Data Engineering in the Cloud

Real-time Data Processing with Azure Databricks

Xuemao Zhang
East Stroudsburg University

January 18, 2025

Outline

- Azure Services Required
- Lab

Azure Services Required

- Data Sources: Streaming data from IoT devices or social media feeds. W use simulated data in Event Hubs.
- Ingestion: Azure Event Hubs for capturing real-time data.
- Processing: Azure Databricks for stream processing using Structured Streaming.
- Storage: Processed data stored Azure Data Lake (Delta Format).
- The project is copied from Real Time Streaming with Azure Databricks and Event Hubs https://youtu.be/pwWIegHgNRw?si=SP1S9r0Z4AZPzGYA

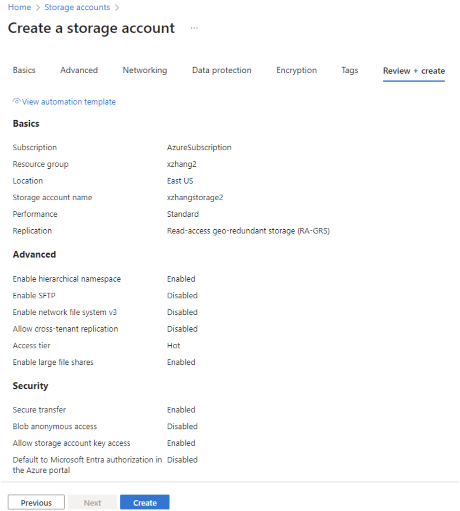
Azure Services Required

Compared to Azure Stream Analytics, Azure Databricks

- Highly scalable, can handle large volumes of data with Spark's distributed computing power.
- Interactive development environment with notebooks for collaborative data analysis and visualization.
    - Flexibility: Supports complex transformations and machine learning tasks using Python, Scala, SQL, and R.
- Advanced Analytics: Suitable for advanced analytics, machine learning, and data science use cases.

Lab

Step 1. Create a datalake (storage account with hierarchical namespace enabled) - To lower the cost, you may choose `Redundancy` as `LRS` - Create a container in the storage account



Basics

Subscription	AzureSubscription
Resource group	xzhang2
Location	East US
Storage account name	xzhangstorage2
Performance	Standard
Replication	Read-access geo-redundant storage (RA-GRS)

Advanced

Enable hierarchical namespace	Enabled
Enable SFTP	Disabled
Enable network file system v3	Disabled
Allow cross-tenant replication	Disabled
Access tier	Hot
Enable large file shares	Enabled

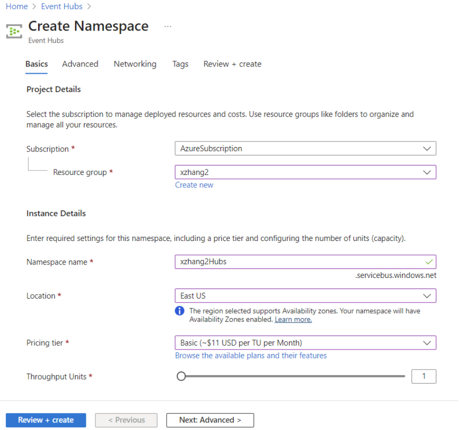
Security

Secure transfer	Enabled
Blob anonymous access	Disabled
Allow storage account key access	Enabled
Default to Microsoft Entra authorization in the Azure portal	Disabled

Lab

Step 2: Search for `Event Hubs` to create an event hub namespace to which the data will be streamed: - Event Hub acts as a message broker for the streaming data.

1. Enter the details shown in the image below and click on the "Review + create" button.



Home > Event Hubs >

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 * AzureSubscription

Resource group * xzhang2

Instance Details

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

Namespace name * xzhang2Hubs ✓

.servicebus.windows.net

Location * East US

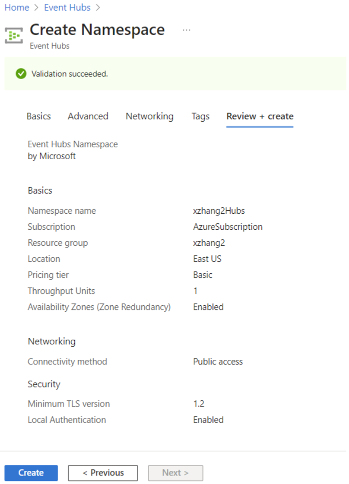
The region selected supports Availability zones. Your namespace will have Availability Zones enabled. Learn more.

Pricing tier * Basic (~$11 USD per TU per Month)

Throughput Units * 1

Lab

2. Validate the configuration and click on the “Create” button.



✓ Validation succeeded.

Event Hubs Namespace
by Microsoft

Basics

Namespace name xzhang2Hubs
Subscription AzureSubscription
Resource group xzhang2
Location East US
Pricing tier Basic
Throughput Units 1
Availability Zones (Zone Redundancy) Enabled

Networking

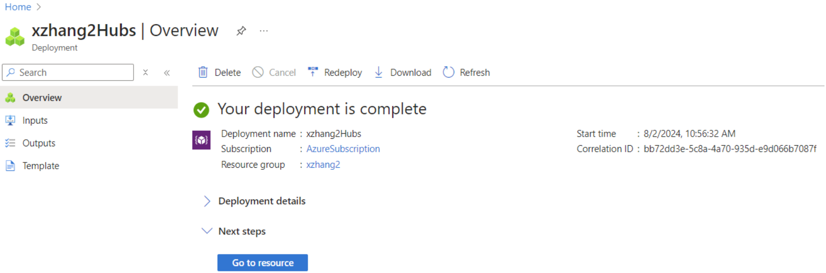
Connectivity method Public access

Security

Minimum TLS version 1.2
Local Authentication Enabled

Lab

3. Once the deployment process is completed, click on the “Go to resource” button.

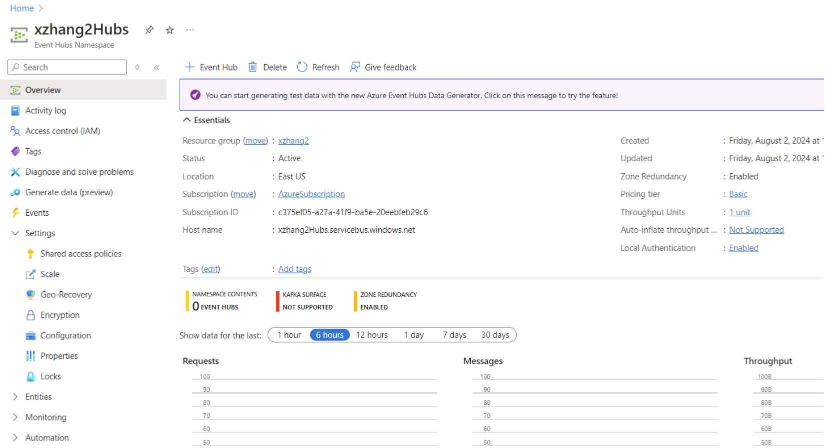


Deployment name : xzhang2Hubs
Subscription : AzureSubscription
Resource group : xzhang2

Start time : 8/2/2024, 10:56:32 AM
Correlation ID : bb72dd3e-5c8a-4a70-935d-e9d066b7087f

Lab

4. The Event Hubs dashboard will be as shown in the image below.



You can start generating test data with the new Azure Event Hubs Data Generator. Click on this message to try the feature!

