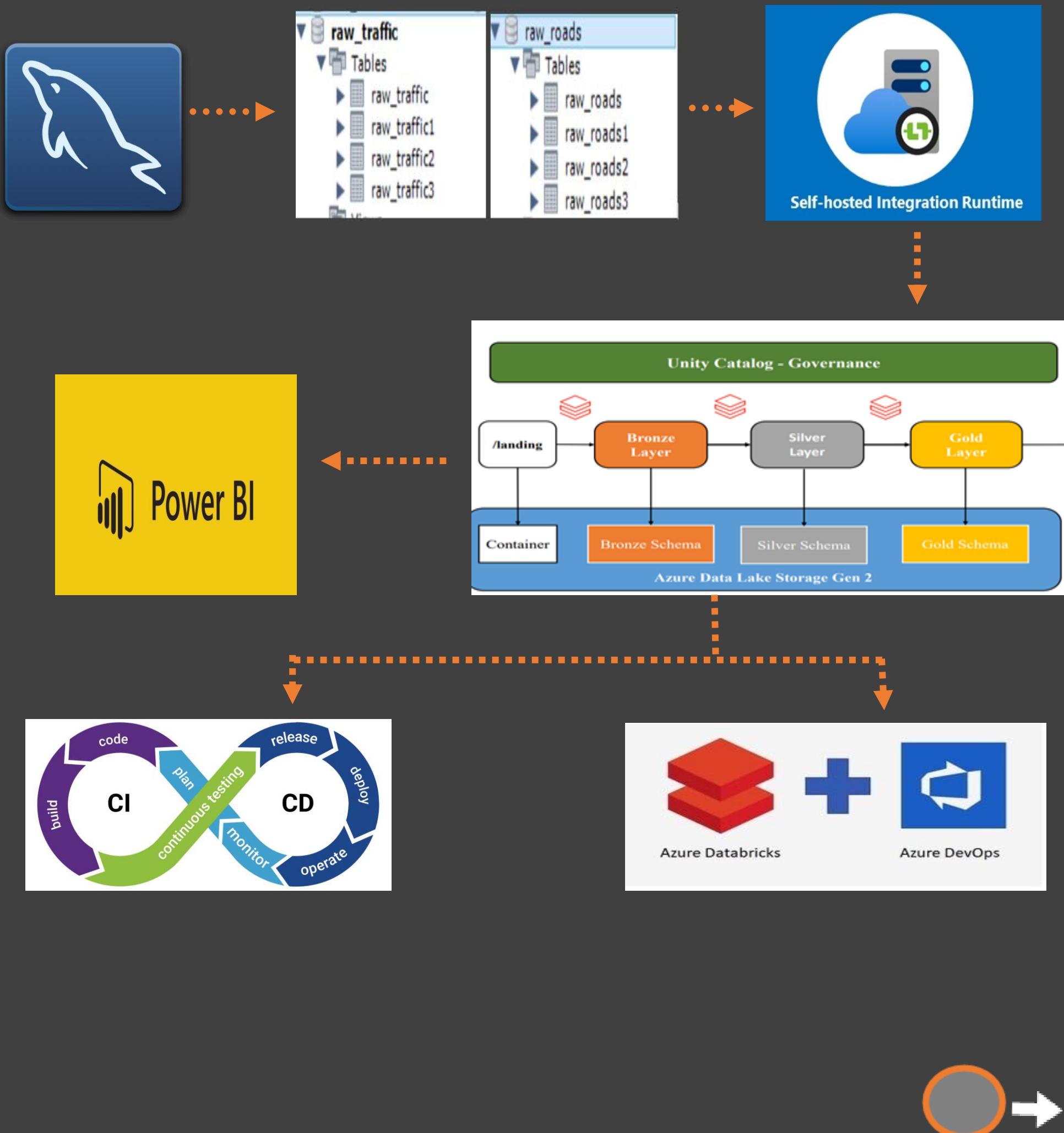


# Azure Databricks Mastery: Hands-on project MySQL(on premises) to Azure Data Factory(cloud), Unity Catalog, Delta lake, CI/CD implementing Medallion Architecture

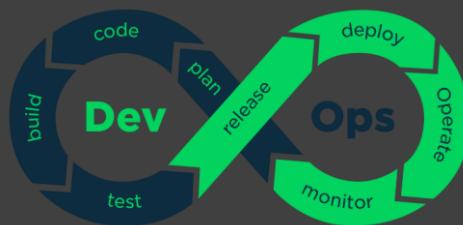
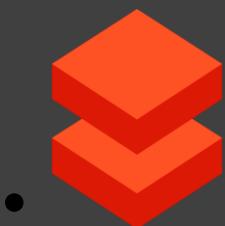


## OUTLINE:

1. Use Case
2. MySQL to Azure Data Factory
3. Configure connection between Databricks and Azure Lake Storage
4. Databricks Working
  - Databricks utilities
  - Delta tables using SQL
  - Schema evolution
  - Upsert
  - Unity catalog
  - Permissions to specific users
  - Cluster pools and policy
  - Storage credential and external location
5. Spark structured streaming
6. Data Architecture
  - Medallion architecture
  - Incremental loading
  - Auto loader with Spark Structured Streaming
  - Ingest data into bronze layer
  - Data from bronze to silver layer
  - Handle newly added data
7. Data Transformation and Quality
  - Automate the transformation process and data quality monitoring
  - Applying transformations, and writing the results to the Silver layer
  - Silver to gold layer transformation
8. Workflow and Orchestration
  - Orchestrate notebook workflows
  - Triggers for Azure
9. Power BI Integration
10. Linking and Repos
  - Linking Databricks and Azure Devops
  - Repos in Databricks
11. CI/CD Pipeline
12. User Management in Azure DevOps (Adding another user in Azure DevOps)
13. Pipeline and Environment Configuration
14. Delta Live Tables
  - DLT
  - Live delta pipeline

# Learning:

- Created queries in mysql, then load them in azure data factory using self hosted integration
- Implemented Unity Catalog and Implemented project with incremental loading
- Learnt the Spark Structured Streaming
- Implement Continuous Integration and Continuous Deployment in project
- Implementation, features and work with Delta Lake
- Implemented Medallion Architecture in my project
- Evolution of Delta lake from Datalake and Workflows in Azure Databricks and Simulate real time environment with Unity Catalog
- Implemented the governance with Unity Catalog and Mastered the compute cluster creation and Management
- How spark structured streaming works and Implemented structured streaming in Azure databricks
- Incremental loading with Autoloader, Code that can run in any environment and Implemented the Unity Catalog Object Model
- Build an end to end CICD pipeline and Implementation of Delta Live tables



# Use Case

This project was designed for companies and organizations that want to process and analyze their data in real-time. In this use case, I took a smart city scenario where IoT devices and sensors collected data, such as traffic cameras, weather stations, and public transportation systems. These devices continuously streamed data, which was analyzed to make real-time decisions, such as traffic management, pollution control, and public safety measures.

## Problem Statement

Real-time data processing is a significant challenge for smart cities. Large quantities of data are constantly being generated, making it difficult to analyze and make timely decisions based on this data. Traditional data processing systems could not efficiently handle this data. The solution to this problem required a system that could process data in real-time, be reliable, and easily scalable.

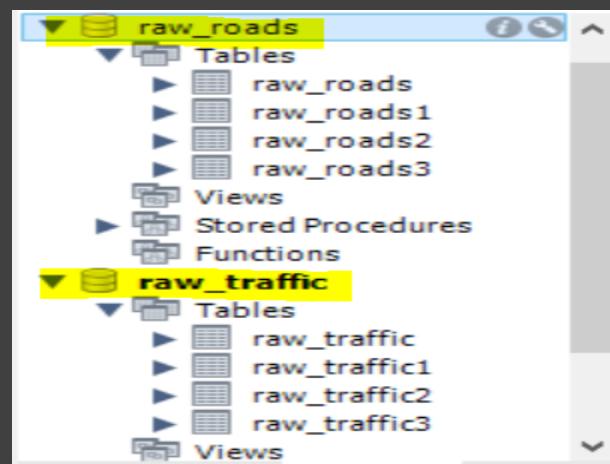
## Proposal

In this proposal, I designed a solution using Databricks live Delta tables and Spark Structured Streaming. I processed continuously streaming data from IoT devices and sensors and stored this data in Delta tables. This data was then analyzed through transformations and aggregations. I used multiple sources and sinks, such as file sources, and table sinks. This solution employed both micro-batch processing and continuous processing modes. This project provided an end-to-end orchestration and streaming solution using Databricks' live Delta tables. The goal of this project was to efficiently solve the challenges of real-time data processing and develop a robust solution that enhanced smart city operations.

# MySQL TO Azure Data Factory

## 1. Created MySQL Tables:

- First, I created a MySQL database and then established three tables for raw road data (raw\_roads1, raw\_roads2, raw\_roads3) and three tables for raw traffic data (raw\_traffic1, raw\_traffic2, raw\_traffic3).



raw\_roads

```

471 • -- Step 4: Call the stored procedure
472 CALL ProcessRawtrafficTables();
473
474 • select * from raw_roads1;
475
476
477
478
479
    
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | raw\_roads1 8 x

RoadID	RoadCategoryID	RoadCategory	RegionID	RegionName	TotalLinkLengthKm	TotalLinkLengthMiles	AllMotorVehicles
1	1	TM	1	South West	301.339	187.24	3465840000
2	3	TA	1	South West	993.586	617.39	3484710000
3	4	PA	1	South West	3874.92	2407.77	7794000000
4	5	M	1	South West	43581.7	27080.4	9112010000
5	1	TM	2	East Midlands	178.609	110.98	2736660000
6	3	TA	2	East Midlands	1219.23	757.6	4936700000
7	4	PA	2	East Midlands	2571.21	1597.68	5535720000
8	5	M	2	East Midlands	26712.7	16598.5	7083360000
9	1	TM	3	Scotland	335.263	208.32	2485470000

raw\_traffic

```

476
477 • select * from raw_traffic1;
    
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | raw\_traffic1 11 x

Record_ID	Count_point_id	Direction_of_travel	Year	Count_date	hour	Region_id	Region_name	Local_authority_name	Road_name	Road_Category_ID	Start_junction_n
1	749	E	2014	2014-06-25	7	3	Scotland	East Ayrshire	A77	3	LA boundary
2	749	E	2014	2014-06-25	8	3	Scotland	East Ayrshire	A77	3	LA boundary
3	749	E	2014	2014-06-25	9	3	Scotland	East Ayrshire	A77	3	LA boundary
4	749	E	2014	2014-06-25	10	3	Scotland	East Ayrshire	A77	3	LA boundary
5	749	E	2014	2014-06-25	11	3	Scotland	East Ayrshire	A77	3	LA boundary
6	749	E	2014	2014-06-25	12	3	Scotland	East Ayrshire	A77	3	LA boundary
7	749	W	2014	2014-06-25	13	3	Scotland	East Ayrshire	A77	3	LA boundary
8	749	W	2014	2014-06-25	14	3	Scotland	East Ayrshire	A77	3	LA boundary
9	749	W	2014	2014-06-25	15	3	Scotland	East Ayrshire	A77	3	LA boundary
10	749	W	2014	2014-06-25	16	3	Scotland	East Ayrshire	A77	3	LA boundary
11	749	W	2014	2014-06-25	17	3	Scotland	East Ayrshire	A77	3	LA boundary

## 2. Set Up Control Table:

- Next, I created a control table to manage these raw data tables. This control table was then loaded into Azure Data Factory.

```

● CREATE TABLE raw_traffic (
    TableName VARCHAR(50) PRIMARY KEY,
    SchemaName VARCHAR(50) DEFAULT 'raw_traffic'
);

● INSERT INTO raw_traffic (TableName) VALUES
    ('raw_traffic1'),
    ('raw_traffic2'),
    ('raw_traffic3');

DELIMITER $$

● CREATE PROCEDURE ProcessRawtrafficTables()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE tblName VARCHAR(50);
    DECLARE cur CURSOR FOR SELECT TableName FROM raw_traffic;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;

```

## 3. Configured Azure Data Factory:

- To connect to the data, I needed a self-hosted integration runtime and set up a linked service in Azure Data Factory.

**Integration runtimes**

The integration runtime (IR) is the compute infrastructure to provide the following data integration capabilities across different cloud environments.

+ New    Refresh

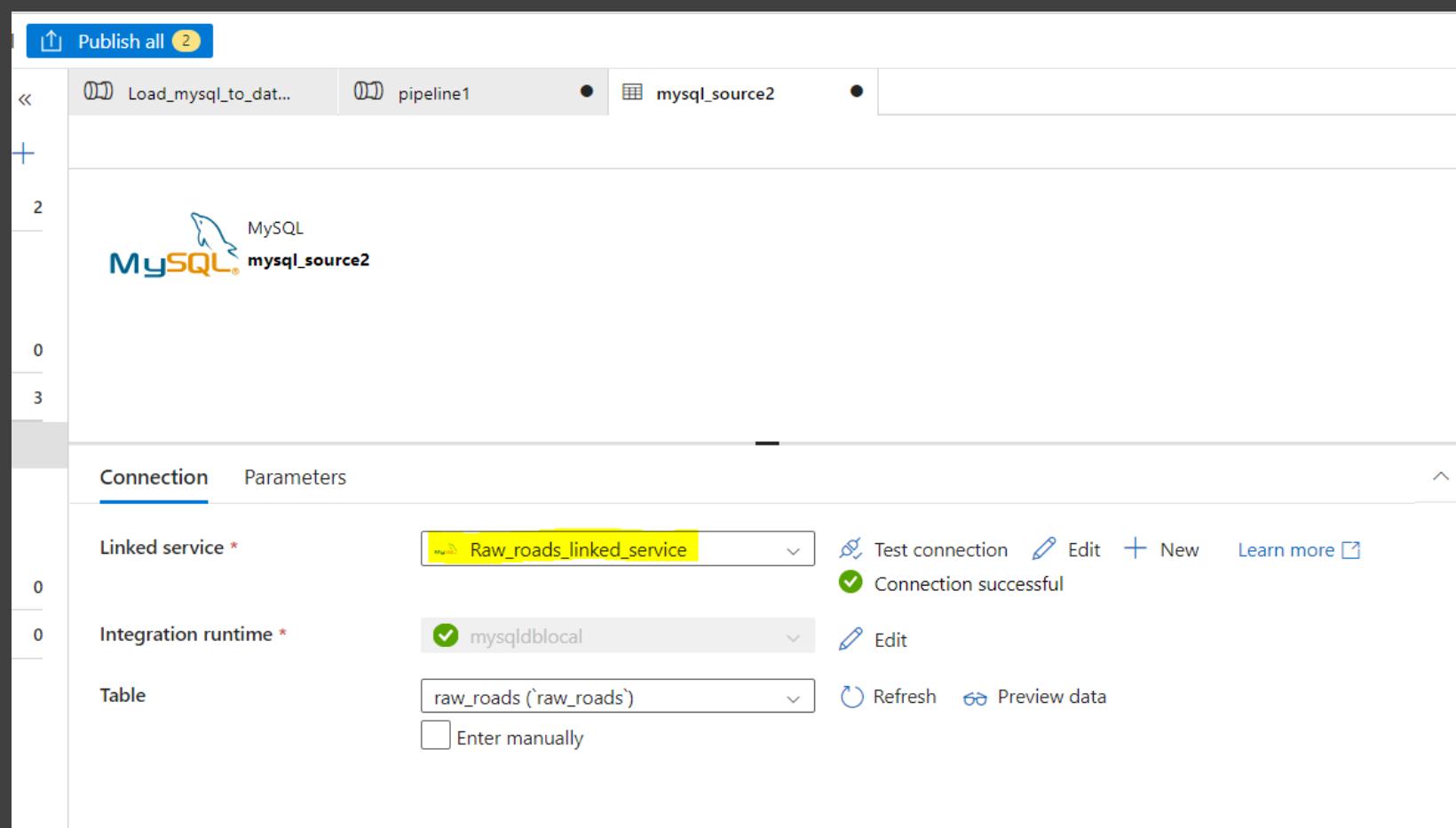
Filter by name

Showing 1 - 2 of 2 items

Name ↑↓	Type ↑↓	Sub-type ↑↓	Status ↑↓
AutoResolveIntegrationR...	Azure	Public	✓ Running
mysqldblocal	Self-Hosted	---	✓ Running

#### 4. Verified Connections:

- I verified both the self-hosted integration runtime and the linked service connections to ensure they were working correctly.



### Edit linked service

MySQL [Learn more](#)

**Name \***  
raw\_roads\_linked\_service

**Description**

**Connect via integration runtime \*** ⓘ  
 mysqldblocal [Edit](#)

**⚠** The credentials are stored in the machines of self-hosted integration runtime if you don't choose to store them in Azure Key Vault.

**Server name \***  
172.0.0.1

**Port**  
3306

**Database name \***  
raw\_roads

**User name \***  
root

**Password** [Azure Key Vault](#)

**Password \***  
.....

**Apply** **Cancel** [Test connection](#)

## 5. Configured Data Pipeline:

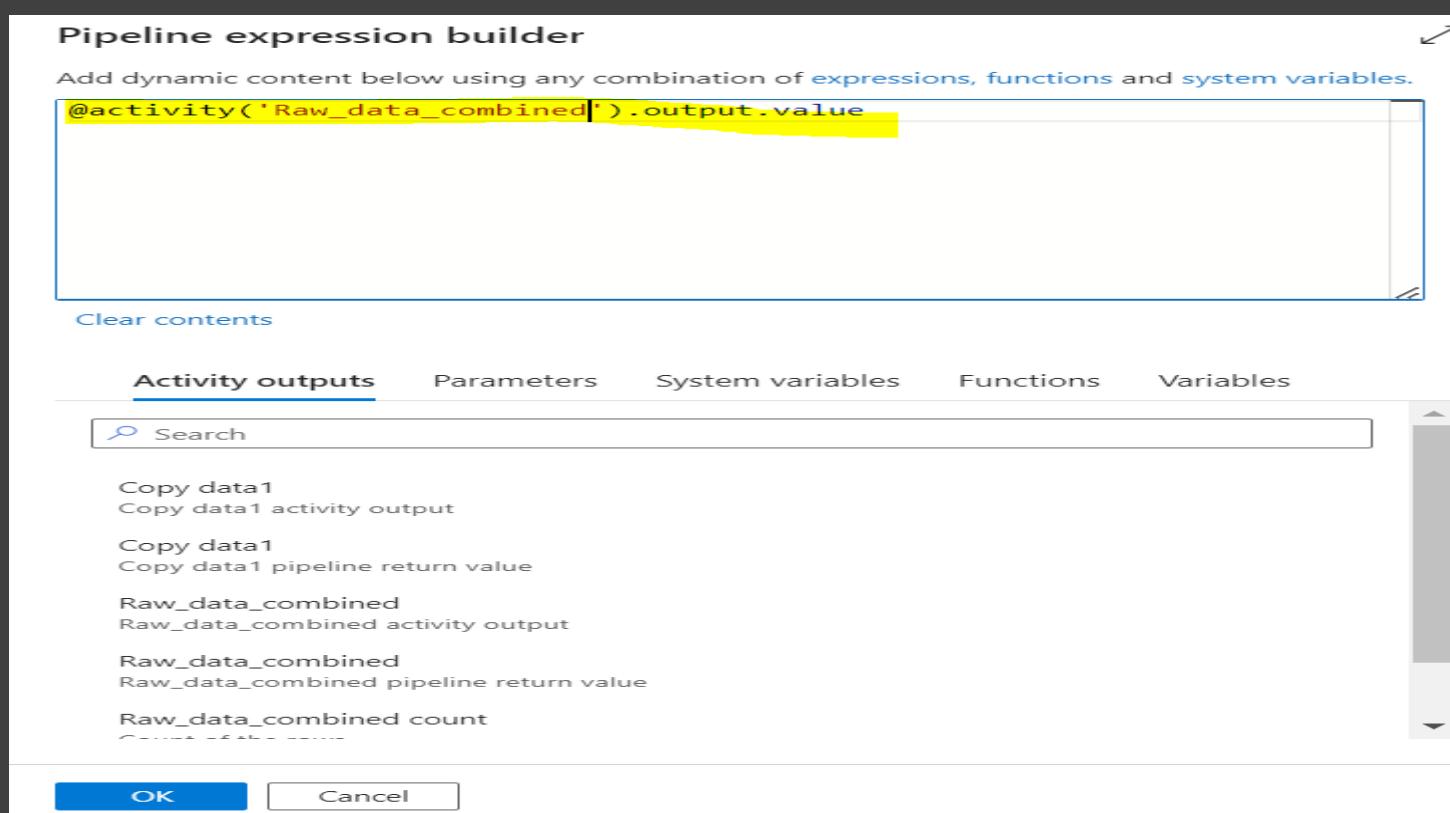
- In Azure Data Factory, I used a **Lookup activity** to select the appropriate source from the control table. This involved dragging and dropping the activity into the pipeline.

## 6. Used ForEach Activity:

- I utilized a **ForEach activity** to iterate over the data sources. Within this activity, I set up data parameters.

## 7. Added Copy Activity:

- Inside the **ForEach activity**, I added a **Copy activity** to handle data movement. I configured the source and sink destinations, selecting the control table as needed.

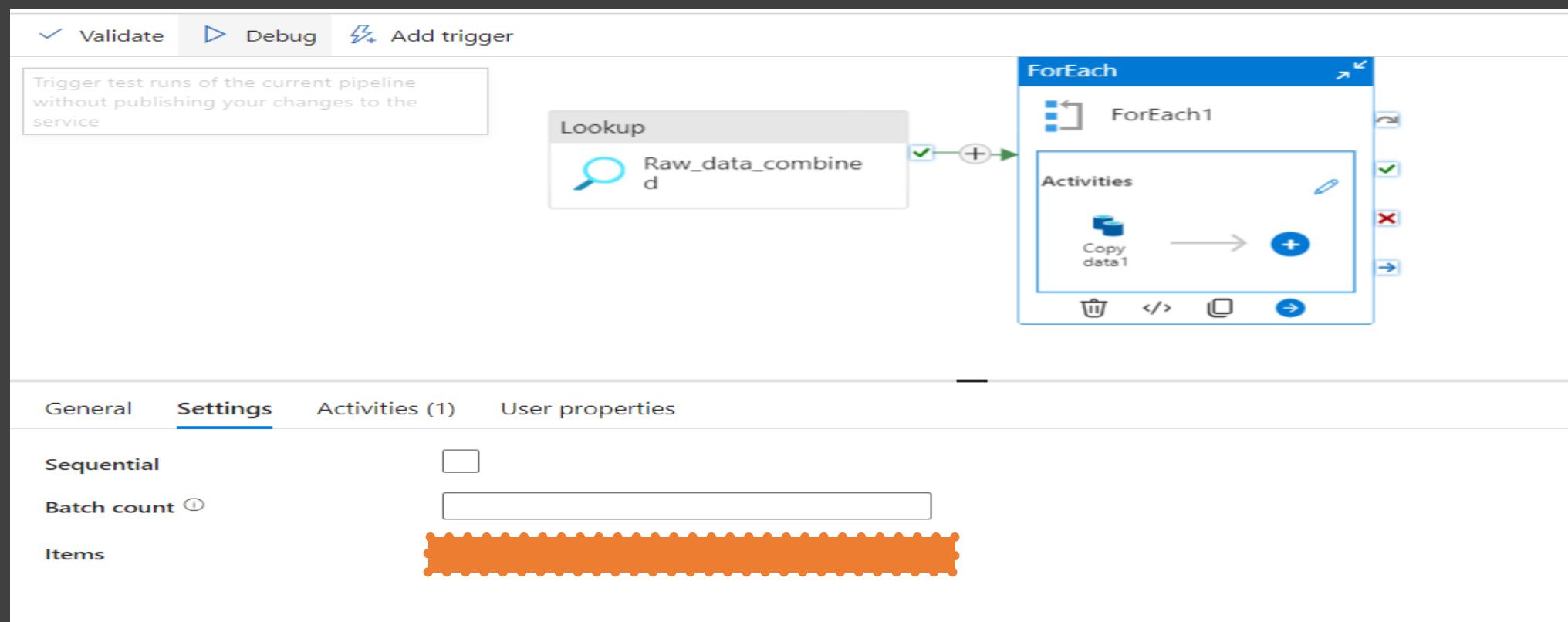


## 8. Debugged and Tested:

- I debugged the pipeline to ensure it was working correctly and that the data was being processed as expected. I provided the path to the landing container, which I had set up in my Azure storage account.

## 9. Loaded CSV Files:

- In the Azure storage account, I created folders named **raw\_roads** and **raw\_traffic**. I loaded three CSV files into each folder: **raw\_roads1.csv**, **raw\_roads2.csv**, and **raw\_roads3.csv** for road data, and similarly for traffic data. Processed Data Using Spark:
- Finally, I used **Spark Structured Streaming with Auto-Loader** to read and process the data from the CSV files



Search factory and documentation

Cloud-first data analytics SaaS platform? Click here to get started with Fabric Data

Validate Debug Add trigger

Lookup Raw\_data\_combine\_d

Pipeline expression builder

Add dynamic content below using any combination of expressions, functions and system variables.

```
@concat('select * from ', item().TableName)
```

Clear contents

ForEach iterator Activity outputs Parameters System variables Functions

Search

ForEach1 Current item

OK Cancel

General   Source   **Sink**   Mapping   Settings   User properties

**Sink dataset \*** StorageDestination [Open](#) [New](#) [Learn more](#)

**Dataset properties**

Name	Value	Type
FileName	[REDACTED]	string

**Copy behavior**

**Max concurrent connections**

**Block size (MB)**

**Metadata** [New](#)

**Quote all text**

**File extension** .txt

**Max rows per file**

## Foreach working

Is this your first time using the Azure Data Factory pipeline editor? Click [here](#) to get started with Fabric Data.

Validate   Debug   Add trigger

Lookup Raw\_data\_combined

**Pipeline expression builder**

Add dynamic content below using any combination of [expressions](#), [functions](#) and [system variables](#).

```
@activity('Raw_data_combined').output.value
```

[Clear contents](#)

**Activity outputs**   Parameters   System variables   Functions   Variables

Search

- Copy data1  
Copy data1 activity output
- Copy data1  
Copy data1 pipeline return value
- Raw\_data\_combined  
Raw\_data\_combined activity output
- Raw\_data\_combined  
Raw\_data\_combined pipeline return value
- Raw\_data\_combined count

**OK**   **Cancel**

Home > [databricksdevstorage0300](#) | Containers >

### landing

Container

Search

Upload Add Directory Refresh Rename Delete

**Authentication method:** Access key ([Switch to Microsoft Entra user account](#))  
**Location:** landing

Search blobs by prefix (case-sensitive)

Name	Modified
raw_roads	
raw_traffic	

# Working:

- I have created different resources in azure. I have created databricks resource group named as databricksdevws, then I created the storage account “Databricksdevstorage0300” with locally redundant, and also enable the herairchical namespace because it will create simple blob storage without heraical namespace. Then I have created folder named Databricks project, then inside this folder I have created notebook “Understanding Markdown” for learning magic commands
- For Markdown , we use  
%md #This is my databricks project.
- For ordered list, -one that would give bullet point with one.

Microsoft Azure

Storage accounts

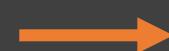
Default Directory

Create Restore Manage view Refresh Export to CSV Open query Assign tags Delete

Filter for any field... Subscription equals all Resource group equals all Location equals all Add filter

Showing 0 to 0 of 0 records.

Name	Type	Status	Actions
No grouping			



Home > [databricks1000](#) | Overview >

### databricks1000

Storage account

Search

Upload Open in Explorer Delete Move Refresh

Overview

Activity log Tags Diagnose and solve problems Access Control (IAM) Data migration Events Storage browser Storage Mover Data storage File shares

Containers

Resource group (move) [sqltodatabricks](#)  
 Location eastus Primary/Secondary Location Primary: East US, Secondary: West US Subscription (move) [Azure subscription\\_1](#)  
 Subscription ID ea8d46ac-c7b3-4f01-a3ea-a860b2f2dca5 Disk state Primary: Available, Secondary: Available Tags (edit)

The screenshot shows the Databricks Workspaces interface. At the top, there is a search bar labeled "Filter workspaces" with a magnifying glass icon. Below the search bar is a table with columns: Name, Status, Resource group, Region, and Sub. There are three rows of data:

Name	Status	Resource group	Region	Sub
deltadbwsp	Running	databricksresource	eastus	ea8d
databricksdevws	Running	databricksresource	eastus	ea8d
databrickworkspace	Running	databricksproject	eastus	ea8d

The screenshot shows a Databricks notebook titled "Understanding markdown" in Python. The notebook interface includes a sidebar with options like New, Workspace, Recents, Search, Catalog, Workflows, Compute, Machine Learning, and Experiments. The main area contains the following text:

This is my databricks project

I am using hash symbol for heading, %md in the beginning and language=markdown

my html how to bold?

Highlight my text

# Q. How to configure connection between databricks and azure lake storage?

There is no connection between databricks and azure data lake storage. We need general service principal. I have selected the azure entra id, then I have chosen app registration and created new registration named; “db\_access”. Now copy the application id, directory tenant id and new client secret from certifications and secrets. Copy the value of client secret and paste them in a notebook file. I have created storage account named “deltadbstg0300” and create a notebook named “test”

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes 'Microsoft Azure', a search bar, and a feedback link. Below the navigation is a breadcrumb trail: 'Home > Default Directory | App registrations > db\_access'. On the left, there's a sidebar with links like 'Overview', 'Quickstart', 'Integration assistant', 'Diagnose and solve problems', 'Manage', and 'Support + Troubleshooting'. The main content area is titled 'db\_access' and shows the following details under the 'Essentials' section:

- Display name: db\_access
- Application (client) ID: 7977d277-d3dc-4846-83df-5f00837fa3a6
- Object ID: 116d3f61-de3d-4f2e-be05-643f8d0a13a1
- Directory (tenant) ID: e890db70-99d7-4f80-b7b9-3ad1bf375a42
- Supported account types: My organization only

## Q. How to create containers in data storage account?

Create a container named test in the storage account “deltadbstg0300”. Create three directories named files, ParquetFolder and sample

The screenshot shows the Microsoft Azure portal interface for a storage account. The top navigation bar includes 'Microsoft Azure', a search bar, and a feedback link. Below the navigation is a breadcrumb trail: 'Home > deltadbstg0300 | Containers > test'. On the left, there's a sidebar with links like 'Overview', 'Diagnose and solve problems', 'Access Control (IAM)', and 'Settings'. The main content area shows the 'test' container details. It includes sections for 'Authentication method' (Access key) and 'Location' (test). A search bar at the bottom allows searching by blob prefix. The list of blobs shows three entries:

Name	Modified
files	
ParquetFolder	
sample	

# Q. What are the main steps for processing data?

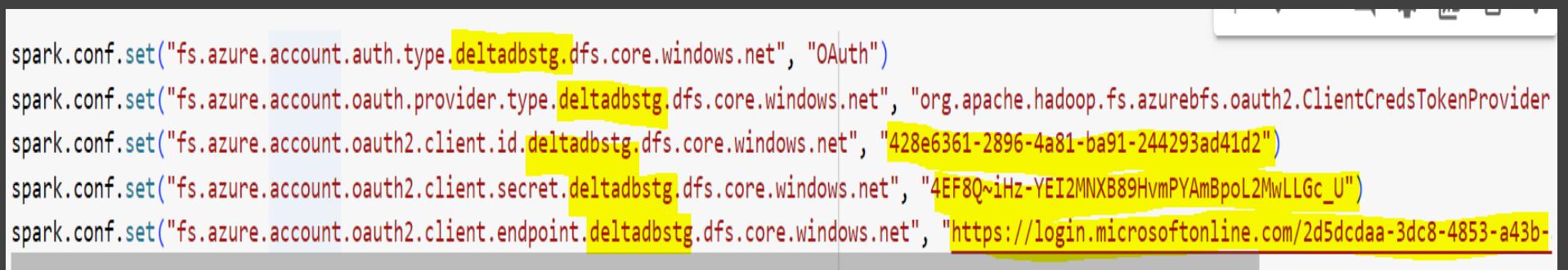
Created a notebook named; Drawbacks of Azure data lake storage and paste the service credential code in first cell

## Step 1: Service credential code

Sample:

```
service_credential = dbutils.secrets.get(scope="", key="")
```

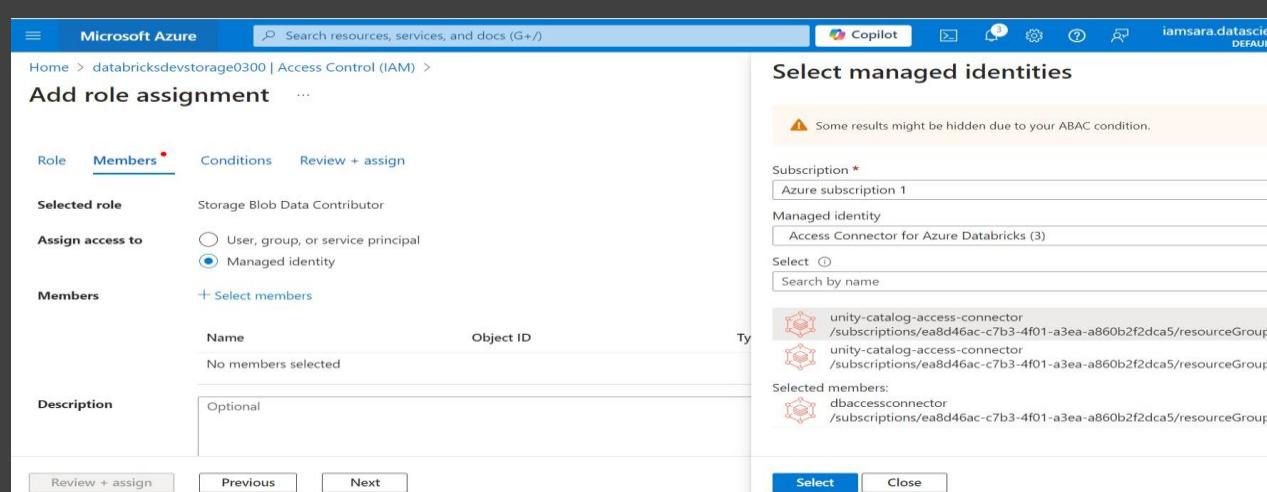
```
spark.conf.set("fs.azure.account.auth.type.<storage-account>.dfs.core.windows.net", "OAuth")
spark.conf.set("fs.azure.account.oauth.provider.type.<storage-account>.dfs.core.windows.net",
"org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")
spark.conf.set("fs.azure.account.oauth2.client.id.<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)
spark.conf.set("fs.azure.account.oauth2.client.endpoint.<storage-
account>.dfs.core.windows.net", "https://login.microsoftonline.com/<directory-
id>/oauth2/token")
```



```
spark.conf.set("fs.azure.account.auth.type.deltadbstg.dfs.core.windows.net", "OAuth")
spark.conf.set("fs.azure.account.oauth.provider.type.deltadbstg.dfs.core.windows.net", "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")
spark.conf.set("fs.azure.account.oauth2.client.id.deltadbstg.dfs.core.windows.net", "428e6361-2896-4a81-ba91-244293ad41d2")
spark.conf.set("fs.azure.account.oauth2.client.secret.deltadbstg.dfs.core.windows.net", "4EF8Q~iHz-YEI2MNXB89HvmPYAmBpoL2MwLLGc_U")
spark.conf.set("fs.azure.account.oauth2.client.endpoint.deltadbstg.dfs.core.windows.net", "https://login.microsoftonline.com/2d5dcdaa-3dc8-4853-a43b-
```

## Step 2: Give azure access permissions

Go to Azure IAM of storage account, add role, add assignment and the choose storage blob data contributor. Select members and choose db\_access



## Step 3: Give the path a suitable variable name, source and create a schema using hard code, named schema1

```
[ ] source = 'abfss://test@deltaadbstg0300.dfs.core.windows.net/'
```

```
from pyspark.sql.types import StructType,StructField, StringType, IntegerType, DateType, FloatType, DoubleType

schema1 = StructType([
    StructField('Education_Level', StringType()),
    StructField('Line_Number', IntegerType()),
    StructField('Employed', IntegerType()),
    StructField('Unemployed', IntegerType()),
    StructField('Industry', StringType()),
    StructField('Gender', StringType()),
    StructField('Date_Inserted', StringType()),
    StructField('dense_rank', IntegerType())
])
```

## Step 4: Read the schema management file

```
df = (spark.read.format('csv')
      .option('header', 'true')
      .schema(schema1)
      .load(f'{source}/files/*.csv'))
```

## Step 5: Writing the data in parquet format and then go to test container files folder and that is parquet folder written. Now we need to read that parquet file written

```
(df.write.format('parquet')
  .mode('overwrite')
  .save(f'{source}/ParquetFolder/'))
```



```
df_parquet = (spark.read.format('parquet')
              .load(f'{source}/ParquetFolder/'))
```

Print the schema, then create a parquet view. The issue arises that update table is not supported with temp. I have used data lake not delta lake and these statements are not supported in data lake. To solve this issue, create the delta lake and copy the writing to format “delta” and save in delta. Now go to test container, open the delta folder and there will be two files, parquet file and delta log. Read this file and there will be parquet content of delta file. Create another notebook named “Transaction log”

# Q. What are databricks utilities?

Databricks Utilities (dbutils) are a set of functions provided by Databricks to help users interact with Databricks notebooks and clusters more efficiently. They facilitate tasks such as managing files, creating widgets, and running notebooks programmatically.

Statements:

1. Use dbutils.help() to see the list of available utilities.
2. Use dbutils.fs.ls(path) to list the contents of a directory.
3. Use dbutils.fs.mkdirs(path) to create new directories.
4. Use dbutils.fs.rm(path, recurse=True) to remove directories or files.
5. Use dbutils.fs.put(path, contents, overwrite=True) to create a file and write contents to it.
6. Use dbutils.fs.head(path) to read the first few lines of a file.
7. Use dbutils.fs.cp(source\_path, destination\_path, recurse=True) to copy files from one location to another.

# Q. How to create delta tables using SQL?

Data lakes give you flexibility at the cost of performance and reliability. Delta Lake is an open-source table format for data storage. Delta Lake improves data storage by supporting ACID transactions, high-performance query optimizations, schema evolution, data versioning and many other features.



Step 1: Create schema IF\_NOT\_EXISTS delta\_table

Step 2: Go to catalog of databricks workspace

Step 3: Create catalog named “metastore”, if unity catalog enabled workspace. It will create metastore in unity catalog

Step 4: Create table `delta`.deltafile (.....), use backticks

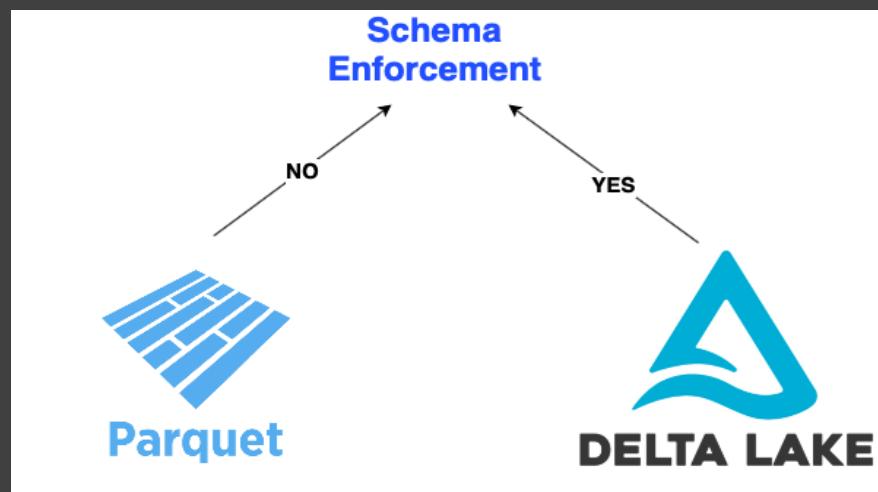
Step 5: Now go to catalog explorer and check delta file under hivemetastore

Step 6: Please select check type “managed” because of metadata nad data management by databricks itself

Step 7: Copy the particular location and list that using dbutils,fs.ls

Step 8: Now check delta log of delta file

# Q. How does schema enforcement works?



If your Delta Table has six columns and new data with an extra column is ingested, Delta Lake will reject this data. Delta Lake vs General Data Lake: In a general data lake, you can overwrite without schema validation, risking schema integrity. Delta Lake enforces schema, ensuring the data matches the defined schema, otherwise, it cancels the write operation and generates an error. Delta lake enforces schema validation on writes.

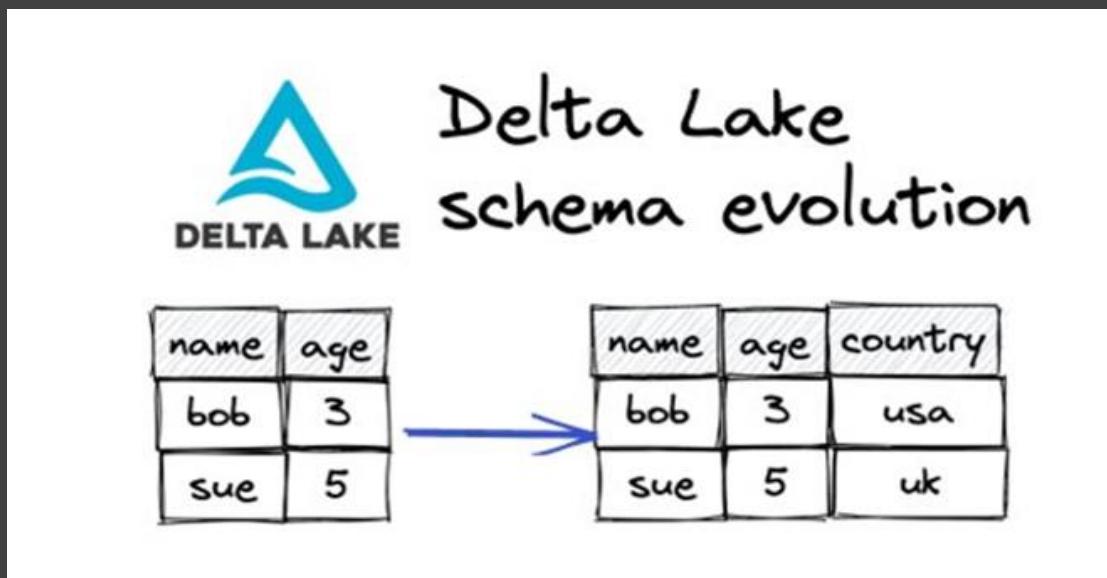
## Errors:

1. If the target table has six columns and incoming data has an extra column, the write operation fails with an error.
2. If the incoming data has fewer columns than the target table, the write operation is allowed, and missing columns are filled with null values.
3. If incoming data has mismatched data types (e.g., string vs. integer), the write operation fails with an error.

# Q. How does schema evolution works?

Schema evolution allows you to modify the schema of your Delta Lake tables to accommodate changes in the data structure over time, such as new columns or different data types.

- When new data arrives with additional columns not in the current schema, Delta Lake can automatically adjust the schema to include these new columns by setting the `mergeSchema` option to true during data write operations.
- If new data has different data types from the existing schema, setting `mergeSchema` to true won't be enough. In this case, you'll need to overwrite the existing schema with the new schema.



