ

UTF-8

Write mode * ⓘ

☐ Once after all results for the time partition are available (preview)

☒ Append as results arrive

Path pattern ⓘ

{date}

Date format

DD/MM/YYYY

Time format

HH

Minimum rows ⓘ

100

Maximum time

Hours ⓘ	Minutes	Seconds
0 ✓	3 ✓	0 ✓

5. Copy the Connection string into python code

- Go to eventhub namespace
- Click on shared access policies → click on policy
- Copy the connection string primary key
- Run the python code.
- Once we start the stream analytics job. It will create folder with date as name.

[Upload](#)
[Add Directory](#)
[Refresh](#)
[Rename](#)
[Delete](#)
[Change tier](#)
[Acqui](#)

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

Shared access tokens

Manage ACL

Access policy

Properties

Metadata

Authentication method: Access key ([Switch to Azure AD User Account](#))
Location: inputdata / stream_input_data

Name	Modified	Access tier
<input type="checkbox"/> 06-07-2023		

Data Transformation in Azure Databricks

1.Create Azure Databricks Workspace

Subscription * ⓘ

Azure for Students

Resource group * ⓘ

projects

[Create new](#)

Instance Details

Workspace name *

datasimulatordbricks

Region *

Central India

Pricing Tier * ⓘ

Trial (Premium - 14-Days Free DBUs)

Managed Resource Group name

Enter name for managed resource group

- Create a notebook and cluster
- Attach cluster to the notebook
- Mount the ADLS into notebook.
- Access key:
 - Click on storage account → Under Security + Networking
 - Click on Access keys → Get key 1

```

'''
1.) Mount acts as a pointer to the ADLS. When we mount ADLS at some mount point
we can access the files from ADLS as they are in databricks.
'''
dbutils.fs.mount(
    #wasbs://<container-name>@<storage-account-name>.blob.core.windows.net

```

```

source="wasbs://inputdata@endtoendprojectssa.blob.core.windows.net",
mount_point="/mnt/datasimulator_input_data/",
extra_configs = {

    #Here we are connecting adls using access key
    #fs.azure.account.key.<container-name>.blob.core.windows.net : access key from the security+networking section
    #in storage account
    "fs.azure.account.key.endtoendprojectssa.blob.core.windows.net": "LpT3mcJvEYkPkVHIXsqtdddddQA+gPn0DTkezIy5yeMxI+2PLgZm0X7G05DcjI//3
x+APgKK09r6h+AST8qJXBQ==",
}
)

#import required modules
import datetime
from pyspark.sql.functions import to_date, from_unixtime, year, col, count, expr

'''
Since the data is available in the folder with today's date.
We have to dynamically pass the today date as folder.
'''
date = datetime.date.today().strftime("%d-%m-%Y")
path = f"/mnt/datasimulator_input_data/stream_input_data/{date}/"

#read data into data frame
bronze_data = spark.read.json(path=path)

#Convert timestamp into date and extract year from it
silver_data = bronze_data.select("ProductId", "Rating", "Timestamp", "UserId")\
    .withColumn("Year", year(to_date(from_unixtime(bronze_data.Timestamp))))\
    .drop("Timestamp")

#Do group by and aggregations to get top 5 years which are getting highest 5 ratings.
gold_data = silver_data.where(expr("Rating = 5")).groupBy("Year")\
    .agg(count("Rating").alias("Total_Ratings"))\
    .orderBy(col("Total_Ratings").desc()).limit(5)

#Since we need to visualize the data, create a delta table in default database.
gold_data.write.mode("overwrite").format("delta").saveAsTable("default.gold_table")

#Command to unmount the ADLS.
dbutils.fs.unmount("/mnt/datasimulator_input_data/")

```

Alternate way to connect ADLS to Databricks(Production level use case) using Service Principal.

- It is a more secured way to connect ADLS to Databricks.
- **Step 1:**
 - Go to Azure Active Directory → Under manage section → Click on App Registration → New Registration.
 - Give a name and keep other things default → click on Register
- **Step 2:**
 - Click on the created service principal → Go to Certificates & secrets
 - Click new client secret and make use you copy the secret value
- **Step 3:**
 - Go to Storage account → Click on Access Control (IAM)
 - Click on Add role assignment → Select Storage Blob Data Contributor
 - select your ADLS service principle.
 - Click on Review+Assign

Add role assignment

Role **Members** Conditions (optional) Review + assign

Selected role Storage Blob Data Contributor

Assign access to ☒ User, group, or service principal ☐ Managed identity

Members [+ Select members](#)

Name	Object ID	Type
No members selected		

Description Optional

[Review + assign](#) [Previous](#) [Next](#)

Select

No users, groups, or service principals found.