Essentials

Resource group (move) : xzhang2
Status : Active
Location : East US
Subscription (move) : AzureSubscription
Subscription ID : c375ef05-a27a-41f9-ba5e-20eebfeb29c6
Host name : xzhang2Hubs.servicebus.windows.net

Created : Friday, August 2, 2024 at
Updated : Friday, August 2, 2024 at
Zone Redundancy : Enabled
Pricing tier : Basic
Throughput Units : 1 unit
Auto-inflate throughput ... : Not Supported
Local Authentication : Enabled

Tags (edit) : Add tags

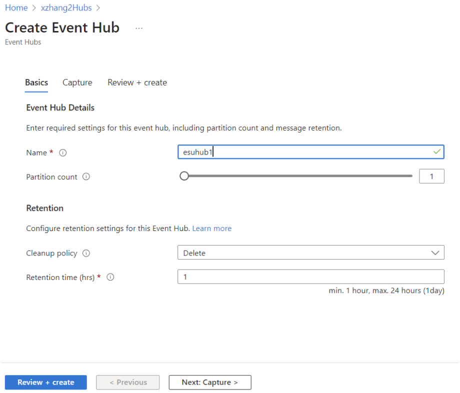
NAMESPACE CONTENTS 0 EVENT HUBS
KAFKA SURFACE NOT SUPPORTED
ZONE REDUNDANCY ENABLED

Show data for the last: 1 hour [✓ 6 hours] 12 hours 1 day 7 days 30 days

Lab

5. click on the "+ Event Hub" button to create an event hub

- Add a name and select retention period as shown in the image below and click on the "Create" button. (Note down the event hub name for future reference)



Home > xzhang2Hubs >

Create Event Hub ...

Event Hubs

Basics   Capture   Review + create

Event Hub Details

Enter required settings for this event hub, including partition count and message retention.

Name * ⓘ esuhub1

Partition count ⓘ 1

Retention

Configure retention settings for this Event Hub. Learn more

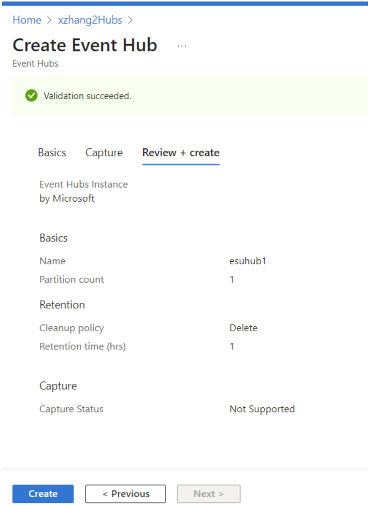
Cleanup policy ⓘ Delete

Retention time (hrs) * ⓘ 1

min. 1 hour, max. 24 hours (1day)

Lab

6. Click Create



Home > xzhang2Hubs >

Create Event Hub

Event Hubs

✓ Validation succeeded.

Event Hubs Instance
by Microsoft

Basics

Name	esuhub1
Partition count	1

Retention

Cleanup policy	Delete
Retention time (hrs)	1

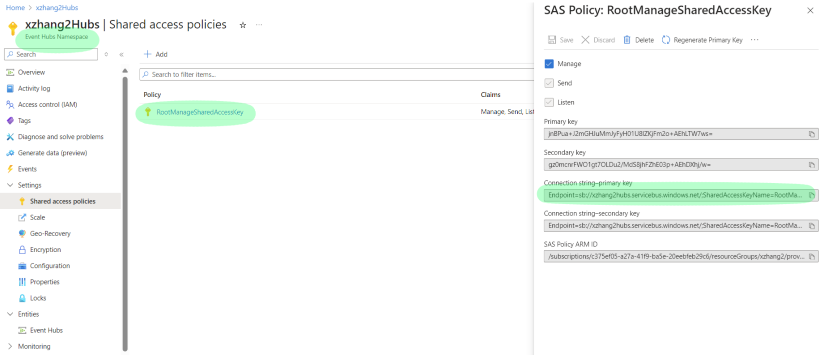
Capture

Capture Status	Not Supported

Lab

7. Go to the `Settings` of the Event Hubs Namespace and go to Shared access policies.

   - Update the policy if necessay



Policy	Claims
RootManageSharedAccessKey	Manage, Send, List

SAS Policy: RootManageSharedAccessKey

☑ Manage

☑ Send

☑ Listen

Primary key: jnBPua+J2mGHJuMmJyFyH01U8IZKjFm2o+AEhLTW7ws=

Secondary key: gz0mcnrFWO1gt7OLDu2/MdS8jhFZhE03p+AEhDXhj/w=

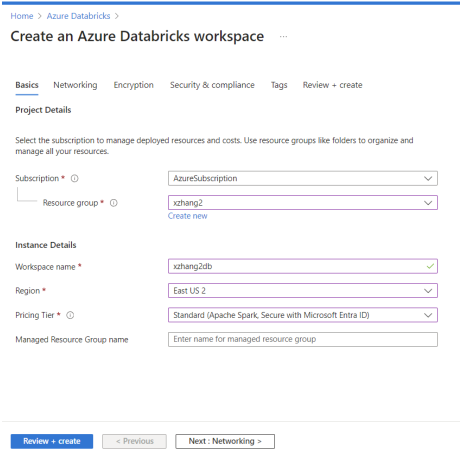
Connection string–primary key: Endpoint=sb://xzhang2hubs.servicebus.windows.net/;SharedAccessKeyName=RootMa...

Connection string–secondary key: Endpoint=sb://xzhang2hubs.servicebus.windows.net/;SharedAccessKeyName=RootMa...

SAS Policy ARM ID: /subscriptions/c375ef05-a27a-41f9-ba5e-20eebfeb29c6/resourceGroups/xzhang2/prov...

Lab

Step 2: Create an Azure Databricks workspace



Create an Azure Databricks workspace

Project Details

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

Subscription * : AzureSubscription

Resource group * : xzhang2

Instance Details

Workspace name * : xzhang2db ✓

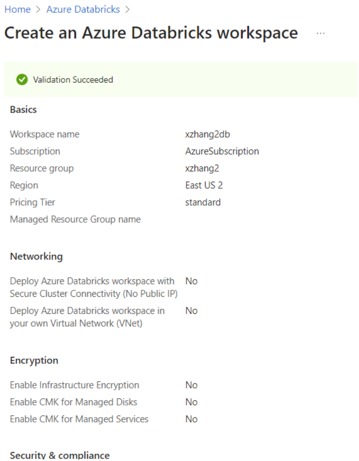
Region * : East US 2

Pricing Tier * : Standard (Apache Spark, Secure with Microsoft Entra ID)

Managed Resource Group name : Enter name for managed resource group

Lab

8. Click Create



Home > Azure Databricks >

Create an Azure Databricks workspace ...