- Use **mergeSchema** option to merge the new columns into the target Delta table schema.

```
df.write.format("delta").option("mergeSchema",
  "true").mode("append").save("path/to/delta/table")
```

- Use **overwriteSchema** option to replace the existing schema with the new one.

```
df.write.format("delta").option("overwriteSchema",
  "true").mode("overwrite").save("path/to/delta/table")
```

## Q. What is vacuum command?

The **VACUUM** command deletes obsolete data files in Delta Lake that are no longer valid, helping manage storage costs. By default, VACUUM retains data for 7 days. Files older than this period are deleted.

- Data Management: Useful for projects needing recent data, such as clickstream data from the past few days.
- Governance: Use cautiously in production environments due to compliance and data governance concerns.

**Step by Step:**

### 1. Create a Delta Table:

```
df.write.format("delta").save("/path/to/delta/table")
```

### 2. Perform Operations:

```
# Insert, update, delete operations here
```

### 3. Check Files:

```
display(dbutils.fs.ls("/path/to/delta/table"))
```

### 4. Run VACUUM:

```
spark.sql("VACUUM '/path/to/delta/table'")
```

# Q. What is UPSERT?



- **Update:** Modify existing records if they match a common column in both source and destination tables.
- **Insert:** Add new records to the destination table if no match is found.

## Implementation Steps:

### 1. Create Tables:

- Source Table

```
CREATE TABLE source_table (line_number INT, employed INT);
INSERT INTO source_table VALUES (100, 4500), (101, 6500), (103, 7000);
```

- Destination Table:

```
sql
CREATE TABLE destination_table (line_number INT, employed INT);
INSERT INTO destination_table VALUES (100, 1500), (101, 2500);
```

### 2. Upsert Operation:

- Use the MERGE statement to perform the upsert.

```
%sql
MERGE INTO destination_table AS dst
USING source_table AS src
ON dst.line_number = src.line_number
WHEN MATCHED THEN UPDATE SET dst.employed = src.employed
WHEN NOT MATCHED THEN INSERT (line_number, employed)
VALUES (src.line_number, src.employed);
```

### 3. Verify Results:

Ensure existing records are updated and new records are inserted.

```
sql
SELECT * FROM destination_table;
```

## Outcome:

(100, 4500)  
(101, 6500)  
(103, 7000)

# Q. What is unity catalog and describe it's working?

Unity catalog is the central place to govern all data and user access in the Databricks environment. This blog is for beginners, as well as intermediate those, who have recently started exploring Databricks and its different modules

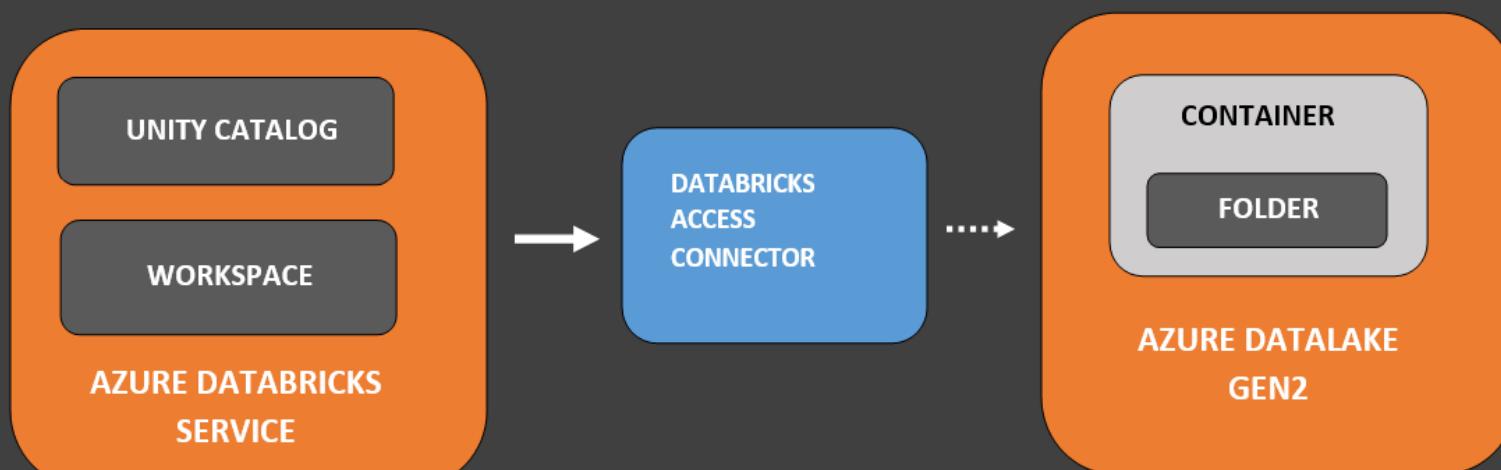
## Q. What is Hive Metastore?

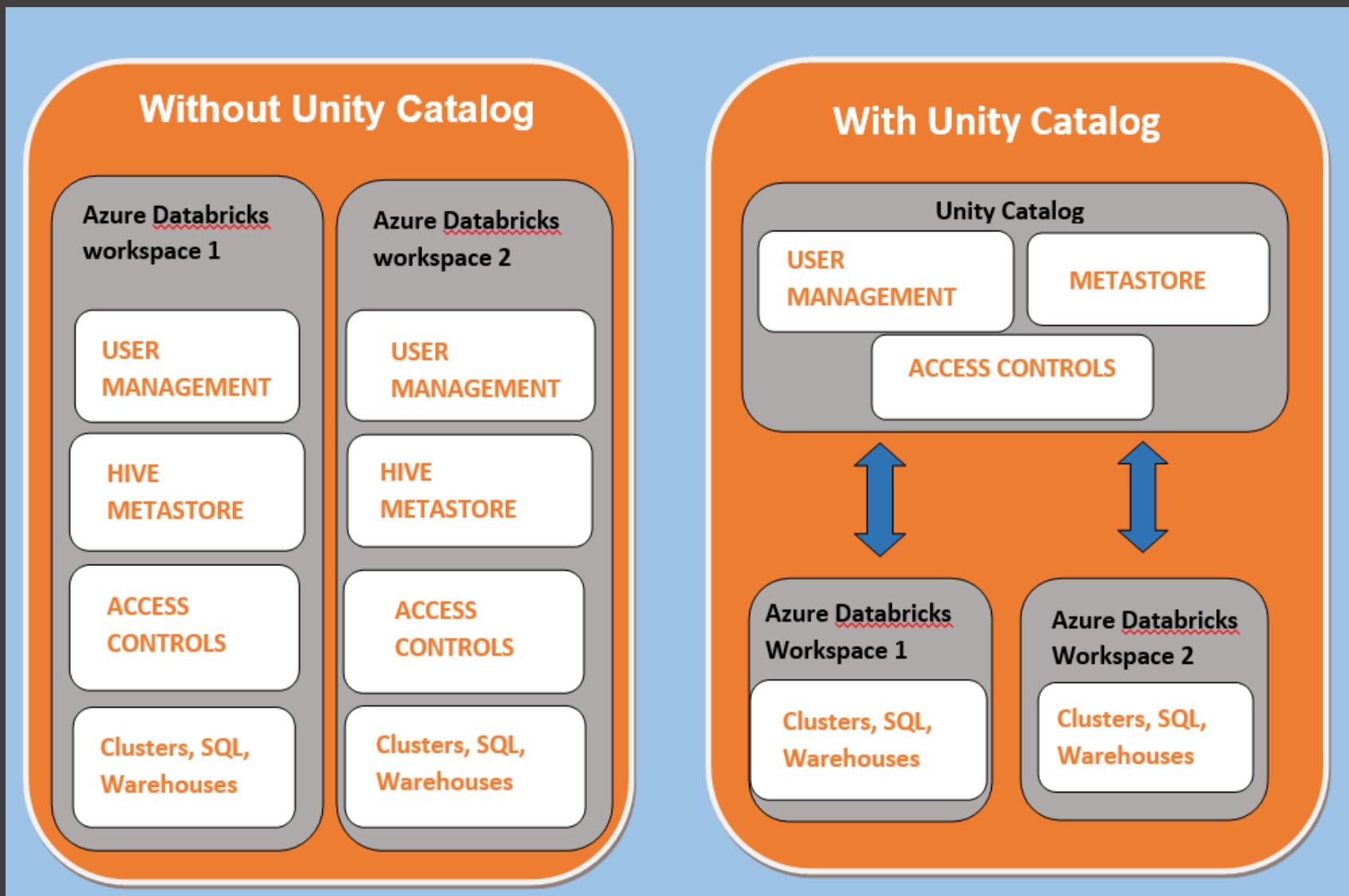
Hive Metastore is a component of the Apache Hive ecosystem that stores metadata for Hive tables, databases, partitions, and other related objects

## Q. Why need of Switching from Hive metastore to Unity Catalog?

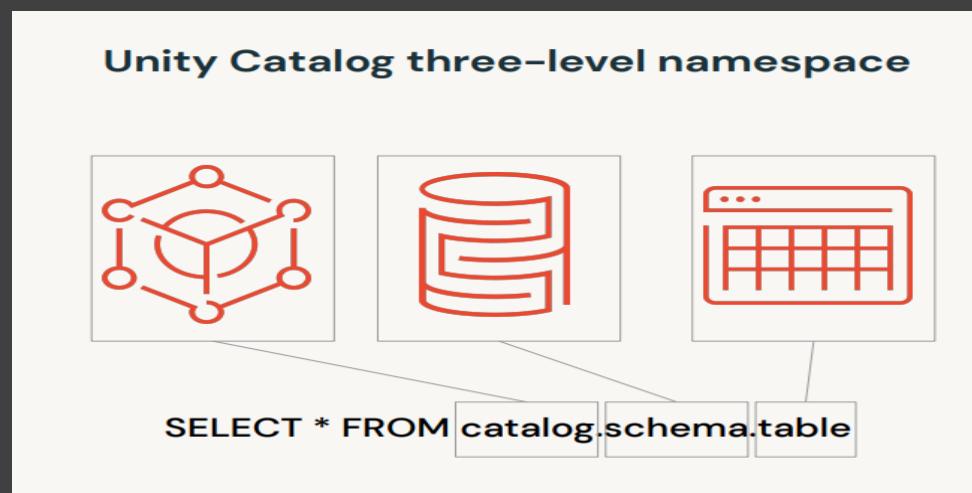
Unity catalog is overcoming the limitations of Traditional Hive metastore and providing more control over metadata and access management.

- 1- Unity Catalog provides native support for Delta Lake, a powerful storage layer that enhances reliability, performance, and data integrity in Databricks environments.
- 2- Delta Lake offers features such as ACID transactions, schema enforcement, time travel queries, and data versioning, which are not available in traditional Hive Metastore setups.
- 3- Unity Catalog offers advanced metadata management capabilities, including metadata versioning, lineage tracking, and data governance features. Users can track changes to data assets over time, understand data lineage, and enforce policies related to data access, security, and compliance more effectively compared to Hive Metastore.





Create databricksdevws, search access connector for databricks. Its resource id is used to manage metadata. The access connector has given to this storage account



Now create metastore in unity catalog by clicking on managed account of workspace. Create metastore named as dbs\_traffic\_project. Open azure and create container named; metastoreroot and create directory inside the container named; meta

**Create metastore**

**1 Create metastore**   **2 Assign to workspaces**

\* Name: dbtrafficproject

\* Region: eastus2  
Select the region for your metastore. You will only be able to assign workspaces in this region to this metastore.

ADLS Gen 2 path (optional) ⓘ  
abfss://metastoreroot@databricksdevstorage0300.dfs.core.windows.net/meta  
Optional location for storing managed tables data across all catalogs in the metastore.  
Once configured this path can't be removed. [Learn more](#)

Access Connector Id ⓘ  
/subscriptions/ea8d46ac-c7b3-4f01-a3ea-a860b2f2dca5/resourceGroups/databricksresource/providers/Microsoft...

Advanced options

## User management:

- 1) Invite and add users to Unity Catalog
- 2) Create groups
  - Workspace admins
  - Developers
- 3) Assign groups to users
  - Workspace admins – Anaya
  - Developers - Mikaeel
- 4) Assign roles to groups
  - Workspace Admin – **Workspace Admins Group**
  - Workspace User – **Developers Group**

Home > Default Directory | Users >

**Users** ...

Default Directory - Microsoft Entra ID

Search New user Download users Bulk operations Refresh Manage view Delete

All users

Azure Active Directory is now Microsoft Entra ID.

3 users found

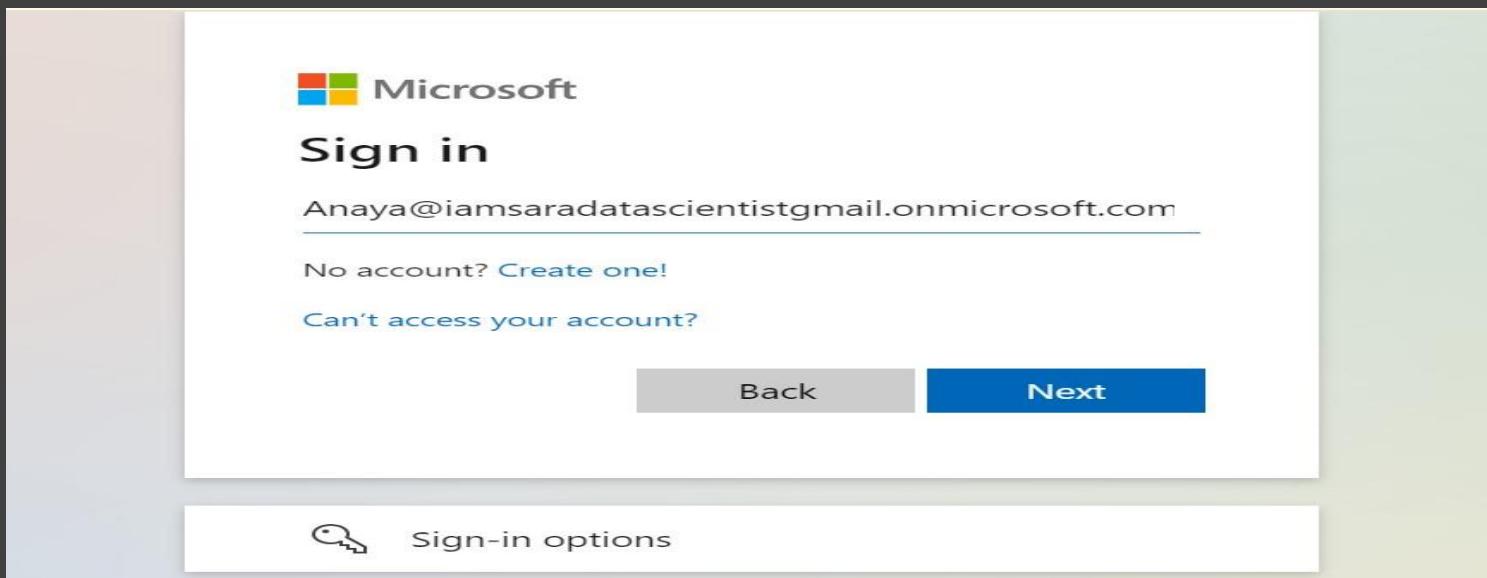
	Display name ↑	User principal name ↑	User type	On-premises sync...
<input type="checkbox"/>	AW Anaya Workspace	Anaya@iamsaradatascien...	Member	No
<input type="checkbox"/>	MD Mikaeel Developer	Mikaeel@iamsaradatascie...	Member	No
<input type="checkbox"/>	SI Sara Irfan	iamsara.datascientist_gm...	Member	No

**User management**

Users Service principals Groups

Add users to the account. Account users can use the account console to view and connect to their workspaces. Account admins can perform all of the management functions.

Name	Email	Status
Anaya Workspace	anaya@iamsaradatascientistgmail.onmicrosoft.com	Active
Sara Irfan	iamsara.datascientist@gmail.com	Active
Mikael Developer	mikaeel@iamsaradatascientistgmail.onmicrosoft.com	Active



**User management**

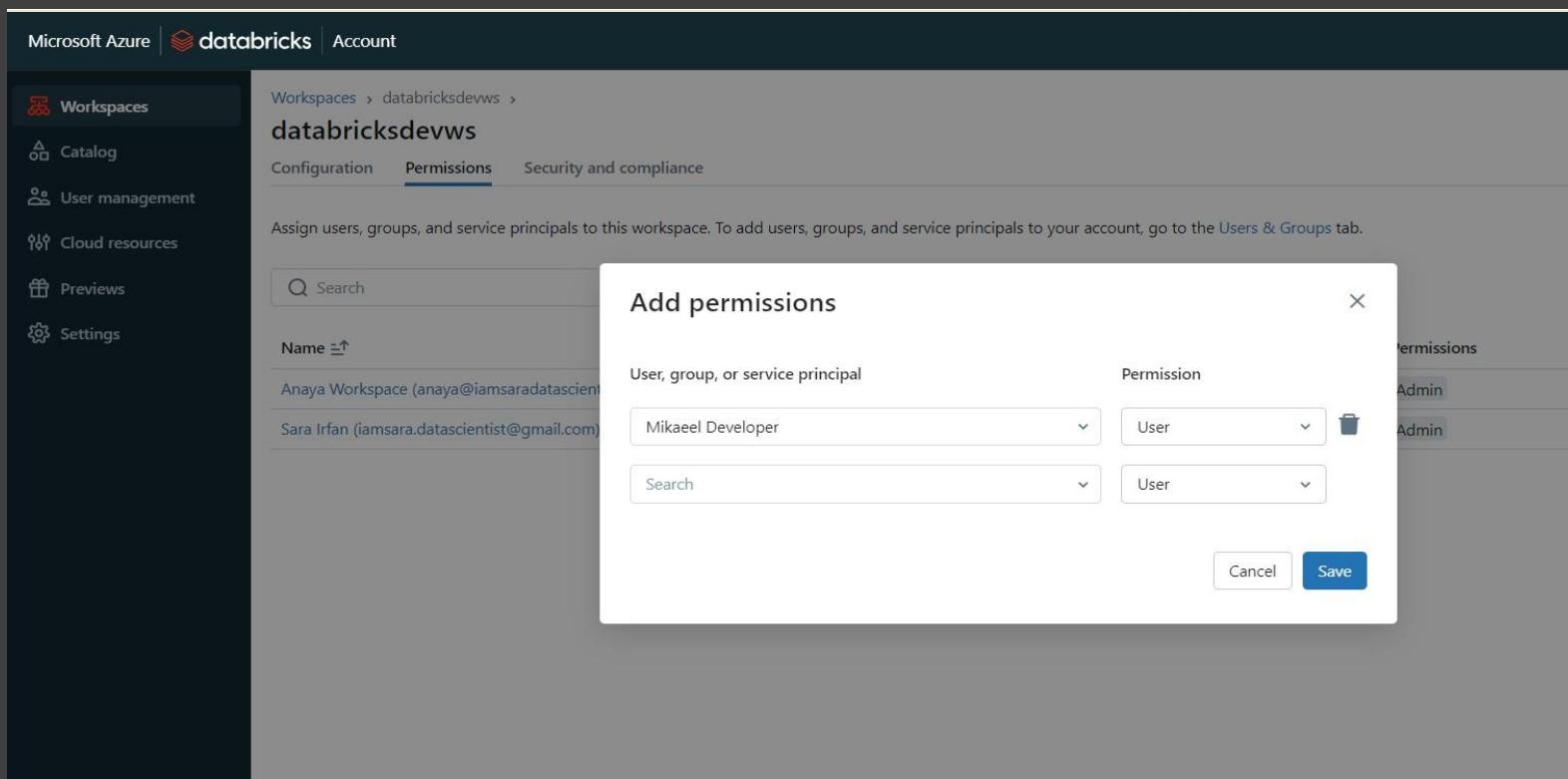
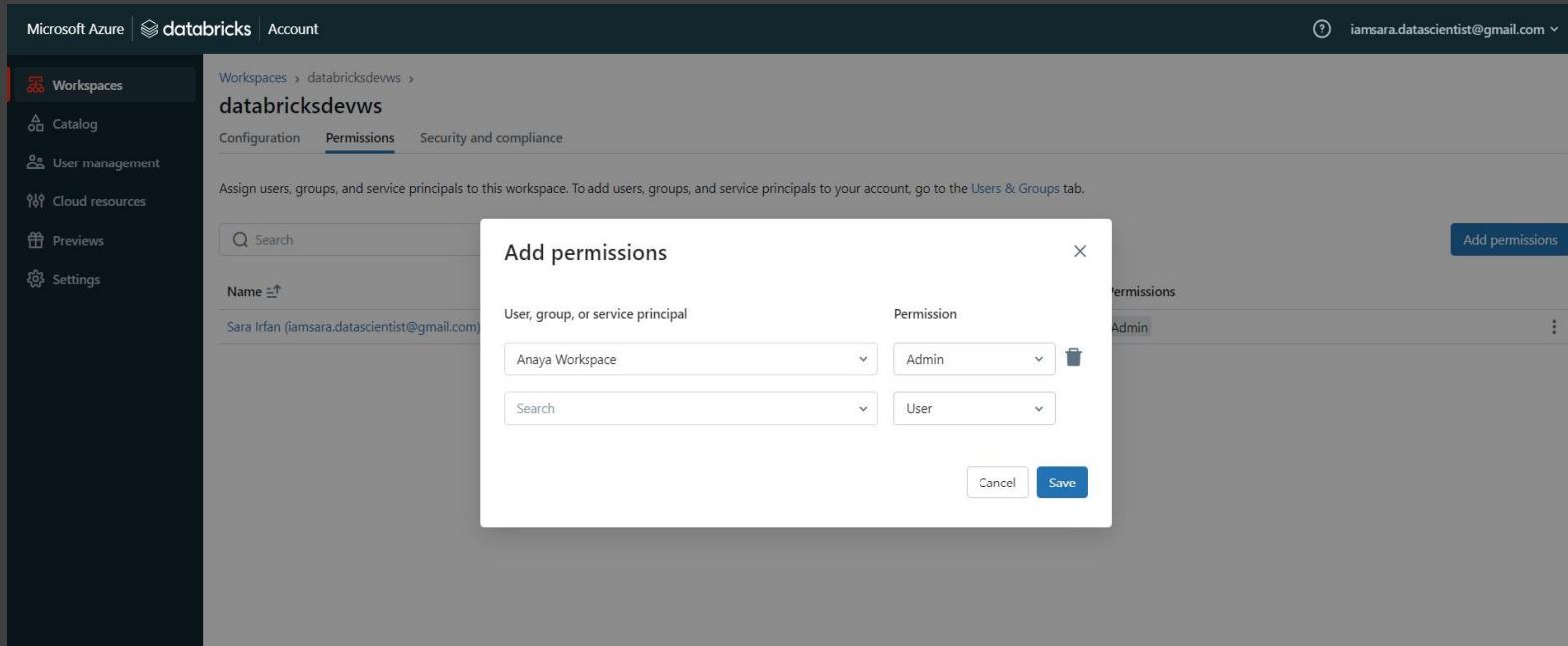
Users Service principals Groups

Filter groups

Name	Members
Developers	1
Workspace Admins	1
account users	3

# Q. How to give permissions to specific users?

By clicking on workspaces, permissions, add permissions



databricksdevws

Configuration Permissions Security and compliance

Assign users, groups, and service principals to this workspace. To add users, groups, and service principals to your account, go to the [Users & Groups tab](#).

Search

Name	Type	Permissions
Anaya Workspace (anaya@iamsaradatascientist@gmail.onmicrosoft.com)	User	Admin
Mikael Developer (mikael@iamsaradatascientist@gmail.onmicrosoft.com)	User	User
Sara Irfan (iamsara.datascientist@gmail.com)	User	Admin

## Q. How to create Cluster Pools and Policy?

Clusters usually take time to start and auto-scale, in order to minimize the time for a cluster to start cluster pools, are used. A cluster pool is basically a set of idle ready-use virtual machines that allows us to reduce the time taken by clusters to start and auto-scaling times

Microsoft Azure | databricks

+ New

- Workspace
- Recents
- Catalog
- Workflows
- Compute**
- SQL
  - SQL Editor
  - Queries
  - Dashboards
  - Alerts
  - Query History
  - SQL Warehouses
- Data Engineering
  - Job Runs
  - Data Ingestion
  - Delta Live Tables
- Machine Learning
  - Playground
  - Experiments
  - Features
  - Models
  - Serving

Compute > Policies

Create policy

Project Defaults

Family

Custom

Description

Optional

Definitions Libraries Permissions

How to define a policy?

```

1 {
2   "node_type_id" : {
3     "type" : "allowlist",
4     "values" : [ "Standard_DS3_v2" ]
5   },
6   "spark_version" : {
7     "type" : "fixed",
8     "Value" : "13.3.x_Scala2.12"
9   },
10  "runtime_engine" : {
11    "type" : "fixed",
12    "value" : "STANDARD",
13    "hidden" : true
14  },
15  "num_workers" : {
16    "type" : "fixed",
17    "value" : 0,
18    ...
19  }
20}

```

Compute > Policies

## Create policy

Definitions   Libraries   Permissions

How to define a policy?

```

17     "value" : 0,
18     "hidden" : true
19   },
20   "data_security_mode" : {
21     "type" : "fixed",
22     "values" : "SINGLE_USER"
23   },
24   "cluster_type" : {
25     "type" : "fixed",
26     "value" : "all-purpose"
27   },
28   "instance_pool_id" : {
29     "type" : "forbidden",
30     "hidden" : true
31   },
32   "azure_attributes.availability" : {
33     "type" : "fixed",
34     "value" : "ON_DEMAND_AZURE",
35     "hidden" : true
36   },
37   "spark_conf.spark.databricks.cluster.profile" : {
38     "type" : "fixed",
39     "value" : "singleNode",
40     "hidden" : true
41   },
42   "autotermination_minutes" : {
43     "type" : "fixed",
44     "Value" : 4000
45   }
46 }
```

Microsoft Azure |  Account

 Workspaces     Catalog     User management     Cloud resources     Previews     Settings

User provisioning   App connections   Feature enablement   Language settings   My preferences   Security and compliance   Account settings

Some features require you to explicitly opt-in before you can use them. Use this page to enable and configure the following account features.

Changes made to the settings on this page affect all workspaces in your Microsoft Entra ID tenant.

**Serverless compute for Workflows, Notebooks, and Delta Live tables**    Disabled

Start up a job, notebook, or pipeline in seconds instead of minutes. At this time, Serverless Jobs, Notebooks, and DLT have unrestricted access to the public Internet.

[Documentation](#)

---

**Personal Compute**    [Enable for all](#) 

Enables all users to create single-machine compute using the Personal Compute default policy. [Learn more](#)

---

**Web Terminal**    [On](#) 

Grants all users access to the web terminal, where they can interact with the command line on their all-purpose compute resources. [Learn more](#)

Microsoft Azure |  databricks

Compute > Policies

## Project Defaults

Name: Project Defaults

Family: Custom

Description: Optional

Definitions Libraries Permissions **Permissions**

Max compute resources per user: Unlimited

NAME	PERMISSION
Admins	Can Use inherited
Mikaeel Developer (mikaeel@iamsaradatascientistgmail...)	Can Use
Anaya Workspace (anaya@iamsaradatascientistgmail.o...)	Can Use
Select user, group or service principal...	Can Use
<b>Add</b>	

Home >

 **databricksdevws** ⚡ ⭐ ...

Azure Databricks Service

Search Delete

**Overview**

- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Settings
  - Virtual Network Peering
  - Encryption
  - Networking

**Essentials**

Status: Active

Resource group: [databrickresource](#)

Location: East US

Subscription: [Azure subscription 1](#)

Subscription ID: ea8d46ac-c7b3-4f01-a3ea-a860b2f2dca5

Tags ([edit](#)) [Add tags](#)

Managed Resource Group: [databricks-rg-databricksdevws-5dtlgclwhxpo](#)

URL: <https://adb-276878930788483.3.azure.databricks.net>

Pricing Tier: Premium (+ Role-based access controls) (Click to change)

Microsoft Azure |  databricks

Compute

All-purpose compute Job compute SQL warehouses Vector Search Pools Policies [①](#)

Create pool

Name	Instance type	Min idle	Max capacity	Idle instances	Used instances
+ Create an instance pool					

Create an instance pool  
Your workspace has no pools. Learn more about instance pools

Create pool

Microsoft Azure | databricks

Search data, notebooks, recents, and more... CTRL + P

databricksdevws ✓ A

**New**

Compute > Pools

### available pool

Overview Configuration

Pool ID: 0727-155102-ward25-pool-m2my10v6

Min Idle: 1 Total instances: 1

Instance Type: Standard\_DS3\_v2, 14 GB Memory, 4 Cores Max Capacity: 2

Used: 0 Idle: 0 (Pending: 1) Max Capacity: 2

Idle Instance Auto Termination: 60 minutes

Attached compute

Name	Nodes from pool	Runtime	Creator	Pool role
[Empty]	[Empty]	[Empty]	[Empty]	[Empty]

Search attached compute

## Q. How to create catalog and how it can be viewable by all other users?

Create devcatalog and choose standard. After creating catalog, there will be two default schemas; default and information schema. Access of viewing this catalog by other users using following query

**GRANT privilege\_type ON securable\_object TO principal**

**Open catalog from Sara(admin account), click on devcatalog, permissions, grant**

The screenshot shows the Databricks Catalog interface. On the left, there's a sidebar with various navigation options like New, Workspace, Recents, Catalog (which is selected), Workflows, Compute, SQL, SQL Editor, Queries, Dashboards, Alerts, Query History, SQL Warehouses, Data Engineering, and Job Runs. The main area is titled 'Catalog' and shows a tree view of catalogs under 'Serverless Starter Warehouse Serverless'. The 'devcatalog' catalog is currently selected. A search bar at the top right says 'Search data, notebooks, recents, and more...' and has a 'CTRL + P' keybinding. Below the search bar are buttons for 'Delta Sharing', 'External Data', and 'Add data'. A 'Quick access' section includes 'Recents', 'Favorites', and 'Catalogs' buttons, along with 'Create catalog' and 'Filter' buttons. To the right, a table lists existing catalogs with columns for Name, Owner, and Created at.

Name	Owner	Created at
databricksdevws	_workspace_admins_databricksd...	2024-07-25
devcatalog	iamsara.datascientist@gmail.com	2024-07-28
hive_metastore		
samples		
system	System user	2024-06-29

## Q. What is the difference between external tables and managed table?

### Managed Tables

- Defined without a specified location.
- Data files stored in managed storage in Delta format.
- Dropping the table removes metadata and deletes actual data, but in Unity Catalog, the underlying data is retained for 30 days.

### External Tables

- Require an EXTERNAL LOCATION and STORAGE CREDENTIALS for access.
- Defined for a custom file location outside managed storage.
- Dropping the table deletes metadata from the catalog but does not affect the data files.

```
SELECT * FROM `catalog`.schema.table
But we need permission for schema too
```

## Q. How to create storage credentials and external locations in Azure Databricks with Unity Catalog?

## 1. Creating storage credential and external location:

- **Performed by a workspace admin through the Anaya Portal.**
- **Go to the catalog section to find external data.**
- **Note: External data options might be missing; this is managed at the metastore level.**
- **Need Metastore admin or account admin role to manage external locations and storage credentials.**
- **Typically, one person in the project will have these roles.**
- **Navigate to the external data section in the catalog.**
- **Existing storage credentials will show up.**
- **For new credentials:**
  - **Create a storage credential with a name and access connector ID.**
  - **Use the Databricks Access connector managed identity.**
  - **Assign the necessary role (e.g., Storage Blob Data Contributor**
- **Now create external location by defining a path for the external location, typically different from the default metastore path.**
- **Ensure the access path is correctly set up to the desired container and folders.**
- **Test the connection to validate the setup.**

## 2. Accessing Data:

- **Create a notebook to run commands and access data using the external location.**
- **Set up a compute cluster and run the commands.**
- **External location checks for storage credentials and retrieves data.**

## 3. Handling Permissions:

- **If accessing data from different containers within the same storage account, create a new external location for each container.**
- **Assign fine-grained access control for different users or roles.**
- **Example: Grant read access to developers.**

## 4. Practical Demonstration:

- **Create a notebook to test data access.**
- **Set up shared compute clusters to allow access for multiple users.**
- **Validate data access permissions and test reading and writing data.**

## 5. Conclusion:

- **External locations in Unity Catalog provide controlled access to data stored in Azure.**
- **Fine-grained access control ensures secure data handling within the team.**
- **Proper role assignment and permissions are crucial for managing external data.**

The screenshot shows the Microsoft Azure Databricks interface. The top navigation bar includes 'Microsoft Azure' and the 'databricks' logo. A search bar says 'Search data, notebooks, recents, and more...' with a 'CTRL + P' keyboard shortcut. The left sidebar has a 'New' button and sections for 'Workspace', 'Recents', 'Catalog' (which is selected), 'Workflows', 'Compute', 'SQL' (with 'SQL Editor', 'Queries', 'Dashboards', 'Alerts', 'Query History', and 'SQL Warehouses' sub-options), 'Data Engineering' (with 'Job Runs' and 'Data Ingestion' sub-options). The main content area shows 'Catalog Explorer > Storage Credentials > storagecredential'. It displays details: 'Owner: Sara Irfan', 'Credential Type: Managed Identity', 'Connector Id: /subscriptions/ea8d46ac-c7b3-4f01-a3ea-a860b2f2dca5/resourceGroups/databricksresource/providers/Microsoft.Databricks/accessConnectors/dbaccessconnector', and 'User Assigned Managed Identity Id:'. Below this, there are tabs for 'Usage', 'Permissions' (which is selected), and 'Workspaces'. Under 'Permissions', there are 'Grant' and 'Revoke' buttons. A table lists 'Principal', 'Privilege', and 'Object', with a note 'No permissions granted yet.'

## Creating an external location:

External Locations >

### Create a new external location

An external location is a cloud storage url (and paired credential) that allows access to data stored on your cloud tenant. [Learn more](#)

**Name**:

**\* Storage credential** [Learn more](#):

Connector Id: /subscriptions/ea8d46ac-c7b3-4f01-a3ea-a860b2f2dca5/resourceGroups/databricksresource/providers...

User Assigned Managed Identity Id:

**\* URL** [Learn more](#):

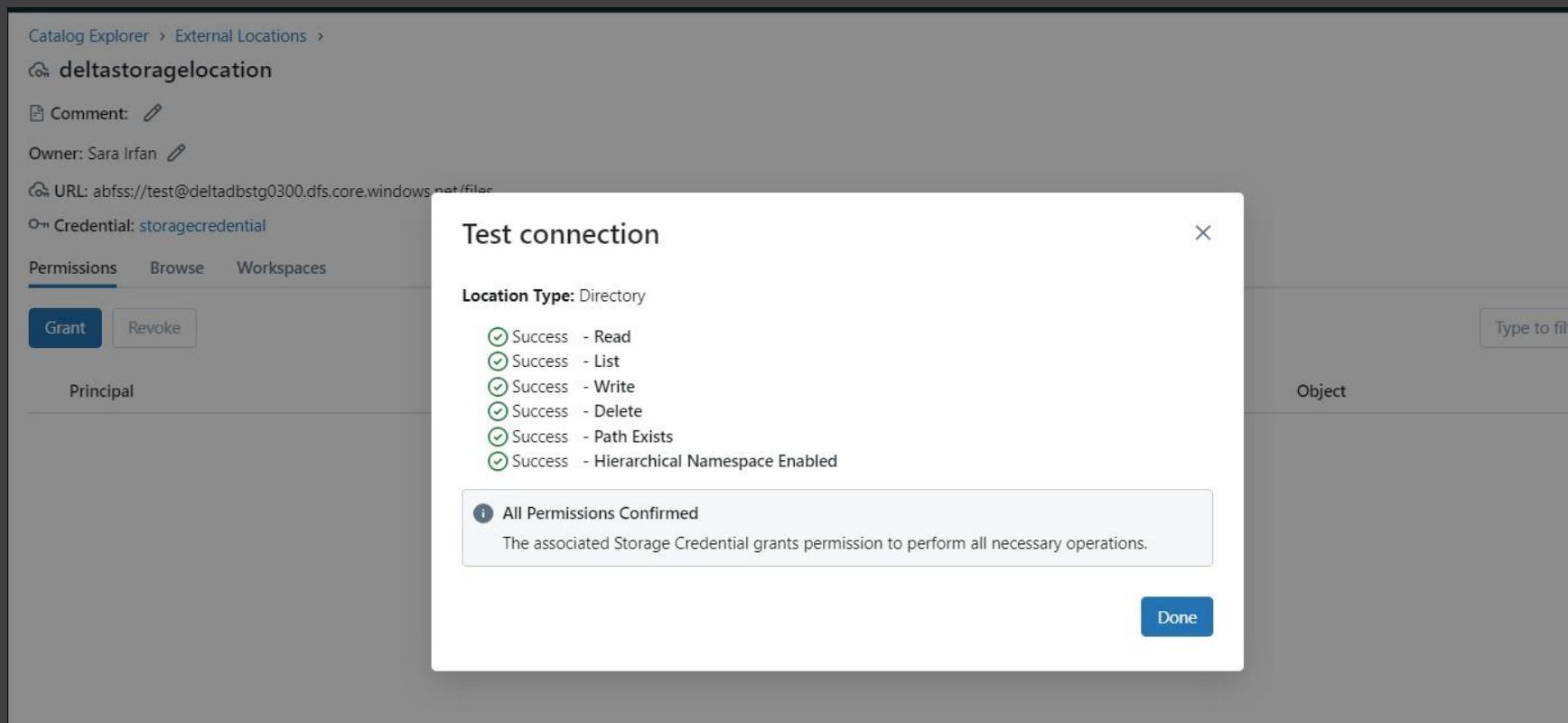
Enter the bucket path that you want to use as the external location

**Comment**:

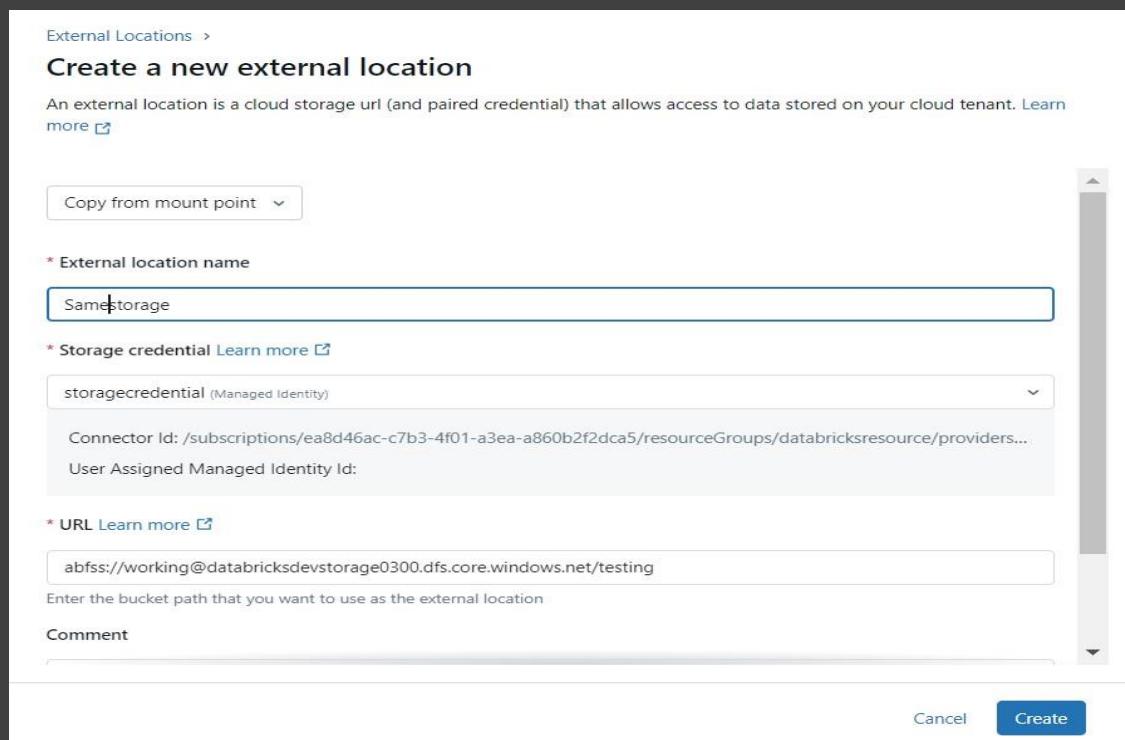
[Advanced Options](#)

[Cancel](#) [Create](#)

## Testing a connection:



## Creating another external location:



# Q. How to handle Spark Structured Streaming?

## 1. What is Streaming Data?

- Streaming data refers to data that is continuously generated and never completely stops because it keeps coming in.

## 2. Conceptualization:

- Understand how structured streaming works:

- Source: Imagine an IoT device collecting car details.
- Data: Data flows from thousands of cars or a single logging system.

## 3. Data Stream and Destination Table:

- Data Stream: Consider the incoming data as a stream. This data is written to an unbounded table (a table without limits).
- Unbounded Table: This means there is no limit to the data. As data arrives, it is written to the destination table.

## 4. Spark Structured Streaming:

- New Data: Each new piece of data is added as a new row.
- Transformation: Streaming data is processed continuously in micro-batches.
- Cluster Creation: Create a cluster and name a notebook "Streaming Basics."

Sara Irfan's Cluster

Configuration Notebooks (1) Libraries Event log Spark UI Driver logs Metrics Apps Spark compute UI - Master More ... Terminate Edit Open in new tab

Jobs Stages Storage Environment Executors SQL / DataFrame JDBC/ODBC Server Structured Streaming

Spark Jobs (?)

User: root  
Total Uptime: 7.4 min  
Scheduling Mode: FAIR  
Completed Jobs: 1

Event Timeline

Completed Jobs (1)

Job Id (Job Group)	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0 (652d1a52-2c0f-4caa-aee5-c3fe86e25ff0)	display_query_1 id = 090b917a-99f6-41fc-ab09-ec195391b117 runId = 652d1a52-2c0f-4caa-aee5-c3fe86e.. start at DriverSparkHooks.scala:147	2024/07/28 09:53:18	2 s	2/2	2/2

## 5. File Source Setup:

- **File Upload:**

- Click on "Data," then "DBFS," and then "File Store."
- Create a new folder named "streaming."
- Upload files like "countries1.csv."
  - countries1.csv:
- This file contains two columns: Country Name and Number of Citizens.

## 6. Data Reading:

- **Read Stream:**

- Copy the path and use spark.readStream to read the data.

```
Spark Structured Streaming Python ▾ ☆
File Edit View Run Help Last edit was 3 minutes ago Provide feedback
Interrupt Sara Irfan's Cluster
```

```
1
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, FloatType

schema = StructType([
    StructField('Country', StringType()),
    StructField('Citizens', IntegerType())
])
```

**Now read file with schema**

```
3
df = spark.readStream.format("csv")\
    .option('header', 'true')\
    .schema(schema)\
    .load('dbfs:/FileStore/tables/streaming')

df: pyparq.sql.DataFrame = [Country: string, Citizens: integer]
```

```
4
display(df)

display_query_1 (id: 68506d23-686c-4622-a45c-7e646e692c6e) Last updated: 5 seconds ago
Query returned no results
```

- Specify the schema explicitly to avoid performance issues. Do not use infer schema.
- Use the display() method to view the data.