Selected members:

adls_db_service_principal [Remove](#)

[Select](#) [Close](#)

- Step 4:
 - Create a Azure Key Vault → Under objects click on secrets.
 - Click on Generate → Add the secret value we created in step2.
- Step 5:
 - Creating a secret scope for the key in databricks
 - Add #secrets/createScope at the end of the databricks url.
 - It will display an UI to create a secret scope
 - Fill th details and create a scope.

<https://adb-6736124552009880.0.azuredatabricks.net/?o=6736124552009880#secrets/createScope>

- Go to Databricks note book and add the below code to configure connection

```
service_credential = dbutils.secret.get(scope="<scope-name>",key="<akv-secret-key-name>")

spark.conf.set("fs.azure.account.auth.type.<storage-account>.dfs.core.windows.net", "OAuth")
spark.conf.set("fs.azure.account.oauth.provider.type.<storage-account>.dfs.core.windows.net", "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")
spark.conf.set("fs.azure.account.oauth2.client.id.<storage-account>.dfs.core.windows.net", "application-id")
spark.conf.set("fs.azure.account.oauth2.client.secret.<storage-account>.dfs.core.windows.net", service_credential)

#here directory id is id of your service principle
#Goto Azure Active Directory -> App Registrations -> Owned applications
#click on your service principle -> Directory(tenant) ID
spark.conf.set("fs.azure.account.oauth2.client.endpoint.<storage-account>.dfs.core.windows.net", "https://login.microsoftonline.com/<directory-id>/oauth2/token")

df = spark.read.csv("abfs://<container-name>@<storage-account>.dfs.core.windows.net/<path-to-file>")
```

Connecting Databricks to Power BI

- Download the PowerBI Desktop App or we can directly use in azure as service (PowerBI Embedded)
- Under Home → Click on Get Data → Click on more
- Select Azure → Databricks

Azure Databricks

Server Hostname ⓘ

HTTP Path ⓘ

Example: sql/protocolv1/o/1814582234607533/7508-187377-agent704

▲ Advanced Options (optional)

Default catalog (optional) ⓘ

Example: abc

Database (optional) ⓘ

Example: abc

Automatic Proxy Discovery (optional) ⓘ

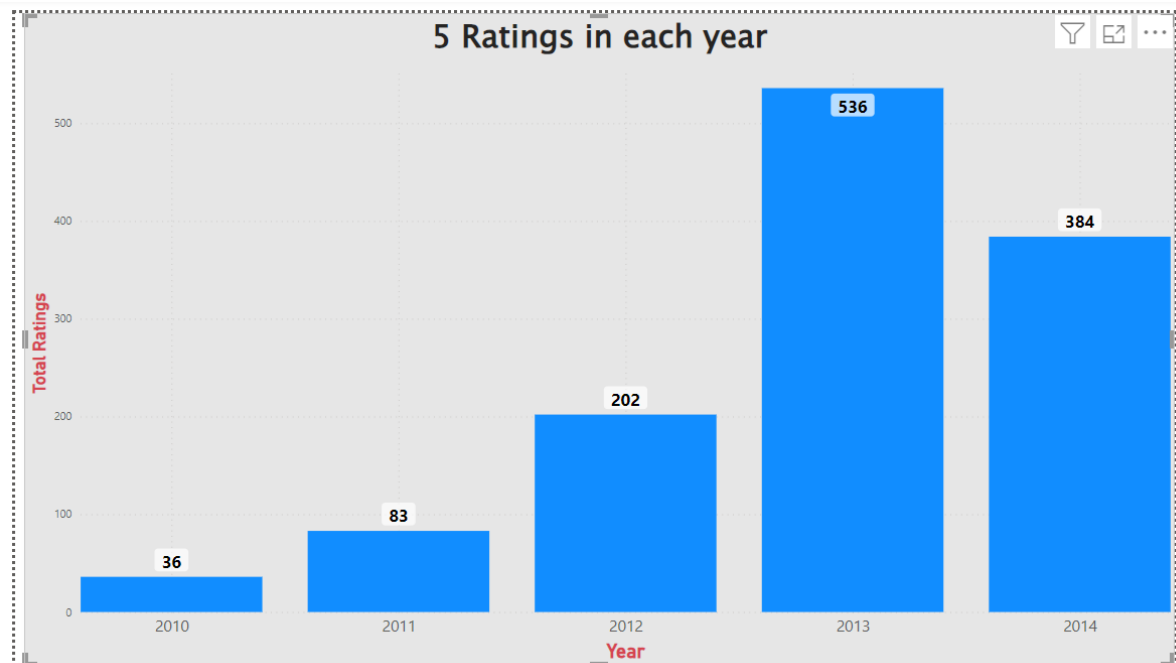
Native query (Requires: Default catalog) (optional) ⓘ