✓ Validation Succeeded

Basics

Workspace name	xzhang2db
Subscription	AzureSubscription
Resource group	xzhang2
Region	East US 2
Pricing Tier	standard
Managed Resource Group name	

Networking

Deploy Azure Databricks workspace with Secure Cluster Connectivity (No Public IP)	No
Deploy Azure Databricks workspace in your own Virtual Network (VNet)	No

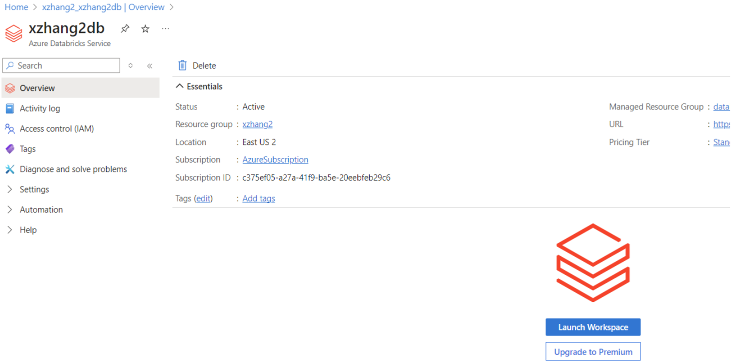
Encryption

Enable Infrastructure Encryption	No
Enable CMK for Managed Disks	No
Enable CMK for Managed Services	No

Security & compliance

Lab

9. Go to Azure Databricks resource and click on “Launch Workspace”.



Status : Active

Resource group : xzhang2

Location : East US 2

Subscription : AzureSubscription

Subscription ID : c375ef05-a27a-41f9-ba5e-20eebfeb29c6

Tags (edit) : Add tags

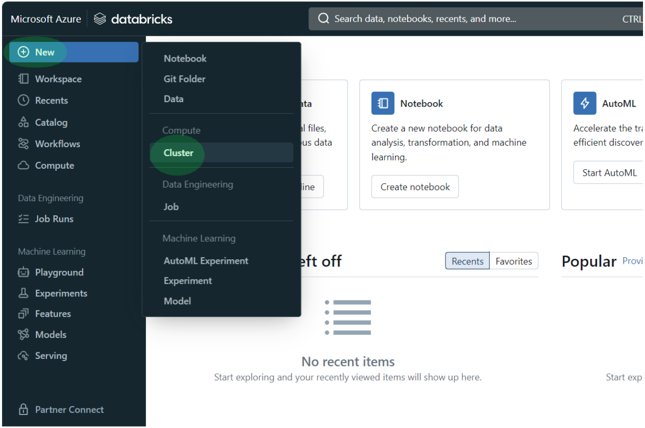
Managed Resource Group : data

URL : https:

Pricing Tier : Stan

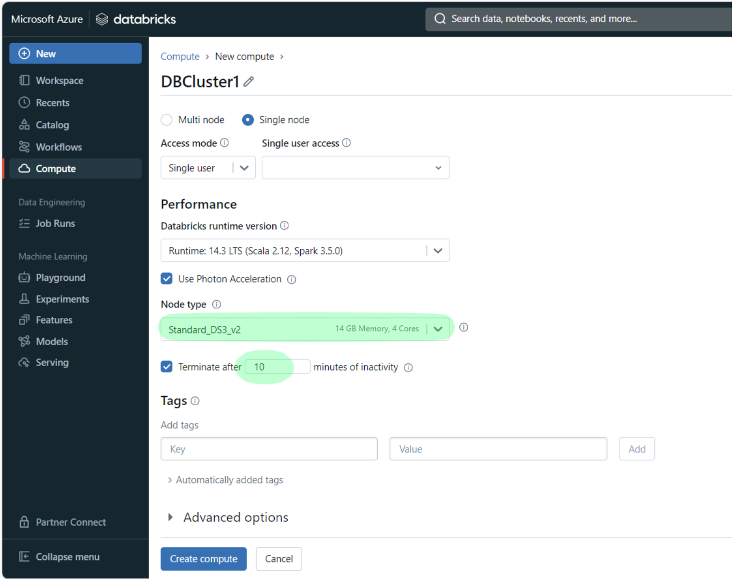
Lab

10. In the workspace, click on “New Cluster” to create a cluster.



Lab

11. Settings of the cluster are shown below.



Compute › New compute ›

DBCluster1

○ Multi node ● Single node

Access mode: Single user

Single user access:

Performance

Databricks runtime version: Runtime: 14.3 LTS (Scala 2.12, Spark 3.5.0)

☑ Use Photon Acceleration

Node type: Standard_DS3_v2 14 GB Memory, 4 Cores

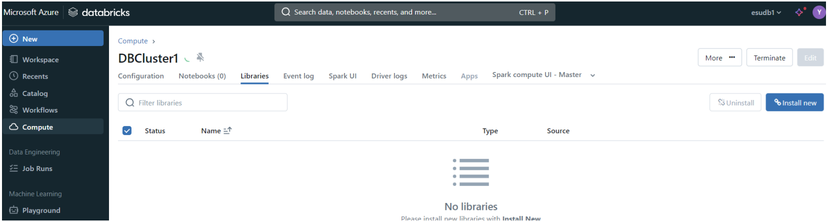
☑ Terminate after 10 minutes of inactivity