	Country	Citizens
1	India	10
2	USA	5
3	China	10
4	India	10
5	Canada	40
6	Brazil	10

6 rows

- Refer to the directory, not the CSV file directly, to avoid errors.

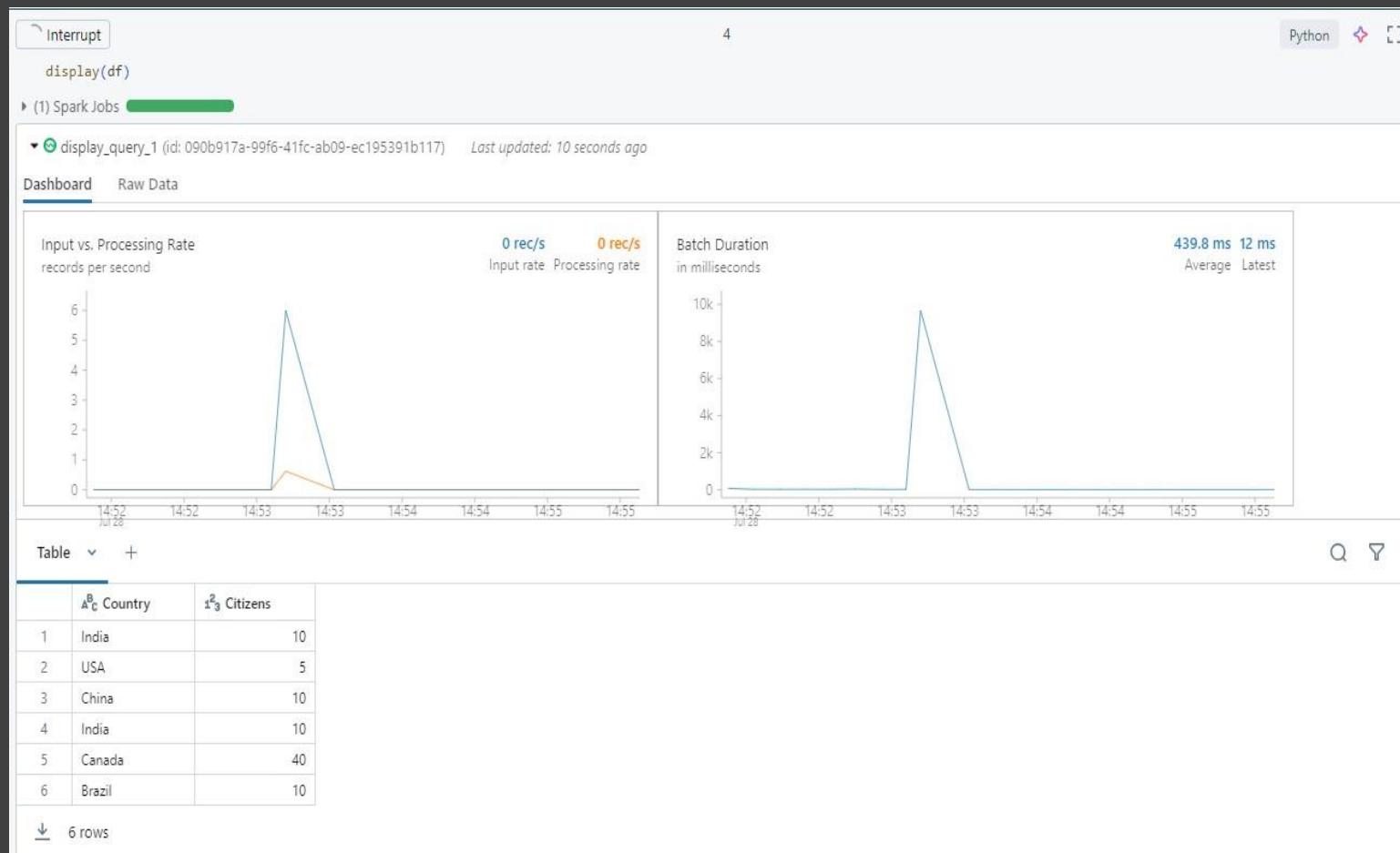
## 7. Streaming Query Monitoring:

- Data is read from the streaming source, transformed, and written to the destination.
- This process happens in micro-batches.

## 8. Streaming Query Acting as a Watcher:

- The streaming query acts as a watcher, continuously checking for new data.
- The streaming query runs continuously in the background.

## 9. Stopping the Streaming Query: To stop the streaming query, click on the "Cancel" button and confirm the action.



Adding more data in the container of data lake storage, the number of rows are increased

( Interrupt

```
display(df)
```

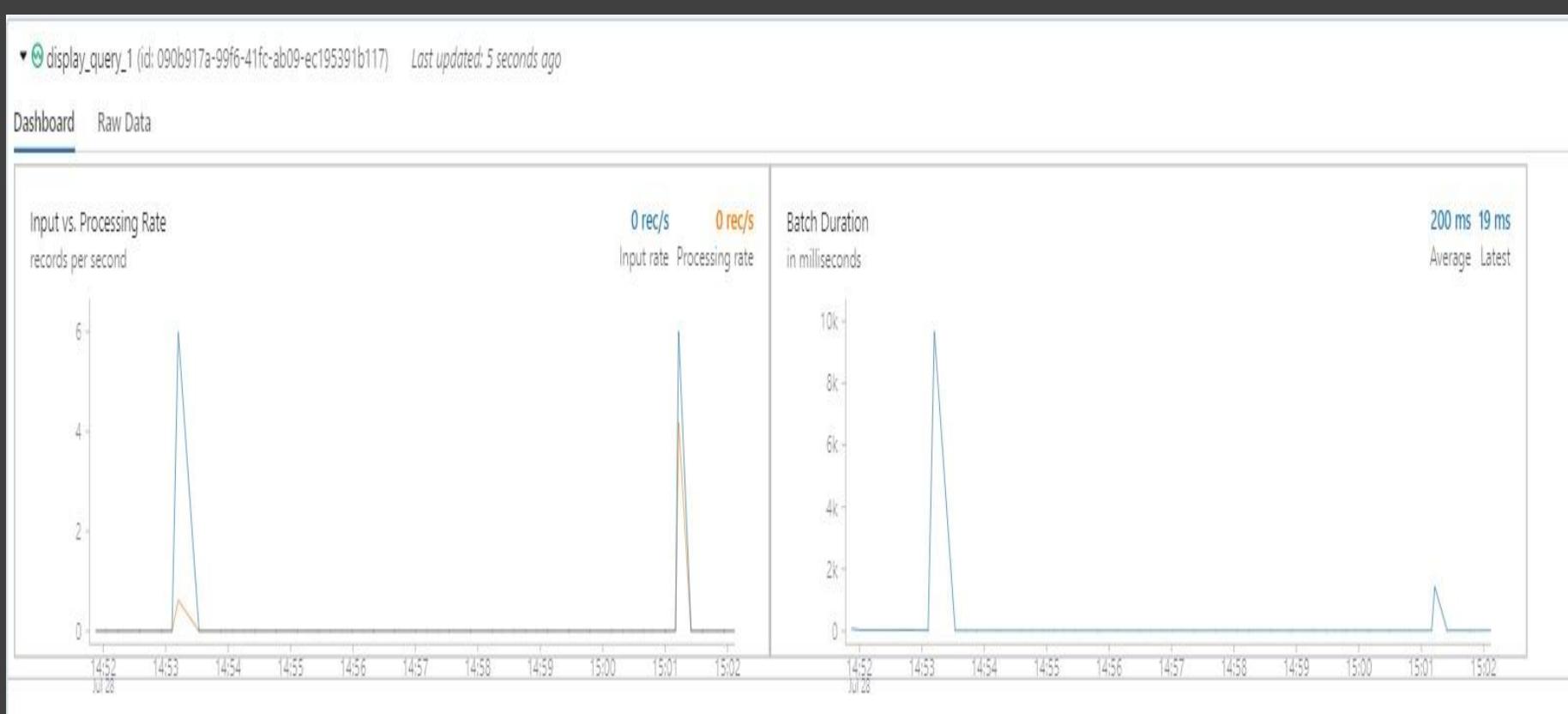
► (1) Spark Jobs [green bar]

display\_query\_1 (id: 090b917a-99f6-41fc-ab09-ec195391b117) Last updated: 10 seconds ago

Table ▾ +

	Country	Citizens
1	India	10
2	USA	5
3	China	10
4	India	10
5	Canada	40
6	Brazil	10
7	India	5
8	USA	10
9	China	5
10	India	5
11	Canada	10
12	Brazil	50

↓ 12 rows



Now another csv file has been uploaded and rows are 15 now

( Interrupt

```
display(df)
```

► (1) Spark Jobs [green bar]

display\_query\_1 (id: 090b917a-99f6-41fc-ab09-ec195391b117) Last updated: 5 seconds ago

Table ▾ +

	Country	Citizens
1	India	10
2	USA	5
3	China	10
4	India	10
5	Canada	40
6	Brazil	10
7	India	5
8	USA	10
9	China	5
10	India	5
11	Canada	10
12	Brazil	50
13	India	15
14	USA	20
15	China	10

↓ 15 rows

## Check out the spark jobs from databricks workspace environment

The screenshot shows the 'Spark Jobs' page in Databricks. It displays the following information:

- User: root
- Total Uptime: 14 min
- Scheduling Mode: FAIR
- Completed Jobs: 3
- Event Timeline (link)
- Completed Jobs (3) (link)
- Page: 1

Job Id (Job Group) *	Description
2 (652d1a52-2c0f-4caa-aee5-c3fe86e25ff0)	display_query_1 id = 090b917a-99f6-41fc-ab09-ec195391b117 runId = 652d1a52-2c0f-4caa-aee5-c3fe86e25ff0 batch = 2 start at DriverSparkHooks.scala:147
1 (652d1a52-2c0f-4caa-aee5-c3fe86e25ff0)	display_query_1 id = 090b917a-99f6-41fc-ab09-ec195391b117 runId = 652d1a52-2c0f-4caa-aee5-c3fe86e25ff0 batch = 1 start at DriverSparkHooks.scala:147
0 (652d1a52-2c0f-4caa-aee5-c3fe86e25ff0)	display_query_1 id = 090b917a-99f6-41fc-ab09-ec195391b117 runId = 652d1a52-2c0f-4caa-aee5-c3fe86e25ff0 batch = 0 start at DriverSparkHooks.scala:147

## Check out the streaming query validation under the strutured streaming of spark UI using specified cluster

The screenshot shows the 'Spark UI' tab in the Databricks interface. It displays the following navigation tabs:

- Configuration
- Notebooks (1)
- Libraries
- Event log
- Spark UI**
- Driver logs
- Metrics
- Apps
- Spark compute UI - Master

Below the tabs, there is a sub-navigation bar:

- Jobs
- Stages
- Storage
- Environment
- Executors
- SQL / DataFrame
- JDBC/ODBC Server
- Structured Streaming**

The main content area is titled 'Streaming Query' and shows the following details:

- Active Streaming Queries (1)
- Page: 1

Name	Status	ID	Run ID
display_query_1	RUNNING	090b917a-99f6-41fc-ab09-ec195391b117	652d1a52-2c0f-4caa-aee5-c3fe86e25ff0

The screenshot shows a metrics table for a streaming query. The table has the following columns:

- Start Time
- Duration
- Avg Input /sec
- Avg Process /sec
- Latest Batch

The data in the table is:

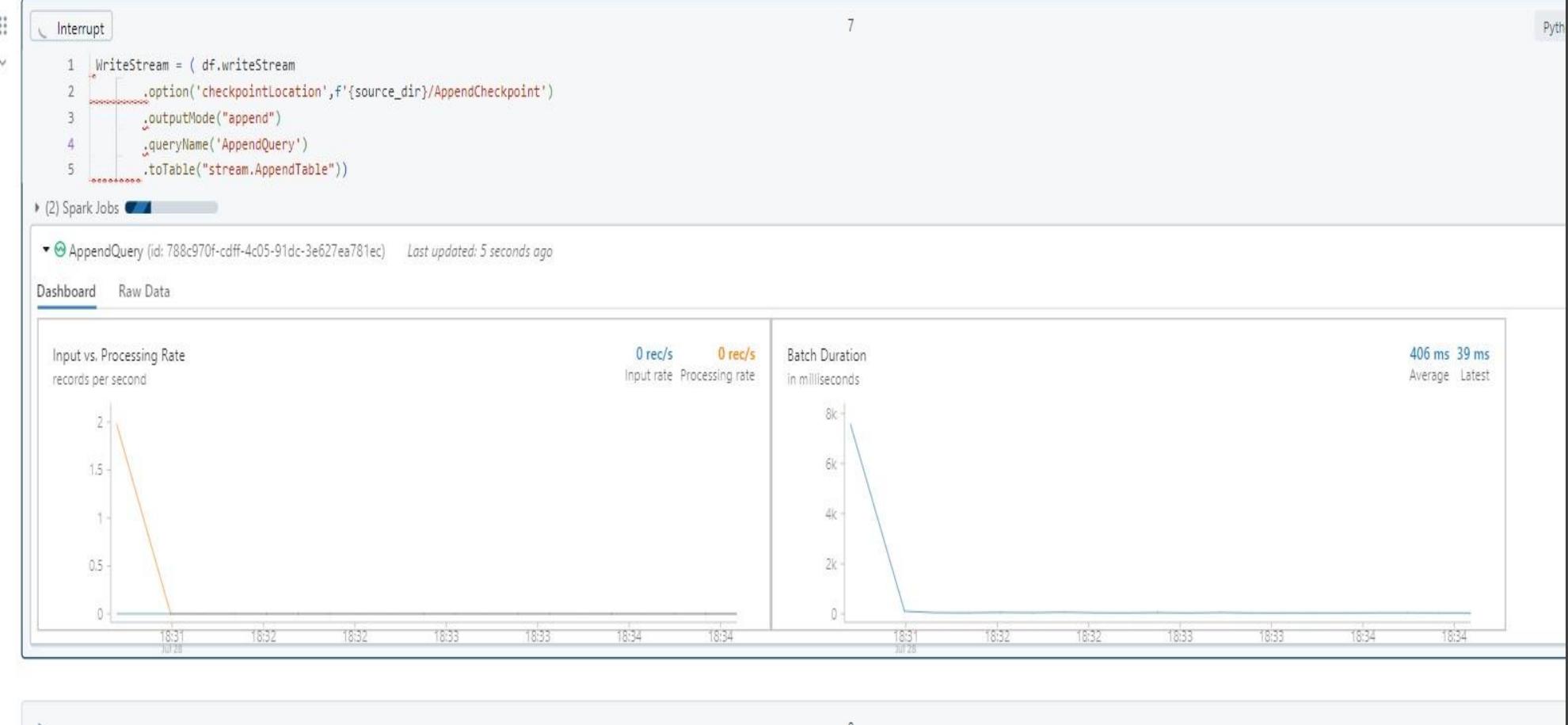
Start Time	Duration	Avg Input /sec	Avg Process /sec	Latest Batch
2024/07/28 09:51:52	13 minutes 7 seconds	5.00	2.71	2

Below the table, there is a pagination control:

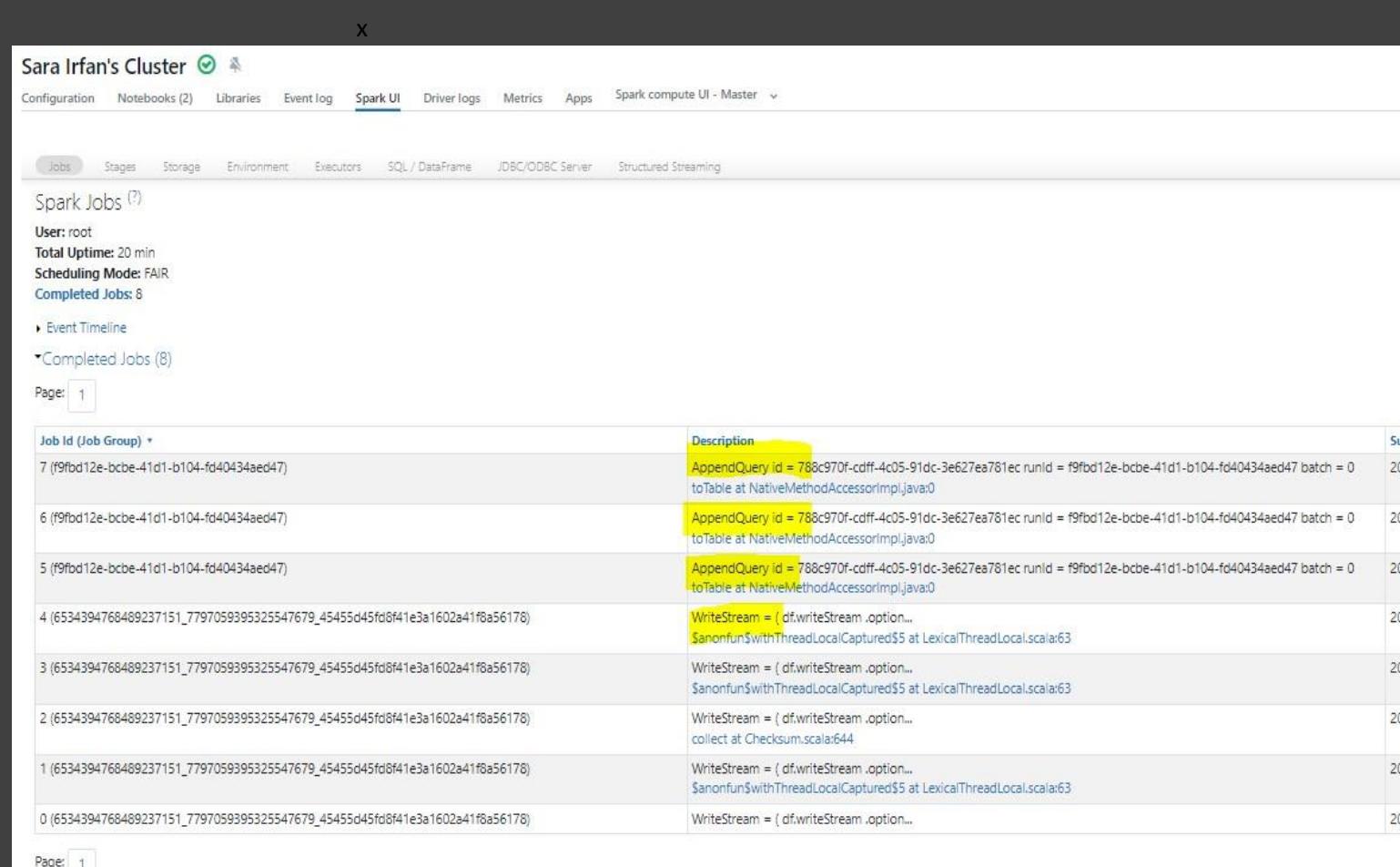
1 Pages. Jump to  . Show  items in a page.

## Writing the data with writeStream

We need specific schema to write this



7



8

# Q. What are the steps for Data Engineering Project in Azure Databricks using three layer architecture?

In this project, we will implement a multi-layered data architecture using Azure Databricks. We will follow a structured approach to create schemas, ingest data, perform transformations, and prepare the data for analytical consumption. The layers we will work with are Bronze, Silver, and Gold.

The screenshot shows the 'Containers' blade for the storage account 'databricksdevstorage0300'. On the left, there's a sidebar with various options like Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, Storage browser, and Data storage. Under Data storage, 'Containers' is selected. In the main area, there's a search bar and a table listing existing containers: '\$logs', 'checkpoints', 'landing', 'medallian', 'metastoreroot', and 'working'. A new container named 'medallian' has just been created, as indicated by the green checkmark icon and the message 'Successfully created' at the top right. The 'Containers' tab is highlighted in yellow.

## Data Sources

1. Traffic Data: Contains raw traffic counts and vehicle type information.
2. Roads Data: Contains information about road categories, types, lanes, lengths, and vehicle counts.

The screenshot shows the 'medallian' container blade. The sidebar includes Overview, Diagnose and solve problems, Access Control (IAM), and Settings. The main area shows the authentication method as 'Access key (Switch to Microsoft)' and the location as 'medallian'. There's a search bar for blobs and a table with columns 'Name', 'Last modified', and 'Anonymous access level'. Three directories are listed: 'bronze', 'gold', and 'silver', all of which are private. The 'bronze' directory is highlighted in yellow.

## Project Workflow

1. Create Schemas
  - o Bronze Schema
  - o Silver Schema

- **Gold Schema**
- 2. **Create Raw Tables in Bronze Layer**
  - Raw Traffic Table
  - Raw Roads Table
- 3. **Data Ingestion**
  - Ingest data from the landing zone to the bronze layer.
- 4. **Transformations and Quality Checks**
  - Apply transformations and quality checks on the bronze layer data.
  - Create Silver Tables.
  - Apply business-level transformations to create Gold Tables.
- 5. **Gold Layer**
  - Create Gold Tables for reporting purposes in Power BI.
- 6. **Delta Tables**
  - Use Delta format for managed tables to leverage transactional capabilities and performance advantages.

## Detailed Steps

### Step 1: Create External Locations

#### 1. Landing Zone

`abfss://landing@databricksdevstorage0300.dfs.core.windows.net/`

#### 2. Checkpoints

`abfss://checkpoints@databricksdevstorage0300.dfs.core.windows.net/`

#### 3. Bronze

`abfss://medallion@databricksdevstorage0300.dfs.core.windows.net/bronze/`

#### 4. Silver

`abfss://medallion@databricksdevstorage0300.dfs.core.windows.net/silver/`

#### 5. Gold

`abfss://medallion@databricksdevstorage0300.dfs.core.windows.net/gold/`

### Step 2: Test Connection for External Locations

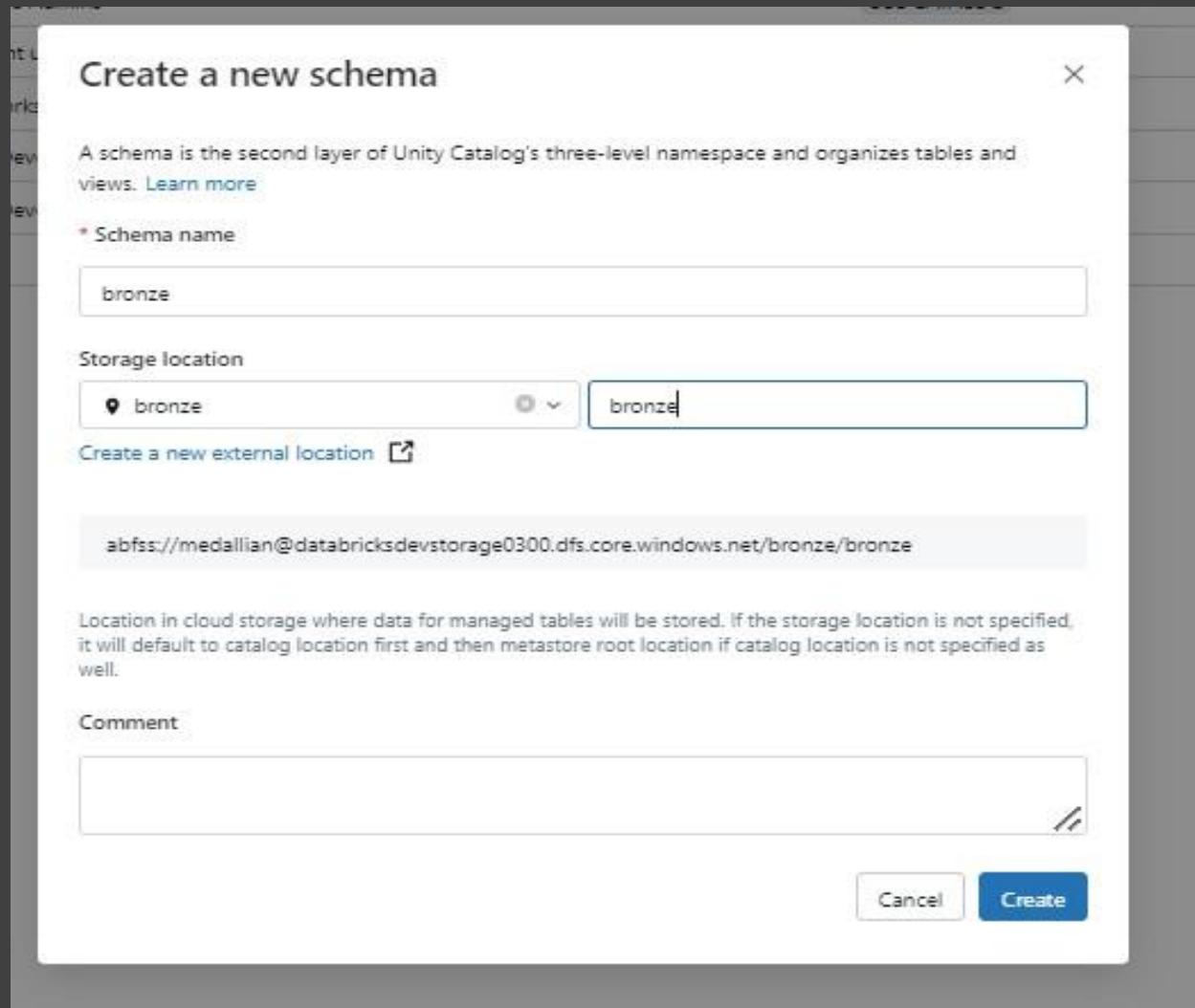
Ensure to test the connection for each external location to confirm the correct setup.

### Step 3: Create Schemas in Azure Databricks

#### A) Creating Schemas using UI

1. Open the Databricks workspace and navigate to the Catalog Explorer.
2. Select the devcatalog.
3. Grant yourself the necessary permissions if the 'Create Schema' option is disabled.
4. **Create the Bronze Schema:**
  - Name: bronze
  - Storage Location: Select the bronze external location.
5. **Create the Silver Schema:**
  - Name: silver
  - Storage Location: Select the silver external location.
6. **Create the Gold Schema:**
  - Name: gold

- Storage Location: Select the gold external location.



## B) Creating Schemas using Commands

1. Open a new notebook in Databricks and attach it to a running cluster.
2. Run the following commands to create the schemas:

```
spark.sql("CREATE SCHEMA IF NOT EXISTS bronze LOCATION 'abfss://medallion@  
databricksdevstorage0300.dfs.core.windows.net/bronze/'")  
spark.sql("CREATE SCHEMA IF NOT EXISTS silver LOCATION 'abfss://medallion@  
databricksdevstorage0300.dfs.core.windows.net/silver/'")  
spark.sql("CREATE SCHEMA IF NOT EXISTS gold LOCATION 'abfss://medallion@  
databricksdevstorage0300.dfs.core.windows.net/gold/'")
```

### Step 4: Create Raw Tables in the Bronze Layer

#### Raw Traffic Table

1. Define the table structure:
2. Create a function to create the raw\_traffic table dynamically:
3. Execute the function:

#### Raw Roads Table

1. Define the table structure:
2. Create a function to create the raw\_roads table dynamically:
3. Execute the function:

### Step 5: Data Ingestion

#### Ingest Data from Landing to Bronze Layer

Use Databricks Auto Loader or a similar tool to move data from the landing zone to the bronze layer.

The screenshot shows a Jupyter Notebook cell with the following code:

```
display(spark.sql(f"SELECT * FROM `{env}catalog`.`bronze`.`raw_traffic`"))
```

Below the code, the resulting DataFrame is displayed. The table has 13 columns:

- e
- longitude
- link\_length\_km
- pedal\_cycles
- two\_wheeled\_motor\_vehicles
- cars\_and\_taxis
- buses\_and\_coaches
- lvg\_type
- hgv\_type
- ev\_car
- ev\_bike
- extract\_time

The 'extract\_time' column is highlighted in yellow. At the bottom of the table, it says "10,000+ rows | Truncated data due to row limit | 2.80 seconds runtime".

## Step 6: Transformations and Quality Checks

### Transform Bronze Data and Create Silver Tables

1. Apply transformations and data quality checks to the bronze layer data.
2. Create Silver Traffic and Silver Roads Tables:

## Step 7: Create Gold Tables

Create Gold Traffic and Gold Roads Tables with minimal transformation from Silver Tables:

## Step 8: Conclusion

- Tables are now created in the bronze, silver, and gold layers.
- Data has been ingested and transformed through each layer.
- The gold layer tables are ready for analytical consumption and reporting in Power BI.

# Q. What is the need for Incremental Data Loading?

### Need for Incremental Loading:

- Example: If 10,000 files are ingested today and 20,000 more files are ingested tomorrow, only the additional 20,000 files should be processed.

### Table Structure:

- Data should be loaded as-is with an additional column for extract\_time.
- No extra fields or transformations are applied.

## Q. What is the Solution with Delta Tables and Auto Loader?

### 1. Delta Tables:

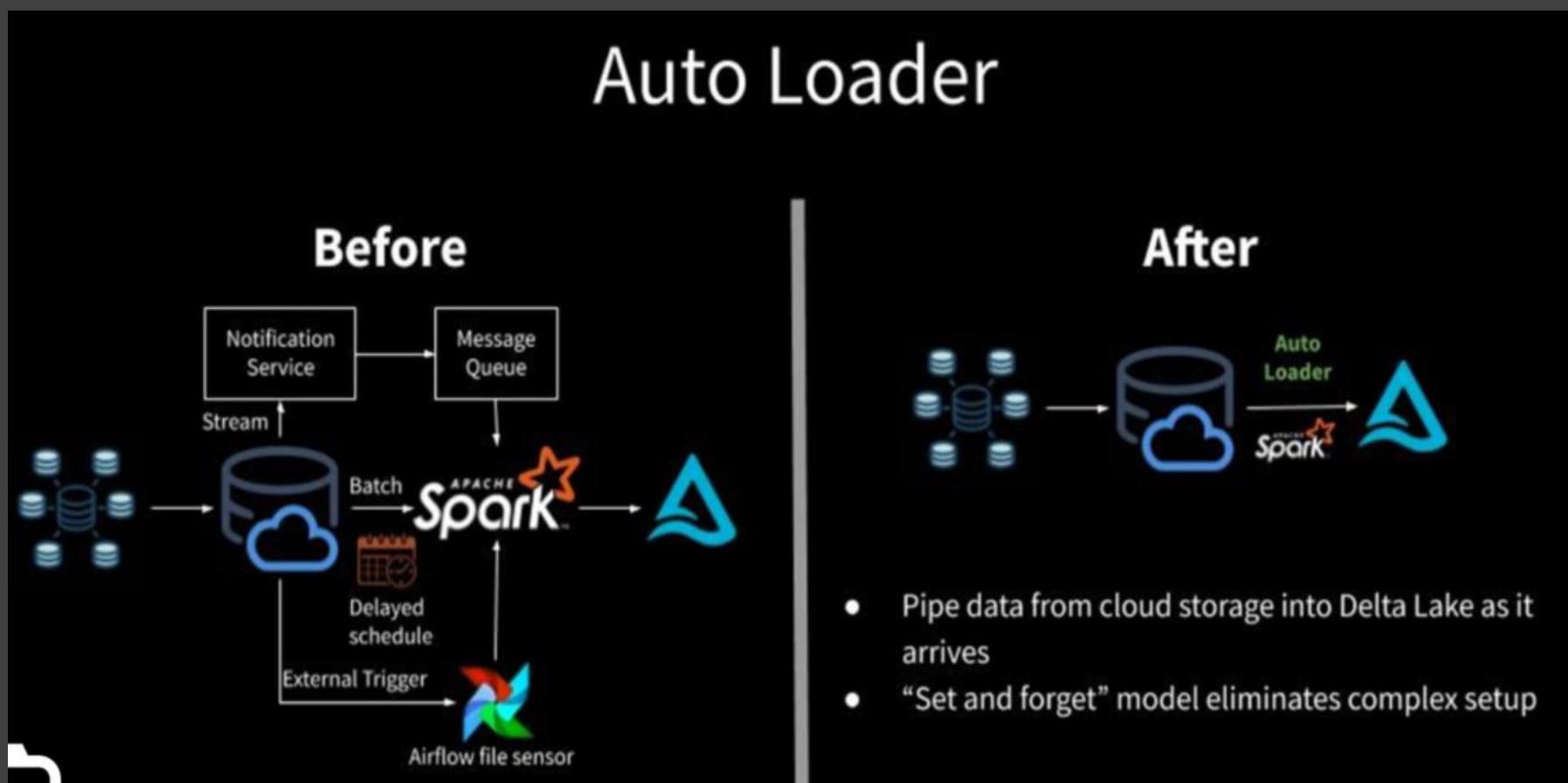
- Use Delta tables for the bronze layer.
- Benefits include ACID transactions and scalable metadata handling.

### 2. Auto Loader:

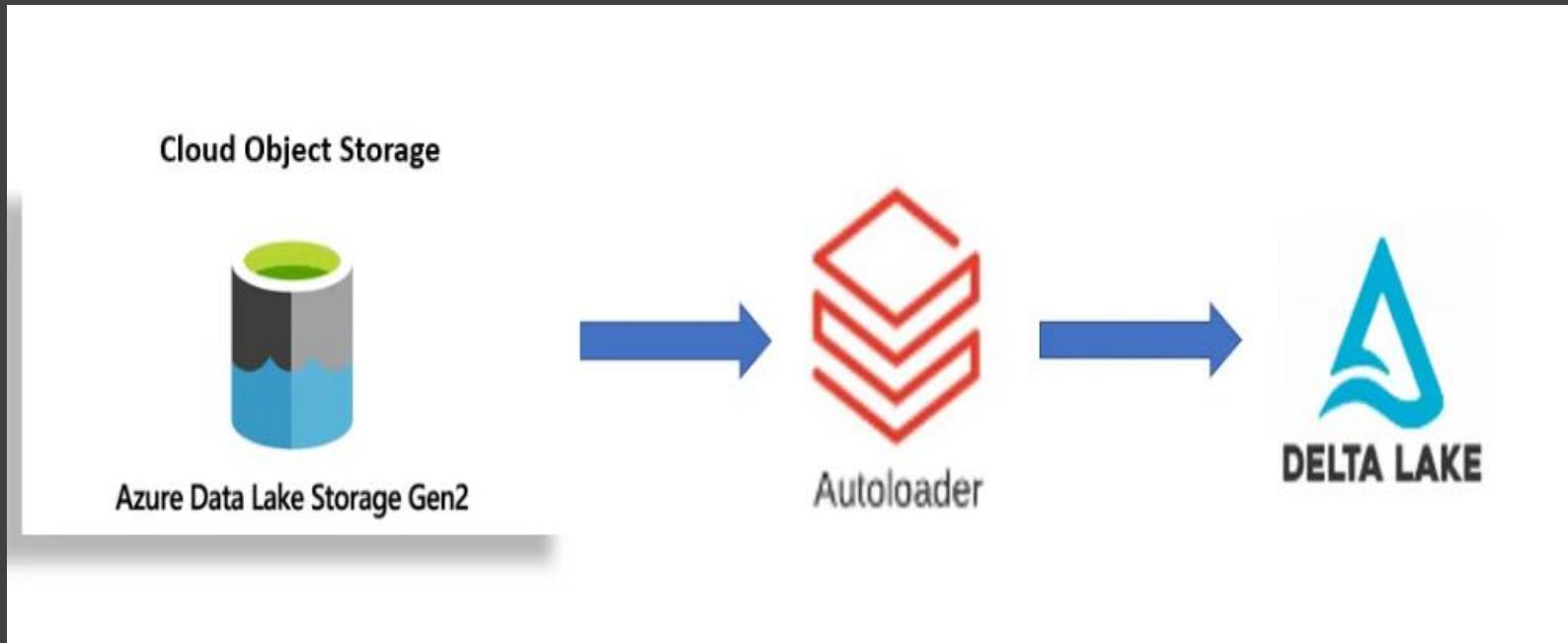
- Auto Loader can process millions of files per hour.
- Automatically handles incremental loading and metadata tracking.

### 3. Benefits of Auto Loader:

- Incremental Loading: Only processes new files added since the last run.
- Checkpointing: Maintains a record of processed files to handle failures gracefully.
- Efficiency: No need for custom solutions for incremental loading.



# Q. How to implement Auto Loader with Spark Structured Streaming?



## 1. Using Auto Loader:

- **Auto Loader Configuration:** Configure it to use Delta format for efficiency.
- **Trigger - Available Now:**
  - Functions like incremental batch loading.
  - Triggers batch processing to process data added after the previous load.
  - Terminates the query after processing the batch.

## 2. Creating Auto Loader Streams:

- **Raw Traffic Data:** Set up an Auto Loader stream to ingest traffic data.
- **Raw Roads Data:** Set up a separate Auto Loader stream for roads data.

# Q. What are the steps Implementing this Strategy?

## 1. Setting Up Auto Loader Streams:

- **Create two Auto Loader configurations:**
  - One for the raw\_traffic table.
  - One for the raw\_roads table.

## 2. Batch Processing with Available Now Trigger:

- **Use the available\_now trigger for batch processing.**
- **Ensure the streaming query stops after processing the batch.**

## 3. Scheduling:

- **Use an external scheduler or scheduling activity to invoke the notebook for periodic data ingestion.**

External Locations >

### Create a new external location

An external location is a cloud storage url (and paired credential) that allows access to data stored on your cloud tenant. Learn more [Learn more](#)

[Copy from mount point](#)

\* External location name  
landing

\* Storage credential [Learn more](#)  
storagecredential (Managed Identity)

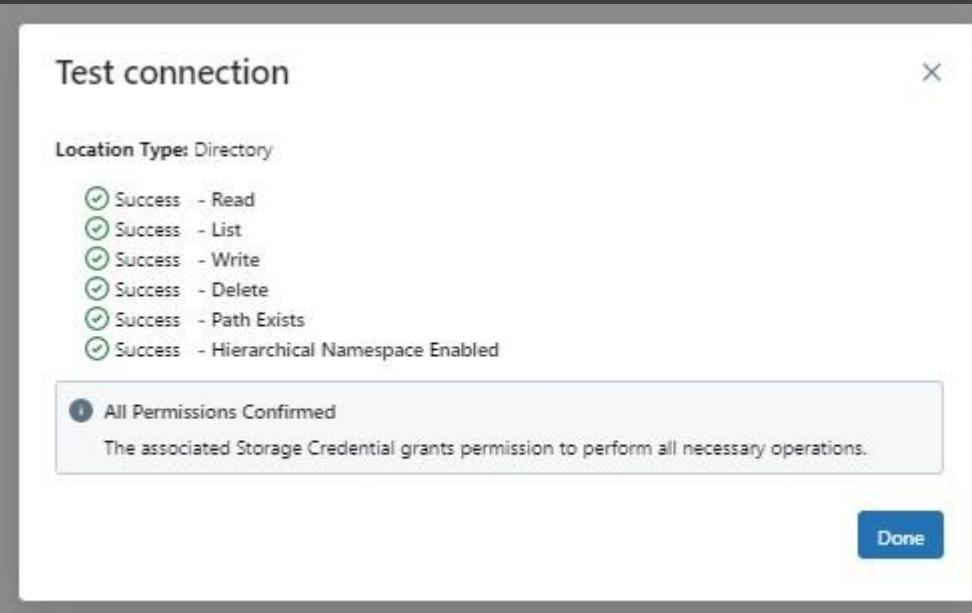
Connector Id: /subscriptions/ea8d46ac-c7b3-4f01-a3ea-a860b2f2dca5/resourceGroups/databricksresource/providers/Mi...  
User Assigned Managed Identity Id:

\* URL [Learn more](#)  
abfss://landing@dataricksdevstorage0300.dfs.core.windows.net

Enter the bucket path that you want to use as the external location

Comment  
this is for landing zone

[Advanced Options](#)



Catalog Explorer >  
**External Data**

External Locations   Storage Credentials

[Filter locations](#)   8 locations

Name	Credential	URL
bronze	storagecredential	abfss://medallian@dataricksdevstorage0300.dfs.core.windows.net/bronze
checkpoints	storagecredential	abfss://checkpoints@dataricksdevstorage0300.dfs.core.windows.net/
databricksdevws	databricksdevws	abfss://unity-catalog-storage@dbstoragepixmslryuphzk.dfs.core.windows.net/276878930788483
deltastorageolocation	storagecredential	abfss://test@deltadbstg0300.dfs.core.windows.net/files
gold	storagecredential	abfss://medallian@dataricksdevstorage0300.dfs.core.windows.net/gold
landing	storagecredential	abfss://landing@dataricksdevstorage0300.dfs.core.windows.net/
samestorage	storagecredential	abfss://working@dataricksdevstorage0300.dfs.core.windows.net/testing
silver	storagecredential	abfss://medallian@dataricksdevstorage0300.dfs.core.windows.net/silver

# Q. How to ingest data into the bronze layer?

## 1. Setting Up the Storage Account and Landing Container

### 1. Open the Storage Account:

- Go to your storage account.
- Click on the landing container.

### 2. Upload Folders and Files:

- There are two folders in the landing container: `raw_traffic` and `raw_roads`.
- First, select the `raw_traffic` folder and upload the traffic data.
- Click the Upload button to upload the file, select the file, and click Open.

### 3. Upload Raw Roads Data:

- Select the `raw_roads` folder and upload the roads data.
- Follow a similar process to upload the file.

## 2. Writing the Auto Loader Code

### 0. Open Databricks Notebook:

- Go to your Databricks notebook and start writing the auto loader code.

### 1. Read Stream Configuration:

- To create a read stream, set the format to `cloud_files`.
- The expected file format is CSV.