*Example: select * from db.schemaname.tablename*

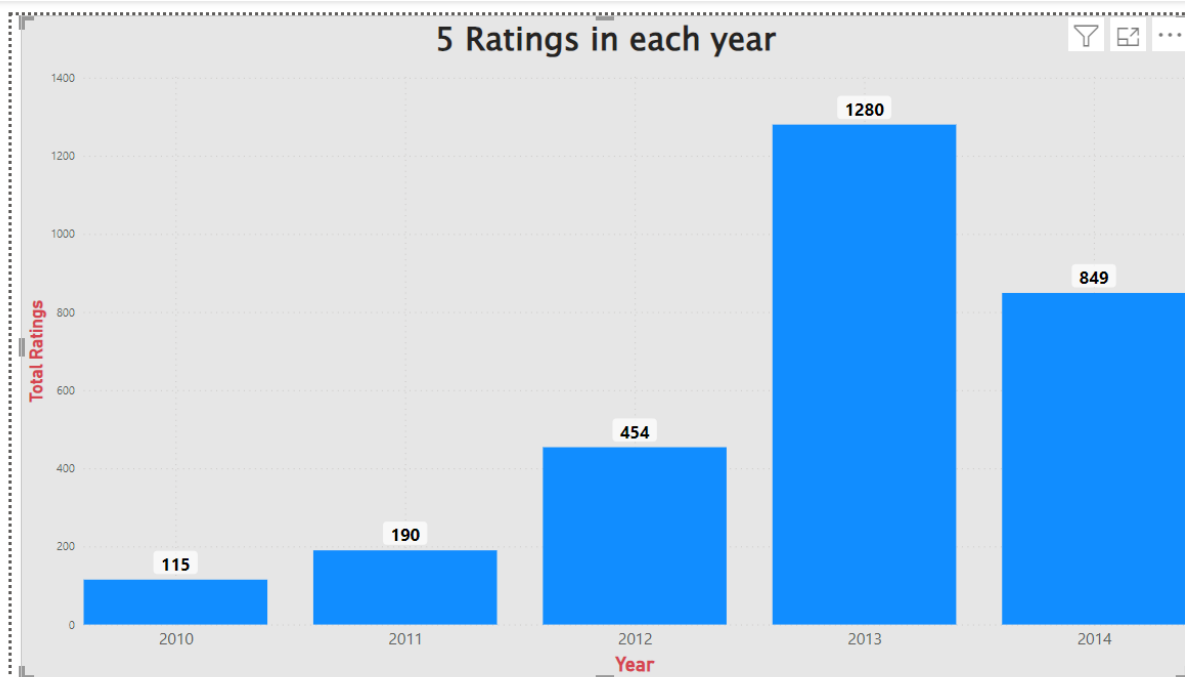
OK

Cancel

- For server Hostname and HTTP path
 - Click on compute → Click on your cluster
 - At bottom of the page under advanced options we can find these details
- Select Personal Access Token for authentication
- To create a Access Token
 - Go to Databricks notebook → Click on Azure databricks username on top right corner
 - On the **Access tokens** tab, click **Generate new token**.
 - Make sure you copy and save it some where.



- After second run, table data will be as show below.



Conclusion

- By running the notebook multiple times. We observed that Year 2013 got highest number of 5 ratings.
- Which indicates the people are happy with the products which are sold in Year 2013.