Tags

Add tags

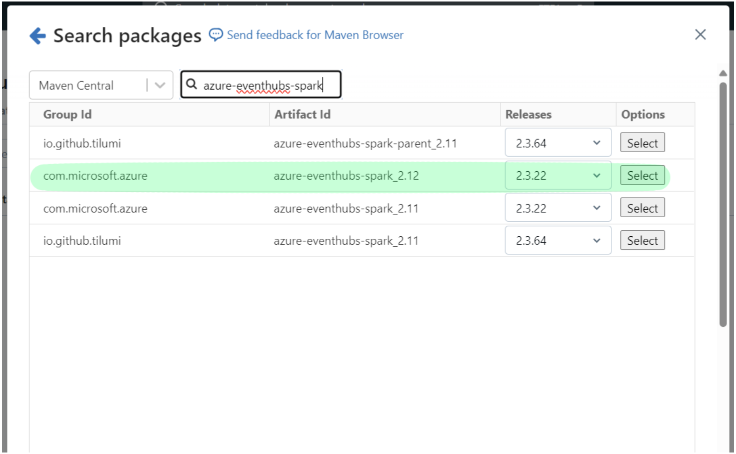
Lab

12. Click on "Libraries" and then Click on "Install New"



Lab

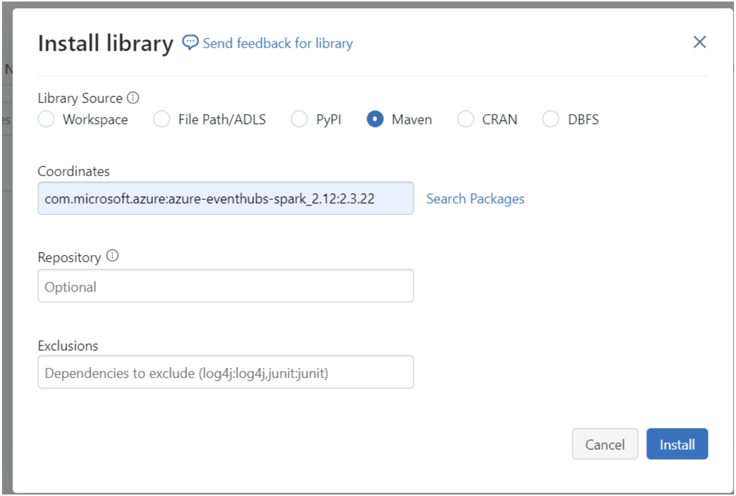
13. Go to Maven and click on ”Search Packages”
14. Select `Maven Central` and search for ‘’azure-eventhubs-spark” and Select “azure-eventhubs-spark_2.12”



Group Id	Artifact Id	Releases	Options
io.github.tilumi	azure-eventhubs-spark-parent_2.11	2.3.64	Select
com.microsoft.azure	azure-eventhubs-spark_2.12	2.3.22	Select
com.microsoft.azure	azure-eventhubs-spark_2.11	2.3.22	Select
io.github.tilumi	azure-eventhubs-spark_2.11	2.3.64	Select

Lab

15. After selecting the package, click on ”Install”



Install library

Send feedback for library

Library Source ⓘ

○ Workspace ○ File Path/ADLS ○ PyPI ● Maven ○ CRAN ○ DBFS

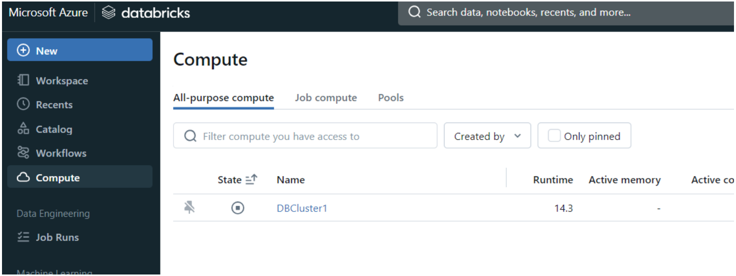
Coordinates: com.microsoft.azure:azure-eventhubs-spark_2.12:2.3.22

Repository ⓘ: Optional

Exclusions: Dependencies to exclude (log4j:log4j,junit:junit)

Lab

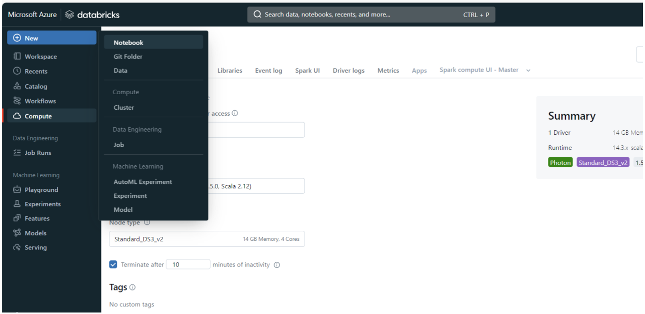
16. Go to Azure Databricks workspace, and click on "Compute" and click on the cluster



	State	Name	Runtime	Active memory	Active co
	⊙	DBCluster1	14.3	-	

Lab

17. Click on "+ New" -> `Notebook` to create a new Notebook



.5.0, Scala 2.12)

Node type

Standard_DS3_v2 14 GB Memory, 4 Cores

☑ Terminate after 10 minutes of inactivity

Tags

No custom tags

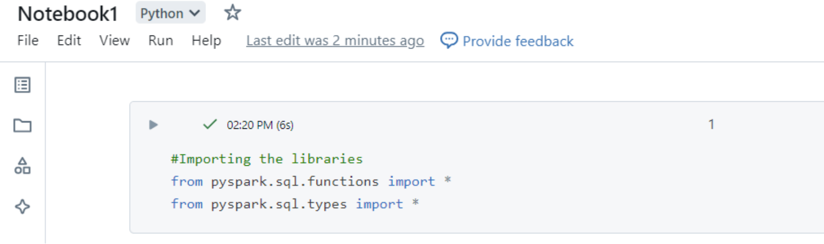
1 Driver 14 GB Mem

Runtime 14.3.x-scala

Lab

18. Type the code in a cell and run the cell

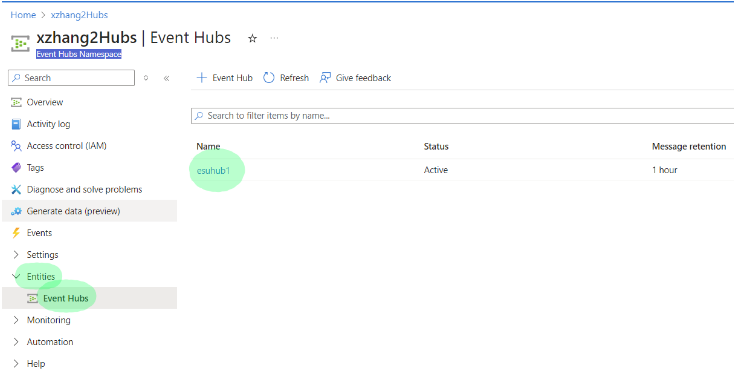
```
#Importing the libraries
from pyspark.sql.functions import *
from pyspark.sql.types import *
```



```
#Importing the libraries
from pyspark.sql.functions import *
from pyspark.sql.types import *
```

Lab

19. Go to the Event Hubs Namespace and click on the event hub



Home > xzhang2Hubs

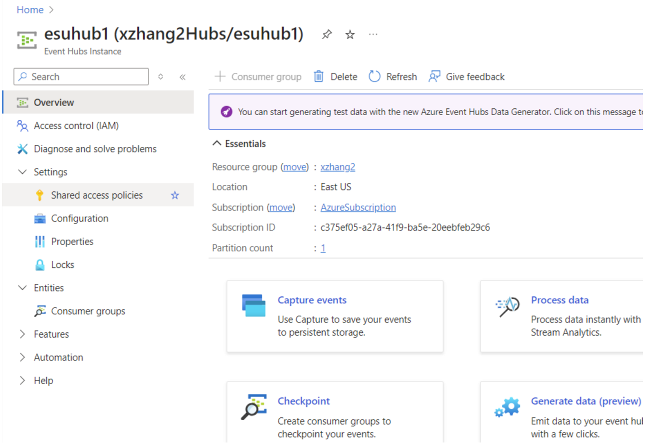
xzhang2Hubs | Event Hubs

Event Hubs Namespace

Name	Status	Message retention
esuhub1	Active	1 hour

Lab

20. Click `Shared access policies`



Home >

esuhub1 (xzhang2Hubs/esuhub1)

Event Hubs Instance

You can start generating test data with the new Azure Event Hubs Data Generator. Click on this message t

Essentials

Resource group (move) : xzhang2

Location : East US

Subscription (move) : AzureSubscription

Subscription ID : c375ef05-a27a-41f9-ba5e-20eebfeb29c6

Partition count : 1

Capture events
Use Capture to save your events to persistent storage.