### 2. Define Schema Location:

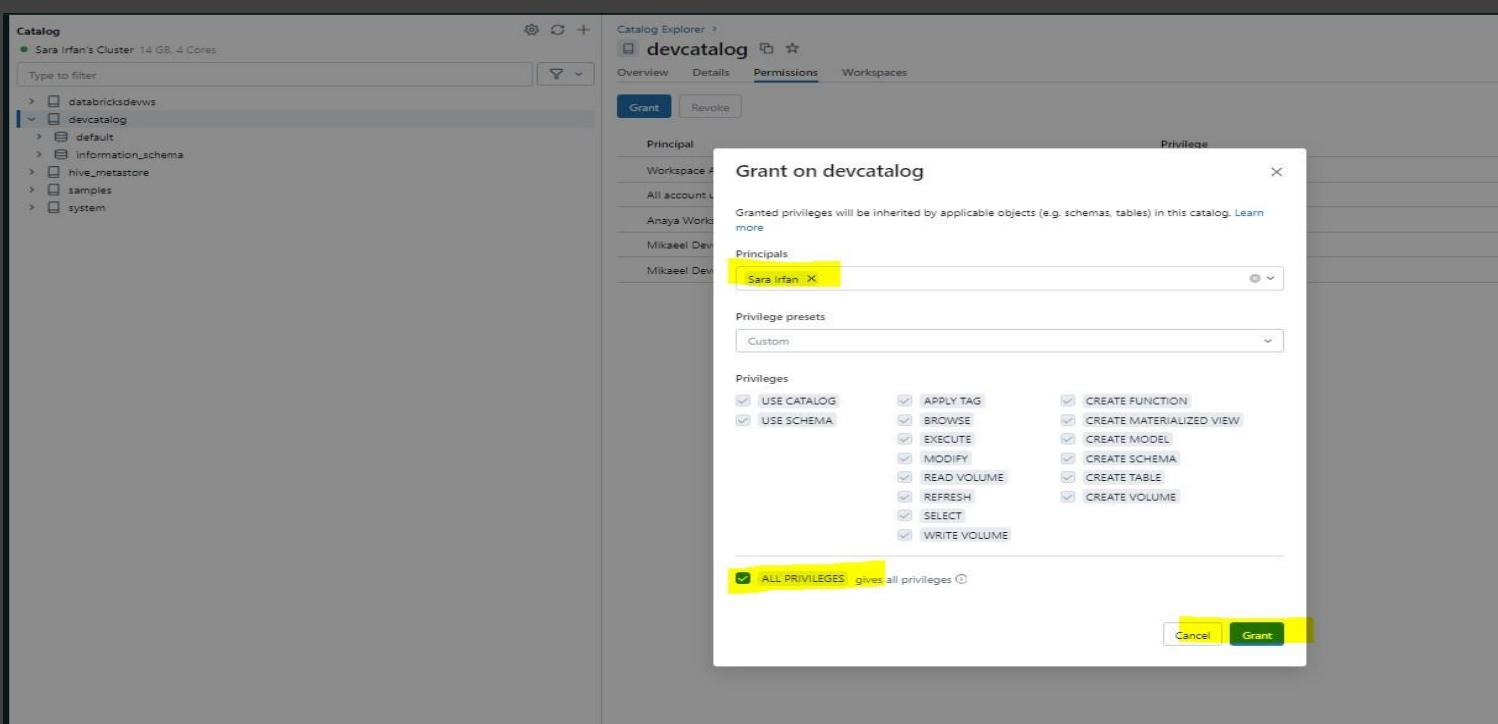
- Define the schema location to record the incoming schema and validate the schema for future runs.
- Use the checkpoint container to avoid hardcoding.

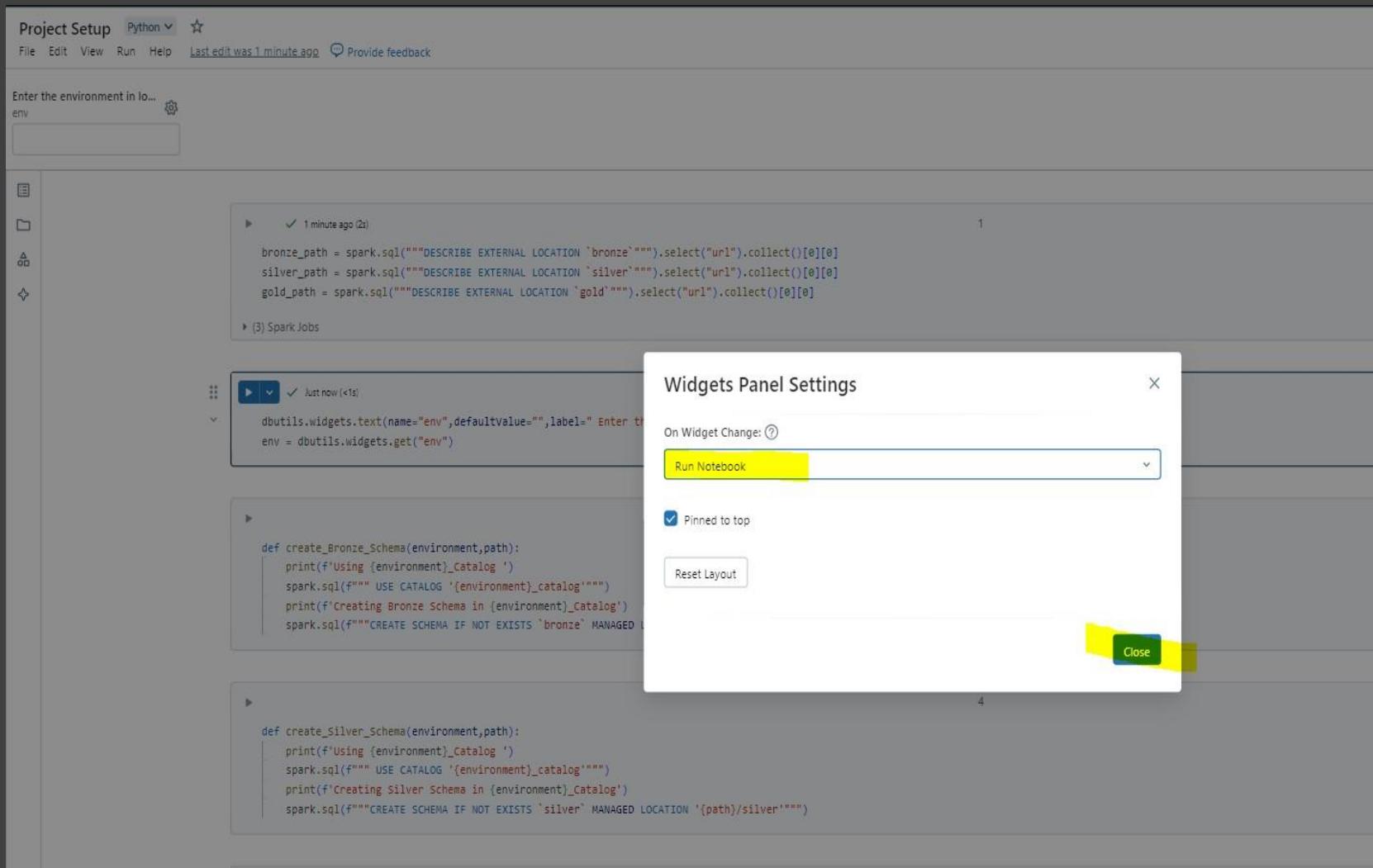
### 3. Checkpoint Container and Schema Folder:

- Select the checkpoints container.
- Create a folder for loading `raw_traffic` where the schema will be stored.

### 4. Obtain Path:

- Write a Spark SQL query to obtain the path. This query will provide you with the particular path.





## Q. How to creating a reusable notebook?

To establish a reusable notebook with commonly used functions and variables for consistency and efficiency.

### 1. Purpose of the Common Notebook:

- To centralize frequently used functions and variables such as URL extraction, duplicate handling, and null handling.

### 2. Creating the Common Notebook:

- Step 1: Define common variables and functions.

```
# Define checkpoints, landing zone, bronze, silver, and gold locations
checkpoints = "/mnt/checkpoints/"
landing_zone = "/mnt/landing/"
bronze = "/mnt/bronze/"
silver = "/mnt/silver/"
gold = "/mnt/gold/"
```

- Step 2: Add commonly used functions for data handling.

```
def extract_url(df):
    # Example function for URL extraction
    return df.withColumn('extracted_url', extract_from_column(df['column_name'])))

def handle_duplicates(df):
    return df.drop_duplicates()

def handle_nulls(df):
    df_string = df.fillna('unknown', subset=['string_column1', 'string_column2'])
```

```
df_clean = df_string.fillna(0)
return df_clean
```

- Step 3: Save and use the notebook in other notebooks for consistency.

## Q. How to call common notebook?

3.

The screenshot shows a Jupyter Notebook interface with the title "01 Project Setup". In the top navigation bar, "Python" is selected. A sidebar on the left shows a file tree with a folder named "env" containing a file named "uat". The main area displays a code cell with the following content:

```
Just now (2s)
%run "./04 Common Notebook"
```

Below the code cell, a large text box contains the following text:

**Giving the path for external location**

**Reuse common notebooks and variables  
external location for every container**

At the bottom of the notebook, another code cell is visible:

```
Just now (<1s)
dbutils.widgets.text(name="env",defaultValue="",label=" Enter the environment in lower case")
env = dbutils.widgets.get("env")
```

4.

## Q. How to demonstrate Incremental Loading with Autoloader?

### 1. Setup and Initial Row Count Verification

Confirm the current row count in the raw\_traffic table before uploading new files.

Steps:

1. Navigate to the Azure portal and locate the landing zone.
2. Query the current row count for the raw\_traffic table.
3. Use SQL syntax to get the count.

```
SELECT COUNT(*) FROM raw_traffic;
```

Expected Result: The current row count is 18546. Verify the last record's ID is 18546.

## 2. Upload New Data File

Upload new data files to the landing zone to test the incremental loading feature.

Steps:

1. Go to the landing container in Azure Storage.
2. Upload a new file to the raw\_traffic folder.

## 3. Running the Notebook

Manually run the notebook to process the newly uploaded data.

Steps:

1. Attach the cluster if it's not already attached.
2. Run the notebook manually to process the newly uploaded file.

## 4. Verify Incremental Load

Confirm that only the new records were added to the raw\_traffic table.

Steps:

1. Query the updated row count in the raw\_traffic table.

```
SELECT COUNT(*) FROM raw_traffic;
```

2. Verify the new row count. It should be 37092 (i.e., twice the original count).
3. To ensure that incremental loading is working as expected, compare the records before and after the original record ID 18546.

```
SELECT * FROM raw_traffic WHERE record_id <= 18546 ORDER BY record_id;  
SELECT * FROM raw_traffic WHERE record_id > 18546 ORDER BY record_id;
```

4. Check the extract\_time column for records before and after 18546.

- Records before 18546 should have one extract\_time.
- Records after 18546 should have a different extract\_time, indicating that only the new records were loaded.

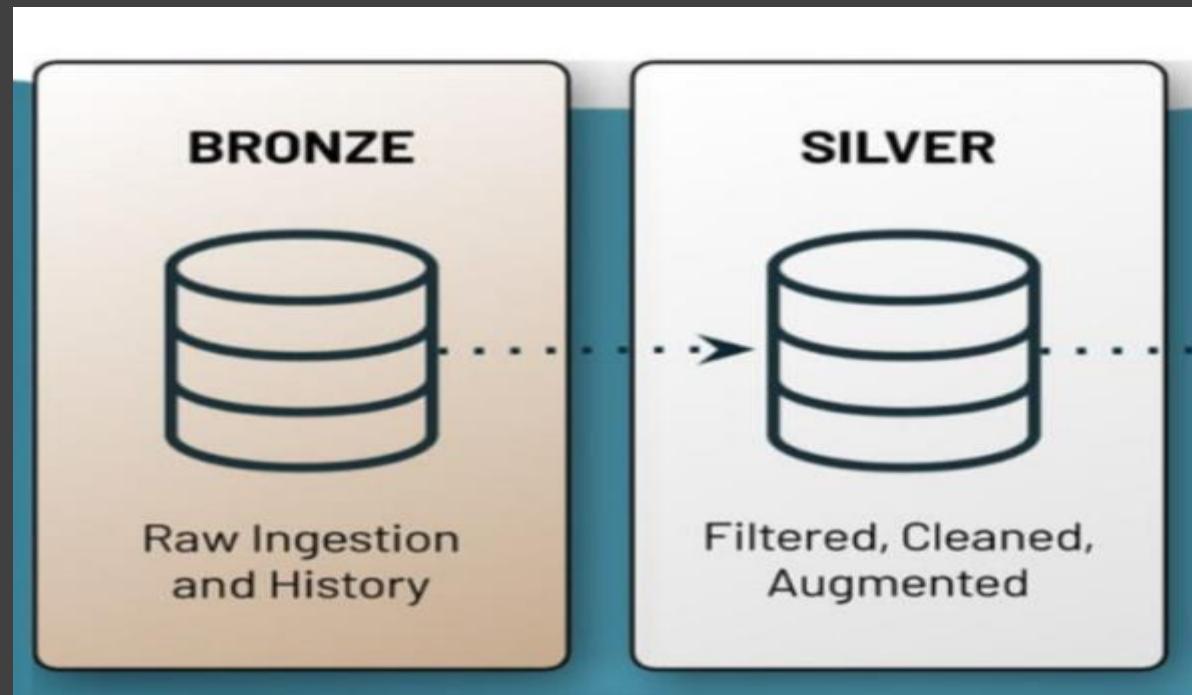
# Q. How to transform Data from Bronze to Silver Layer?

Understand the transition from the bronze layer to the silver layer, focusing on data quality checks and transformations.

Tasks:

- Perform data quality checks.
- Apply business transformations.
- Handle only the newly added data using streaming and Delta Lake.

Ensure the data in the bronze layer meets quality standards before moving to the silver layer.



### Steps:

#### 1. Read Data from Bronze Layer:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

spark = SparkSession.builder.getOrCreate()

# Read data from bronze layer
bronze_df = spark.read.format("delta").load("path_to_bronze_table")
```

#### 2. Check for Null Values:

```
# Check for null values in critical columns
bronze_df.filter(col("important_column").isNull()).show()
```

#### 3. Check for Duplicate Records:

```
# Identify duplicate records
duplicate_records = bronze_df.groupBy("unique_key_column").count().filter("count > 1")
duplicate_records.show()
```

#### 4. Verify Data Ranges and Consistency:

```
# Check for valid data ranges
invalid_records = bronze_df.filter(col("numeric_column") < 0)
invalid_records.show()
```

## Q. How to apply Business Transformations?

Apply necessary transformations to clean and enrich the data, making it suitable for business analysis.

#### 1. Define Transformation Logic:

```
from pyspark.sql.functions import when

# Example transformation logic
transformed_df = bronze_df.withColumn(
    "new_column",
    when(col("existing_column") > 100, "High").otherwise("Low")
)
```

### 2. Apply Aggregations or Calculations:

```
# Example aggregation
aggregated_df = transformed_df.groupBy("group_column").agg({"value_column": "sum"})
```

### 3. Write Transformed Data to Silver Layer:

```
# Define path for silver layer
silver_path = "path_to_silver_table"

# Write transformed data to silver layer
transformed_df.write.format("delta").mode("append").save(silver_path)
```

## Q. How to handle newly added data?

Ensure that only newly added data is processed and transformed.

Steps:

### 1. Read Only New Data Using Incremental Loading:

```
# Define path to read only new data
new_data_df = spark.read.format("delta").option("startingVersion",
last_checkpoint_version).load("path_to_bronze_table")
```

### 2. Apply the Same Transformations:

```
# Apply transformations to new data
new_transformed_df = new_data_df.withColumn(
    "new_column",
    when(col("existing_column") > 100, "High").otherwise("Low")
)
```

### 3. Write Transformed New Data to Silver Layer:

```
# Write the new transformed data to the silver layer
new_transformed_df.write.format("delta").mode("append").save(silver_path)
```

## Q. How to create silver tables?

Create two silver tables, one for quality-checked data and another for transformed data.

Steps:

### 1. Create Silver Table for Quality-Checked Data:

```
# Quality check transformations
```

```
quality_checked_df = bronze_df.filter(col("important_column").isNotNull())

# Write quality-checked data to silver table
quality_checked_df.write.format("delta").mode("overwrite").save("path_to_quality_checked_silver_table")
```

## 2. Create Silver Table for Transformed Data:

# Transformations

```
transformed_df = quality_checked_df.withColumn(
    "new_column",
    when(col("existing_column") > 100, "High").otherwise("Low")
)
```

# Write transformed data to silver table

```
transformed_df.write.format("delta").mode("overwrite").save("path_to_transformed_silver_table")
```

The screenshot shows a Jupyter Notebook interface with three code cells and one output cell.

**Cell 4:** A code cell titled "Waiting" containing Python code to create a Silver schema. It uses PySpark's SQL API to switch to a specific catalog and create a schema if it doesn't exist.

```
def create_Silver_Schema(environment,path):
    print(f'Using {environment}Catalog ')
    spark.sql(f""" USE CATALOG '{environment}catalog'""")
    print(f'Creating Silver Schema in {environment}Catalog')
    spark.sql(f"""CREATE SCHEMA IF NOT EXISTS `silver` MANAGED LOCATION '{path}/silver'""")
```

**Cell 5:** A code cell titled "Waiting" containing Python code to create a Gold schema. Similar to Cell 4, it switches to a catalog and creates a schema if it doesn't exist.

```
def create_Gold_Schema(environment,path):
    print(f'Using {environment}Catalog ')
    spark.sql(f""" USE CATALOG '{environment}catalog'""")
    print(f'Creating Gold Schema in {environment}Catalog')
    spark.sql(f"""CREATE SCHEMA IF NOT EXISTS `gold` MANAGED LOCATION '{path}/gold'""")
```

**Cell 6:** An output cell titled "Waiting" showing the execution of the code from Cell 5. It prints "create\_Bronze\_Schema(env,bronze\_path)" and then "Using devCatalog Creating Bronze Schema in devCatalog".

```
create_Bronze_Schema(env,bronze_path)

Using devCatalog
Creating Bronze Schema in devCatalog
```

The screenshot shows the Microsoft Azure Databricks interface. On the left, the navigation sidebar includes options like New, Workspace, Recents, Catalog (which is selected), Workflows, Compute, SQL Editor, Queries, Dashboards, Alerts, Query History, SQL Warehouses, Data Engineering, Job Runs, Data Ingestion, and Delta Live Tables. The main area is titled 'Catalog' and shows 'Sara Irfan's Cluster 14 GB, 4 Cores'. A search bar at the top right says 'Search data, notebooks, recents'. Below the search bar, the 'Catalog Explorer' section is open for 'devcatalog'. It has tabs for Overview, Details, Permissions, and Workspaces. Under 'Overview', there is a 'Filter schemas' input field and a table with columns 'Name' and 'Type'. The table lists five schemas: bronze, default, gold, information\_schema, and silver. The 'bronze', 'gold', and 'silver' entries are highlighted with yellow boxes.

**Calling all functions to check the execution of these functions  
To check its working or not?**

The screenshot shows a Databricks notebook cell with the status 'Waiting' and a timeout of 15 minutes. The cell contains the following Python code:

```
create_Bronze_Schema(env,bronze_path)
createTable_rawTraffic(env)
createTable_rawRoad(env)

create_Silver_Schema(env,silver_path)
create_Gold_Schema(env,gold_path)
```

Below the code, the notebook displays the logs of the schema creation process:

```
Using devCatalog
Creating Bronze Schema in devCatalog
*****
Creating raw_Traffic table in devcatalog
*****
Creating raw_roads table in devcatalog
*****
Using devCatalog
Creating Silver Schema in devCatalog
*****
Using devCatalog
Creating Gold Schema in devCatalog
*****
```

## Q. How to automate the transformation process and data quality monitoring?

To automate the transformation process from the bronze to the silver layer, implement regular data quality checks, and ensure the reliability of the data pipeline using Databricks.

### Silver Layer Transformation Steps

#### 1. Raw Data to Silver Layer Transformation:

- Our raw traffic and raw roads tables are in the bronze layer.
- We will clean and transform these tables to extract business value.
- The silver schema has already been created.

## 2. Starting Transformation:

- First, perform basic data quality checks. This is crucial because bad data leads to poor output.
- General checks include removing duplicates and handling null values.

## 3. Basic Data Quality Checks:

- Removing Duplicates:

```
df = df.drop_duplicates()
```

- Handling Null Values:

```
def handle_nulls(df):
    df_string = df.fillna('unknown', subset=['string_column1', 'string_column2'])
    df_clean = df_string.fillna(0)
    return df_clean
```

Use the handle\_nulls function to process your dataframe.

## 4. Column Renaming:

- Replace spaces in column names with underscores.
- Replace raw traffic with raw\_traffic.

```
df = df.withColumnRenamed('raw traffic', 'raw_traffic')
```

## 5. Specific Transformation for Traffic Data Set:

- Electric Vehicles Count:

```
df = df.withColumn('electric_vehicles_count', col('EV_car') + col('EV_bike'))
```

## 6. Additional Transformation:

- Motor Vehicles Count:

```
df = df.withColumn('motor_vehicles_count', col('EV_car') + col('EV_bike') + col('bus') + col('truck'))
```

## 7. Adding Transform Time Column:

- Add a column to record the transformation time.

```
from pyspark.sql.functions import current_timestamp
df = df.withColumn('transform_time', current_timestamp())
```

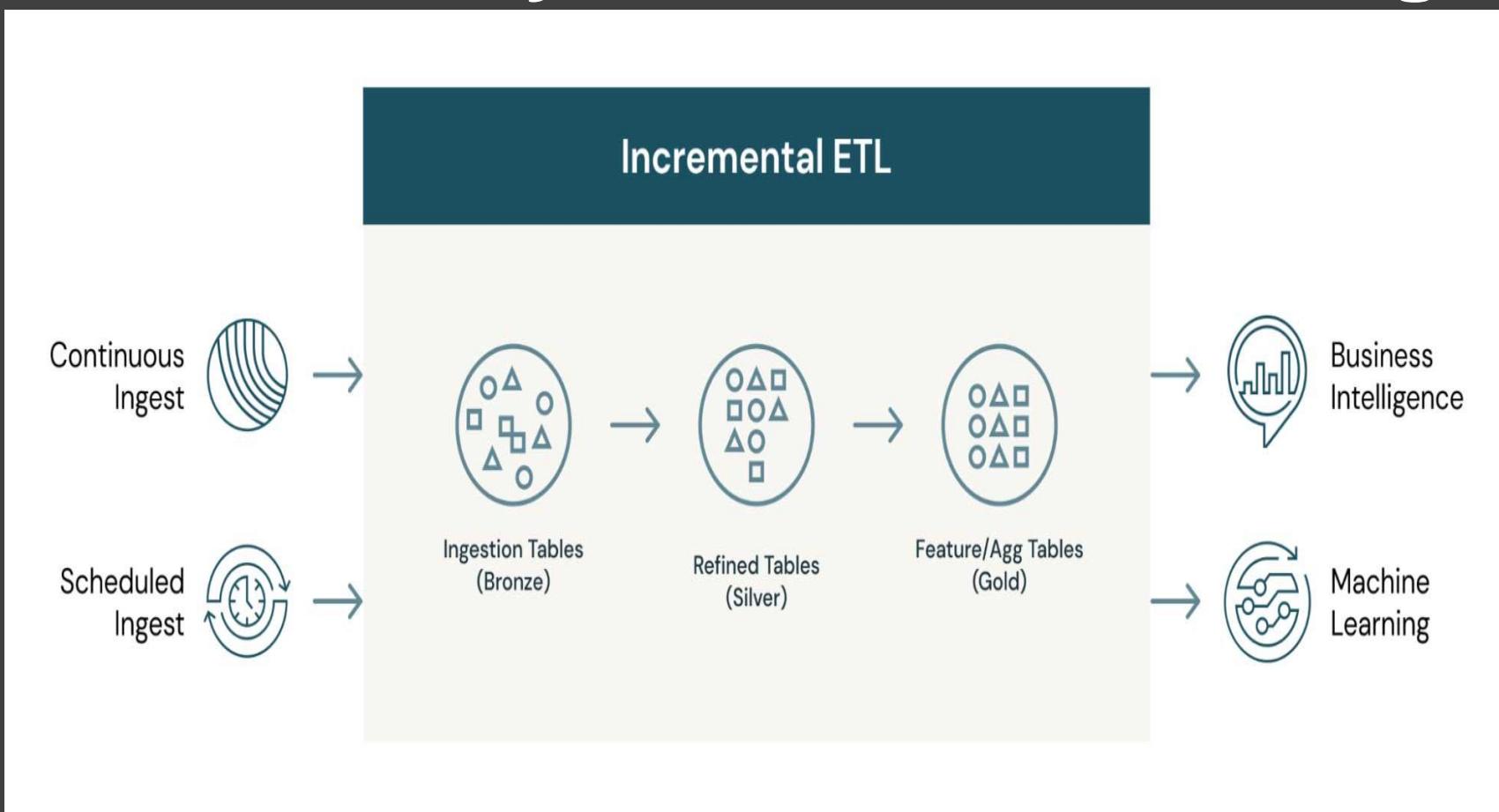
## 8. Write Transformed Data to Silver Layer:

```
df.writeStream.format('delta').outputMode('append').table('silver_traffic')
```

## 9. Verification:

- Go to the Databricks Catalog Explorer, refresh the Silver Traffic table, and verify the data.

# Q. How to verify incremental loading?



To verify that only newly added records are processed during the data transformation stage.

## 1. Initial Setup:

- The bronze table may contain thousands of records. Incremental loading means only new data gets appended to the bronze table.
- As the bronze table grows, ensure that transformations in the silver layer apply only to newly added records.

## 2. Verification Process:

- Current Record Count:

```
silver_count = spark.table('silver_traffic').count()
print("Silver Table Record Count:", silver_count)
```

Assume this count is 37,092.

- Data Upload and Incremental Loading:
  - Upload a new raw traffic file and run the transformation process.
- Verify Incremental Data:

```
updated_silver_count = spark.table('silver_traffic').count()
```

- Detailed Analysis:
  - Pre-Transformation State:

```
old_records = spark.table('silver_traffic').filter(col('record_id') < 37092).count()
new_records = spark.table('silver_traffic').filter(col('record_id') >= 37092).count()
print("Old Records Count:", old_records)
print("New Records Count:", new_records)
```

- Transformation Verification:

```
# Command to get transformation time for old and new records
transformation_time_old = spark.table('silver_traffic').filter(col('record_id') <
37092).select('transform_time').distinct().collect()
```

```
transformation_time_new = spark.table('silver_traffic').filter(col('record_id') >= 37092).select('transform_time').distinct().collect()
print("Old Records Transformation Time:", transformation_time_old)
print("New Records Transformation Time:", transformation_time_new)
```

- The transformation times for records before and after the specified ID will differ, demonstrating that only new records are processed.

## Q. How do you perform transformations on the roads dataset in Azure Databricks, and what steps are involved in ensuring data quality, applying transformations, and writing the results to the Silver layer?

### Steps for Transformations:

#### 1. Read Data:

- We will read the roads data from the bronze table named "raw\_roads."

```
df_roads = spark.table('raw_roads')
```

#### 2. Initial Data Quality Check:

- It's important to check the data quality by handling duplicate records and null values.

```
df_roads_clean = handle_nulls(df_roads)
df_roads_clean = df_roads_clean.drop_duplicates()
```

#### 3. Use Common Functions:

- Copy and use functions from the common notebook.

#### 4. Transformations on Roads Data:

- New Column Creation:

**Adding a column Road Category Name**

```
✓ 01:39 PM {<1s}

def road_Category(df):
    print('Creating Road Category Name Column: ', end='')
    from pyspark.sql.functions import when,col

    df_road_Cat = df.withColumn("Road_Category_Name",
        when(col('Road_Category') == 'TA', 'Class A Trunk Road')
        .when(col('Road_Category') == 'TM', 'Class A Trunk Motor')
        .when(col('Road_Category') == 'PA', 'Class A Principal road')
        .when(col('Road_Category') == 'PM', 'Class A Principal Motorway')
        .when(col('Road_Category') == 'M', 'Class B road')
        .otherwise('NA')
    )
    print('Success!! ')
    print('*****')
    return df_road_Cat
```

Road_Category_Name	Road_Type
Class A Trunk Motor	Major
Class B road	Minor
Class A Principal road	Major
Class A Principal road	Major
Class A Trunk Motor	Major
Class B road	Minor
Class A Trunk Road	Major
Class A Principal road	Major
Class A Trunk Motor	Major
Class A Trunk Motor	Major
Class A Trunk Road	Major
Class A Principal Motorway	Major
Class A Trunk Road	Major
Class A Trunk Motor	Major

```
from pyspark.sql.functions import when, col
```

```
def create_road_category(df):
```

```
return df.withColumn(  
    'road_category_name',  
    when(col('road_code') == 'TO', 'Class A Trunk Road')  
    .when(col('road_code') == 'TM', 'Class A Trunk Motor')  
    .when(col('road_code') == 'PA', 'Class A Principal Road')  
    .when(col('road_code') == 'PM', 'Motorway')  
    .when(col('road_code') == 'M', 'Class B Road')  
    .otherwise('Unknown')  
)
```

```
df_roads_transformed = create_road_category(df_roads_clean)
```

- Additional Column Creation:

```
df_roads_transformed = df_roads_transformed.withColumn(  
    'road_type',  
    when(col('road_category_name').contains('Class A'), 'Major Road')  
    .otherwise('Minor Road')  
)
```

#### 5. Visual Display:

- Display the data visually to make the results of the transformations clear.

```
df_roads_transformed.select('road_category_name', 'road_type').show()
```

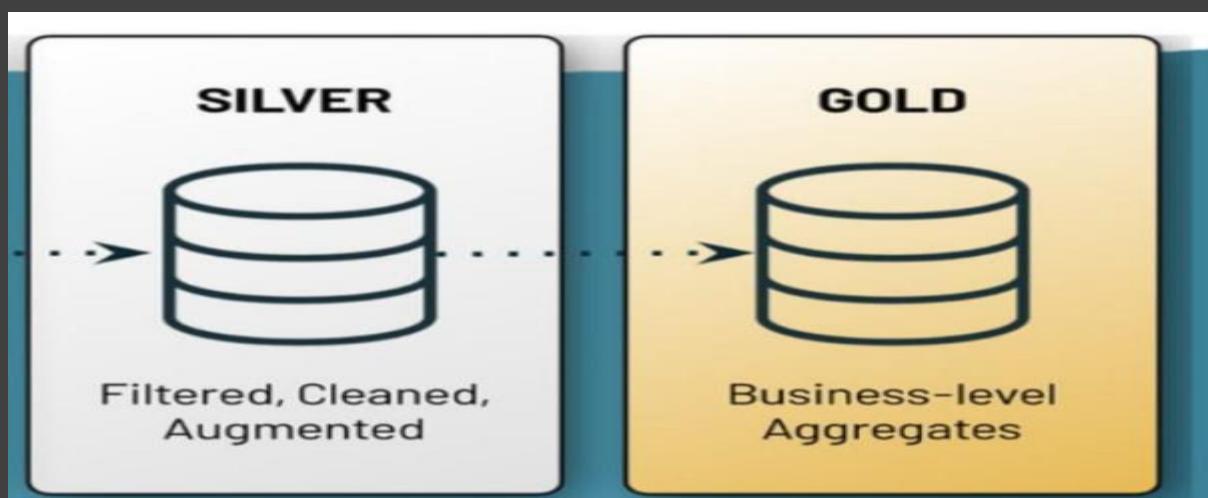
#### 6. Write to Silver Roads Table:

```
df_roads_transformed.writeStream.format('delta').outputMode('append').table('silver_roads')
```

#### 7. Verification:

- Query the Silver layer tables to verify that the transformations have been applied correctly

## Q. What are the key steps involved in loading data into the Gold layer in Azure Databricks?



# Steps:

1. Initialize Environment:
  - Run the common variables notebook to set paths and configurations for the Gold layer.
  - Define widgets to specify the environment.
2. Read Silver Tables:
  - Read data from the Silver schema using Spark.
3. Perform Minimal Transformations:
  - Add a "vehicle intensity" column by dividing motor\_vehicles\_count by road\_length\_km.
  - Add a "load time" column with the current timestamp.
4. Write Data to Gold Tables:
  - Write the transformed Silver traffic data to the Gold Traffic table.
  - Write the transformed roads data to the Gold Roads table.
  -

The screenshot shows a Jupyter Notebook interface with the title '06 Gold Layer Final Transformation'. The environment is set to 'uat'. The code cell contains the following Python code:

```
dt_vehicle = create_VehicleIntensity(df_SilverTraffic)
df_FinalTraffic = create_LoadTime(df_vehicle)
df_FinalRoads = create_LoadTime(df_SilverRoads)

## Writing to gold tables
write_Traffic_GoldTable(df_FinalTraffic,env)
write_Roads_GoldTable(df_FinalRoads,env)
```

The output pane shows the execution results:

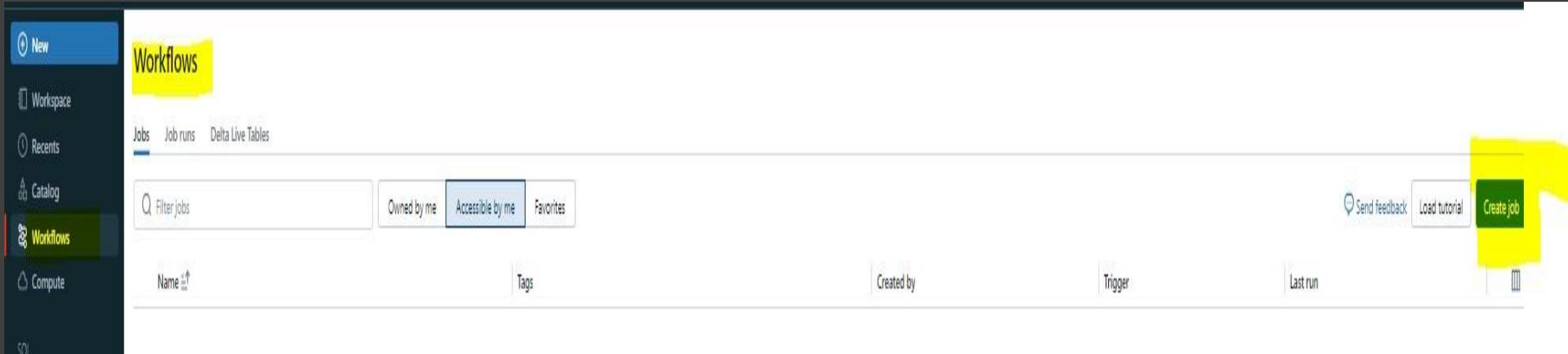
```
df_SilverTraffic: pyspark.sql.dataframe.DataFrame = [Record_ID: integer, Count_point_id: integer ... 26 more fields]
df_SilverRoads: pyspark.sql.dataframe.DataFrame = [Road_ID: integer, Road_Category_Id: integer ... 8 more fields]
df_vehicle: pyspark.sql.dataframe.DataFrame = [Record_ID: integer, Count_point_id: integer ... 27 more fields]
df_FinalTraffic: pyspark.sql.dataframe.DataFrame = [Record_ID: integer, Count_point_id: integer ... 28 more fields]
df_FinalRoads: pyspark.sql.dataframe.DataFrame = [Road_ID: integer, Road_Category_Id: integer ... 9 more fields]

Reading the Silver Traffic Table Data : Reading uatcatalog.silver.silver_traffic Success!
*****
Reading the Silver Table Silver_roads Data : Reading uatcatalog.silver.silver_roads Success!
*****
Creating Vehicle Intensity column : Success!!!
*****
Creating Load Time column : Success!!
*****
Creating Load Time column : Success!!
*****
Writing the gold_traffic Data : Writing `uatcatalog`.`gold`.`gold_traffic` Success!
Writing the gold_roads Data : Writing `uatcatalog`.`gold`.`gold_roads` Success!
```

5. Run Functions:
  - Define and apply necessary functions in a notebook cell.
6. Validate and Verify:
  - Run the notebook to ensure correctness.
  - Refresh and query the Gold tables in the Catalog Explorer to verify the data.

# Q. How to orchestrate workflows in Azure Databricks?

## Orchestrating Workflows in Azure Databricks

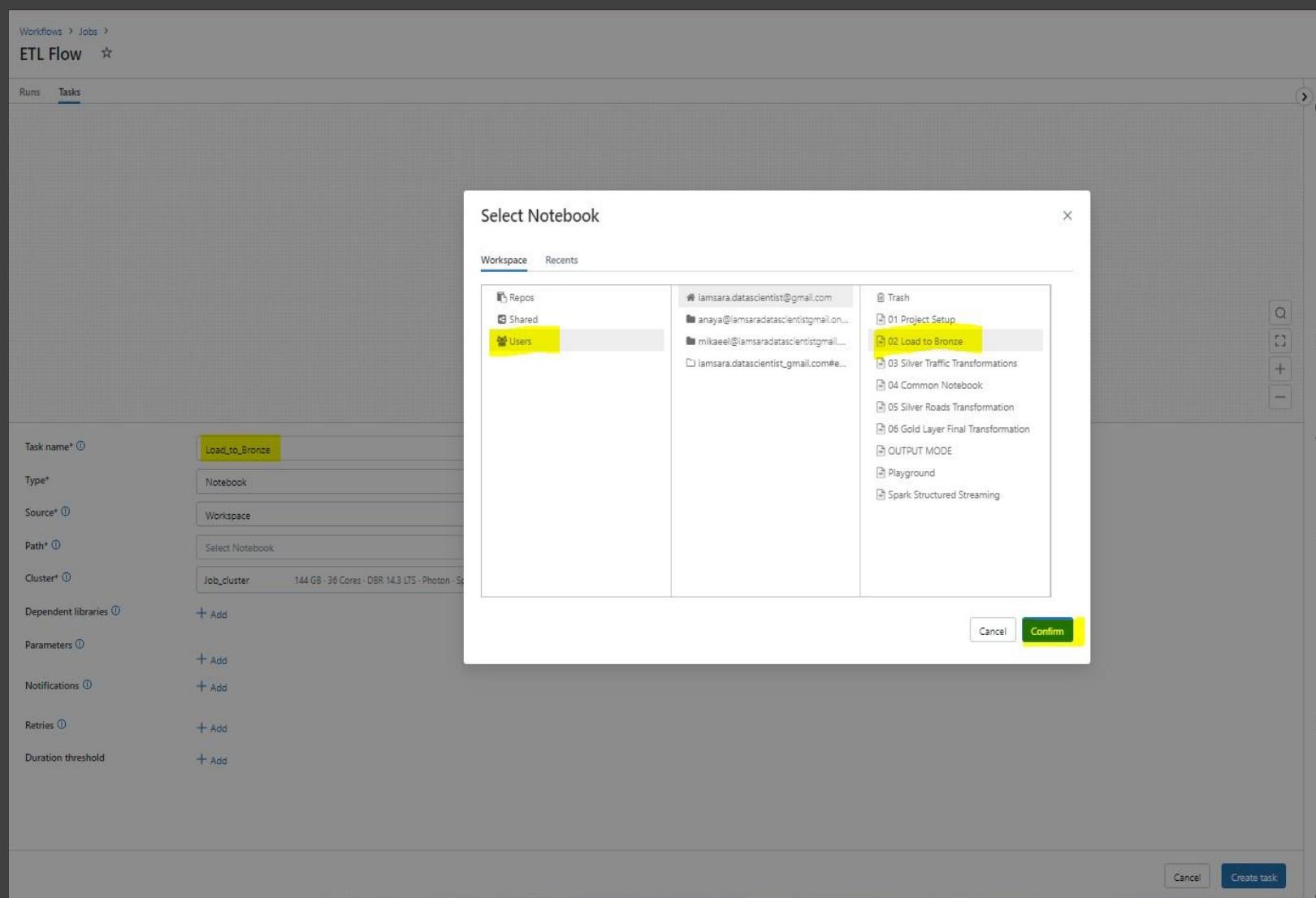


### 1. Introduction:

- Automate notebook execution to manage data from bronze to gold layers.
- Use Case: Utilize Azure Databricks workflows for automation and scheduling.

### 2. Link Common Notebook to All Notebooks:

- Ensure notebooks can access shared variables and configurations.
- Steps:
  - Open the common notebook.
  - Copy and link code to other notebooks.



- Remove unnecessary cells from the linked notebooks.
- Run the notebooks to verify access to the common notebook.



### 3. Design Orchestration Flow:

- Steps:
  - Create a new job in Azure Databricks Workflows.
  - Add tasks for each notebook with dependencies.
  - Set parameters for each task.

### 4. Configure Parameters:

- Control notebook behavior via parameters.
- Add key-value pairs for parameters (e.g., env=dev).

### 5. Run the Workflow:

- Execute the entire workflow.
- Query tables to verify data.
- Compare record counts in different layers.

### 6. Adjust and Optimize:

- Objective: Improve and troubleshoot.
- Steps:
  - Review performance.
  - Adjust clusters or parameters.
  - Set up triggers for automatic execution.



## Q. How to set Up Triggers for Azure Databricks Jobs?

Triggers automate the execution of tasks based on specific conditions or events. They ensure timely and consistent processing without manual intervention, improving efficiency and responsiveness. By setting up triggers, you can run jobs automatically when files arrive or at scheduled intervals, ensuring data pipelines operate seamlessly and in real-time. This helps in maintaining data integrity and meeting real-time processing requirements.

The screenshot shows the 'External Data' section of the Catalog Explorer. On the left, there's a list of external locations: bronze, checkpoints, databricksdevws, deltastoragelocation, gold, landing (which is highlighted with a yellow box), samestorage, and silver. On the right, the 'Schedules & Triggers' panel is open. It has tabs for 'Trigger Status' (Active) and 'Trigger type' (File arrival). A note says: 'This job does not have any failure notifications. Consider adding email or webhook notifications to alert if trigger evaluation fails.' Below that, it says 'File arrival triggers monitor cloud storage paths of up to 10,000 files. These paths are either Unity Catalog volumes or external locations managed through Unity Catalog.' The 'Storage location' field contains the path 'abfss://landing@databricksdevstorage0300.dfs.core.windows.net/raw\_traffic/'. At the bottom right are 'Success', 'Cancel', and 'Save' buttons.

- Ensure the cluster is running or swap to a new cluster if needed.
- Open the job workflow in Azure Databricks.
- Click on "Add Trigger."
- Choose Trigger Type:
- Scheduled Trigger: For periodic execution.
- File Arrival Trigger: Executes when new files appear.
- Continuous Trigger: For real-time processing.
- Configuring File Arrival Trigger:
- Steps:
  - Select "File Arrival" trigger type.
  - Specify the path to monitor.
  - Set monitoring frequency.
  - Test the connection
- Add notification destinations for job failures.
- Verify the trigger by uploading a test file.
- Setting Up a Scheduled Trigger: Clone the job if needed.
- Configure the schedule using Cron syntax or specific times.
- Handling Multiple Locations: Create separate jobs for monitoring multiple locations if necessary.

The screenshot shows the Azure Storage Blob container interface for the 'landing' container. The left pane shows blob details like Overview, Diagnose and solve problems, Access Control (IAM), and Settings. The right pane shows an 'Upload blob' dialog. It has a file input field showing '1 file(s) selected: raw\_roadsl.csv', a checkbox for 'Overwrite if files already exist', and an 'Upload' button. The URL for the blob is shown as 'https://databricksdevstorage0300.blob.core.windows.net/raw\_roads/raw\_roadsl.csv'.

Catalog

Sara Irfan's Cluster 14 GB, 4 Cores

Type to filter

- bronze
  - raw\_roads
  - raw\_traffic
- default
- gold
- information\_schema
- silver
- hive\_metastore
- samples
- system

**raw\_roads**

Overview Sample Data Details Permissions History Lineage Insights Quality

Filter columns...

Column	Type	Comment	Tags
Road_ID	int	⊕	⊕
Road_Category_Id	int	⊕	⊕
Road_Category	varchar(255)	⊕	⊕
Region_ID	int	⊕	⊕
Region_Name	varchar(255)	⊕	⊕
Total_Link_Length_Km	double	⊕	⊕
Total_Link_Length_Miles	double	⊕	⊕
All_Motor_Vehicles	double	⊕	⊕

Please analyze the increase in records after inserting a file in a container

Just now (4s)

```
%sql
select COUNT(*) from `devcatalog`.`bronze`.`raw_traffic`;
```

(3) Spark Jobs

```
_sqldf: pyspark.sql.DataFrame = [count(1): long]
```

	count(1)
1	18546

Waiting