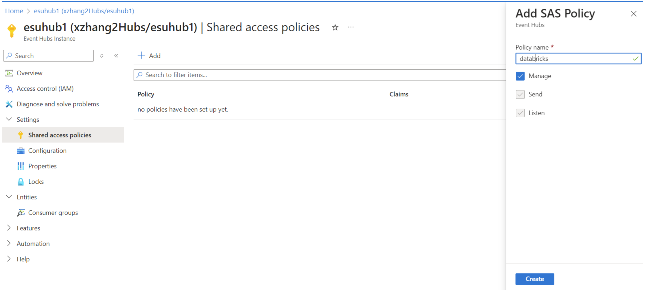
Process data
Process data instantly with Stream Analytics.

Checkpoint
Create consumer groups to checkpoint your events.

Generate data (preview)
Emit data to your event hul with a few clicks.

Lab

21. Create the policy



Home > esuhub1 (xzhang2Hubs/esuhub1)

esuhub1 (xzhang2Hubs/esuhub1) | Shared access policies

Event Hubs Instance

Policy	Claims
no policies have been set up yet.	

Add SAS Policy

Event Hubs

Policy name * databricks

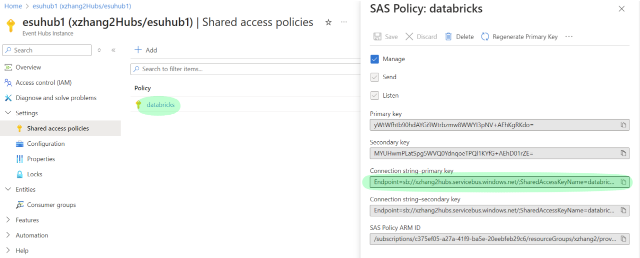
☑ Manage

☑ Send

☑ Listen

Lab

22. Click on the policy and Copy the connection string



Policy

databricks

SAS Policy: databricks

☑ Manage

☑ Send

☑ Listen

Primary key

yWtWfhtb90hdAYGi9Wtrbzmw8WWYI3pNV+AEhKgRKdo=

Secondary key

MYUHwmPLatSpg5WVQ0YdnqoeTPQl1KYfG+AEhD01rZE=

Connection string–primary key

Endpoint=sb://xzhang2hubs.servicebus.windows.net/;SharedAccessKeyName=databric...

Connection string–secondary key

Endpoint=sb://xzhang2hubs.servicebus.windows.net/;SharedAccessKeyName=databric...

SAS Policy ARM ID

/subscriptions/c375ef05-a27a-41f9-ba5e-20eebfeb29c6/resourceGroups/xzhang2/prov...

Lab

23. Go back to the notebook in the databricks

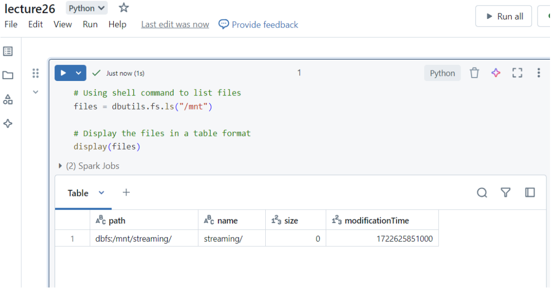
```
# create a schema
spark.sql("create schema streaming;")
```

```
#Set up Azure Event hubs connection string
#Replace with your Event Hub namespace, name, and key
connectionString = "Endpoint=sb://xzhang2hubs.servicebus.windows.net/;
SharedAccessKeyName=databricks;
SharedAccessKey=yWtWfhtb90hdAYGi9Wtrbzmw8WWYI3pNV+AEhKgRKdo=;
EntityPath=esuhub1"
eventHubName = "esuhub1"

ehConf = {
  'eventhubs.connectionString' : sc._jvm.org.apache.spark.eventhubs.EventHubsUtils.e
  'eventhubs.eventHubName': eventHubName
}
```

Lab

- We can display the contents of a DBFS mount point or directory in Azure Databricks by running

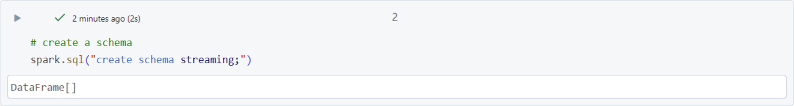
```
# Using shell command to list files
files = dbutils.fs.ls("/mnt")

# Display the files in a table format
display(files)
```



```
# Using shell command to list files
files = dbutils.fs.ls("/mnt")

# Display the files in a table format
display(files)
```

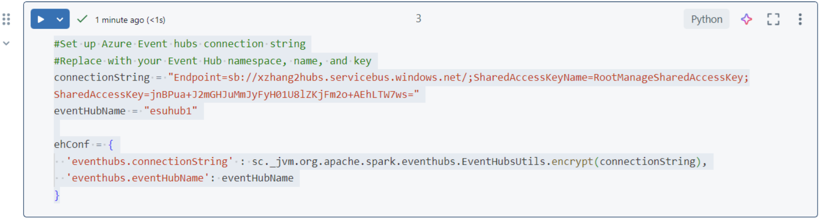
	path	name	size	modificationTime
1	dbfs:/mnt/streaming/	streaming/	0	1722625851000

Lab



```
# create a schema
spark.sql("create schema streaming;")
```

```
DataFrame[]
```



```
#Set up Azure Event hubs connection string
#Replace with your Event Hub namespace, name, and key
connectionString = "Endpoint=sb://xzhang2hubs.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;
SharedAccessKey=jnBPua+J2mGHJuMmJyFyH01U8lZKjFm2o+AEhLTW7ws="
eventHubName = "esuhub1"

ehConf = {
  'eventhubs.connectionString' : sc._jvm.org.apache.spark.eventhubs.EventHubsUtils.encrypt(connectionString),
  'eventhubs.eventHubName': eventHubName
}
```

Lab

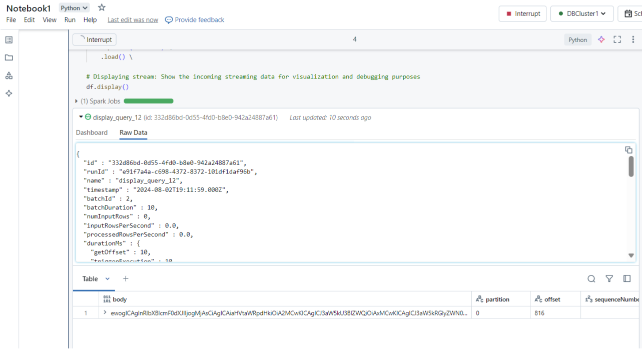
24. Run the following cell

- Send data from the event hub while the stream is running

```
{
    "temperature": 20,
    "humidity": 60,
    "windSpeed": 10,
    "windDirection": "NW",
    "precipitation": 0,
    "conditions": "Partly Cloudy"
}
```

```python
# Reading stream: Load data from Azure Event Hub into DataFrame 'df' using the previ
df = spark.readStream \
    .format("eventhubs") \
    .options(**ehConf) \
    .load() \

# Displaying stream: Show the incoming streaming data for visualization and debuggin
df.display()
```

Lab



```
        .load() \

# Displaying stream: Show the incoming streaming data for visualization and debugging purposes
df.display()
```

(1) Spark Jobs

display_query_12 (id: 332d86bd-0d55-4fd0-b8e0-942a24887a61) *Last updated: 10 seconds ago*

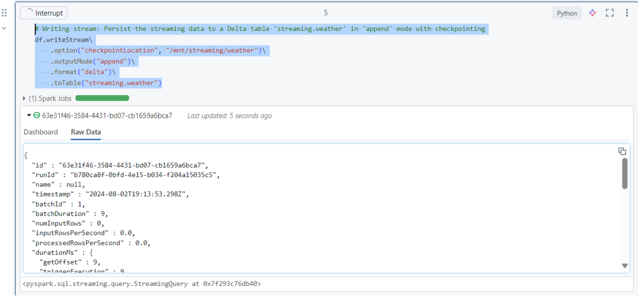
```
{
  "id" : "332d86bd-0d55-4fd0-b8e0-942a24887a61",
  "runId" : "e91f7a4a-c698-4372-8372-101df1daf96b",
  "name" : "display_query_12",
  "timestamp" : "2024-08-02T19:11:59.000Z",
  "batchId" : 2,
  "batchDuration" : 10,
  "numInputRows" : 0,
  "inputRowsPerSecond" : 0.0,
  "processedRowsPerSecond" : 0.0,
  "durationMs" : {
    "getOffset" : 10,
    "triggerExecution" : 10
```

	body	partition	offset	sequenceNumbe
1	ewogICAgInRlbXBlcmF0dXJlIjogMjAsCiAgICAiaHVtaWRpdHkiOiA2MCwKICAgICJ3aW5kU3BlZWQiOiAxMCwKICAgICJ3aW5kRGlyZWN0...	0	816	

Lab

25. Writing stream

```
# Writing stream: Persist the streaming data to a Delta table 'stre
df.writeStream\
    .option("checkpointLocation", "/mnt/streaming/weather")\
    .outputMode("append")\
    .format("delta")\
    .toTable("streaming.weather")
```



```
# Writing stream: Persist the streaming data to a Delta table 'streaming.weather' in 'append' mode with checkpointing
df.writeStream\
    .option("checkpointLocation", "/mnt/streaming/weather")\
    .outputMode("append")\
    .format("delta")\
    .toTable("streaming.weather")
```

63e31f46-3584-4431-bd07-cb1659a6bca7 *Last updated: 5 seconds ago*

```
{
  "id" : "63e31f46-3584-4431-bd07-cb1659a6bca7",
  "runId" : "b780ca8f-0bfd-4e15-b034-f204a15035c5",
  "name" : null,
  "timestamp" : "2024-08-02T19:13:53.298Z",
  "batchId" : 1,
  "batchDuration" : 9,
  "numInputRows" : 0,
  "inputRowsPerSecond" : 0.0,
  "processedRowsPerSecond" : 0.0,
  "durationMs" : {
    "getOffset" : 9,
    "triggerExecution" : 9
```

```
<pyspark.sql.streaming.query.StreamingQuery at 0x7f293c76db40>
```

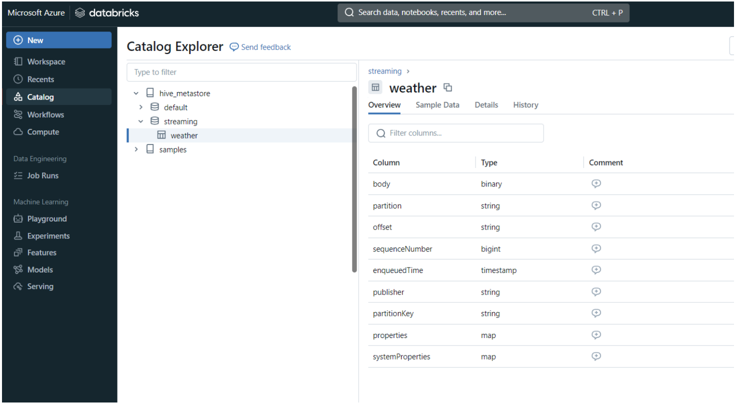
Lab

In Spark Structured Streaming, the outputMode specifies how the results of a streaming query should be written to the output sink. There are three output modes: append, complete, and update. Each mode has different behaviors and use cases:

- `Append` Mode: Only the new rows that are appended to the result table since the last trigger are written to the sink
- `Complete` Mode: The entire result table is written to the sink every time there is a trigger. This means all rows of the result are written out, not just the new rows.
- `Update` Mode: Only the rows that have changed since the last trigger are written to the sink. This includes both newly added rows and updated row

Lab

26. Check the `Catalog`, you should be able to see the table `weather`

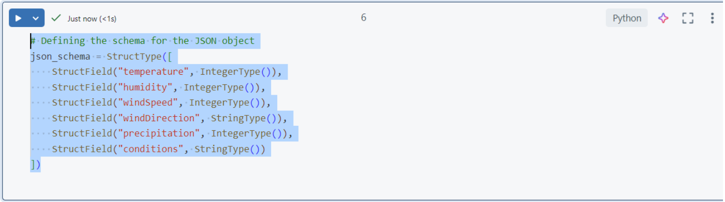


Column	Type	Comment
body	binary	
partition	string	
offset	string	
sequenceNumber	bigint	
enqueuedTime	timestamp	
publisher	string	
partitionKey	string	
properties	map	
systemProperties	map	

Lab

27. Defining the schema for the JSON object.

```python
# Defining the schema for the JSON object
json_schema = StructType([
    StructField("temperature", IntegerType()),
    StructField("humidity", IntegerType()),
    StructField("windSpeed", IntegerType()),
    StructField("windDirection", StringType()),
    StructField("precipitation", IntegerType()),
    StructField("conditions", StringType())
])
```