```
%sql
select COUNT(*) from `devcatalog`.`bronze`.`raw_traffic`;
```

```
_sqldf: pyspark.sql.DataFrame = [count(1): long]
```

	count(1)
1	37092

landing Container

Search

Upload Add Directory Refresh Rename

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

Authentication method: Access key (Switch to Microsoft Entra user access)

Location: landing / raw\_traffic

Search blobs by prefix (case-sensitive)

Name

[.]

raw\_traffic1.csv

Overwrite if files already exist

Advanced

Upload

```

▶ ✓ 04:16 PM (3s)

%sql
-- Databricks notebook source
-- Count of Bronze Traffic Rows

SELECT COUNT(*) FROM `devcatalog`.`bronze`.raw_traffic

▶ (4) Spark Jobs
▶ _sqldf: pyspark.sql.dataframe.DataFrame = [count(1): long]

Table + 
| 1 | 37092 |

SELECT COUNT(*) FROM `devcatalog`.`gold`.gold_roads

▶ (4) Spark Jobs
▶ _sqldf: pyspark.sql.dataframe.DataFrame = [count(1): long]

Table + 
| 1 | 76 |

SELECT COUNT(*) FROM `devcatalog`.gold.gold_traffic

▶ (4) Spark Jobs
▶ _sqldf: pyspark.sql.dataframe.DataFrame = [count(1): long]

Table + 
| 1 | 37092 |

```

```

SELECT COUNT(*) FROM `devcatalog`.`bronze`.raw_roads

▶ (4) Spark Jobs
▶ _sqldf: pyspark.sql.dataframe.DataFrame = [count(1): long]

Table + 
| 1 | 76 |

SELECT COUNT(*) FROM `devcatalog`.`silver`.silver_traffic

▶ (5) Spark Jobs
▶ _sqldf: pyspark.sql.dataframe.DataFrame = [count(1): long]

Table + 
| 1 | 37092 |

SELECT COUNT(*) FROM `devcatalog`.`silver`.silver_roads

▶ (5) Spark Jobs
▶ _sqldf: pyspark.sql.dataframe.DataFrame = [count(1): long]

Table + 
| 1 | 76 |

```

Name	Type	Owner
01 Project Setup	Notebook	Sara Irfan
02 Load to Bronze	Notebook	Sara Irfan
03 Silver Traffic Transformations	Notebook	Sara Irfan
04 Common Notebook	Notebook	Sara Irfan
05 Silver Roads Transformation	Notebook	Sara Irfan
06 Gold Layer Final Transformation	Notebook	Sara Irfan
OUTPUT MODE	Notebook	Sara Irfan
Spark Structured Streaming	Notebook	Sara Irfan
Untitled Notebook 2024-07-30 16:03:57	Notebook	Sara Irfan

Landing Container

Search Overview Diagnose and solve problems Access Control (IAM) Settings

**Authentication method:** Access key ([Switch to Microsoft Entra user access](#))  
**Location:** landing / raw\_traffic

Search blobs by prefix (case-sensitive)

Name	Modified
[..]	7/29/2023 10:44:11 AM
raw_traffic1.csv	7/29/2023 10:44:11 AM
raw_traffic2.csv	7/29/2023 10:44:11 AM
raw_traffic3.csv	7/29/2023 10:44:11 AM
raw_traffic4.csv	7/29/2023 10:44:11 AM

1 file(s) selected: raw\_traffic5.csv  
 Drag and drop files here or [Browse for files](#)

Overwrite if files already exist

Advanced

**Upload**

## Please analyze the change in values, with insertion of files

```

Just now (4s)
-- Databricks notebook source
-- Count of Bronze Traffic Rows

SELECT COUNT(*) FROM `devcatalog`.`bronze`.raw_traffic
(4) Spark Jobs
+---+sqldf: pyspark.sql.DataFrame = [count(1): long]
Table +---+
| 1 | 74184 |
+---+

```

```

SELECT COUNT(*) FROM `devcatalog`.`bronze`.raw_roads
(4) Spark Jobs
+---+_sqldf: pyspark.sql.DataFrame = [count(1): long]
Table +---+
| 1 | 152 |
+---+

```

```

SELECT COUNT(*) FROM `devcatalog`.`silver`.silver_traffic
(5) Spark Jobs
+---+sqldf: pyspark.sql.DataFrame = [count(1): long]
Table +---+
| 1 | 74184 |
+---+

```

```

SELECT COUNT(*) FROM `devcatalog`.`gold`.gold_roads
(4) Spark Jobs
+---+_sqldf: pyspark.sql.DataFrame = [count(1): long]
Table +---+
| 1 | 152 |
+---+

```

Start time	Run ID	Launched	Duration	Status
Jul 30, 2024, 08:01 PM	299860350075303	By file arrival	1m 5s	Running
Jul 30, 2024, 05:19 PM	965390140979113	Manually	2m 59s	Succeeded

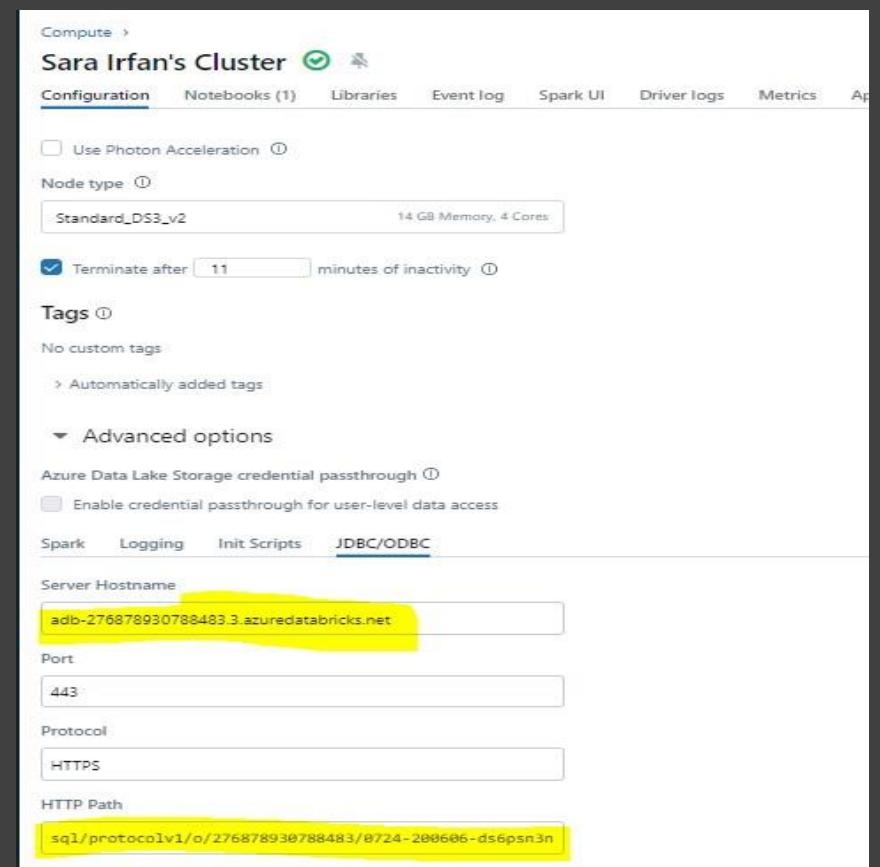
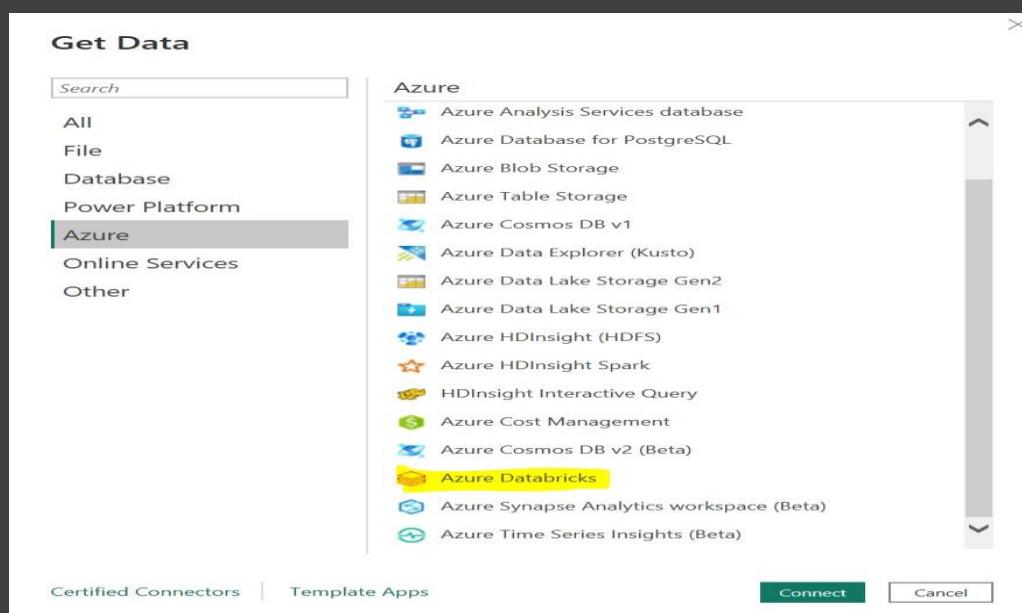
## Q. How to report Data to Power BI from Azure Databricks?

### 1. Open Power BI Desktop:

### 2. Connect to Azure Databricks:

- Get Data:

- Select Azure Databricks as the data source.
- Configure connection with Server Hostname and HTTP Path from Azure Databricks.

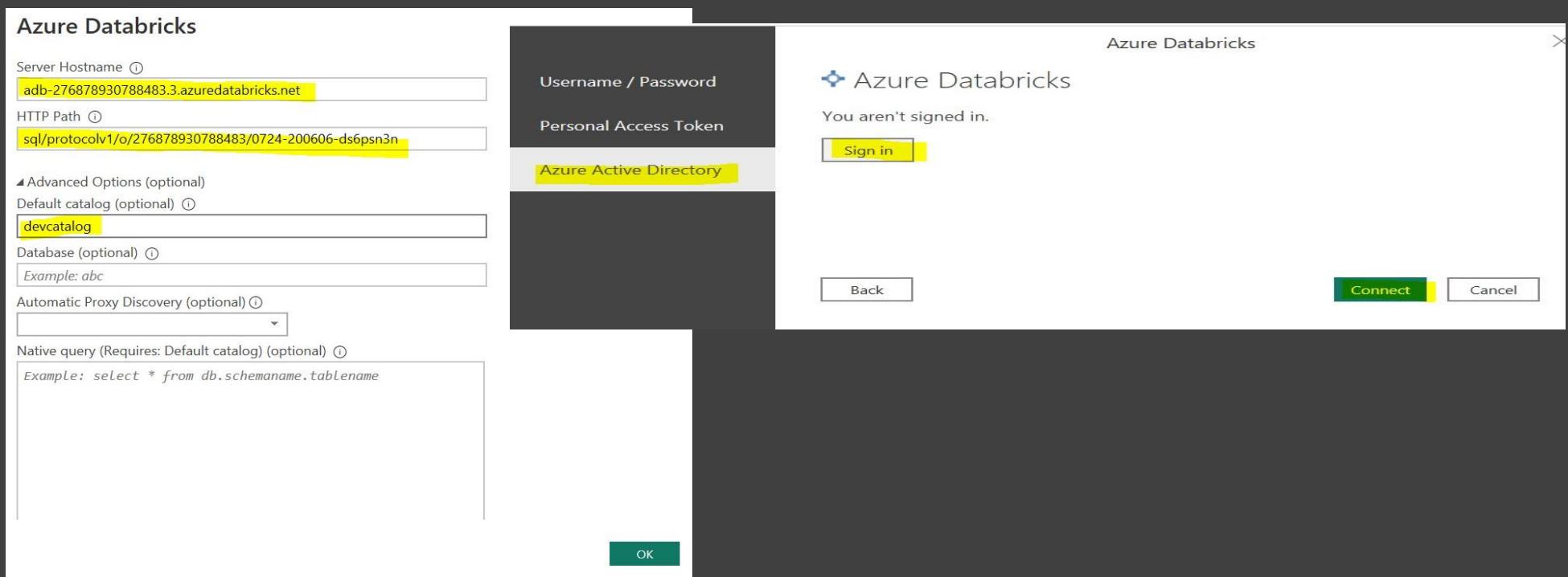


### 3. Log In to Azure Databricks:

- Authenticate using Azure Active Directory.

### 4. Select and Load Data:

- Choose tables from the Gold Schema and load them into Power BI



## 5. Build Data Model:

- Define relationships between tables.
- Create visuals like charts and cards.

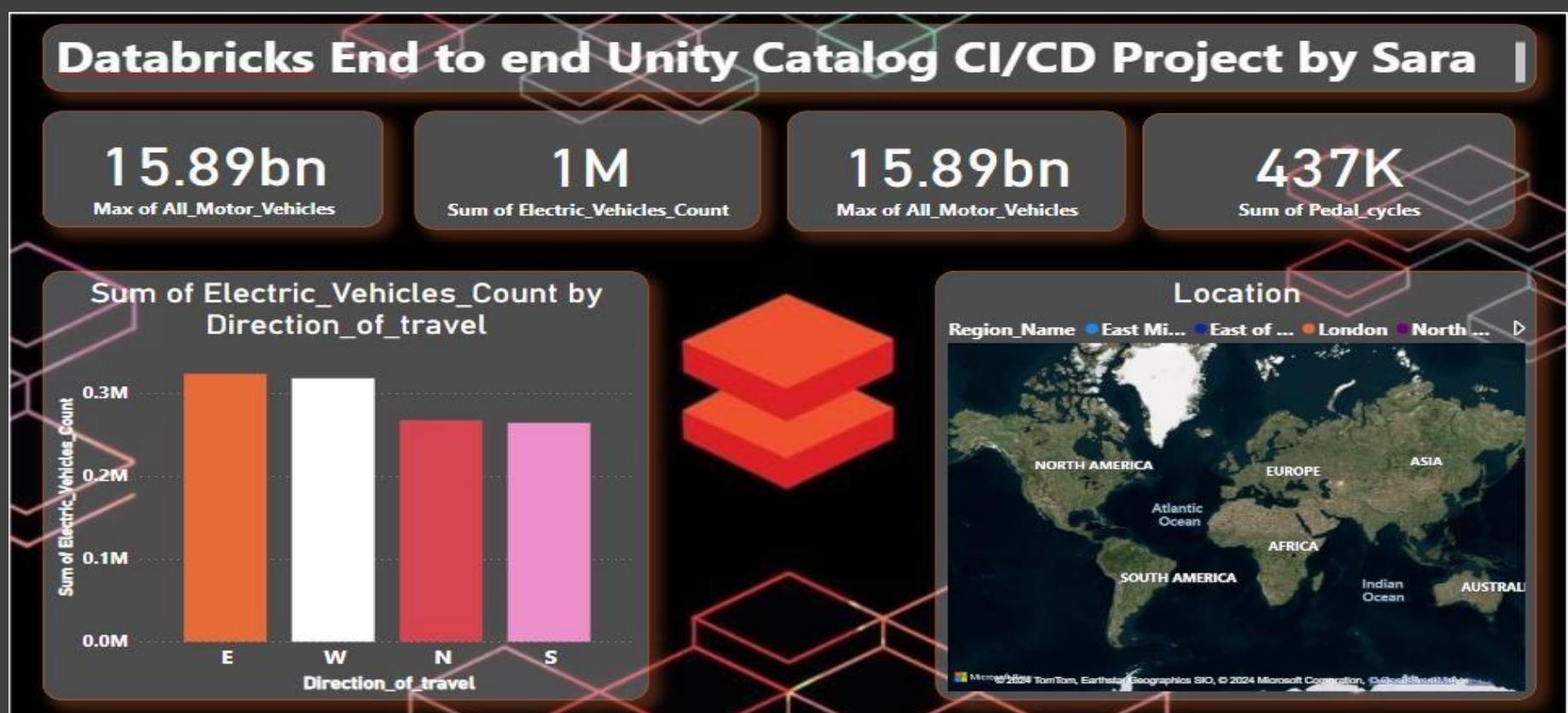
## 6. Update Data:

- Run Databricks jobs to update data.
- Refresh Power BI data to reflect the latest updates.

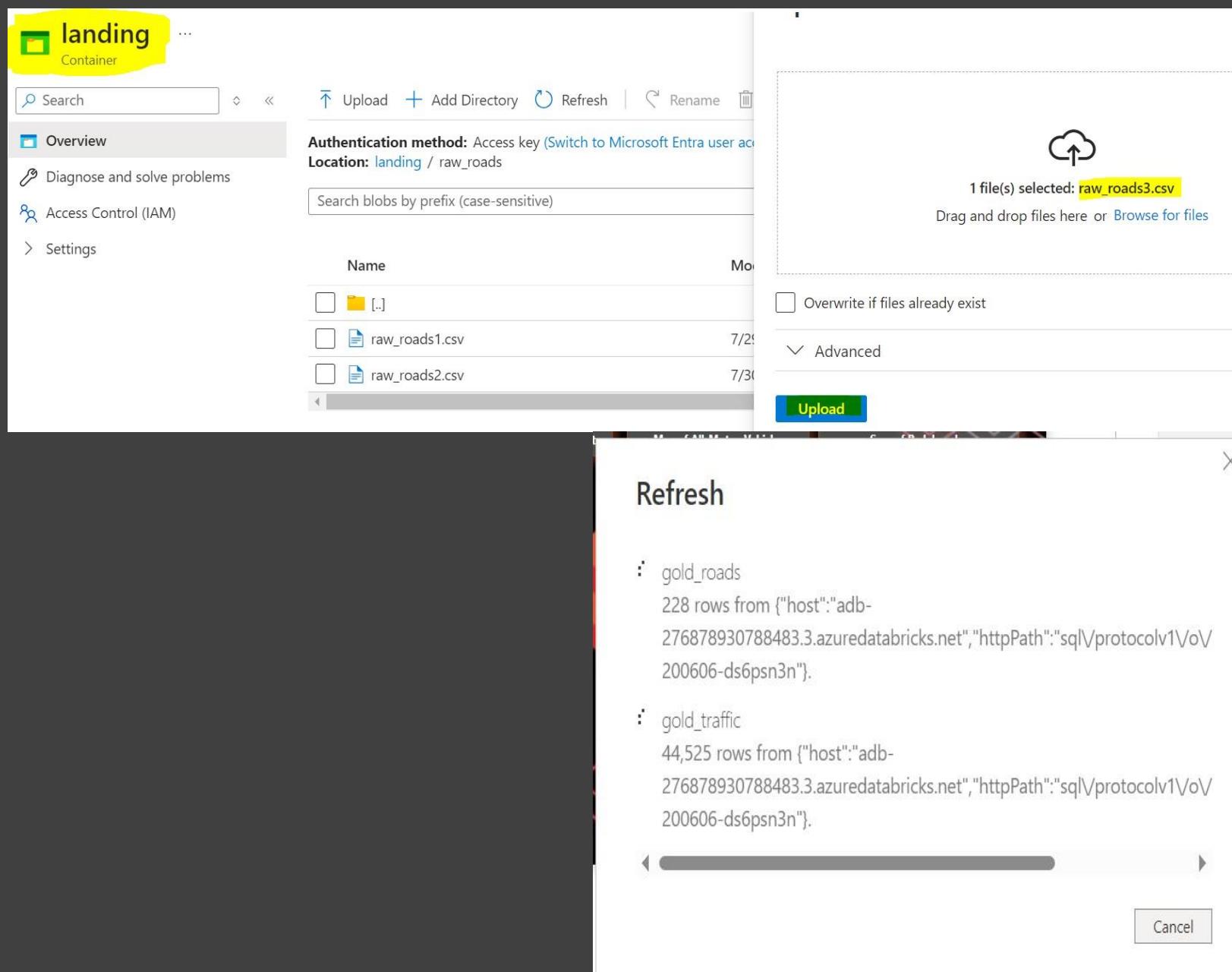
## 7. Publish Power BI Report:

- Publish to Power BI Service
- Configure scheduled refresh for automatic updates

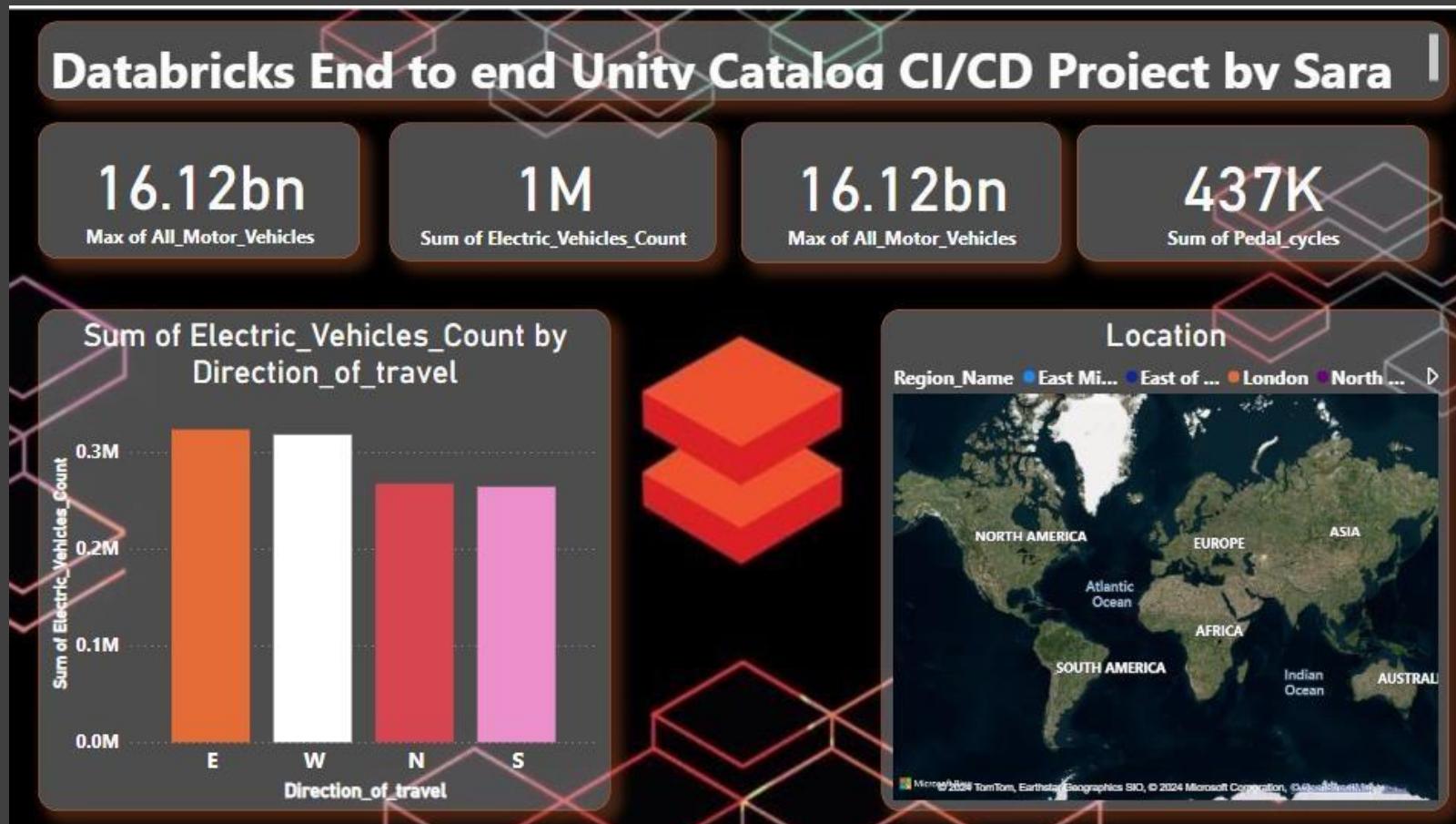
## Power BI Dashboard



Uploading raw\_roads3.csv in landing container, the values of power BI dashboard has been changed



## Updated Power BI Dashboard



# Q. How do you configure a UAT environment in Azure, and what are the essential steps to integrate it with Azure DevOps?

To configure a UAT environment in Azure and integrate it with Azure DevOps, follow these essential steps:

## 1. Create the UAT Environment:

- In the Azure Portal, navigate to Resource Groups, create a new one (e.g., UAT-ResourceGroup), and select a region.
- Databricks Workspace: Go to "Create a resource," find Azure Databricks, and create a workspace (e.g., UAT-Databricks-Workspace) within the chosen resource group.

- Storage Account: Set up a storage account (e.g., uatstorageaccount) with hierarchical namespace enabled for Data Lake Storage Gen2.

## 2. Configure UAT Databricks Workspace:

- Assign the managed identity of the Databricks workspace to the Storage Account with the "Storage Blob Data Contributor" role.
- Manage and assign the Unity Catalog to the UAT Databricks workspace from the Dev Databricks Workspace.

**Create a new catalog**

A catalog is the first layer of Unity Catalog's three-level namespace and is used to organize your data assets. [Learn more](#)

\* Catalog name:

\* Type:

Storage location:

[Create a new external location](#)

Location in cloud storage where data for managed tables will be stored. If not specified, the location will default to the metastore root location.

[Cancel](#) [Create](#)

Catalog Explorer > **uat\_catalog**

[Overview](#) [Details](#) [Permissions](#) [Workspaces](#)  [Create schema](#)

About this catalog

Owner: Sara Irfan

Tags: [Add tags](#)

Description: [Add description](#)

2 schemas

Name	Owner	Created at
default	iamsara.datascientist@gmail.co...	2024-07-31 09:05:00
information_schema	System user	2024-07-31 09:05:00

### 3. Create External Locations:

- Set up storage credentials and define external locations for containers in your storage account (e.g., landing-UAT, bronze-UAT).

### 4. Implement Continuous Deployment:

- In Azure DevOps, create a multi-stage pipeline for Dev, UAT, and Prod environments, configure approvals, and test the pipeline.

Account

[Workspaces](#) [Catalog](#) [User management](#) [Cloud resources](#) [Previews](#) [Settings](#)

**Account console**  
Manage your Databricks account at scale

**Workspaces**: Configure workspace settings. Workspaces contain notebooks, libraries, queries, and workflows.

**Catalog**: Manage metastores as your top-level container for catalogs, schemas (also called databases), views and tables.

**Users & groups**: Manage identities for use with jobs, automated tools and systems.

**Cloud resources**: Manage network connectivity configurations for your resources.

**Settings**: Configure your Databricks account user provisioning and other settings.

### 5. Validate and Test:

- Ensure the UAT Databricks workspace is correctly set up and run sample jobs.
- Confirm stability in UAT before moving to production.

Home > databricksuatstg0300

**databricksuatstg0300 | Containers**

Storage account

Search Container Change access level Restore containers Refresh Delete Give feedback

Access Control (IAM)

Data migration

Events

Storage browser

Data storage

Containers

File shares

Search containers by prefix

Show deleted containers

Name	Last modified	Anonymous access level	Lease state
\$logs	7/30/2024, 10:22:25 PM	Private	Available
checkpoints	7/31/2024, 8:32:53 AM	Private	Available
landing	7/31/2024, 8:32:34 AM	Private	Available
medallian	7/31/2024, 8:32:44 AM	Private	Available

## 6. Azure DevOps Setup:

- **Sign Up and Log In:** Go to the Azure DevOps products page
- **Set up a new project (e.g., Databricks Traffic), and explore repositories, pipelines, and other sections.**
- **Configure Git integration with Azure Databricks, set up authentication, and manage repositories.**

Catalog

Metastores

A metastore is the top-level container for catalog in Unity Catalog. Within a metastore, Unity Catalog provides a 3-level namespace for organizing data: catalogs, schemas (also called databases), and tables / views. Learn More

Filter metastores

Create metastore

Name	Region	Path	Created at	Updated at
dbtrafficproject	eastus2	abfss://metastoreroot@databricksdevstorage0300.dfs.core.windows...	last Friday at 10:34 AM	last Friday at 10:34 AM
metastore_azure_eastus	eastus		06/29/2024	06/29/2024

Catalog > dbtrafficproject >

**dbtrafficproject**

Configuration Workspaces

Filter workspaces

Assign to workspace

Name	Resource group	Region	Subscription	Created

## 7. Branch Management and Pull Requests:

- Establish branch policies for code reviews and access management.
- Create a feature branch, add notebooks, and commit changes.
- Open a pull request, review, approve, and merge changes to the main branch.

## 8. Mikael's Integration:

- Mikael configures a repository, creates a feature branch, and commits changes.
- Pull request is reviewed, approved, and merged.
- Set up a live folder for automatic updates and integration pipeline.

The screenshot shows the Databricks Container interface. At the top, it displays the path: Home > databricksuatstg0300 | Containers >. Below this, there's a sidebar with links: Overview, Diagnose and solve problems, Access Control (IAM), and Settings. The main area shows three storage containers listed under 'Name': bronze, gold, and silver. Each container has a small checkbox icon to its left.

### Create a new storage credential

A storage credential represents an authentication and authorization mechanism for accessing data stored on your cloud tenant. [Learn more](#)

\* Credential Type: Azure Managed Identity

\* Storage credential name: UAT-Access

\* Access connector ID: /subscriptions/ea8d46ac-c7b3-4f01-a3ea-a860b2f2dca5/resourceGroups/databricksresource/providers...

User assigned managed identity ID (optional):

[Cancel](#) [Create](#)

### Create a new external location

An external location is a cloud storage url (and paired credential) that allows access to data stored on your cloud tenant. [Learn more](#)

\* External location name: landing-uat

\* Storage credential: uat-access (Managed Identity)

Connector Id: /subscriptions/ea8d46ac-c7b3-4f01-a3ea-a860b2f2dca5/resourceGroups/databricksresource/providers...  
User Assigned Managed Identity Id:

\* URL: abfss://landing@databricksuatstg0300.dfs.core.windows.net/

Enter the bucket path that you want to use as the external location

Comment:

[Cancel](#) [Create](#)

External Locations >

### Create a new external location

An external location is a cloud storage url (and paired credential) that allows access to data stored on your cloud tenant. Learn more [↗](#)

[Copy from mount point](#)

\* External location name  
bronze-uat

\* Storage credential [Learn more ↗](#)  
uat-access (Managed Identity)

Connector Id: /subscriptions/ea8d46ac-c7b3-4f01-a3ea-a860b2f2dca5/resourceGroups/databricksresource/providers...

User Assigned Managed Identity Id:

\* URL [Learn more ↗](#)  
abfss://medallian@databricksuatstg0300.dfs.core.windows.net/bronze/

Enter the bucket path that you want to use as the external location

-

[Cancel](#) [Create](#)

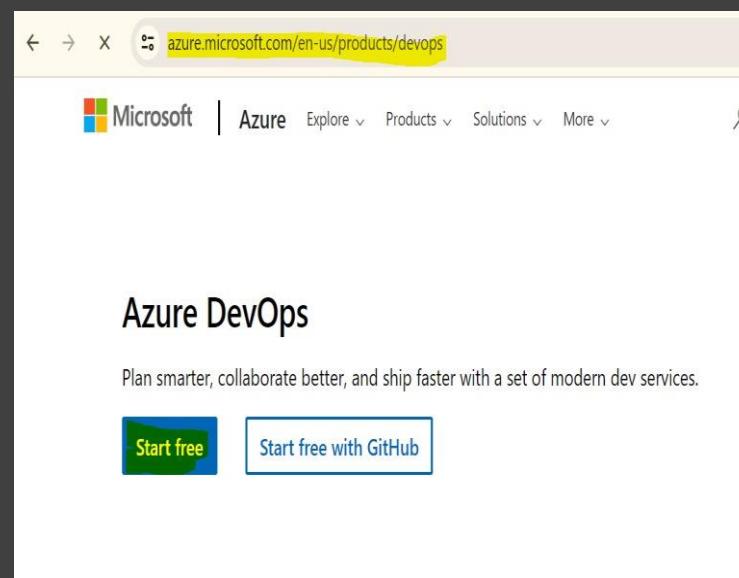
External Data

Name	Credential
bronze	storagecredential
bronze-uat	uat-access
checkpoints	storagecredential
checkpoints-uat	uat-access
databricks_uat_ws	databricks_uat_ws
deltastorageolocation	storagecredential

Name	Credential
gold	storagecredential
gold-uat	uat-access
landing	storagecredential
landing-uat	uat-access
samestorage	storagecredential
silver	storagecredential
silver-uat	uat-access
uat-catalog	storagecredential

## Q. How to sign in and use the Azure Devops?



A screenshot of the Azure DevOps interface. On the left, there's a sidebar with a 'databricks-traffic' project selected. The main area shows a message: 'databricks-traffic is empty. Add some code!'. A modal window titled 'Create a repository' is open on the right. It has a 'Repository type' dropdown set to 'Git', a 'Repository name \*' field containing 'dbproject', and a checked 'Add a README' checkbox. Below these are dropdowns for 'Add a .gitignore' (set to 'None') and a note about initializing the repository with a main branch. At the bottom of the modal are 'Cancel' and 'Create' buttons, with 'Create' being highlighted.

## Create a project to get started

Project name \*

Description

Visibility

Public  
Anyone on the internet can view the project. Certain features like TFVC are not supported.

Private  
Only people you give access to will be able to view this project.

Public projects are disabled for your organization. You can turn on public visibility with [organization policies](#).

# Q. How to link databricks and azure devops?

To link Databricks and Azure DevOps, use the Azure DevOps services to create a new pipeline. Configure the pipeline to use Databricks CLI or REST API for deploying and managing Databricks resources. Secure the connection with service principals and store secrets in Azure Key Vault

The left screenshot shows the Microsoft Azure Databricks interface. In the sidebar, under 'New', there is a 'Linked accounts' option highlighted with a yellow box. The main area is titled 'Settings' and contains sections for 'Workspace admin' (Appearance, Identity and access, Security, Compute, Development, Notifications, Advanced), 'User' (Profile, Preferences, Developer), and 'Linked accounts'. The 'Linked accounts' section is also highlighted with a yellow box. The right screenshot shows the 'Linked accounts' page with the heading 'Git integration'. It discusses co-versioned repos and individual notebooks. A section for 'Git provider' is shown, with 'Azure DevOps Services (Azure Active Direct...' selected, which is also highlighted with a yellow box. A 'Save' button is at the bottom.

# Q. How to add repo in Databricks?

Using a repository in Databricks is essential for effective version control. It allows you to track changes in your code over time, making it easier to manage different versions and revert to previous states if needed. Repos enable you to create branches for working on new features or experiments without affecting the main codebase. This collaborative approach ensures that multiple team members can work on the same project simultaneously, enhancing productivity and reducing conflicts. Overall, repos streamline the development process and help maintain a clean and organized codebase

The left screenshot shows the 'Create Git folder' dialog in Databricks. It has fields for 'Git repository URL' (containing 'https://example.com/organization/project.git') and 'Git provider' (a dropdown set to 'Select a Git provider'). There are also fields for 'Git folder name' and 'Sparse checkout mode'. The right screenshot shows the 'Clone Repository' dialog. It has tabs for 'Command line' (selected), 'HTTPS' (selected), and 'SSH'. The URL 'https://iamsaradatascientist@dev.azure.com' is entered in the HTTPS field. Below are sections for 'Generate Git Credentials' and 'IDE' (with 'Clone in VS Code' selected). A note at the bottom suggests using the latest version of Git for Windows or its plugins for IntelliJ, Eclipse, Android Studio, or Windows command line.

The image shows two screenshots from the Azure DevOps interface. The top screenshot is a 'Create Git folder' dialog box. It contains fields for 'Git repository URL' (https://iamsaradatascientist@dev.azure.com/iamsaradata), 'Git provider' (Azure DevOps Services), 'Git folder name' (dbproject), and a checkbox for 'Sparse checkout mode'. The bottom screenshot shows the 'Branches' page for a repository named 'main'. It displays a list of branches with 'main' selected as the default branch. On the right, the 'Policies' tab is active for the 'main' branch, showing 'Branch Policies'. A note states: 'Note: If any required policy is enabled, this branch cannot be deleted and changes must be made via pull request.' A toggle switch is set to 'On' for the 'Require a minimum number of reviewers' policy. The 'Minimum number of reviewers' input field is highlighted with a yellow border. Below the input field are four optional checkboxes: 'Allow requestors to approve their own changes', 'Prohibit the most recent pusher from approving their own changes', 'Allow completion even if some reviewers vote to wait or reject', and 'When new changes are pushed:'.

**Q. How can we effectively set up and manage a CI/CD pipeline for deploying code changes from a feature branch to a live folder, and ensure smooth integration with the UAT environment using Azure DevOps and Azure Databricks?**

To effectively set up and manage a CI/CD pipeline for deploying code changes from a feature branch to a live folder, and ensure smooth integration with the UAT environment using Azure DevOps and Azure Databricks, you need to establish a Git repository for version control and branching. Azure

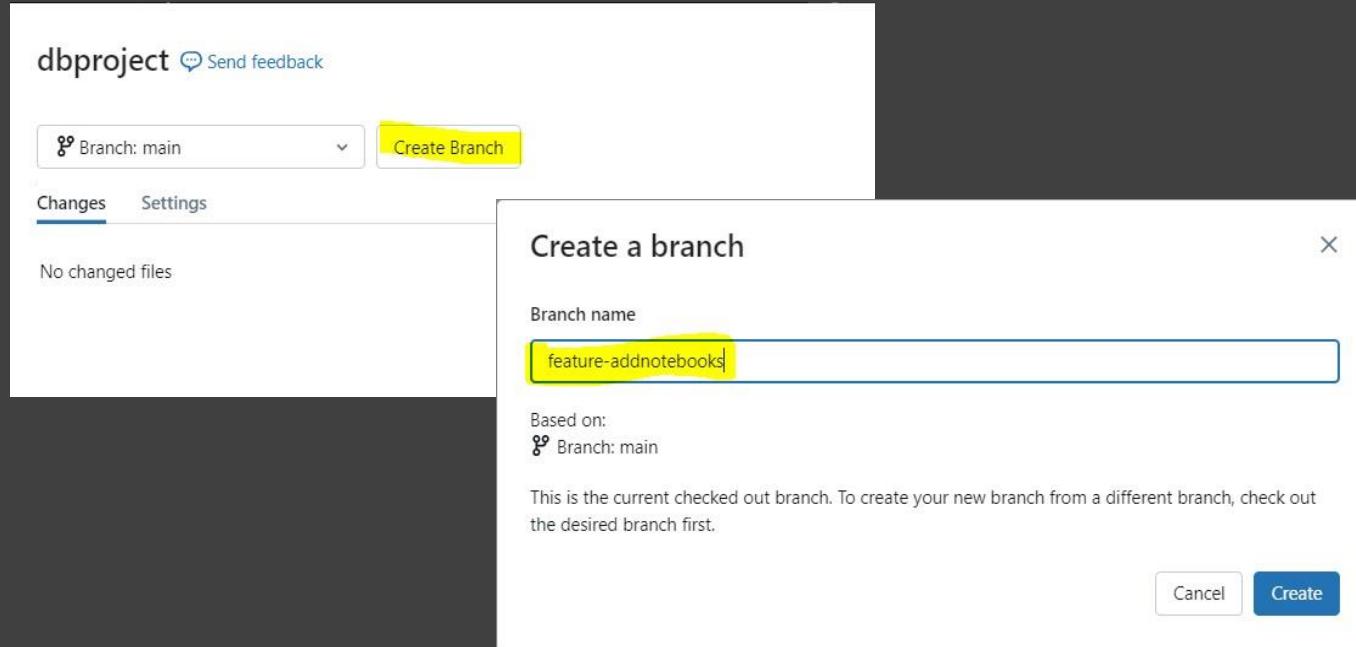
DevOps pipelines should be configured for Continuous Integration (CI) to automate building and testing of code changes. Continuous Deployment (CD) pipelines should then deploy the tested code to the Databricks environment, ensuring that changes are smoothly propagated to the UAT environment. This

approach allows for automated and reliable updates, promoting seamless development and integration workflows.

## 1. Feature Branch Creation and Pull Request:

- o Create a Feature Branch:

- In Azure DevOps, navigate to your repository and select the main branch.
- Click on "Branch" and create a new branch (e.g., feature/add-new-notebooks) from the main branch.



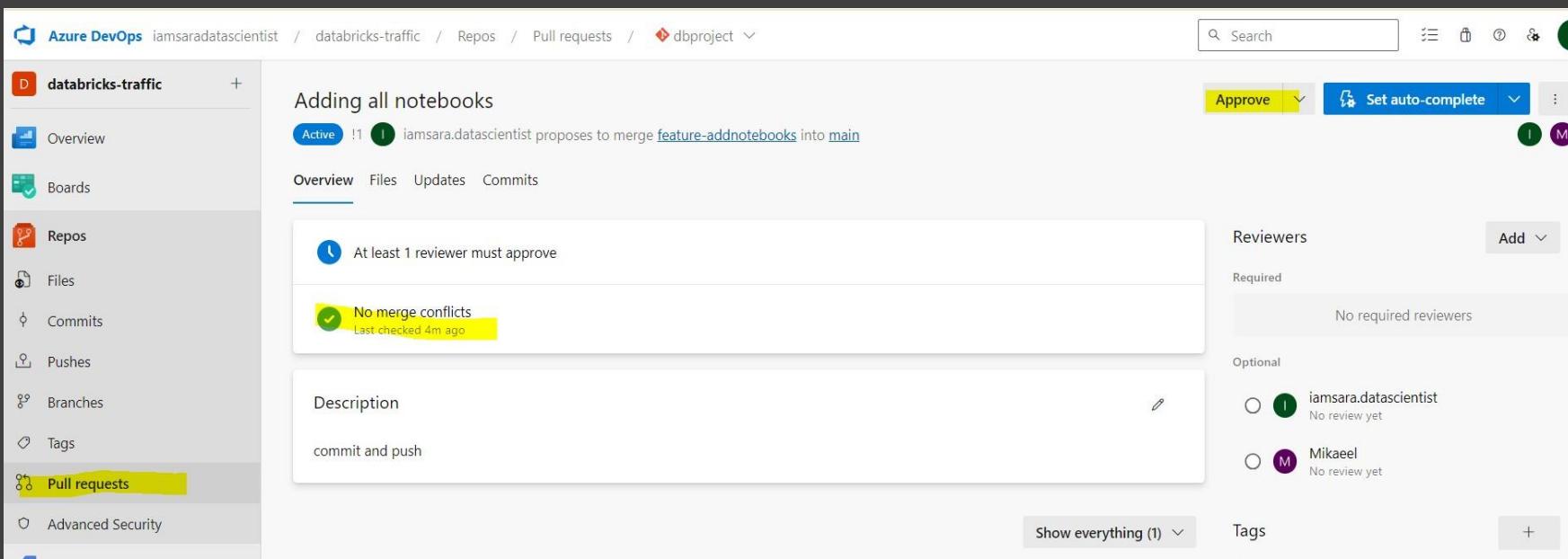
- o Commit Changes:

- Make necessary changes or additions to your code in the feature branch.
- Commit these changes with a clear message.

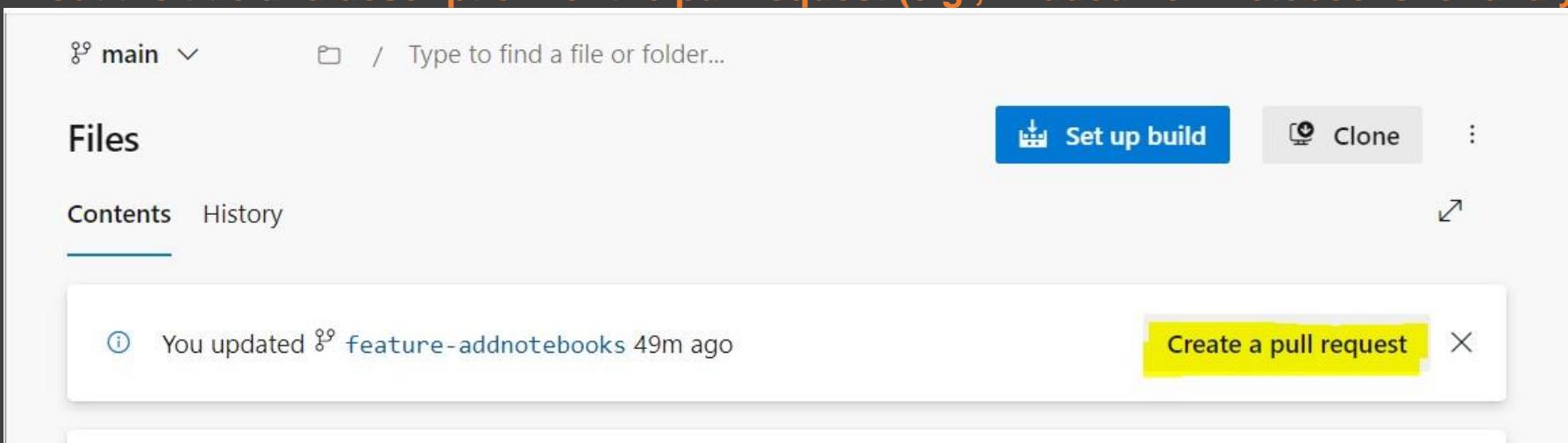
Name	Type	Owner
01 Project Setup	Notebook	Sara Irfan
02 Load to Bronze	Notebook	Sara Irfan
03 Silver Traffic Transformations	Notebook	Sara Irfan
04 Common Notebook	Notebook	Sara Irfan
05 Silver Roads Transformation	Notebook	Sara Irfan
06 Gold Layer Final Transformation	Notebook	Sara Irfan
OUTPUT MODE	Notebook	Sara Irfan
Playground	Notebook	Sara Irfan
README.md	File	Sara Irfan
Spark Structured Streaming	Notebook	Sara Irfan

- o Create a Pull Request:

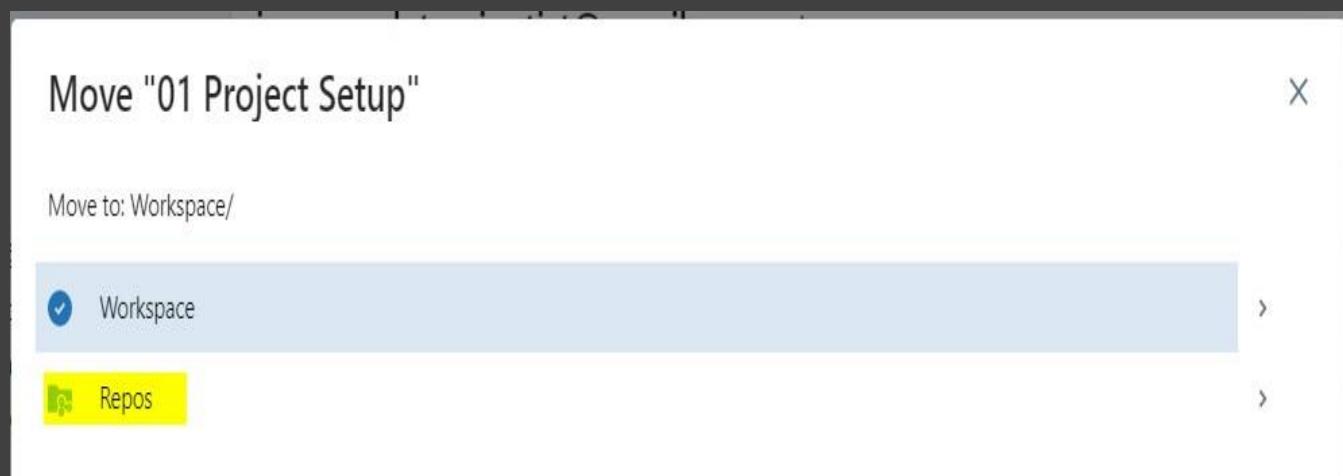
- Go to the "Pull Requests" section and click on "New Pull Request."
- Set the source branch to your feature branch and the target branch to main.



- Fill out the title and description for the pull request (e.g., "Added new notebooks for analysis").



- Reviewer and Approvals:
- Assign reviewers who are responsible for approving changes (e.g., a senior developer or team lead).
- Reviewers will assess the changes and provide feedback. The pull request must be approved before it can be merged.



## Q. How to add another user with same Working?

Users			
All users	Group rules	Export users	
<input type="button"/> Filter users			Access Level <input type="button"/>
Total 2		Summary	Add users
<input type="checkbox"/> Name ↑	Access Level	Date Added	Last Accessed
<input type="checkbox"/>  Mikaeel@iamsaradatascientist@gmail.onmicrosoft.com Mikaeel@iamsaradatascientist@gmail.onmicrosoft.com	Basic	7/31/2024	Never
<input type="checkbox"/>  iamsara.datascientist iamsara.datascientist@gmail.com	Basic	7/31/2024	7/31/2024

Databricks

Search data, notebooks, recents, and more... CTRL + P

databricksdevws M

**Settings**

- User
- Profile
- Preferences
- Developer
- Linked accounts**
- Notifications

**Linked accounts**

Connect your Databricks account to other services

**Git integration**

With co-versioned repo

Databricks Git folders and Repos allow you to clone a remote Git repo, which you can specify when you add a Git folder. Learn more [🔗](#)

With individual notebooks

Although we recommended using co-versioned repo for Git integration, Databricks supports individual notebook version control integration with GitHub [🔗](#), Bitbucket Cloud [🔗](#), or Azure DevOps Services [🔗](#) (using AAD authentication only).

Set your Git provider and credentials

You can also set your Git provider credentials via API. Learn more [🔗](#)

 **Azure DevOps Services (Personal access token)**  
iamsara.datascientist

Edit  Delete

**Credentials update**  
Successfully saved.

**Add Repo**

**Info** You can now create Git folders (previously Repos) outside the Repos folder. Go to home folder and create Git folder.

Create repo by cloning a Git repository

**Git repository URL**

**Git provider**

**Repository name**

Sparse checkout mode

Cancel

Merged PR 1: Adding all notebooks...

ab3ef810 iamsara.datascientist authored and Mikael committed Just now

Files Details

Parent 1 → This commit Filter 9 changed files

dbproject

- PY 01 Project Setup.py
- PY 02 Load to Bronze.py
- PY 03 Silver Traffic Transformatio...
- PY 04 Common Notebook.py
- PY 05 Silver Roads Transformatio...
- PY 06 Gold Layer Final Transform...
- PY OUTPUT MODE.py
- PY Playground.py
- PY Spark Structured Streaming.py

01 Project Setup.py +135 /01 Project Setup.py

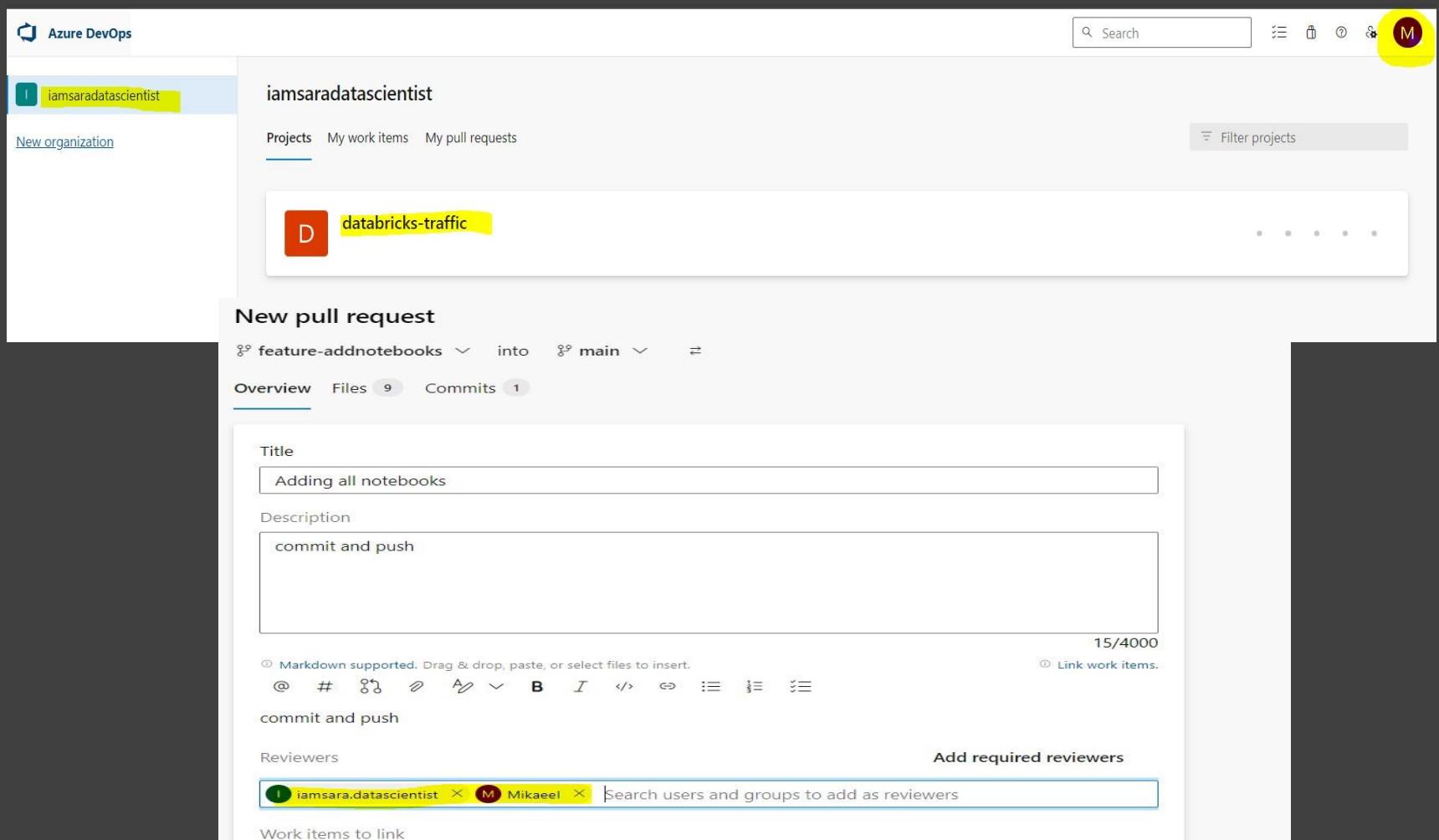
```

1 + # Databricks notebook source
2 + # MAGIC %md
3 + # MAGIC #Giving the path for external location
4 +
5 + # COMMAND -----
6 +
7 + # MAGIC %run "/Users/iamsara.datascientist@gmail.com/04 Common Notebook"
8 +
9 + # COMMAND -----
10 +
11 + dbutils.widgets.text(name="env",defaultValue="",label=" Enter the environment in lower case")
12 + env = dbutils.widgets.get("env")
13 +
14 + # COMMAND -----
15 +
16 + def create_Bronze_Schema(environment,path):
17 +     print(f'Using {environment}Catalog ')
18 +     spark.sql(f""" USE CATALOG '{environment}catalog'""")
19 +     print(f'Creating Bronze Schema in {environment}Catalog')
20 +     spark.sql(f"""CREATE SCHEMA IF NOT EXISTS `bronze` MANAGED LOCATION '{path}/bronze'""")
21 +     print("*****")
22 +
23 + # COMMAND -----
24 +
25 + def create_Silver_Schema(environment,path):

```

## Creating feature branch and adding users to the service principal

The screenshot shows two side-by-side windows. On the left is a pull request interface for a 'dbproject' repository. The title is 'Adding all notebooks'. The description is 'commit and push'. Under 'Reviewers', the email 'iamsara.datascientist' is listed. On the right is a modal dialog titled 'Add new users'. It contains fields for 'Users or Service Principals \*' (with 'Mikael@iamsaradatascientist@gmail.onmicrosoft.com' entered), 'Access level' (set to 'Basic'), 'Add to projects' (set to 'databricks-traffic'), and 'Azure DevOps Groups' (set to 'Project Contributors'). A checked checkbox says 'Send email invites (to Users only)'. At the bottom are 'Cancel' and 'Add' buttons.



## 2. Live Folder Creation:

- Create a Live Folder:
  - In Azure DevOps, configure a pipeline to deploy code to a live folder where the latest code from the main branch will be reflected.
  - Ensure the folder is set up in a location accessible to all relevant workflows.
- CI Pipeline Role:
  - The CI pipeline triggers whenever changes are made to the main branch.
  - It automatically updates the live folder with the latest code to ensure continuous integration and up-to-date workflows.

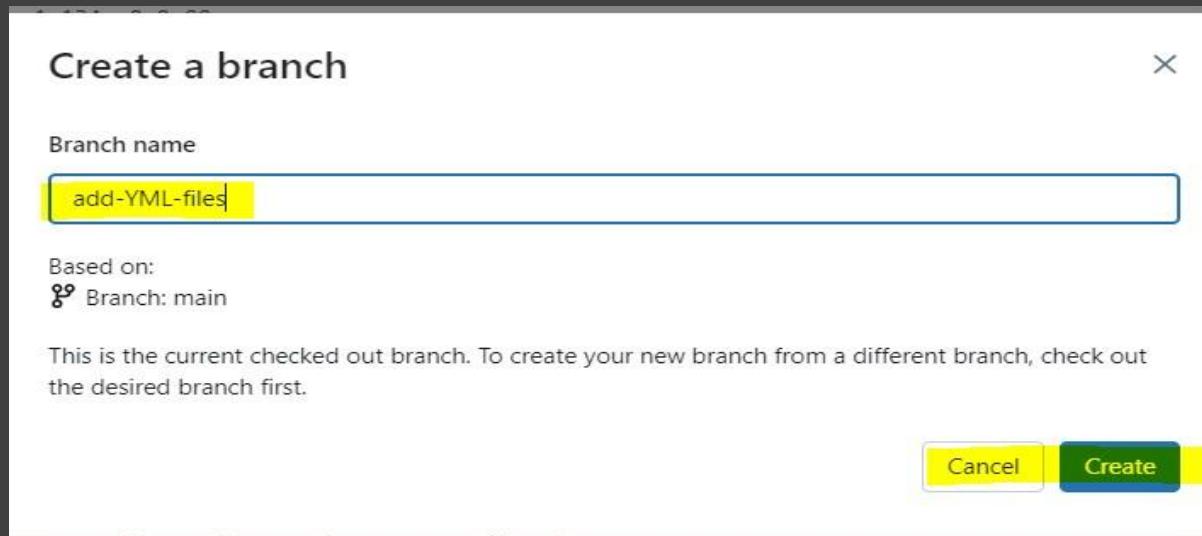
## 3. CI/CD Pipeline Configuration:

### Using Yaml and PowerShell Scripts:

- Yaml Files:
  - Define pipeline configurations using Yaml files. This includes specifying triggers, variables, agents, and stages.
  - For instance, `ci-cd-main.yaml` can be used to set up the main CI pipeline.
- PowerShell Scripts:
  - Use PowerShell scripts for specific tasks like creating live folders or handling configuration settings.
  - For example, a script can be written to automate the creation of the live folder.
- Managing Pipeline Variables:
  - Set up variables in Azure DevOps for different environments (e.g., UAT-ResourceGroup, UAT-ServiceConnection).
  - Define these variables in the pipeline settings to ensure the pipeline operates correctly across environments.

## 4. Integration with Databricks:

- Uploading and Reviewing Yaml Files:



dbproject			
Name	Type	Owner	Created at
notebooks	Folder	Sara Irfan	2024-08-01 01:15:12
README.md	File	Sara Irfan	2024-07-31 11:55:12

- Upload Yaml files to Databricks for pipeline configuration and review them within the Databricks workspace.

Name	Type
DBToken.ps1	File

Name	Type
notebooks	Folder
shellscrip	Folder
cicd-main.yml	File
deploy.yml	File
README.md	File

- Configuring Notebooks and External Locations:

- Set up external locations in Databricks for different stages (e.g., bronze-UAT, silver-UAT).
- Ensure notebooks are configured to use these locations and handle data appropriately.

## 5. Testing and Verification:

- Create a cluster in Databricks for testing. Ensure it's configured correctly for the UAT environment.
- Check that UAT catalogs and external locations are set up and pointing to the correct data sources.
- Verify that all notebooks use the correct paths and configurations for UAT.
- Upload test data to the UAT storage and run pipelines to verify that data processing is working as expected.

Repos > iamsara.datascientist@gmail.com >

## dbproject

[add-YML-files](#)

Name	Type
CICD	Folder
notebooks	Folder
README.md	File

Branch: add-YML-files

Changes Settings

21 changed files

- DBToken.ps1
- notebooks/
  - 01 Project Setup.py
  - 02 Load to Bronze.py
  - 03 Silver Traffic Transformations.py
  - 04 Common Notebook.py
  - 05 Silver Roads Transformation.py
  - 06 Gold Layer Final Transformation.py
  - OUTPUT MODE.py
  - Playground.py
  - Spark Structured Streaming.py

commit files

Description (optional)

Commit & Push

## 6. Permissions and Access Control:

- Ensure that only authorized users have access to the live folder and Databricks workspace.
- For users who need to manage or view the live folder, assign them appropriate permissions (e.g., admin roles).

## 7. Deployment to Production:

- Configure the pipeline to require approval before deploying changes to production.
- Verify that the UAT environment is stable before proceeding with the production deployment.
- Ensure that changes are deployed in a controlled manner to prevent disruptions in the production environment.

### New pull request

add-YML-files into main

Overview Files 3 Commits 1

Title: Added YAML Files

Description: commit files

12/4000

Markdown supported. Drag & drop, paste, or select files to insert.

commit files

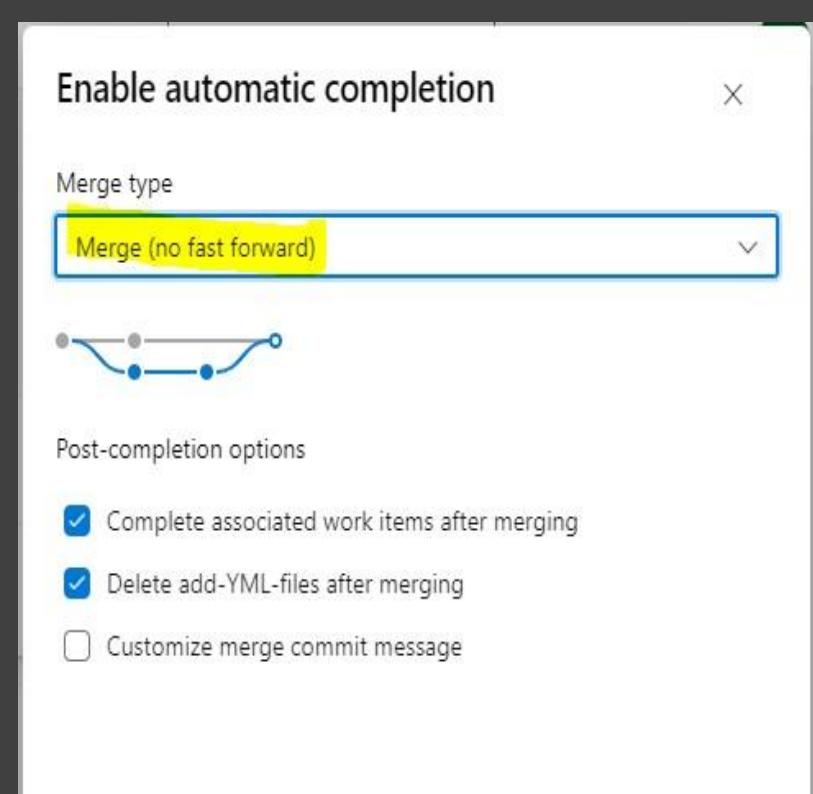
Reviewers: Search users and groups to add as reviewers

Add required reviewers

Work items to link: Search work items by ID or title

Tags

Create



Azure DevOps interface showing a completed pull request for the repository 'databricks-traffic'. The pull request, titled 'Merged PR 3: Added YAML Files', has been merged into the 'main' branch. The status bar indicates '13' changes and '1 reviewer approved'. The merge was completed 'Just now' by 'iamsara.datascientist'. There are 'No merge conflicts' and it was last checked 'Just now'. Buttons for 'Cherry-pick' and 'Revert' are visible.

## Working in Mikael's Account, creating feature branch, commit and push, then pull request to implement changes. Furthermore, pull request the main feature branch

Azure DevOps interface showing the contents of the 'dbproject' repository under 'mikaeel@iamsaradatascientistgmail.onmicrosoft.com'. The repository contains several notebooks and files, including 'README.md'. The table below lists the items:

Name	Type	Owner	Created at
01 Project Setup	Notebook	Mikael Developer	2024-07-31 18:06:15
02 Load to Bronze	Notebook	Mikael Developer	2024-07-31 18:06:15
03 Silver Traffic Transformations	Notebook	Mikael Developer	2024-07-31 18:06:15
04 Common Notebook	Notebook	Mikael Developer	2024-07-31 18:06:15
05 Silver Roads Transformation	Notebook	Mikael Developer	2024-07-31 18:06:15
06 Gold Layer Final Transformation	Notebook	Mikael Developer	2024-07-31 18:06:15
OUTPUT MODE	Notebook	Mikael Developer	2024-07-31 18:06:15
Playground	Notebook	Mikael Developer	2024-07-31 18:06:15
README.md	File	Mikael Developer	2024-07-31 18:06:15
Spark Structured Streaming	Notebook	Mikael Developer	2024-07-31 18:06:15

## Create a folder named test

Azure DevOps interface showing the contents of the 'dbproject' repository under 'mikaeel@iamsaradatascientistgmail.onmicrosoft.com'. A new folder named 'test' has been added to the list of items.

Create a feature branch, feature-addfolder, new folder named as notebooks, in the dbprojct of repos

The screenshot shows two windows. On the left is a 'Branches' interface with branches 'feature-addfolder' and 'main'. On the right is a 'New folder' dialog box where 'notebooks' is typed into the input field. Below the input field are 'Cancel' and 'Create' buttons. The 'Create' button is highlighted with a yellow box.

The bottom part of the screenshot shows a 'Workspace' sidebar and a 'dbproject' details view. The 'Workspace' sidebar includes 'Home', 'Shared', 'Users', 'Repos', and 'Favorites'. The 'dbproject' view shows a table with one item: 'notebooks' (Type: Folder) and 'README.md' (Type: File). The 'notebooks' row is highlighted with a yellow box.

## Commit and Push

The screenshot shows the 'dbproject' interface. At the top, it says 'Branch: feature-addfolder' and has 'Create Branch' and 'Changes' tabs. Under 'Changes', there are 19 changed files listed under the 'notebooks/' folder, including '01 Project Setup.py', '02 Load to Bronze.py', etc. Below the file list is a text input field containing 'create a folder call notebook' and a 'Description (optional)' input field. At the bottom is a 'Commit & Push' button. To the right, a preview of the '01 Project Setup.py' notebook code is shown.

```

@@ -1,134 +0,0 @@
-# Databricks notebook source
-# MAGIC %md
-# MAGIC #Giving the path for external
-
-# COMMAND -----
-
-# MAGIC %run "/Users/iamsara.datascientist@gmail.com/Shared/dbproject/notebooks/01 Project Setup.py"
-
-# COMMAND -----
-
-dbusutils.widgets.text(name="env", default="dev")
-env = dbusutils.widgets.get("env")
-
-# COMMAND -----
-
-def create_Bronze_Schema(environment, path):
-    print(f'Using {environment} Catalog')
-    spark.sql(f""" USE CATALOG '{environment}' """)
-    print(f'Creating Bronze Schema in {path}')
-    spark.sql(f"""CREATE SCHEMA IF NOT EXISTS {environment}""")
-    print("*****")
-
-# COMMAND -----
-
-def create_Silver_Schema(environment, path):
-    print(f'Using {environment} Catalog')
-    spark.sql(f""" USE CATALOG '{environment}' """)
-
```

## Create a pull request

New pull request

feature-addfolder into main

Overview Files 10 Commits 1

Title: create a folder call notebook

Description: create a folder call notebook

29/4000

Markdown supported. Drag & drop, paste, or select files to insert.

Link work items.

create a folder call notebook

Reviewers: Add required reviewers

Search users and groups to add as reviewers

Work items to link: Search work items by ID or title

Tags:

**Create**

Approve from Sara's account(Workspace Admin), the click on approve and complete from Mikael's Account (a developer), then click on auto complete from Admin's account

create a folder call notebook

Active |2 M Mikael proposes to merge feature-addfolder into main

Overview Files Updates Commits

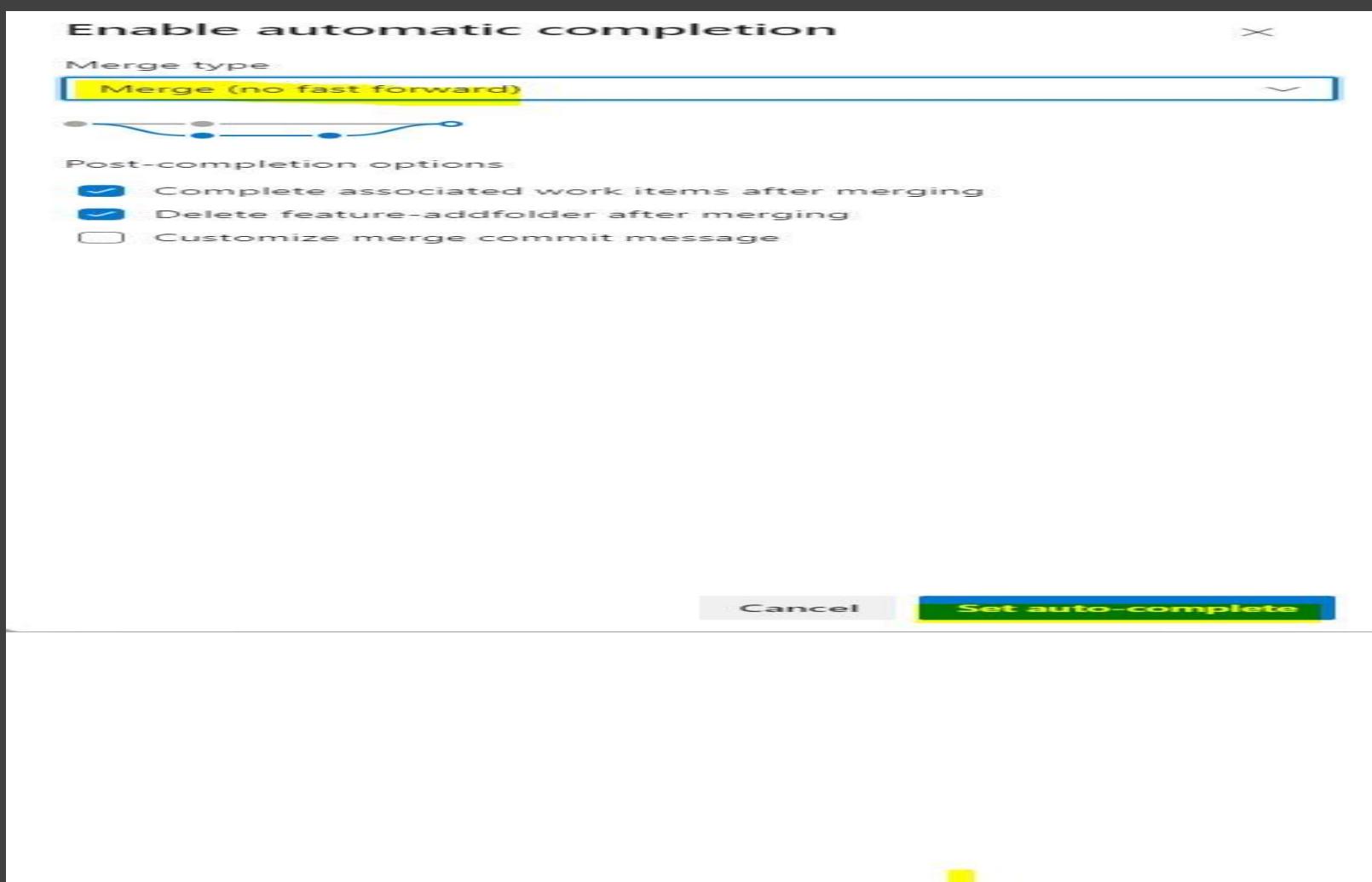
At least 1 reviewer besides author must approve

No merge conflicts

Last checked Just now

Reviewers

Required



# Q. How to get notifications?

To receive email notifications for approvals and other events in Azure DevOps, navigate to your Azure DevOps project and click on your user settings (usually represented by your profile picture). From there, select "Notifications." In the notification settings, you can create a new subscription by clicking on "New subscription." You can either choose a predefined template or create a custom subscription tailored to your needs. Configure the conditions under which you want to receive notifications, such as approvals or work item changes, and specify your email address as the destination. This setup ensures you stay informed about important activities and approvals in your Azure DevOps workflows.

The screenshot shows the 'Job notifications' configuration page. A subscription for 'Azure Databricks job ETL Flow updates' has been confirmed via email to 'iamsara.datascientist@gmail.com'. The notification type is set to 'Failure'. The message body includes a confirmation message from Microsoft Azure and a link to 'View this job now >'. The interface also shows options to mute notifications for skipped or canceled runs.

The screenshot shows an approval request for a stage named 'Deploy\_to\_UAT\_Env' in a dbproject. The stage requires approval from 'Mikaeel'. A 'Review approval' button is visible.

The screenshot shows a pull request review status. It indicates that 'Mikaeel approved the changes'. Two reviewers, 'iamsara.datascientist' and 'Mikaeel', have both approved the pull request, which is marked as optional.

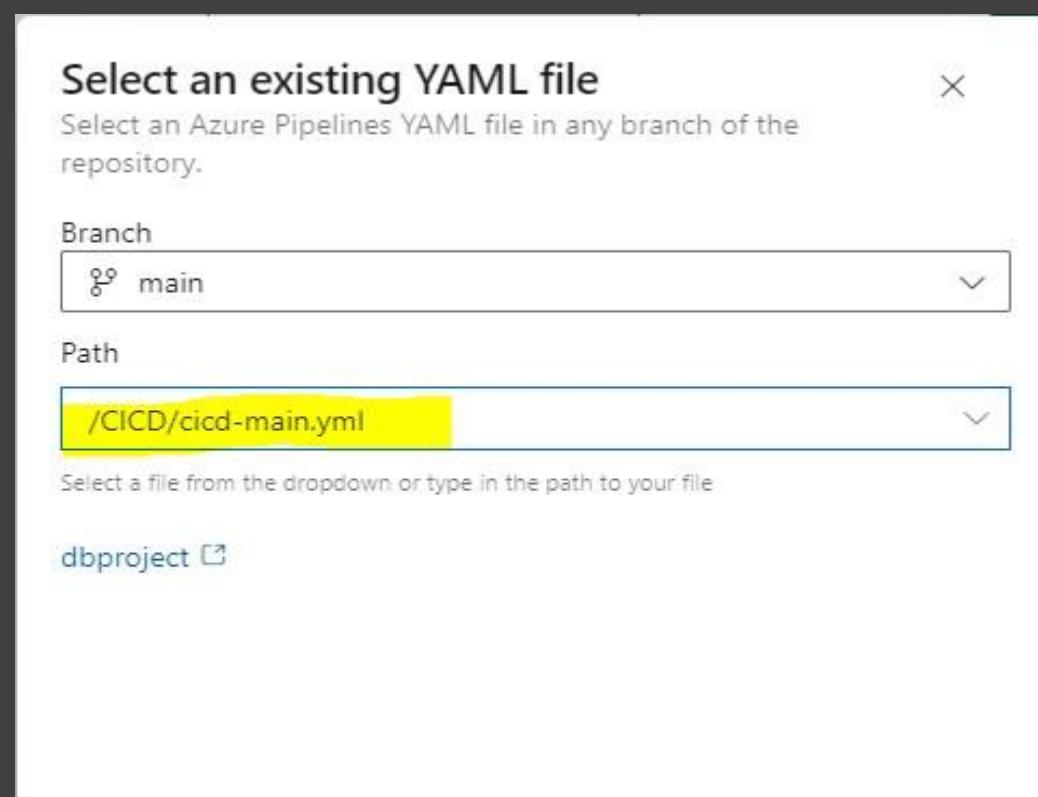
# Q. How to create CI pipeline to have live folder?

Creating a Continuous Integration (CI) pipeline to manage a live folder is a crucial practice in modern software development and operations. A CI pipeline automates the process of integrating code changes from multiple contributors, running tests, and deploying updates to a live folder. This ensures that the live environment is always up-to-date with the latest code, reducing the risk of human error and increasing the reliability of deployments. In real-life scenarios, this approach is vital for maintaining consistent and reliable software operations, as it allows teams to detect and address issues early, streamlines the development process, and ensures that new features and fixes are delivered quickly and efficiently to users. Implementing a CI pipeline for a live folder ultimately leads to improved productivity, faster release cycles, and higher quality software.

The screenshot shows the 'Connect' step of creating a new pipeline in Azure DevOps. On the left, there's a sidebar with options like Overview, Boards, Repos, Pipelines (which is highlighted), Pipelines (another highlighted item), Environments, Library, Test Plans, and Artifacts. The main area has tabs for Connect, Select, Configure, and Review. It asks 'Where is your code?' and lists several options: 'Azure Repos Git - YAML' (selected and highlighted in yellow), 'Bitbucket Cloud - YAML' (hosted by Atlassian), 'GitHub - YAML' (home to the world's largest community of developers), and 'GitHub Enterprise Server - YAML' (the self-hosted version of GitHub Enterprise). The 'Azure Repos Git - YAML' option is described as 'Free private Git repositories, pull requests, and code search'.

The screenshot shows the 'Configure' step of creating a new pipeline. It follows the same structure as the previous screen, with tabs for Connect, Select, Configure (which is highlighted in blue), and Review. It asks 'Configure your pipeline' and lists four options: 'Python package' (create and test a Python package on multiple Python versions), 'Python to Linux Web App on Azure' (build your Python project and deploy it to Azure as a Linux Web App), 'Starter pipeline' (start with a minimal pipeline that you can customize to build and deploy your app), and 'Existing Azure Pipelines YAML file' (select an Azure Pipelines YAML file in any branch of the repository). A 'Show more' button is at the bottom.

## Select an existing cicd-main.yaml file



## Review your pipeline code

New pipeline

### Review your pipeline YAML

```
dbproject / CICD/cicd-main.yml
```

```
1 trigger:
2   - main
3
4 variables:
5   group: dev-cicd-grp
6   name: notebookPath
7   value: "notebooks"
8
9 pool:
10  vmImage: "windows-latest"
11
12 stages:
13   template: deploy.yml
14   parameters:
15     stageId: "Deploy_to_Dev_Env"
16     env: "dev"
17     environmentName: $(dev-env-name)
18     resourceGroupName: $(dev-resource-grp-name)
19     serviceConnection: $(dev-service-connection-name)
20     notebookPath: $(notebookPath)
```

## Three important steps: Enviornment name through library

Pipelines

Pipelines

Environments

**Library**

Test Plans

Artifacts

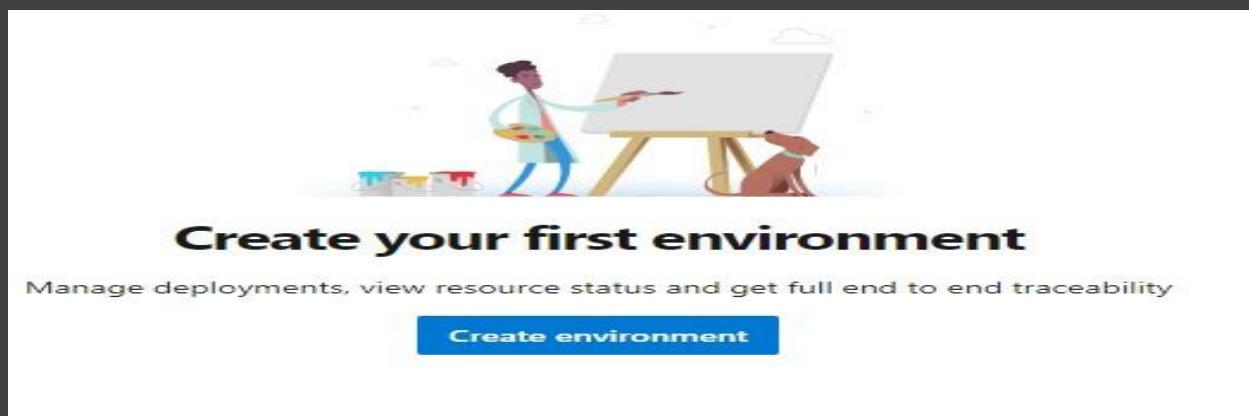
(x)

New variable group

Create groups of variables that you can share across multiple pipelines.

+ Variable group

Learn more about variable groups.



This screenshot shows the Azure DevOps Library interface. On the left, a sidebar lists project navigation options: Overview, Boards, Repos, Pipelines, Pipelines (selected), Environments, Library (selected), Test Plans, and Artifacts. The main content area displays a "Variable group" named "dev-cicd-grp". The "Properties" section includes fields for "Variable group name" (set to "dev-cicd-grp") and "Description". A toggle switch is set to "Link secrets from an Azure key vault as variables". The "Variables" section lists three entries: "dev-env-name" with value "dev-env-name", "dev-resource-grp-name" with value "databricksresource", and "dev-service-connector-name" with value "dev-service-name".

## Pipeline permissions are required

This screenshot shows the "Pipeline permissions" dialog for the "dev-cicd-grp" variable group. It lists the YAML pipelines allowed to use this resource. One pipeline, "dbproject Pipeline", is shown and highlighted with a yellow box. There are buttons for adding more pipelines (+) and more options (:).

## Resource group is same as used in azure for azure devops

Library > dev-cicd-grp

**Variable group**

**Properties**

Variable group name  
dev-cicd-grp

Description

Link secrets from an Azure key vault as variables

**Variables**

Name ↑	Value
dev-env-name	dev-cicd-name
dev-resource-grp-name	databricksresource
dev-service-connector-name	dev-service-name

+ Add

## Q. How to give service connection?

Creating a service connection in Azure DevOps is essential because it allows your pipelines to securely and seamlessly interact with external resources and services, such as cloud providers, databases, or container registries. It ensures that your CI/CD processes can access and manage resources, facilitating automated deployments, integrations, and infrastructure management.

Service connections

databricksresource | Access control (IAM) ...

Resource group

Filter by keywords

Search

Add Download role assignments Edit columns Refresh

dev-service-name

Overview Activity log Access control (IAM) Tags Resource visualizer Events Settings Cost Management Monitoring Automation Help

All Job function (0) Privileged (2)

Search by name or email Type : All Roles

2 items (1 Users, 1 Service Principals)

Name	Type	Role
Owner (1)	Sara Irfan iamsara.datascientist... User	Owner
Contributor (1)	iamsaradatascientist-dev App	Contributor

Home > Default Directory

## Default Directory | App registrations

+ New registration Endpoints Troubleshoot Refresh Download Preview features

Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL). We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications using ADAL will need to migrate to Microsoft Identity Platform (MSAL) and Microsoft Graph. [Learn more](#)

All applications Owned applications Deleted applications Applications from personal account

Start typing a display name or application (client) ID to filter these results... Add filters

2 applications found

Display name ↑	Application (client) ID
DB db_access	7977d277-d3dc-4846-83df-5f008...
IA iamsaradatascientist-databricks-traffic-ea8d46ac-c7b3-4f01-a3ea-a860b2f2d...	178fc50f-b8b4-4ab0-92ae-c8efac...

**iamsaradatascientist-databricks-traffic-ea8d46ac-c7b3-4f01-a3ea-a860b2f2d**

Search Got feedback?

Overview Quickstart Integration assistant Diagnose and solve problems Manage Branding & properties Authentication Certificates & secrets Token configuration API permissions Expose an API App roles Owners

Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens (using a client secret or certificate). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a means of authentication.

Application registration certificates, secrets and federated credentials can be found in the tabs below.

Certificates (0) Client secrets (0) Federated credentials (1)

Allow other identities to impersonate this application by establishing a trust with an external OpenID Connect provider. This application can then get tokens to access Microsoft Entra ID protected resources that this application has access to like Azure Active Directory.

Add credential

Name	Description
80a3e0d4-9a4c-4f98-a00c-ed564d562eaf	Federation for Service Connection dev-service-...

## Permissions for user is also required

**User permissions**

- [databricks-traffic]\Contributors Access: Inherited
- [databricks-traffic]\Project Administrators Access: Inherited
- [databricks-traffic]\Project Valid Users Access: Inherited
- iamsara.datascientist Access: Assigned

**Pipeline permissions**

The following YAML pipelines are allowed to use this resource. YAML pipelines from other projects are not shown in this list. All Classic pipelines can use this resource.

Pipeline

dbproject Pipeline

**User permissions**

User	Role	Access
[databricks-traffic]\Endpoint Administrators	Administrator	Inherited
iamsara.datascientist	Administrator	Assigned

**Pipeline permissions**

The following YAML pipelines are allowed to use this resource. YAML pipelines from other projects are not shown in this list. All Classic pipelines can use this resource.

Pipeline

dbproject Pipeline

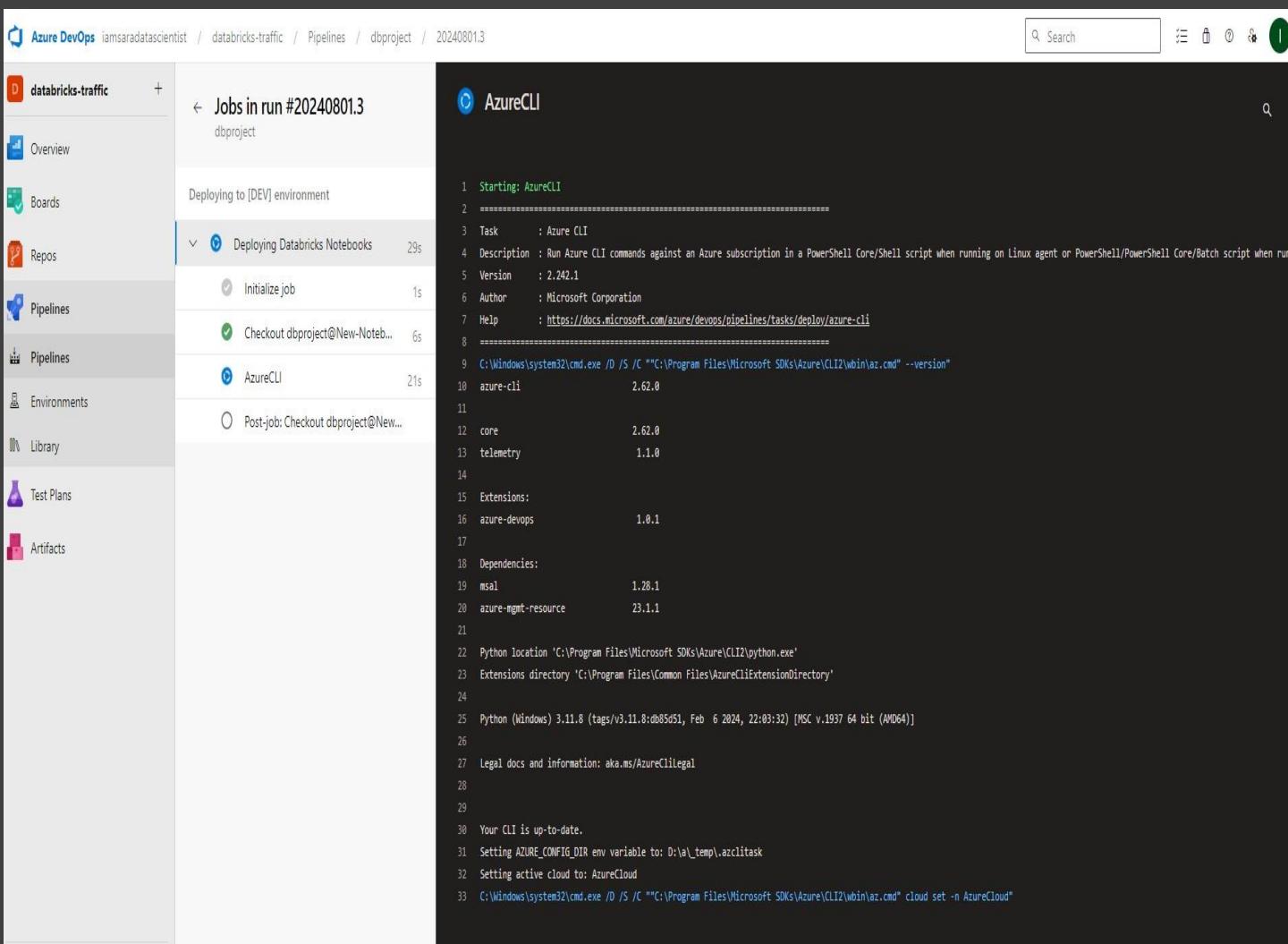
**User management**

Service principals are identities for use with automated tools, running jobs, and applications. Learn more.

Filter service principals Q

Name	Application ID	Status	Roles
Devops Service Principal	84da7a63-c1b1-4f7d-9cb6-a7bc5219b047	Active	

## Deploying databricks notebooks



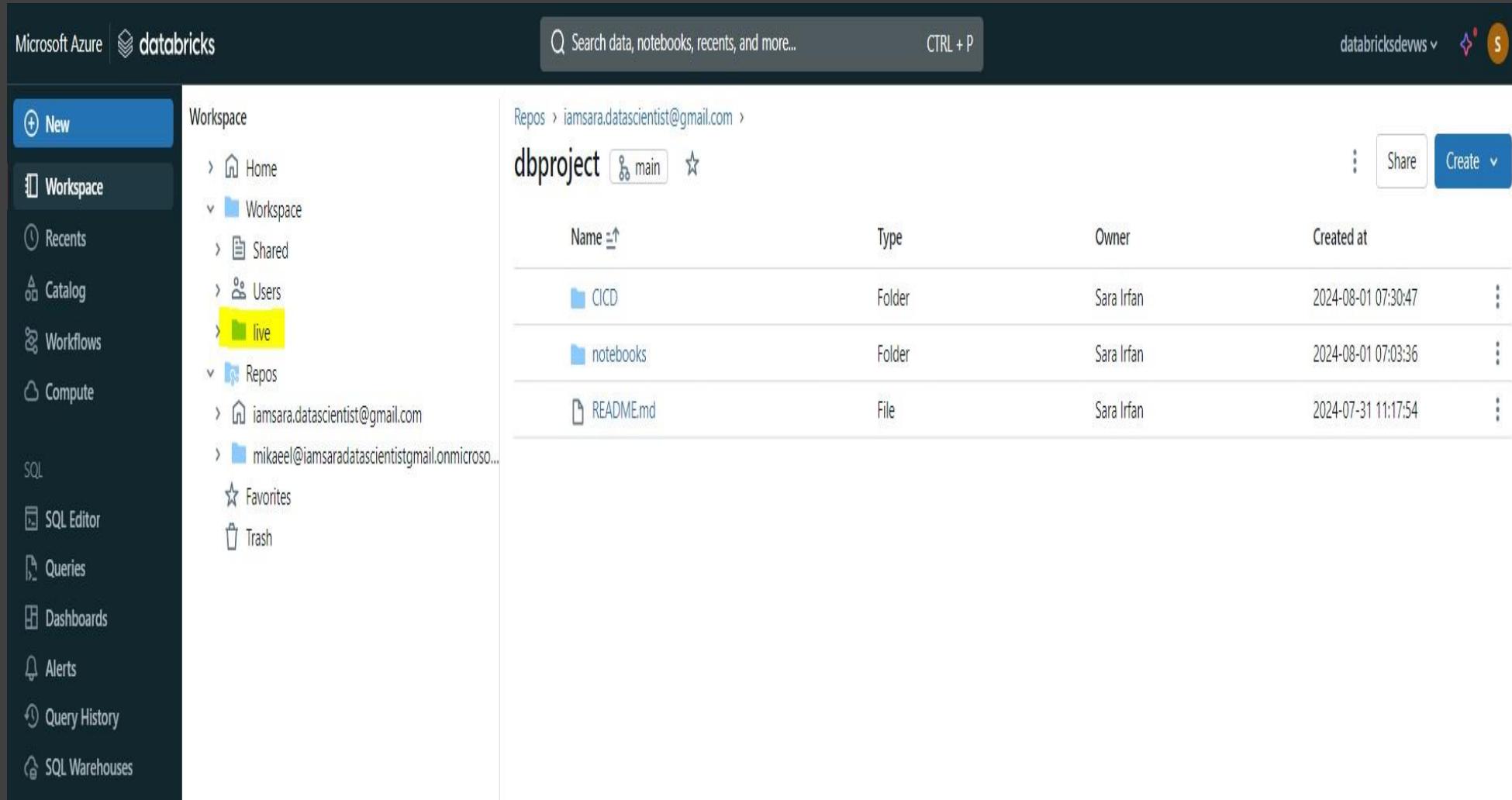
The screenshot shows the Azure DevOps interface for a pipeline named 'databricks-traffic' under the 'Pipelines' tab. The pipeline has a single job named 'dbproject'. The job is currently running and is described as 'Deploying to [DEV] environment'. The task within the job is 'Deploying Databricks Notebooks', which includes steps like 'Initialize job' and 'Checkout dbproject@New-Note...'. The output of the 'AzureCLI' task is displayed on the right, showing the command-line logs for the deployment process. The logs include details about the Azure CLI version (2.242.1), dependencies (msal 1.28.1, azure-mgmt-resource 23.1.1), and Python location ('C:\Program Files\Microsoft SDKs\Azure\CLI2\python.exe'). The logs also mention setting the AZURE\_CONFIG\_DIR env variable to 'D:\a\\_temp\azclitask' and setting the active cloud to 'AzureCloud'.