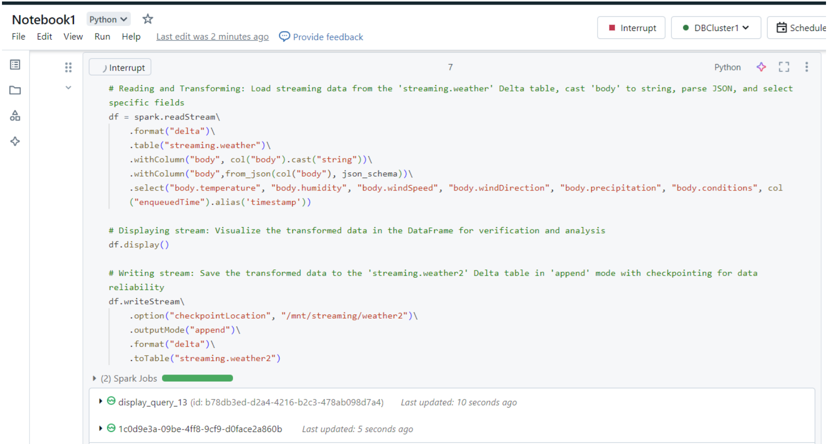
```python
# Defining the schema for the JSON object
json_schema = StructType([
    StructField("temperature", IntegerType()),
    StructField("humidity", IntegerType()),
    StructField("windSpeed", IntegerType()),
    StructField("windDirection", StringType()),
    StructField("precipitation", IntegerType()),
    StructField("conditions", StringType())
])
```

Lab

28. Reading and Transforming

```
# Reading and Transforming: Load streaming data from the 'streaming.weather' Delta table, cast '
df = spark.readStream\
    .format("delta")\
    .table("streaming.weather")\
    .withColumn("body", col("body").cast("string"))\
    .withColumn("body",from_json(col("body"), json_schema))\
    .select("body.temperature", "body.humidity", "body.windSpeed",
            "body.windDirection", "body.precipitation",
            "body.conditions", col("enqueuedTime").alias('timestamp'))

# Displaying stream: Visualize the transformed data in the DataFrame for verification and analys
df.display()

# Writing stream: Save the transformed data to the 'streaming.weather2' Delta table in 'append' 
df.writeStream\
    .option("checkpointLocation", "/mnt/streaming/weather2")\
    .outputMode("append")\
    .format("delta")\
    .toTable("streaming.weather2")
```

Lab

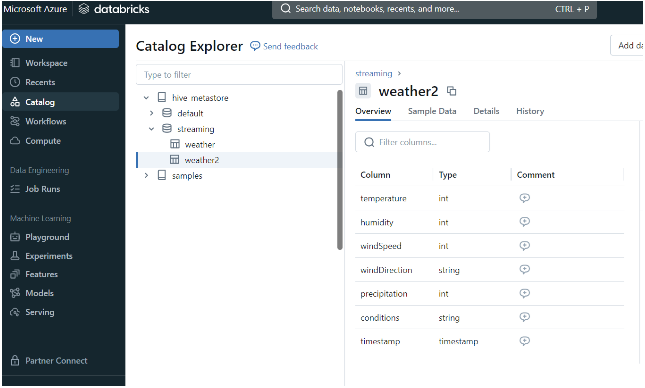


```
# Reading and Transforming: Load streaming data from the 'streaming.weather' Delta table, cast 'body' to string, parse JSON, and select specific fields
df = spark.readStream\
    .format("delta")\
    .table("streaming.weather")\
    .withColumn("body", col("body").cast("string"))\
    .withColumn("body",from_json(col("body"), json_schema))\
    .select("body.temperature", "body.humidity", "body.windSpeed", "body.windDirection", "body.precipitation", "body.conditions", col("enqueuedTime").alias('timestamp'))

# Displaying stream: Visualize the transformed data in the DataFrame for verification and analysis
df.display()

# Writing stream: Save the transformed data to the 'streaming.weather2' Delta table in 'append' mode with checkpointing for data reliability
df.writeStream\
    .option("checkpointLocation", "/mnt/streaming/weather2")\
    .outputMode("append")\
    .format("delta")\
    .toTable("streaming.weather2")
```

```
(2) Spark Jobs
display_query_13 (id: b78db3ed-d2a4-4216-b2c3-478ab098d7a4)    Last updated: 10 seconds ago
1c0d9e3a-09be-4ff8-9cf9-d0face2a860b    Last updated: 5 seconds ago
```

Lab

29. Check the Catalogy again



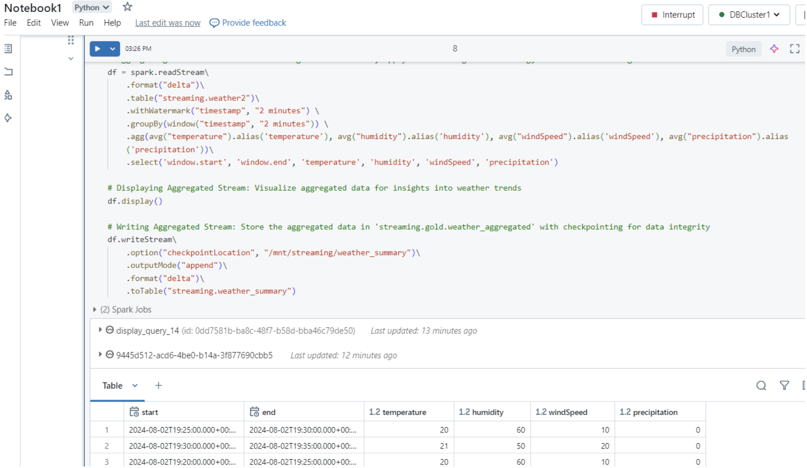
Column	Type	Comment
temperature	int	
humidity	int	
windSpeed	int	
windDirection	string	
precipitation	int	
conditions	string	
timestamp	timestamp	

Lab

30. Reading, aggregating and writing the stream

```
# Aggregating Stream: Read from 'streaming.silver.weather', apply watermarking and windowing, and
df = spark.readStream\
    .format("delta")\
    .table("streaming.weather2")\
    .withWatermark("timestamp", "1 minutes") \
    .groupBy(window("timestamp", "1 minutes")) \
    .agg(avg("temperature").alias('temperature'), avg("humidity").alias('humidity'),
      avg("windSpeed").alias('windSpeed'), avg("precipitation").alias('precipitation'))\
    .select('window.start', 'window.end', 'temperature', 'humidity', 'windSpeed', 'precipitatio

# Displaying Aggregated Stream: Visualize aggregated data for insights into weather trends
df.display()

# Writing Aggregated Stream: Store the aggregated data in 'streaming.gold.weather_aggregated' wi
df.writeStream\
    .option("checkpointLocation", "/mnt/streaming/weather_summary")\
    .outputMode("append")\
    .format("delta")\
    .toTable("streaming.weather_summary")
```