```

1 Starting: AzureCLI
2 =====
3 Task : Azure CLI
4 Description : Run Azure CLI commands against an Azure subscription in a PowerShell Core/Shell script when running on Linux agent or PowerShell/PowerShell Core/Batch script when running on Windows agent
5 Version : 2.242.1
6 Author : Microsoft Corporation
7 Help : https://docs.microsoft.com/azure/devops/pipelines/tasks/deploy/azure-cli
8 =====
9 C:\Windows\system32\cmd.exe /D /S /C "C:\Program Files\Microsoft SDKs\Azure\CLI2\wbin\az.cmd" --version
10 azure-cli          2.62.0
11
12 core               2.62.0
13 telemetry          1.1.0
14
15 Extensions:
16   azure-devops      1.0.1
17
18 Dependencies:
19   msal              1.28.1
20   azure-mgmt-resource 23.1.1
21
22 Python location 'C:\Program Files\Microsoft SDKs\Azure\CLI2\python.exe'
23 Extensions directory 'C:\Program Files\Common Files\AzureCliExtensionDirectory'
24
25 Python (Windows) 3.11.8 (tags/v3.11.8:db08d61, Feb 6 2024, 22:03:32) [MSC v.1937 64 bit (AMD64)]
26
27 Legal docs and information: aka.ms/AzureCliLegal
28
29
30 Your CLI is up-to-date.
31 Setting AZURE_CONFIG_DIR env variable to: D:\a\_temp\azclitask
32 Setting active cloud to: AzureCloud
33 C:\Windows\system32\cmd.exe /D /S /C "C:\Program Files\Microsoft SDKs\Azure\CLI2\wbin\az.cmd" cloud set -n AzureCloud"

```

Live folder has been created:



The screenshot shows the Databricks workspace interface. On the left, the navigation sidebar includes options like 'New', 'Workspace', 'Recents', 'Catalog', 'Workflows', 'Compute', 'SQL', 'SQL Editor', 'Queries', 'Dashboards', 'Alerts', 'Query History', and 'SQL Warehouses'. The 'Workspace' option is currently selected. The main area displays a file tree under the 'dbproject' workspace. The 'live' folder, which was created in the previous step, is highlighted with a yellow border. The file tree also includes 'Home', 'Shared', 'Users', 'Repos' (containing 'iamsara.datascientist@gmail.com' and 'mikaeel@iamsaradatascientist@gmail.onmicrosoft.com'), 'Favorites', and 'Trash'. The 'Repos' section shows a list of files and folders: 'CICD' (Folder, Owner: Sara Irfan, Created at: 2024-08-01 07:30:47), 'notebooks' (Folder, Owner: Sara Irfan, Created at: 2024-08-01 07:03:36), and 'README.md' (File, Owner: Sara Irfan, Created at: 2024-07-31 11:17:54).

**Now please notice, all the notebooks are in live folder:**

The screenshot shows the Azure DevOps Pipelines interface. On the left, there's a sidebar with a 'databricks-traffic' project selected. The main area is titled 'Pipelines' and shows a table of recently run pipelines. One pipeline, 'dbproject', is highlighted with a yellow box. It has a green checkmark icon, the ID '#20240801.3 • all', and a note that it was 'Manually triggered for 1 New-Notebook'.

The screenshot shows the Databricks workspace. The left sidebar is the navigation menu. In the center, under the 'Workspace' section, the 'live' folder is selected and highlighted with a blue box. The right side shows a list of notebooks in the 'live' folder, each with its name, type (Notebook), owner, and creation date. The list includes: 01 Project Setup, 02 Load to Bronze, 03 Silver Traffic Transformations, 04 Common Notebook, 05 Silver Roads Transformation, 06 Gold Layer Final Transformation, OUTPUT MODE, Playground, Spark Structured Streaming, and Test Notebook.

Name	Type	Owner	Created at
01 Project Setup	Notebook	ab49c357-ee37-4ae9-bdf6-bee8d69f4...	2024-08-01 22:39:46
02 Load to Bronze	Notebook	ab49c357-ee37-4ae9-bdf6-bee8d69f4...	2024-08-01 22:39:47
03 Silver Traffic Transformations	Notebook	ab49c357-ee37-4ae9-bdf6-bee8d69f4...	2024-08-01 22:39:48
04 Common Notebook	Notebook	ab49c357-ee37-4ae9-bdf6-bee8d69f4...	2024-08-01 22:39:49
05 Silver Roads Transformation	Notebook	ab49c357-ee37-4ae9-bdf6-bee8d69f4...	2024-08-01 22:39:50
06 Gold Layer Final Transformation	Notebook	ab49c357-ee37-4ae9-bdf6-bee8d69f4...	2024-08-01 22:39:51
OUTPUT MODE	Notebook	ab49c357-ee37-4ae9-bdf6-bee8d69f4...	2024-08-01 22:39:52
Playground	Notebook	ab49c357-ee37-4ae9-bdf6-bee8d69f4...	2024-08-01 22:39:53
Spark Structured Streaming	Notebook	ab49c357-ee37-4ae9-bdf6-bee8d69f4...	2024-08-01 22:39:54
Test Notebook	Notebook	ab49c357-ee37-4ae9-bdf6-bee8d69f4...	2024-08-01 22:39:56

**Give admin permissions to Mikael:**

The screenshot shows the Databricks access control interface. It lists two users: 'Mikael Developer (mikael@iamsaradatascientistgmail.onmicrosoft.com)' and 'Sara Irfan (sara.data.insights@gmail.com)'. Both users are currently listed as 'User'. To the right of each user, there is a button labeled 'Admin' which is highlighted with a yellow box, indicating that clicking it will promote the user to Admin status.

## Live folder is also in Mikael's folder:

The screenshot shows a workspace interface. On the left, there is a sidebar with options like Home, Workspace, Shared, Users, and a highlighted 'live' folder. The main area shows a repository named 'dbproject' under 'mikael@iamsaradatascientist@gmail.onmicrosoft.com'. The repository contains three items: 'CICD' (Folder), 'notebooks' (Folder), and 'README.md' (File). The 'CICD' folder is expanded, showing its contents.

## Create live folder for uat resource group:

### Create a branch:

The screenshot shows a 'Create a branch' dialog box. The 'Branch name' field contains 'feature-CD1'. Below it, 'Based on:' is set to 'Branch: main'. A note at the bottom states: 'This is the current checked out branch. To create your new branch from a different branch, check out the desired branch first.' There are 'Cancel' and 'Create' buttons at the bottom right.

The screenshot shows the 'cicd-main.yml' file in the 'feature-CD1' branch. The code defines a pipeline with variables and a pool:

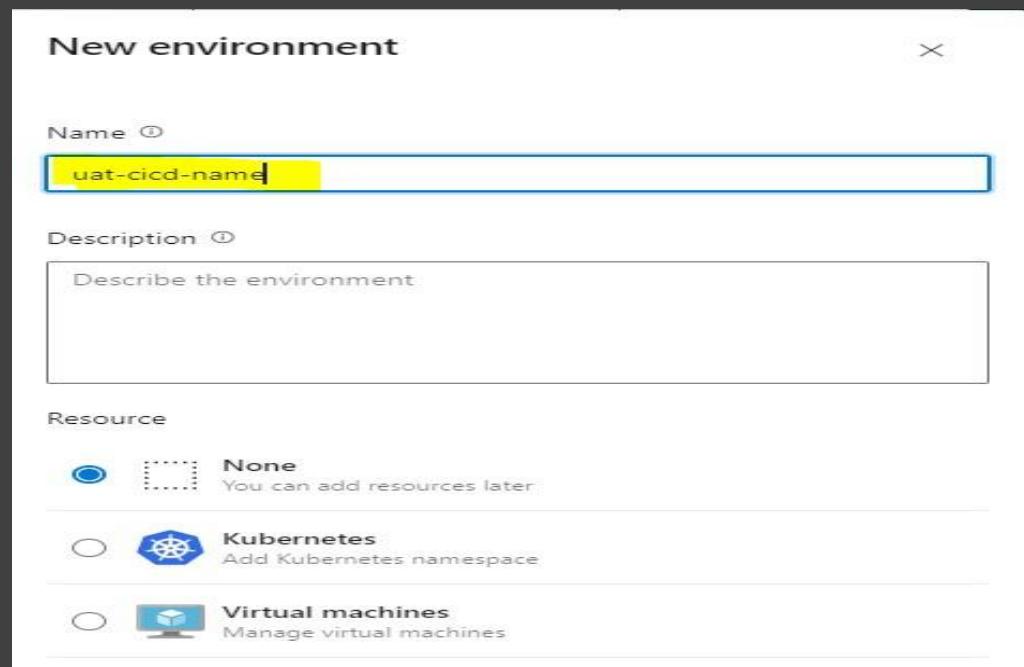
```
2 - main
3
4 variables:
5 - group: dev-cicd-grp
6 - group: uat-cicd-grp
7 - name: notebookPath
8   value: "notebooks"
9
10 pool:
11   vmImage: "windows-latest"
```

## Do these modifications in cicd-yaml file:

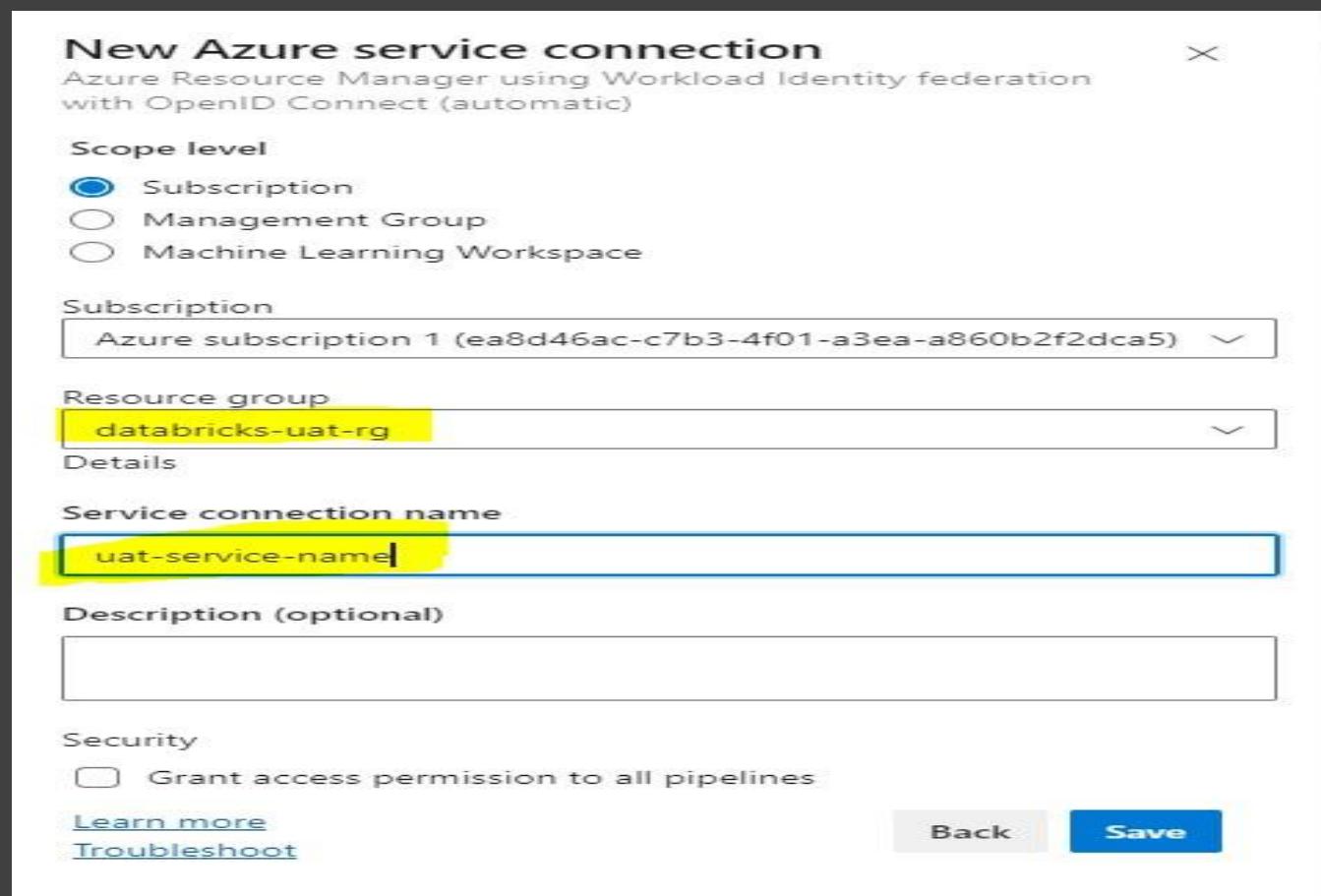
```
stages:
- template: deploy.yml
parameters:
  stageId: "Deploy_to_Dev_Env"
  env: "dev"
  environmentName: $(dev-env-name)
  resourceGroupName: $(dev-resource-grp-name)
  serviceConnection: $(dev-service-connection-name)
  notebookPath: $(notebookPath)