Lab



```
df = spark.readStream\
    .format("delta")\
    .table("streaming.weather2")\
    .withWatermark("timestamp", "2 minutes") \
    .groupBy(window("timestamp", "2 minutes")) \
    .agg(avg("temperature").alias('temperature'), avg("humidity").alias('humidity'), avg("windSpeed").alias('windSpeed'), avg("precipitation").alias
    ('precipitation'))\
    .select('window.start', 'window.end', 'temperature', 'humidity', 'windSpeed', 'precipitation')

# Displaying Aggregated Stream: Visualize aggregated data for insights into weather trends
df.display()

# Writing Aggregated Stream: Store the aggregated data in 'streaming.gold.weather_aggregated' with checkpointing for data integrity
df.writeStream\
    .option("checkpointLocation", "/mnt/streaming/weather_summary")\
    .outputMode("append")\
    .format("delta")\
    .toTable("streaming.weather_summary")
```

(2) Spark Jobs

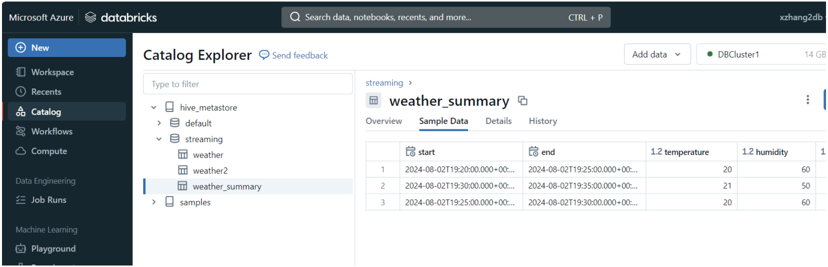
display_query_14 (id: 0dd7581b-ba8c-48f7-b58d-bba46c79de50) *Last updated: 13 minutes ago*

9445d512-acd6-4be0-b14a-3f877690cbb5 *Last updated: 12 minutes ago*

	start	end	1.2 temperature	1.2 humidity	1.2 windSpeed	1.2 precipitation
1	2024-08-02T19:25:00.000+00:...	2024-08-02T19:30:00.000+00:...	20	60	10	0
2	2024-08-02T19:30:00.000+00:...	2024-08-02T19:35:00.000+00:...	21	50	20	0
3	2024-08-02T19:20:00.000+00:...	2024-08-02T19:25:00.000+00:...	20	60	10	0

Lab

31. Check the catalog again



	start	end	1.2 temperature	1.2 humidity	1
1	2024-08-02T19:20:00.000+00:...	2024-08-02T19:25:00.000+00:...	20	60	
2	2024-08-02T19:30:00.000+00:...	2024-08-02T19:35:00.000+00:...	21	50	
3	2024-08-02T19:25:00.000+00:...	2024-08-02T19:30:00.000+00:...	20	60	

Lab

32. Summary of the three steps

```
json_schema = StructType([
    StructField("temperature", IntegerType()),
    StructField("humidity", IntegerType()),
    StructField("windSpeed", IntegerType()),
    StructField("windDirection", StringType()),
    StructField("precipitation", IntegerType()),
    StructField("conditions", StringType())
])

# Reading stream: Load data from Azure Event Hub into DataFrame 'df' using the previously config
df1 = spark.readStream \
    .format("eventhubs") \
    .options(**ehConf) \
    .load() \

# Writing stream: Persist the streaming data to a Delta table 'streaming.weather' in 'append' mo
df1.writeStream\
    .option("checkpointLocation", "/mnt/streaming/weather")\
    .outputMode("append")\
    .format("delta")\
    .toTable("streaming.weather")
```

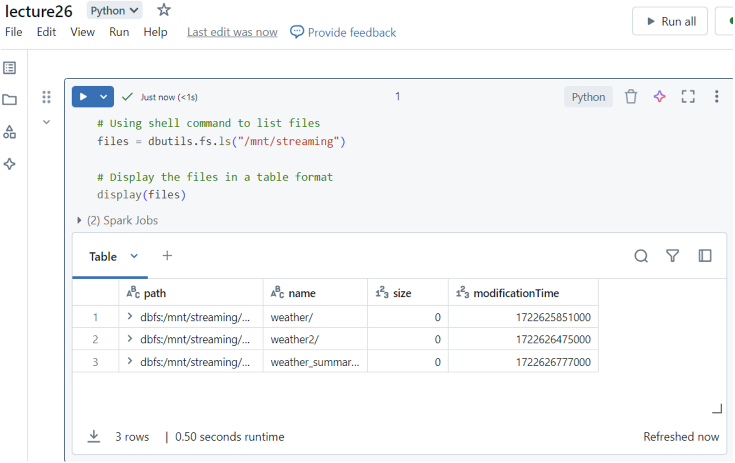
Lab

```
# Reading and Transforming: Load streaming data from the 'streaming.weather' Delta table, cast '
df2 = spark.readStream\
    .format("delta")\
    .table("streaming.weather")\
    .withColumn("body", col("body").cast("string"))\
    .withColumn("body",from_json(col("body"), json_schema))\
    .select("body.temperature", "body.humidity", "body.windSpeed", "body.windDirection",
            "body.precipitation", "body.conditions", col("enqueuedTime").alias('timestamp'))

# Displaying stream: Visualize the transformed data in the DataFrame for verification and analys
df2.display()

# Writing stream: Save the transformed data to the 'streaming.weather2' Delta table in 'append'
df2.writeStream\
    .option("checkpointLocation", "/mnt/streaming/weather2")\
    .outputMode("append")\
    .format("delta")\
    .toTable("streaming.weather2")
```

Lab

```
# Aggregating Stream: Read from 'streaming.silver.weather', apply watermarking and windowing, an
df3 = spark.readStream\
    .format("delta")\
    .table("streaming.weather2")\
    .withWatermark("timestamp", "2 minutes") \
    .groupBy(window("timestamp", "2 minutes")) \
    .agg(avg("temperature").alias('temperature'), avg("humidity").alias('humidity'),
         avg("windSpeed").alias('windSpeed'), avg("precipitation").alias('precipitation'))\
    .select('window.start', 'window.end', 'temperature', 'humidity', 'windSpeed', 'precipitatio

# Displaying Aggregated Stream: Visualize aggregated data for insights into weather trends
df3.display()

# Writing Aggregated Stream: Store the aggregated data in 'streaming.gold.weather_aggregated' wi
df3.writeStream\
    .option("checkpointLocation", "/mnt/streaming/weather_summary")\
    .outputMode("append")\
    .format("delta")\
    .toTable("streaming.weather_summary")
```

Lab



```
# Using shell command to list files
files = dbutils.fs.ls("/mnt/streaming")

# Display the files in a table format
display(files)
```

	path	name	size	modificationTime
1	dbfs:/mnt/streaming/...	weather/	0	1722625851000
2	dbfs:/mnt/streaming/...	weather2/	0	1722626475000
3	dbfs:/mnt/streaming/...	weather_summar...	0	1722626777000

3 rows | 0.50 seconds runtime

License