- template: deploy.yml
parameters:
  dependsOn: "Deploy_to_Dev_Env"
  stageId: "Deploy_to_UAT_Env"
  env: "uat"
  environmentName: $(uat-env-name)
  resourceGroupName: $(uat-resource-grp-name)
  serviceConnection: $(uat-service-connection-name)
  notebookPath: $(notebookPath)|
```

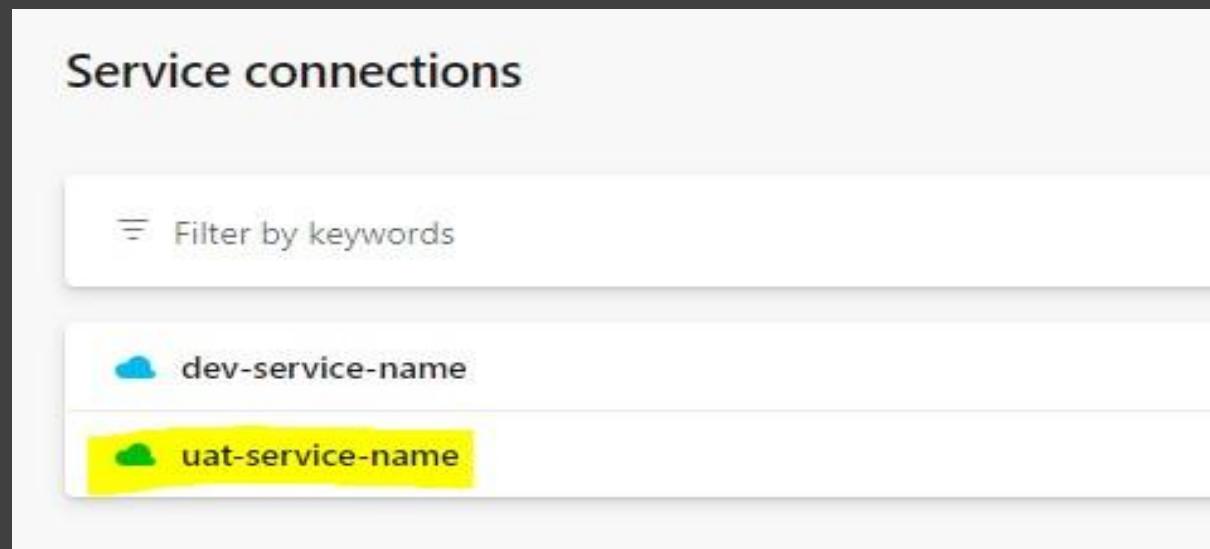
### Create an environment uat-cicd-name:



### Create azure service connection, with service principals:



### Check out the service connection:



Now go to manage account of specified workspace, open user management, service principal:

A screenshot of the 'User management' page in Azure DevOps. The 'Service principals' tab is selected. It shows a list of service principals with columns for Name, Application ID, Status, and Roles. Two entries are listed: 'Devops Service Principal' and another entry with a long GUID.

Click on SAVE:

A screenshot of the 'Variable group' edit page for 'uat-cicd-grp'. The 'Save' button is highlighted. The 'Properties' section shows the variable group name 'uat-cicd-grp'. The 'Variables' section lists three variables: 'uat-env-name' with value 'uat-cicd-name', 'uat-resource' with value 'databricks-uat-rg', and 'uat-service-connection-name' with value 'uat-service-name'. The 'uat-service-connection-name' row has a trash can icon and a lock icon.

Home > databricks-uat-rg

## databricks-uat-rg | Access control (IAM) ...

Resource group

Search Add Download role assignments Edit columns Refresh Delete Feedback

- Overview
- Activity log
- Access control (IAM)**
- Tags
- Resource visualizer
- Events
- > Settings
- > Cost Management
- > Monitoring
- > Automation
- > Help

All Job function (0) Privileged (2)

Search by name or email Type : All Role : All Scope : All scopes Group by : Role

2 items (1 Users, 1 Service Principals)

Name	Type	Role	Scope	Condition
Owner (1)				
Sara Irfan <span>iamsara.datascientist...</span>	User	Owner <span>Subscription (Inherited)</span>	Subscription (Inherited)	None
Contributor (1)				
iamsaradatascientist-d... <span>App</span>	App	Contributor <span>This resource</span>	This resource	None

All applications    Owned applications    Deleted applications    Applications from personal account

Start typing a display name or application (client) ID to filter these r... Add filters

3 applications found

Display name ↑↓	Application (client) ID	Created on ↑↓
db_access	7977d277-d3dc-4846-83df-5f00837f...	7/25/2024
iamsaradatascientist-databricks-traffic-ea8d46ac-c7b3-4f01-a3ea-a...	ab49c357-ee37-4ae9-bdf6-bee8d69f...	8/1/2024
iamsaradatascientist-databricks-traffic-ea8d46ac-c7b3-4f01-a3ea-a...	910f87f3-c8a9-4591-a692-36c16823...	8/2/2024

## User management

Users    **Service principals**    Groups

Service principals are identities for use with automated tools, running jobs, and applications. [Learn more](#).

Filter service principals Q

Name	Application ID
Devops UAT SP	1d1bcb47-e16c-4da6-90e1-d3d3d6d3ec66
Devops Service Principal	84da7a63-c1b1-4f7d-9cb6-a7bc5219b047

**Creating feature branch, Giving approvals, commit and push, create a pull request:**

## Approvals:

The screenshot shows a pull request merge screen. At the top, it says "select feature cd1" and indicates it's "Completed" by user "Mikael". Below this, there are tabs for "Overview", "Files", "Updates", and "Commits", with "Overview" being the active tab. A message states "Mikael completed this pull request Just now". A box highlights "Merged PR 6: select feature cd1" with commit hash "eb64fdd3" and author "Mikael Just now". Below this, there are two green checkmarks: "1 reviewer approved" and "No merge conflicts Last checked Just now". The "Description" section contains the text "select feature cd1". At the bottom, there's a comment input field with placeholder "Add a comment..." and a history of recent actions: "Mikael completed the pull request" and "iamsara.datascientist approved the pull request".

The screenshot shows the "Pipelines" page. It has tabs for "Recent", "All", and "Runs", with "Recent" selected. Below this, it says "Recently run pipelines". It lists a pipeline named "dbproject" which has just completed its last run. The run information includes the ID "#20240802.1", the status "Merged PR 6: select feature cd1", and a note about an "Individual CI for main".

Now you can see, another pipeline is deploying:

The screenshot shows a pipeline deployment screen. It has columns for "Description" and "Stages". The "Description" column shows the pipeline run "#20240802.1" and notes about an "Individual CI for main" and commit hash "eb64fdd3". The "Stages" column shows a yellow progress bar with a green checkmark icon and the number "1" inside it, indicating the first stage is complete.

#20240802.1 • Merged PR 6: select feature cd1

dbproject

Summary Environments Code Coverage

Triggered by M Mikael

Repository and version

dbproject  
main eb64fdd3

Time started and elapsed

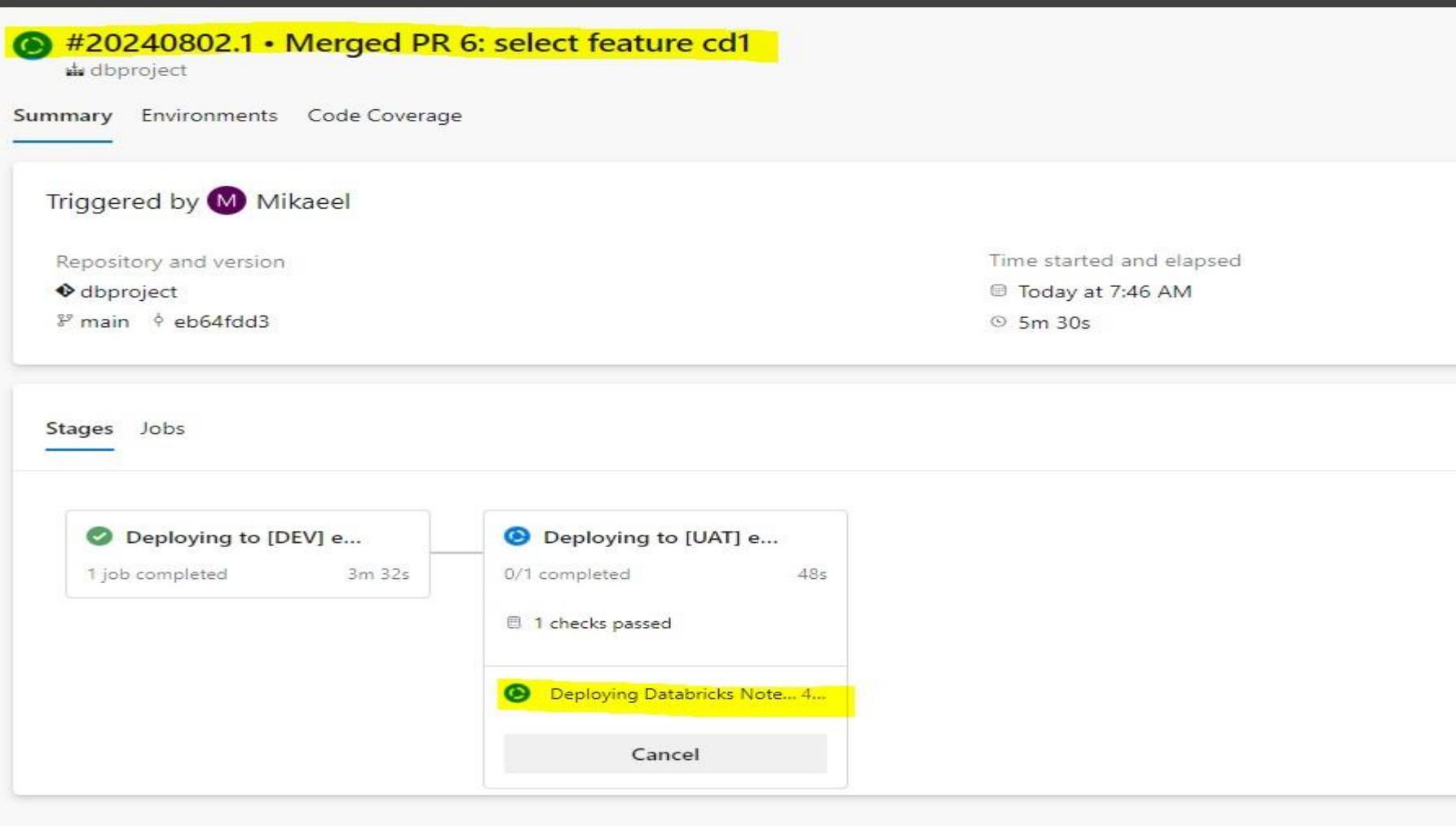
Today at 7:46 AM  
5m 30s

Stages Jobs

Deploying to [DEV] e...  
1 job completed 3m 32s

Deploying to [UAT] e...  
0/1 completed 48s  
1 checks passed

Deploying Databricks Note... 4...  
Cancel



Green tick sign shows, its succeeded:

Azure DevOps iamsaradatascientist / databricks-traffic / Pipelines / dbproject / 20240802.1

databricks-traffic

Overview Boards Repos Pipelines Pipelines Environments Library Test Plans Artifacts

#20240802.1 • Merged PR 6: select feature cd1

dbproject

This run is being retained as one of 3 recent runs by main (Branch).

Summary Environments Code Coverage

Triggered by M Mikael

Repository and version

dbproject  
main eb64fdd3

Time started and elapsed

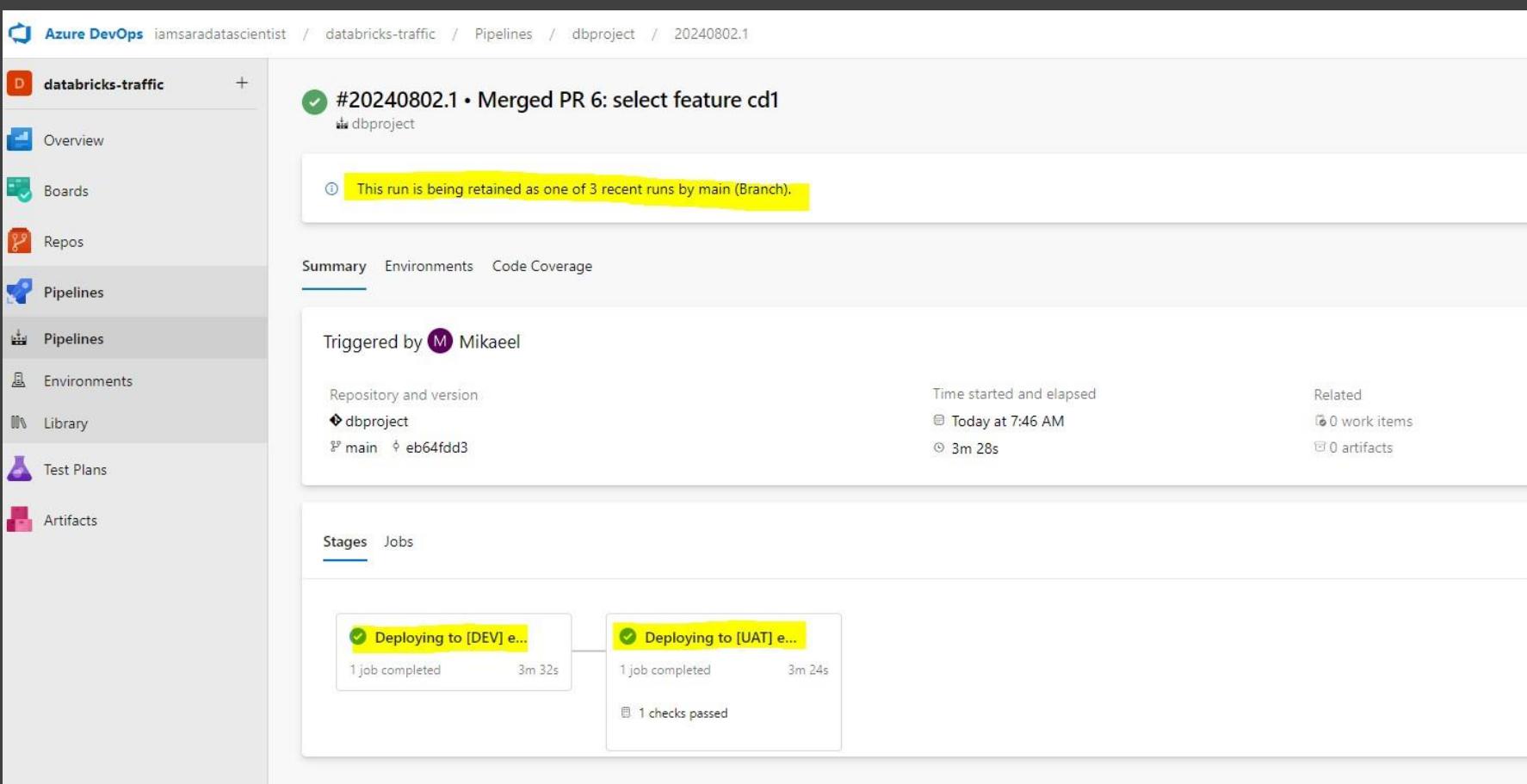
Today at 7:46 AM  
3m 28s

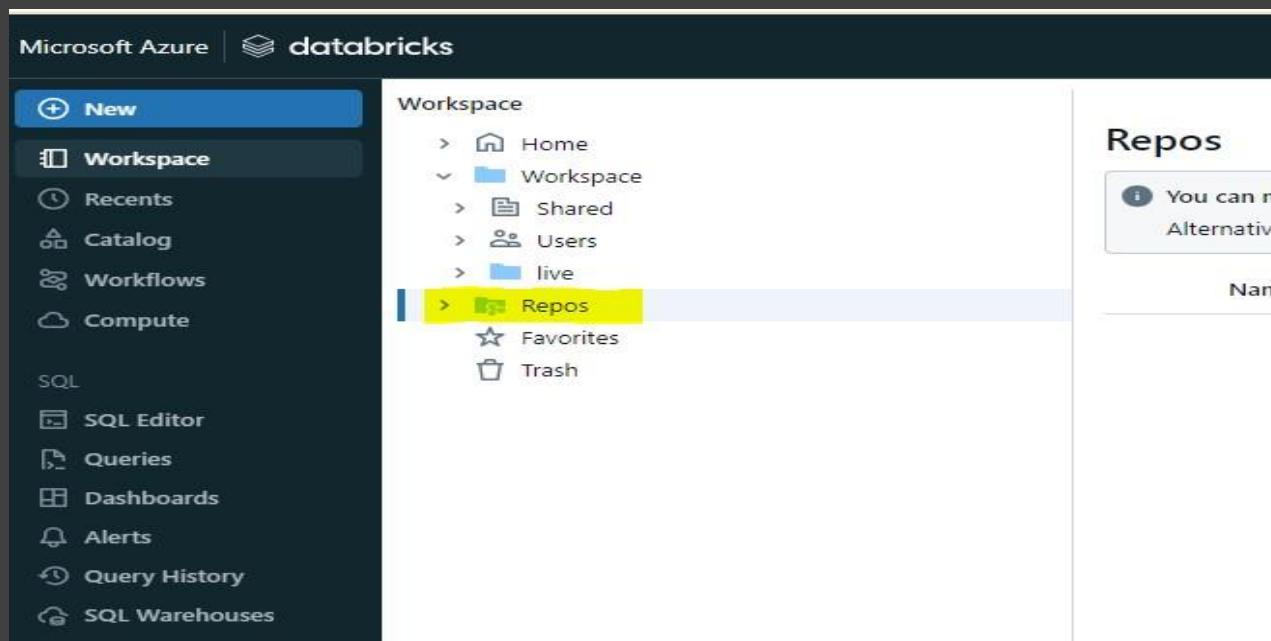
Related 0 work items  
0 artifacts

Stages Jobs

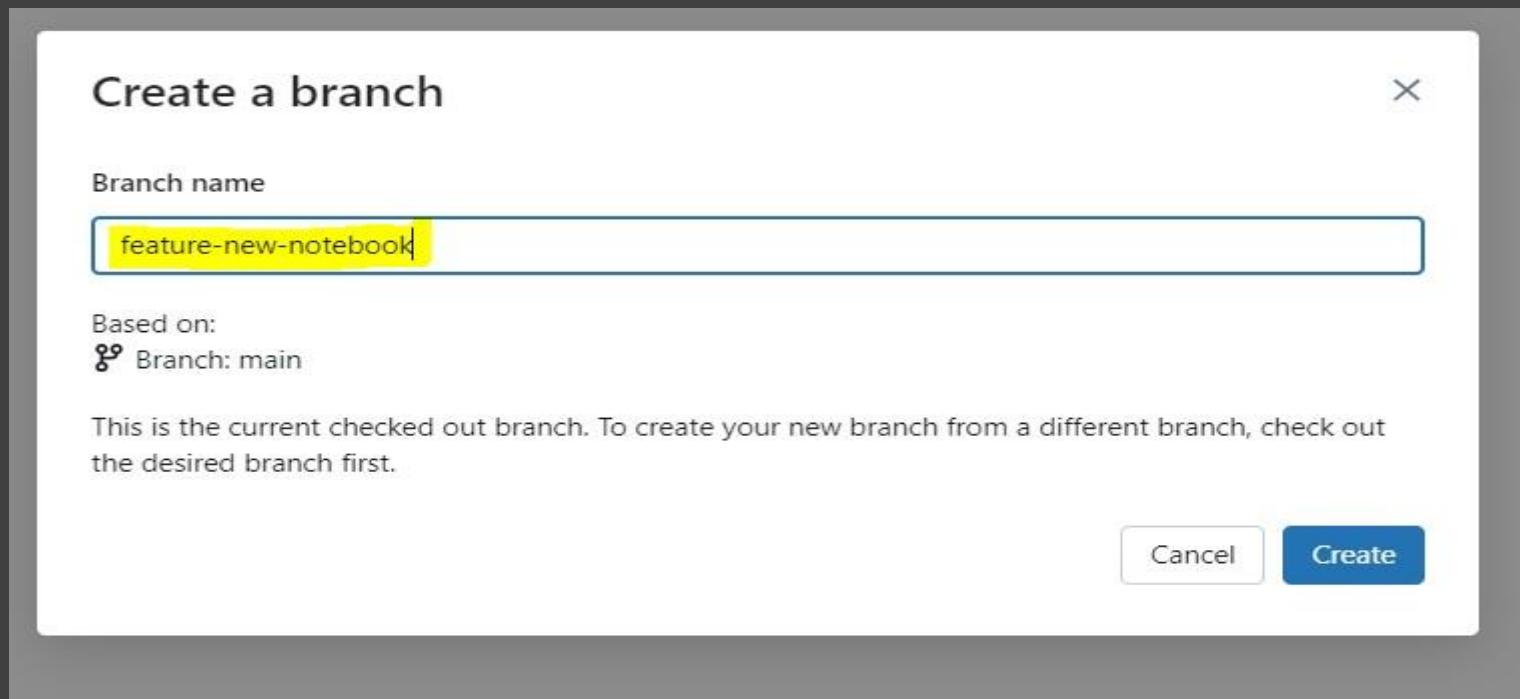
Deploying to [DEV] e...  
1 job completed 3m 32s

Deploying to [UAT] e...  
1 job completed 3m 24s  
1 checks passed

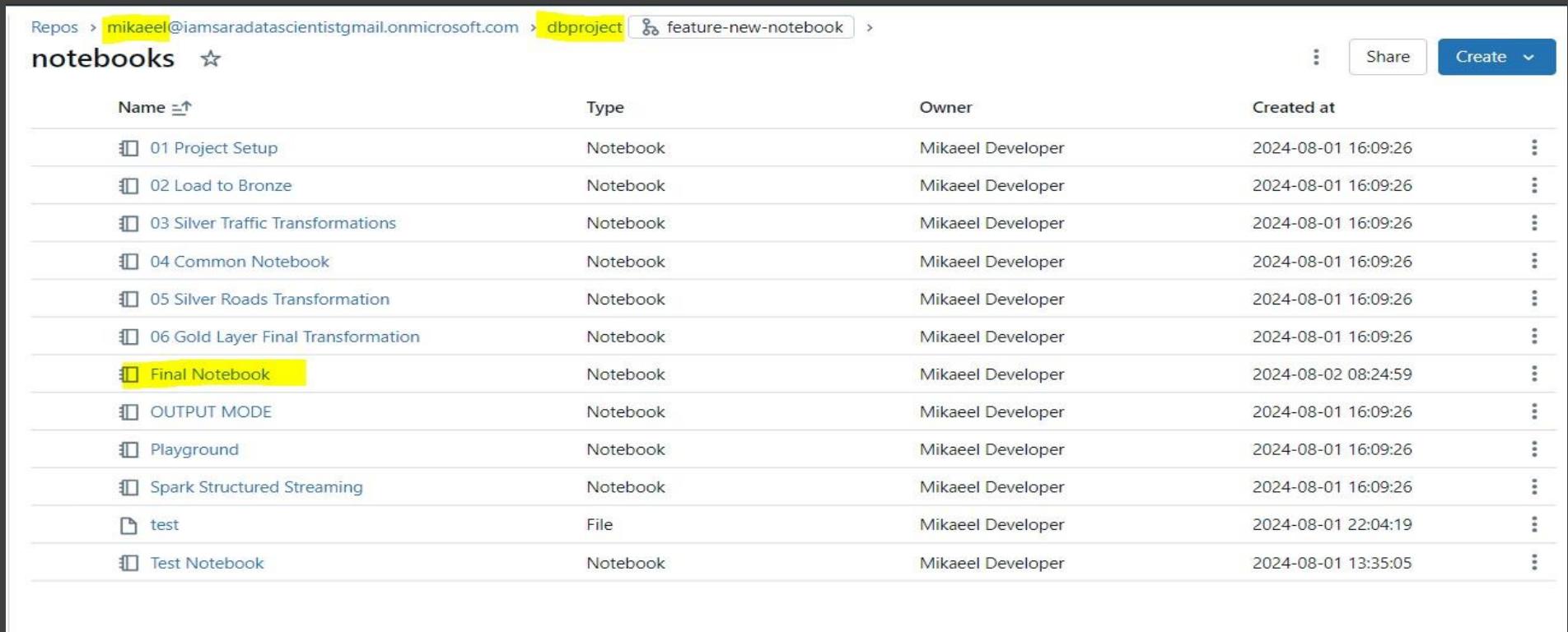




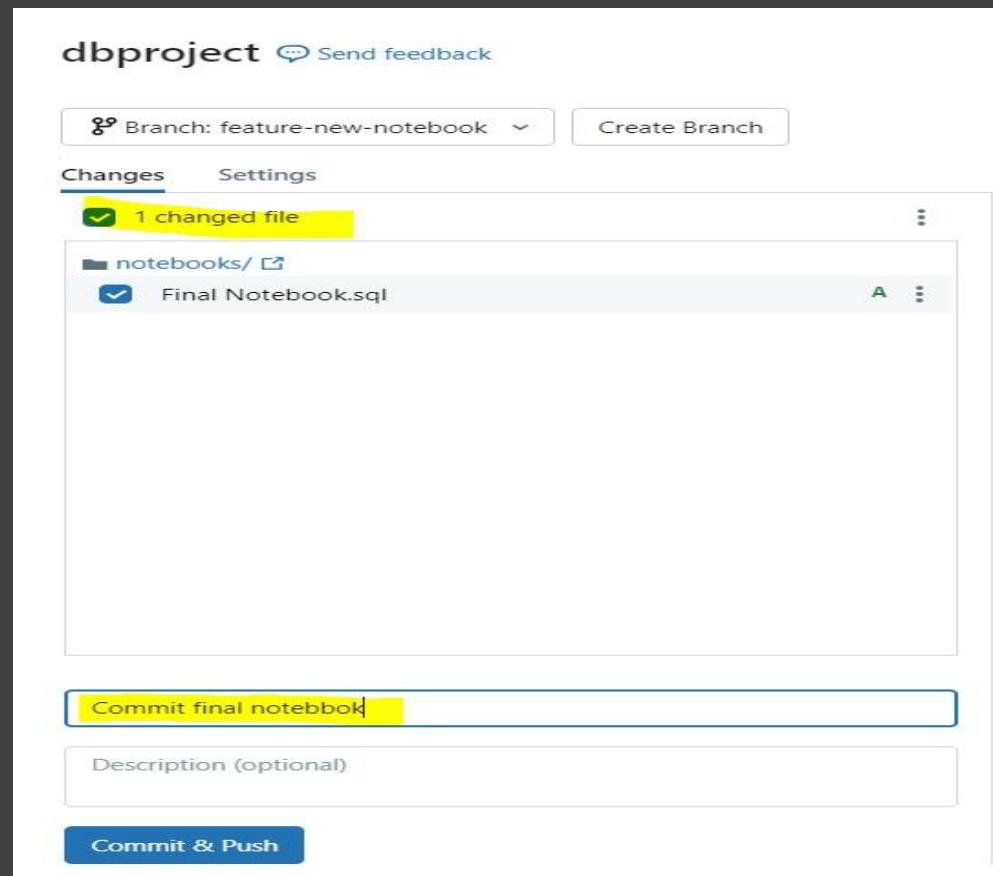
**Now create another feature notebook for final results:**

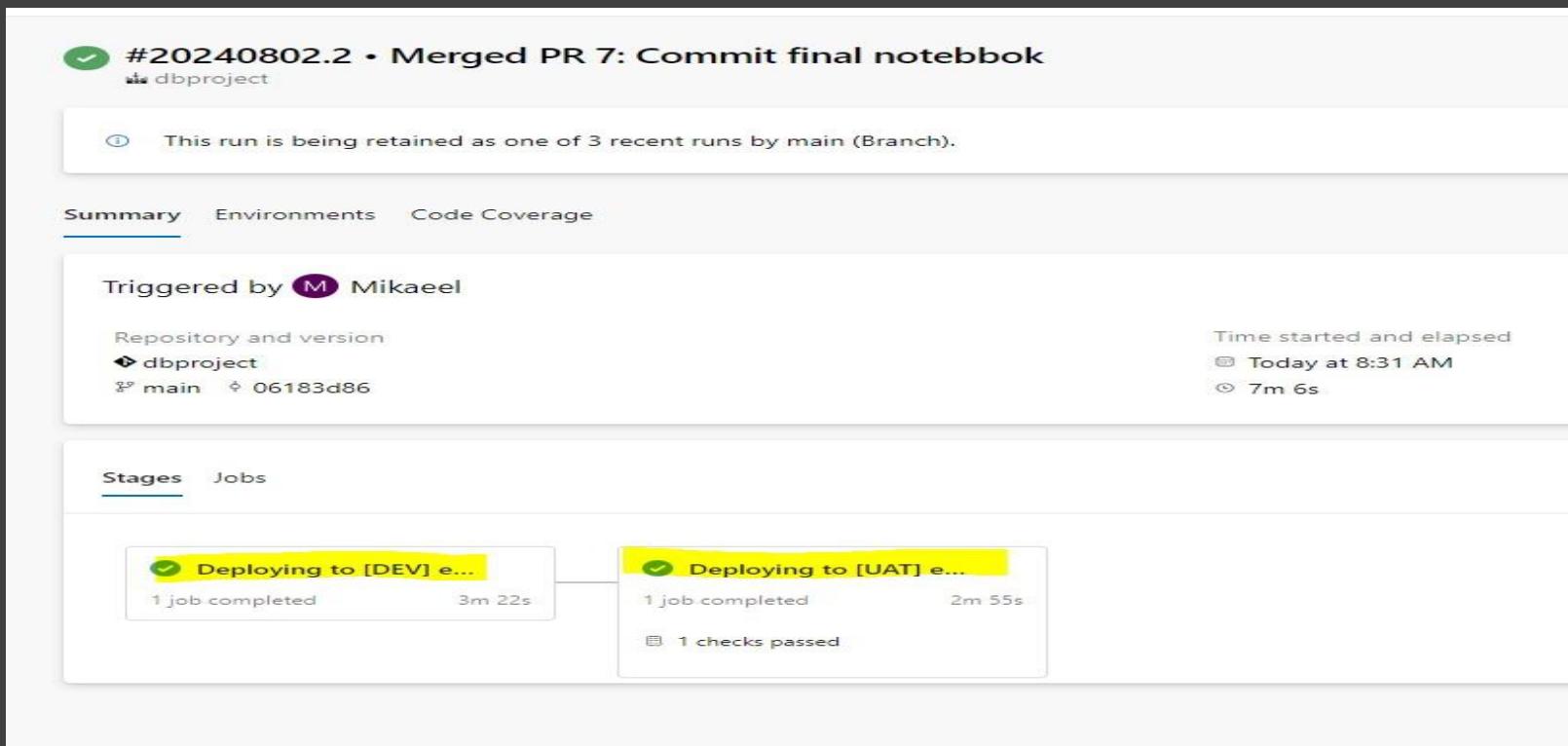


## Create a final notebook inside notebooks folder:



Name	Type	Owner	Created at
01 Project Setup	Notebook	Mikaeel Developer	2024-08-01 16:09:26
02 Load to Bronze	Notebook	Mikaeel Developer	2024-08-01 16:09:26
03 Silver Traffic Transformations	Notebook	Mikaeel Developer	2024-08-01 16:09:26
04 Common Notebook	Notebook	Mikaeel Developer	2024-08-01 16:09:26
05 Silver Roads Transformation	Notebook	Mikaeel Developer	2024-08-01 16:09:26
06 Gold Layer Final Transformation	Notebook	Mikaeel Developer	2024-08-01 16:09:26
Final Notebook	Notebook	Mikaeel Developer	2024-08-02 08:24:59
OUTPUT MODE	Notebook	Mikaeel Developer	2024-08-01 16:09:26
Playground	Notebook	Mikaeel Developer	2024-08-01 16:09:26
Spark Structured Streaming	Notebook	Mikaeel Developer	2024-08-01 16:09:26
test	File	Mikaeel Developer	2024-08-01 22:04:19
Test Notebook	Notebook	Mikaeel Developer	2024-08-01 13:35:05





The screenshot shows the Microsoft Azure Databricks Catalog Explorer. A new catalog named 'uat-catalog' has been created. The catalog details include:

- Comment: (empty)
- Owner: Sara Irfan
- URL: abfss://metastoreroot@databricksuatstg0300.dfs.core.windows.net/meta
- Credential: storagecredential

The Permissions tab is selected, showing the 'Grant' button is active. The table below lists the permissions granted:

Principal	Privilege	Object
		No permissions granted yet.

## DELTA LIVE TABLES:

**Q. What is the current status of Delta Live Tables, and what are its limitations?**

- **Delta Live Tables (DLT) is still evolving and is not yet fully matured. The Databricks team continues to enhance it and add new features to prepare it for production use.**

**Limitations:**

- **Not Fully Mature: The service has some limitations and may not be completely ready for all production environments.**

- **Ongoing Updates:** New features and improvements are being added, which may affect stability and functionality.

## Q. How does Delta Live Tables simplify the ETL process, and what is meant by declarative ETL?

**Simplification of ETL:**

- Delta Live Tables aims to streamline the ETL (Extract, Transform, Load) process by allowing users to define data pipelines declaratively.
- It reduces the need for detailed coding of data processing steps by handling them automatically based on high-level instructions.

**Declarative ETL:**

- Declarative ETL focuses on specifying "what" needs to be done rather than "how" to accomplish it.
- Users define the desired outcomes, and the system takes care of the underlying data processing logic, such as transformations and loading procedures.

## Q. What are the key elements of the Medallion Architecture used with Delta Live Tables?

**Medallion Architecture Layers:**

- **Bronze Layer:**
  - Contains raw, unprocessed data from various sources.
- **Silver Layer:**
  - Data is cleaned and partially transformed to improve its quality.
- **Gold Layer:**
  - Fully cleaned and transformed data that is ready for reporting and production use.

## Q. What prerequisites are needed for using Delta Live Tables effectively in Azure Databricks?

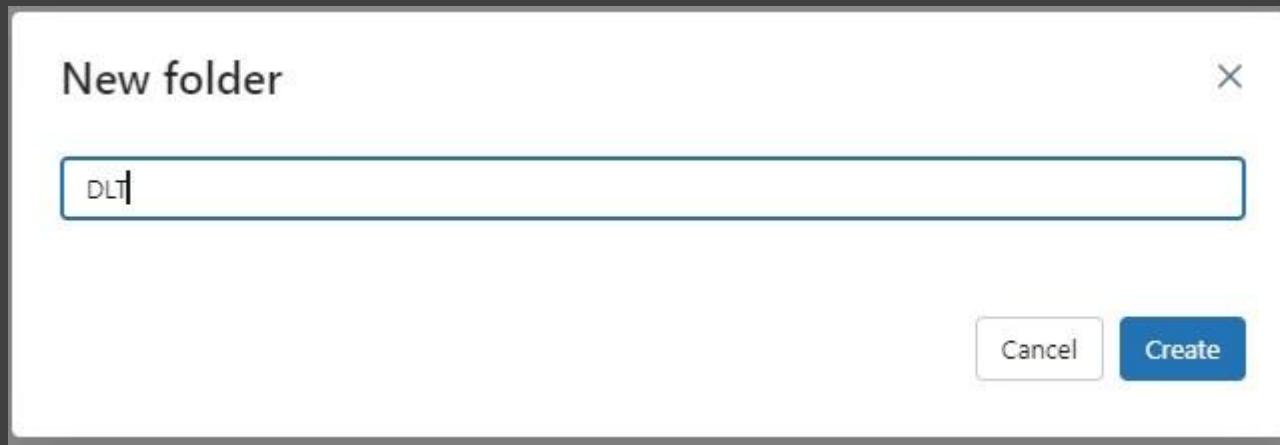
- A premium workspace is required to execute Delta Live Tables, as it supports the necessary features and integrations.
- Delta Live Tables supports Python and SQL. Interaction through magic commands is not supported, and only one language can be used per notebook.
- Delta Live Tables uses declarative definitions only.

# Q. How to create a Delta Live Table pipeline and run it?

## Step-by-Step Answer:

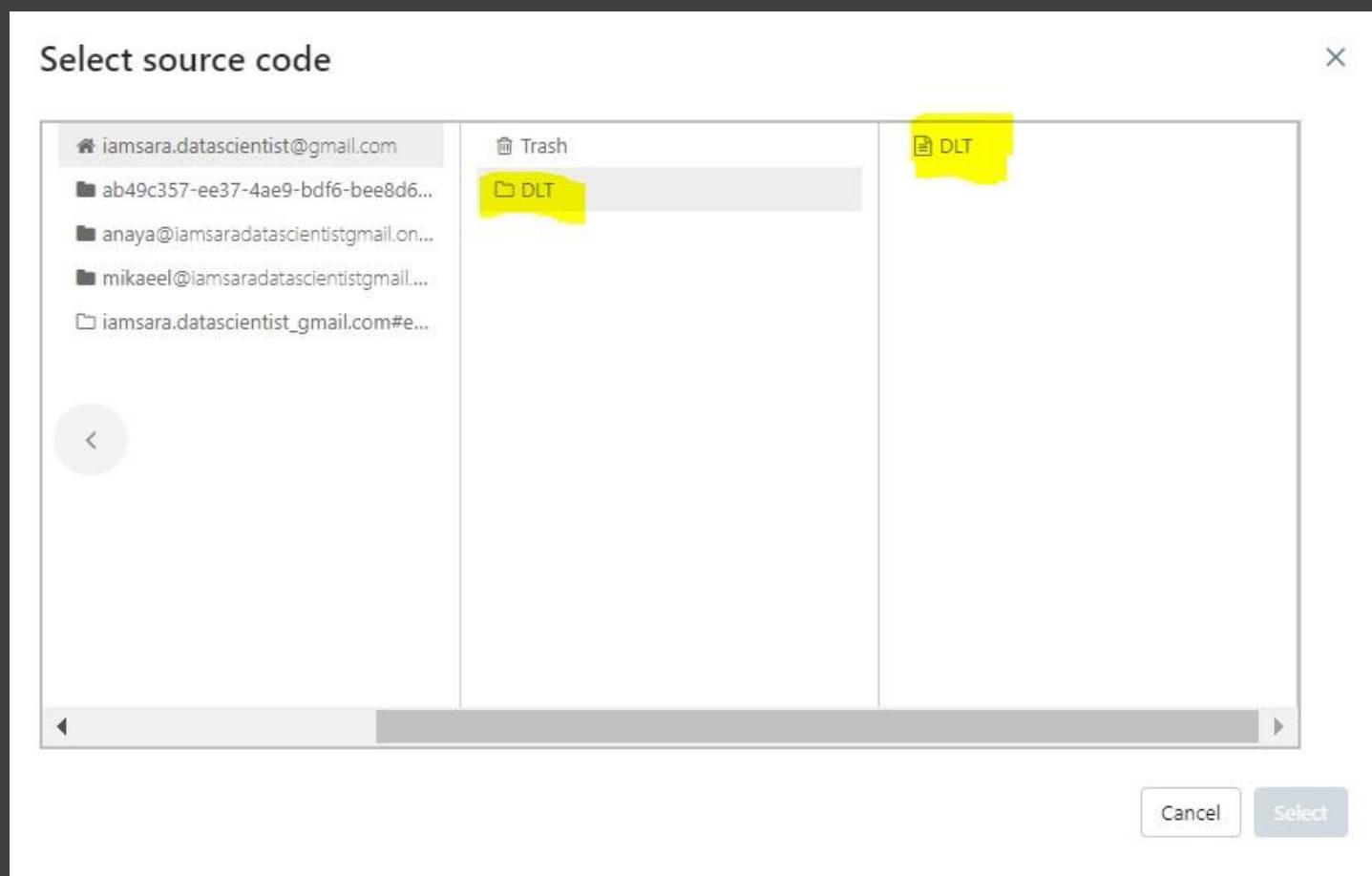
### 1. To create a Delta Live Table pipeline:

- Create a notebook in the workspace:
  - Navigate to the workspace and create a new folder and notebook.
- Write the code in the notebook:
  - Write the code that will create the Delta Live Tables.
  - Use the Autoloader to load the data.
  - Define tables with the @dlt.table decorator.



### 2. To run the pipeline:

- Go to the Delta Live Tables tab in the Workflows section:
  - In the Workflows section, select the Delta Live Tables tab.



- Create the pipeline:
  - Choose the option to create a Delta Live Table pipeline.
  - Name the pipeline and specify the path to the notebook.

- Select the Unity Catalog and schema.

Create pipeline

General

\* Pipeline name  
DLT execution

Product edition  
Advanced

Help me choose ↗

Pipeline mode ⓘ  
 Triggered  Continuous

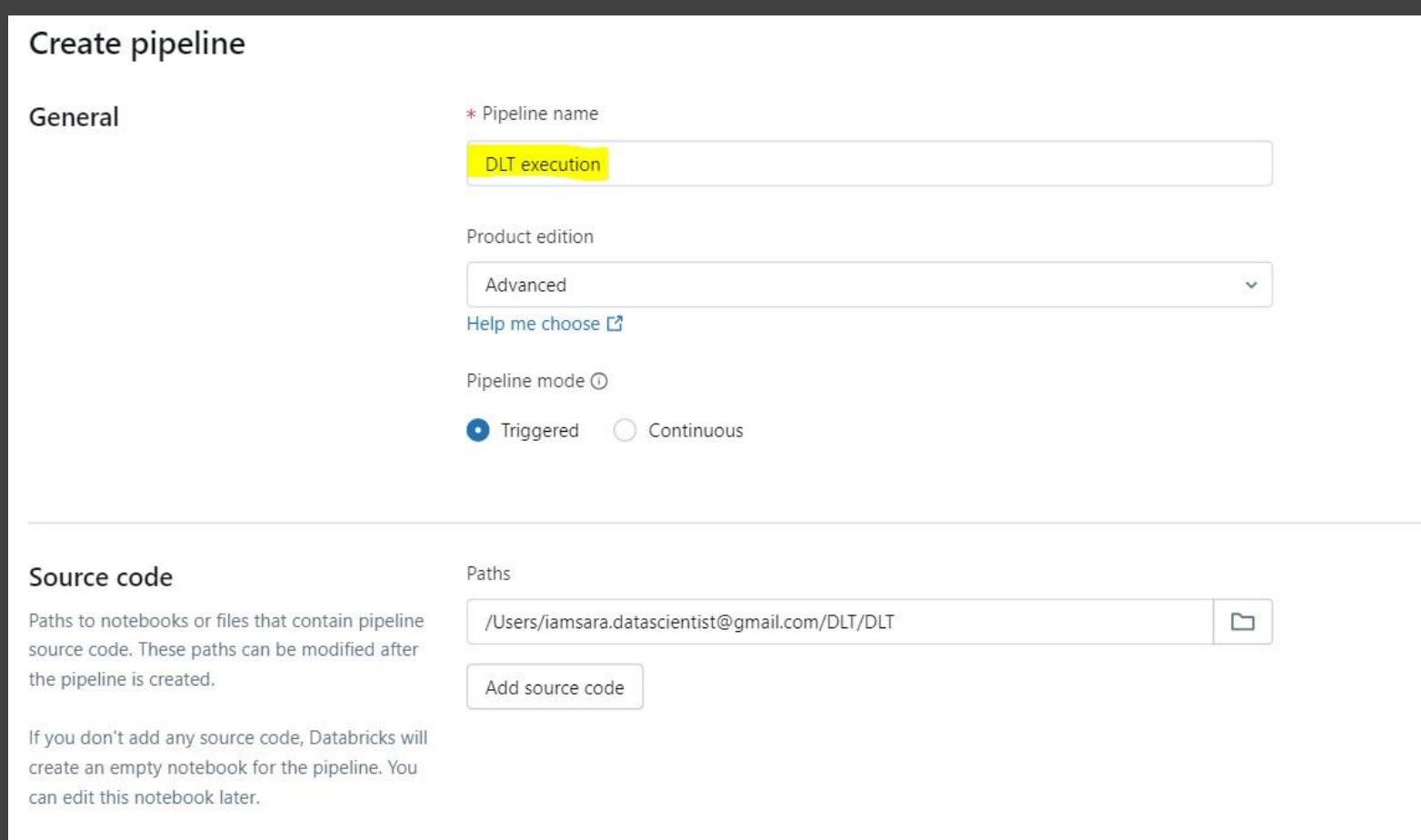
---

Source code

Paths  
/Users/iamsara.datascientist@gmail.com/DLT/ DLT

Add source code

If you don't add any source code, Databricks will create an empty notebook for the pipeline. You can edit this notebook later.



- Configure the compute settings:
  - Specify the cluster node and node type.
  - Use the JSON editor to specify the node type ID.
- Start the pipeline:
  - Click the Start button to run the pipeline.
  - Choose between development or production mode and monitor the execution.

### Selection of destination and compute:

Destination

Storage options  
 Hive Metastore  Unity Catalog Preview

Catalog ⓘ  
devcatalog

Target schema ⓘ  
default

---

Compute

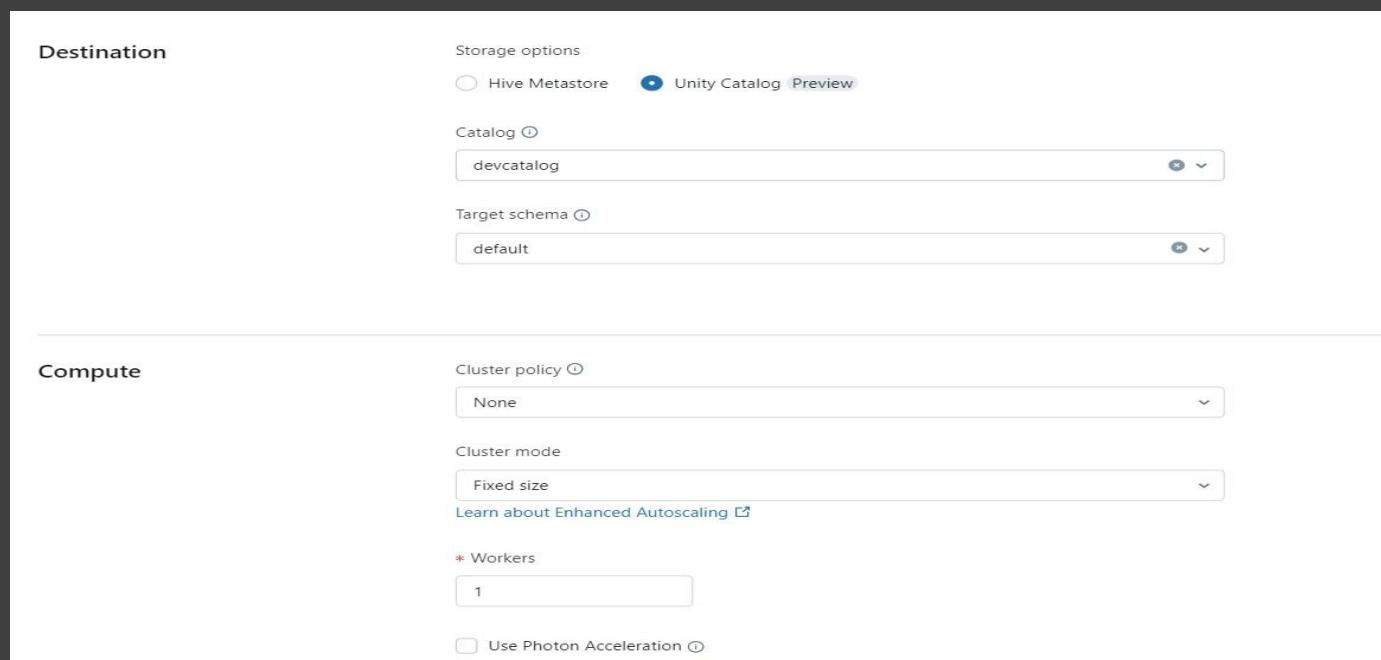
Cluster policy ⓘ  
None

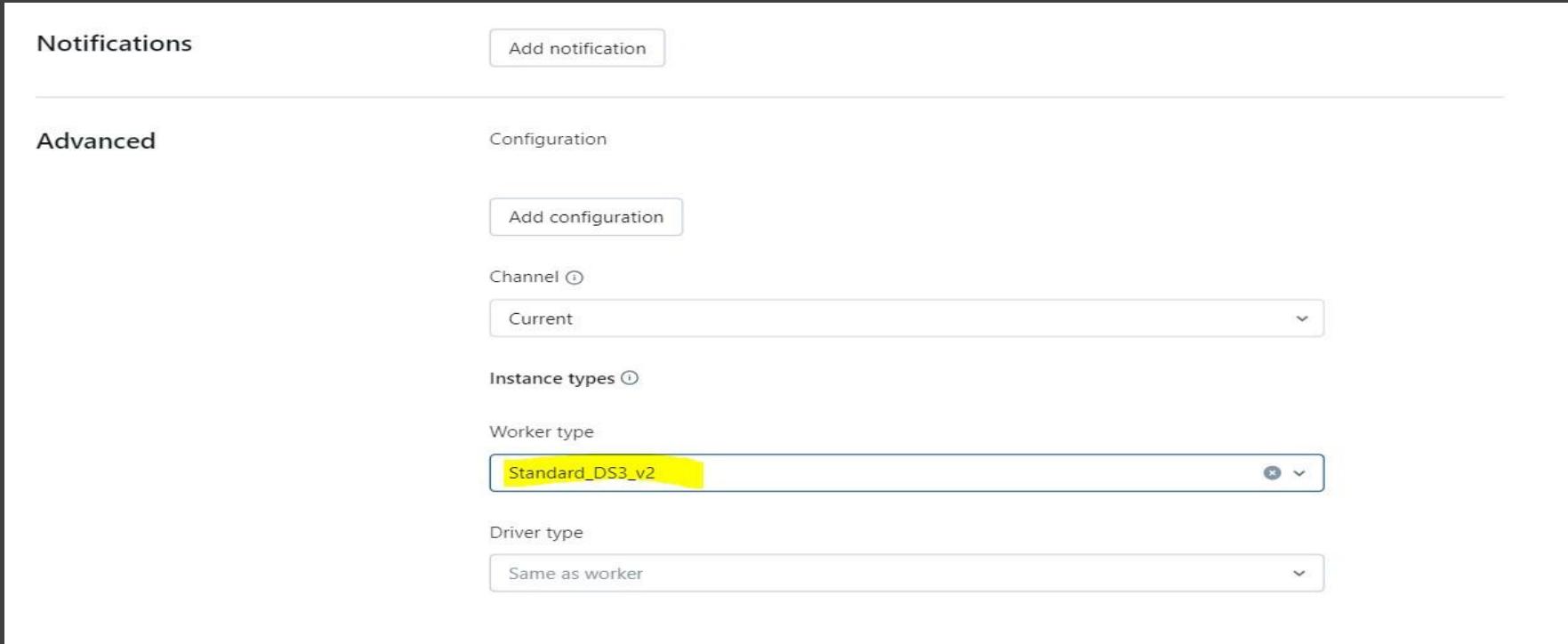
Cluster mode  
Fixed size

Learn about Enhanced Autoscaling ↗

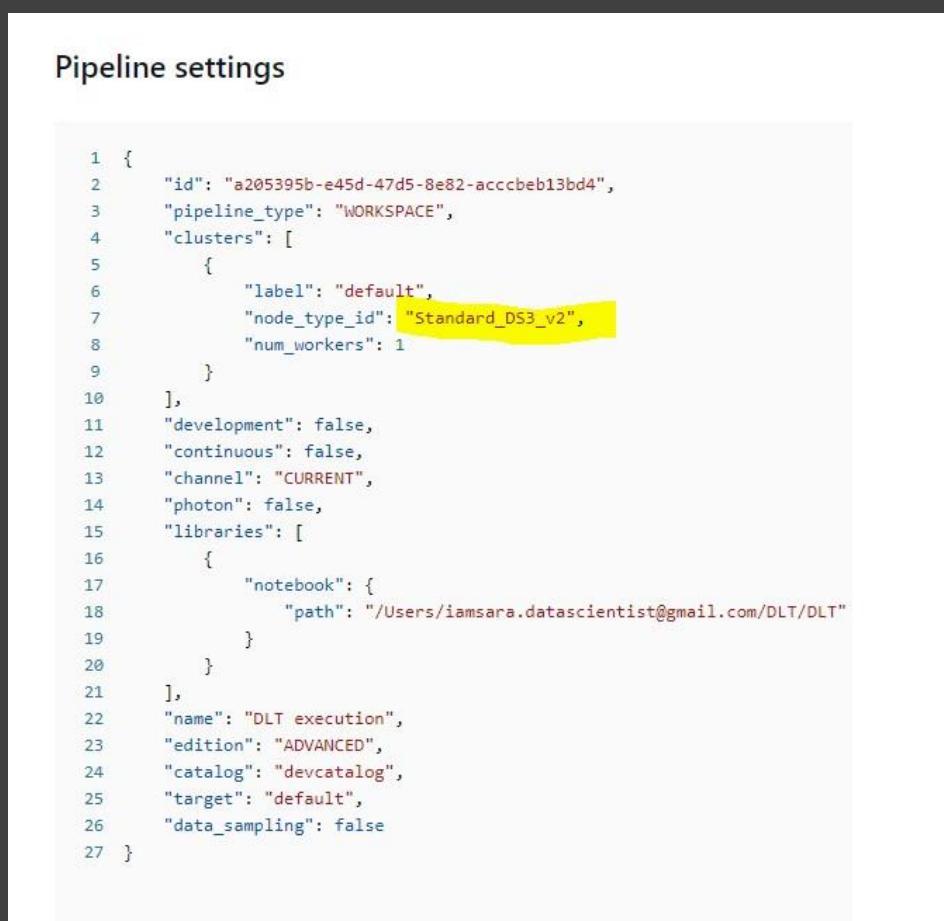
\* Workers  
1

Use Photon Acceleration ⓘ





The screenshot shows the 'Pipeline settings' page in the Databricks UI. At the top, there's a 'Notifications' section with a 'Add notification' button. Below it is an 'Advanced' section with a 'Configuration' tab selected. Under 'Configuration', there are fields for 'Channel' (set to 'Current'), 'Instance types' (set to 'Standard\_DS3\_v2'), 'Worker type' (set to 'Standard\_DS3\_v2'), and 'Driver type' (set to 'Same as worker').



```

1  {
2    "id": "a205395b-e45d-47d5-8e82-acccbeb13bd4",
3    "pipeline_type": "WORKSPACE",
4    "clusters": [
5      {
6        "label": "default",
7        "node_type_id": "Standard_DS3_v2", // Worker type
8        "num_workers": 1
9      }
10 ],
11 "development": false,
12 "continuous": false,
13 "channel": "CURRENT",
14 "photon": false,
15 "libraries": [
16   {
17     "notebook": {
18       "path": "/Users/iamsara.datascientist@gmail.com/DLT/DLT"
19     }
20   }
21 ],
22 "name": "DLT execution",
23 "edition": "ADVANCED",
24 "catalog": "devcatalog",
25 "target": "default",
26 "data_sampling": false
27 }

```

# Q. How to implement data quality checks in a Delta Live Table pipeline?

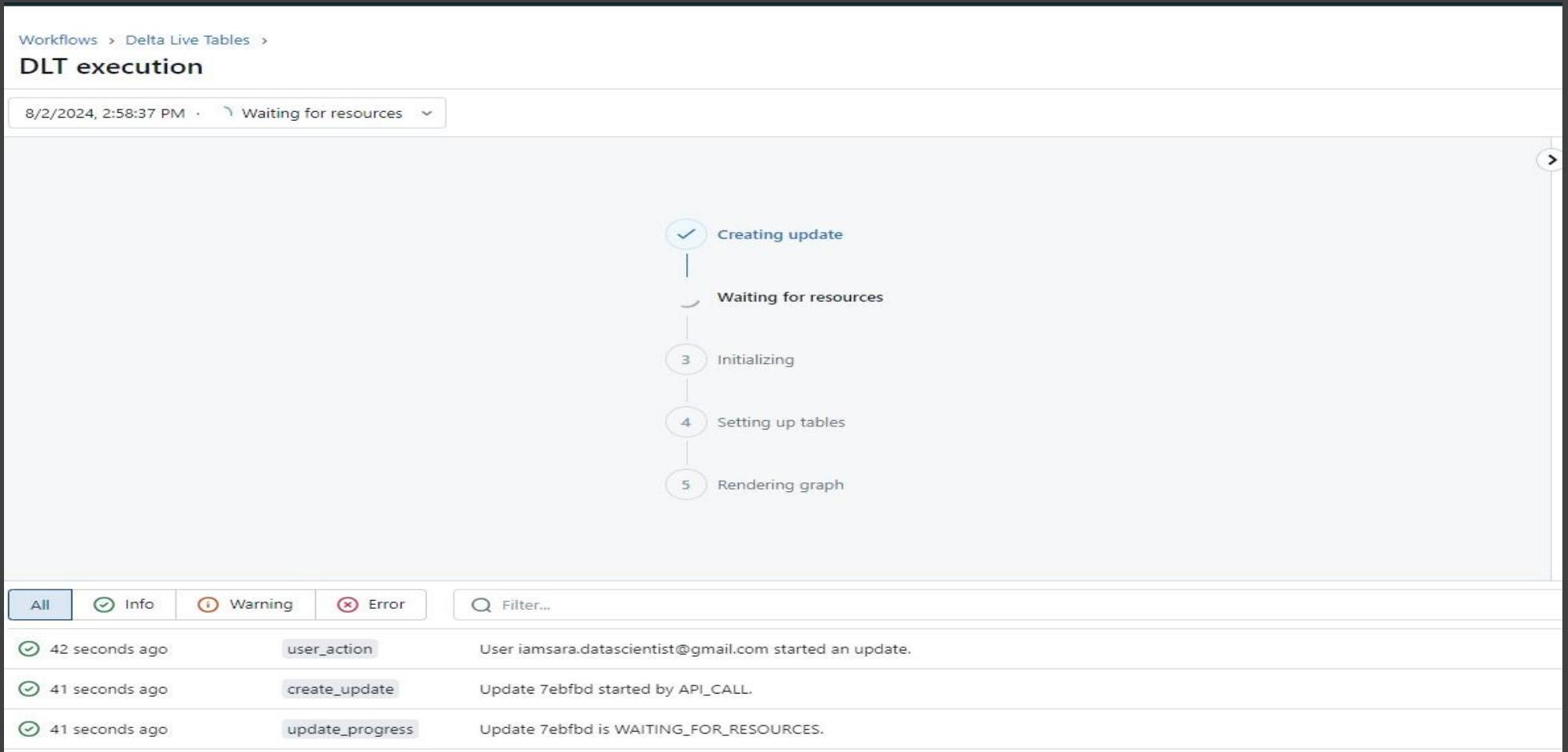
## Step-by-Step Answer:

### 1. To define data quality checks:

- Use the `@dlt.expect` decorator to define data quality rules.
- For example, `@dlt.expect("valid_record", "year IS NOT NULL")` will check that the year column does not have any null values.
  - Specify violation actions:
- Define `onViolation="drop"` to drop rows if the data quality rule is violated.
- Define `onViolation="fail_update"` to fail the pipeline if there is a violation.

### 2. To run data quality checks in the pipeline:

- After defining data quality checks in the notebook, run the pipeline.
- Monitor the pipeline execution and check the data quality metrics.
- Check the status of violations and dropped rows.

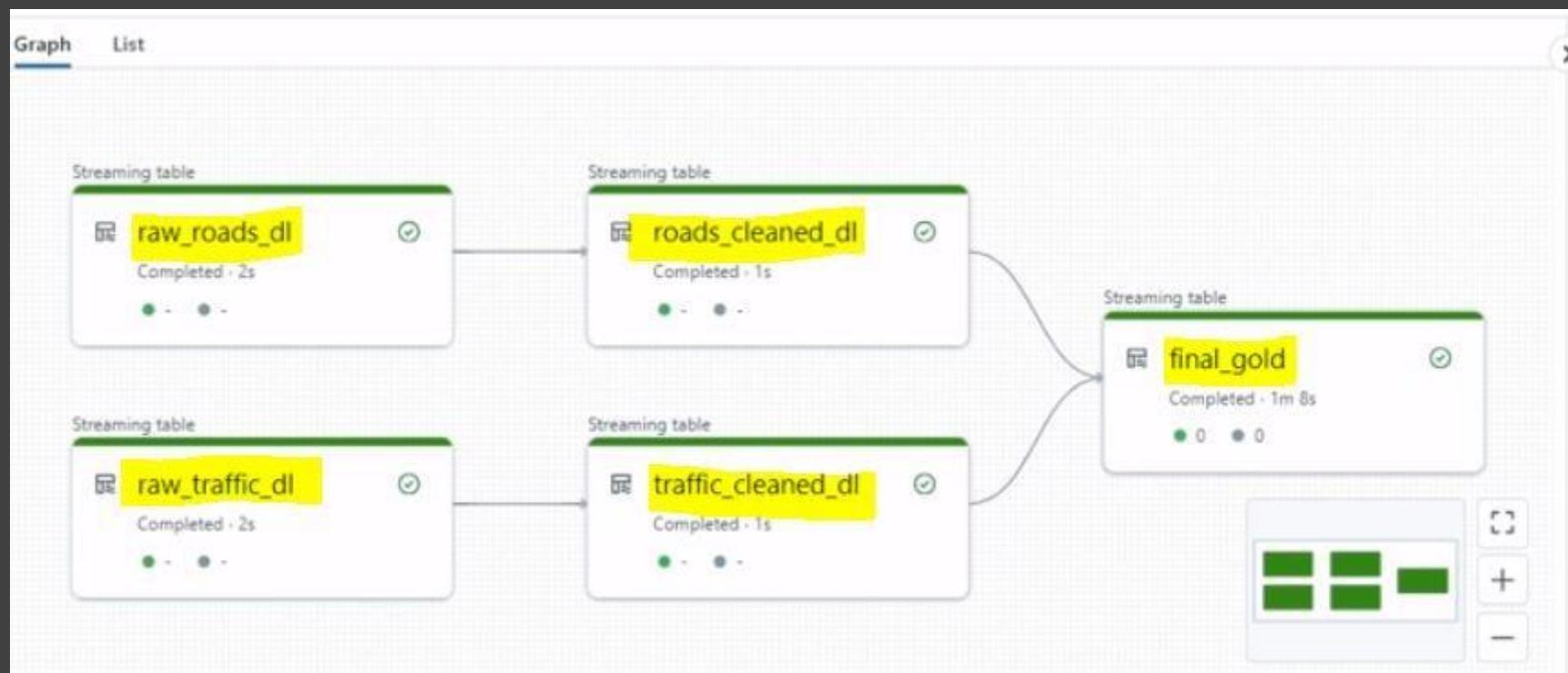


# Q. How to create and interpret a dependency graph in a Delta Live Table pipeline?

## Step-by-Step Answer:

1. To create a dependency graph:
  - Write the code for each table that performs transformations and creates new tables.
  - For example, define multiple tables with the `@dlt.table` decorator that depend on each other.
  - After writing the notebook code, run the pipeline.
  - Delta Live Tables automatically creates a dependency graph.
2. To interpret the dependency graph:
  - Monitor the pipeline execution:
  - In the pipeline UI, view the dependency graph.
  - Click on each table to check its source and destination dependencies.
  - Understand the data flow and dependencies:
  - Look at the arrows and connections in the graph that show data flow and dependencies.
  - For example, see the dependency between the `raw_traffic_dl` table and the `traffic_clean_dl` table.

# Live Delta Pipeline



## Resources used in my project:

Home > Recent

Manage view Refresh Export to CSV Clear Assign tags

Filter for any field... Subscription equals all Resource Group equals all Type equals all Location equals all Add filter

	Name ↑	Type	Location	Resource Group	Subscription	Last accessed
<input type="checkbox"/>	Azure subscription 1	Subscription			Azure subscription 1	42 minutes ago
<input type="checkbox"/>	databricksuatstg0300	Storage account		databricks-uat-rg	Azure subscription 1	4 hours ago
<input type="checkbox"/>	databricksdevstorage0300	Storage account		databricksresource	Azure subscription 1	4 hours ago
<input type="checkbox"/>	databricksdevws	Azure Databricks Service	East US	databricksresource	Azure subscription 1	4 hours ago
<input type="checkbox"/>	databricks-uat-rg	Resource group	East US	databricks-uat-rg	Azure subscription 1	9 hours ago
<input type="checkbox"/>	databricksresource	Resource group	East US	databricksresource	Azure subscription 1	18 hours ago
<input type="checkbox"/>	dbaccessconnector	Access Connector for Azure Dat...		databricksresource	Azure subscription 1	2 days ago
<input type="checkbox"/>	databricks-uat-ws	Azure Databricks Service	East US	databricks-uat-rg	Azure subscription 1	2 days ago
<input type="checkbox"/>	deltadbstg0300	Storage account	East US	databricksresource	Azure subscription 1	5 days ago
<input type="checkbox"/>	deltadbwsp	Azure Databricks Service	East US	databricksresource	Azure subscription 1	8 days ago

Microsoft Azure | databricks

Search data, notebooks, recents, and more... CTRL + P

Workspace

- New
- Workspace
- Recents
- Catalog
- Workflows
- Compute
- Data Engineering
- Job Runs
- Machine Learning

Workspace > Users > iamsara.datascientist@gmail.com ☆

Name ↑	Type	Owner

This folder is empty

✓ delta dbwsp  
eastus  
  
databricks-uat-ws  
eastus  
  
databricksdevws  
eastus  
  
databrickswspace  
eastus

Manage account

The screenshot shows the Databricks interface with the Catalog tab selected in the sidebar. The main area displays a list of catalogs under the heading "Catalog". A search bar at the top right allows users to "Search data, notebooks, recents, and more..." and includes keyboard shortcuts like "CTRL + P". Below the search bar are three buttons: "Delta Sharing", "External Data", and "Add data".

The catalog list includes:

- Serverless Starter Warehouse
- Serverless
- databricksdevws
- devcatalog
- hive\_metastore
- samples
- system
- uatcatalog

A "Quick access" sidebar on the right lists recent items, favorites, and catalogs. The "Catalogs" tab is currently selected. The list of catalogs in the sidebar matches the ones listed in the main catalog view.

**FEEL FREE TO CONTACT ME  
SARA IRFAN**

**MICROSOFT AZURE DATA ENGINEER CERTIFIED  
MICROSOFT POWER BI-DATA ANALYST CERTIFIED  
MICROSOFT FABRIC ANALYTIC CERTIFIED  
EMAIL: sara.data.insights@gmail.com**

