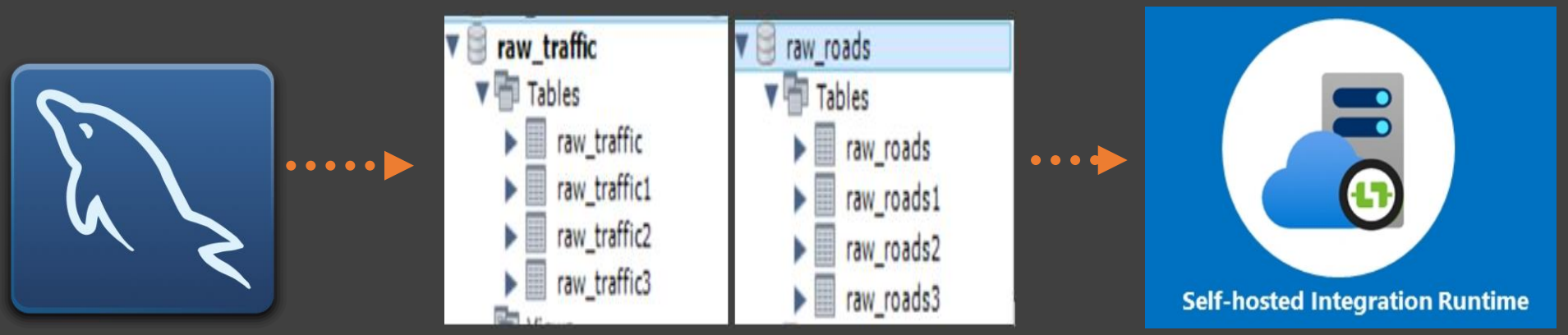
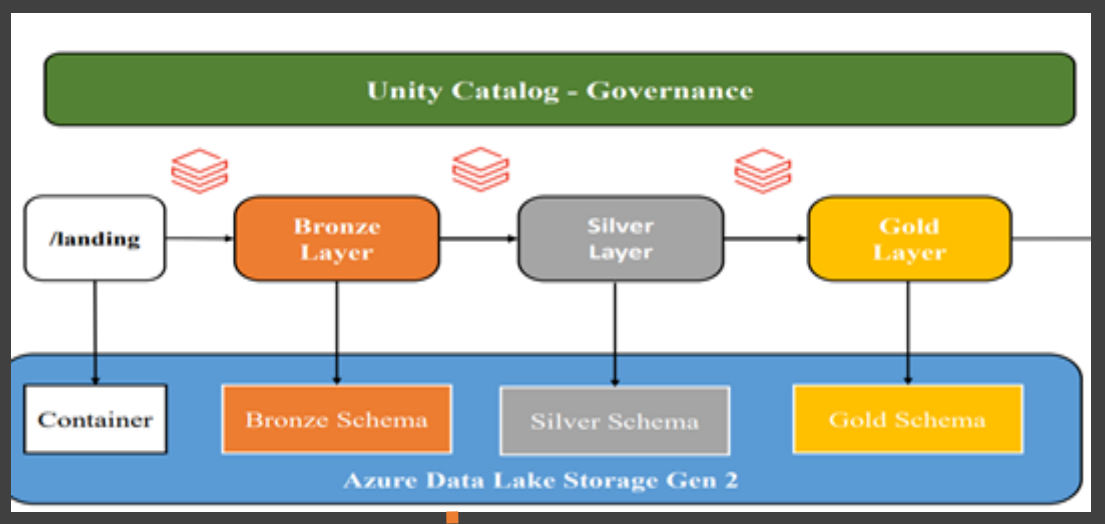
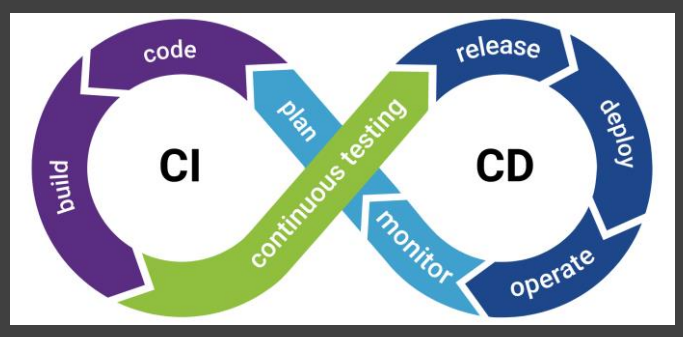
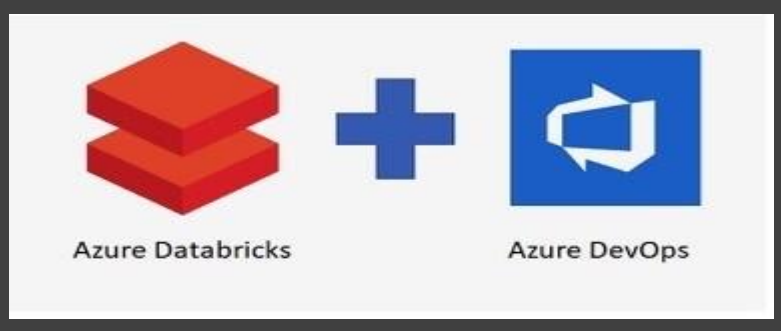
Sara Irfan (sara.data.insights@gmail.com)

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







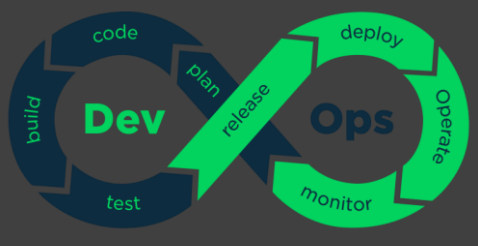


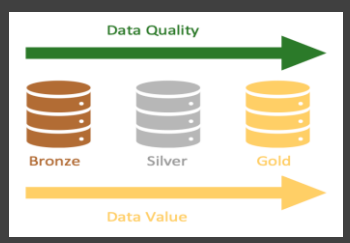
Sara Irfan (sara.data.insights@gmail.com)

OUTLINE:

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.

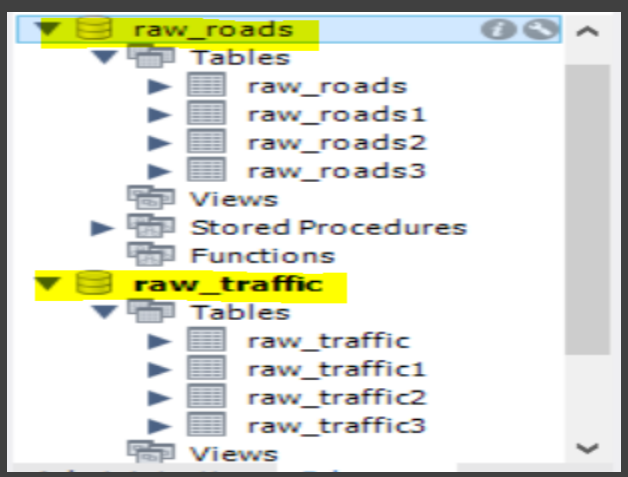
Proposal

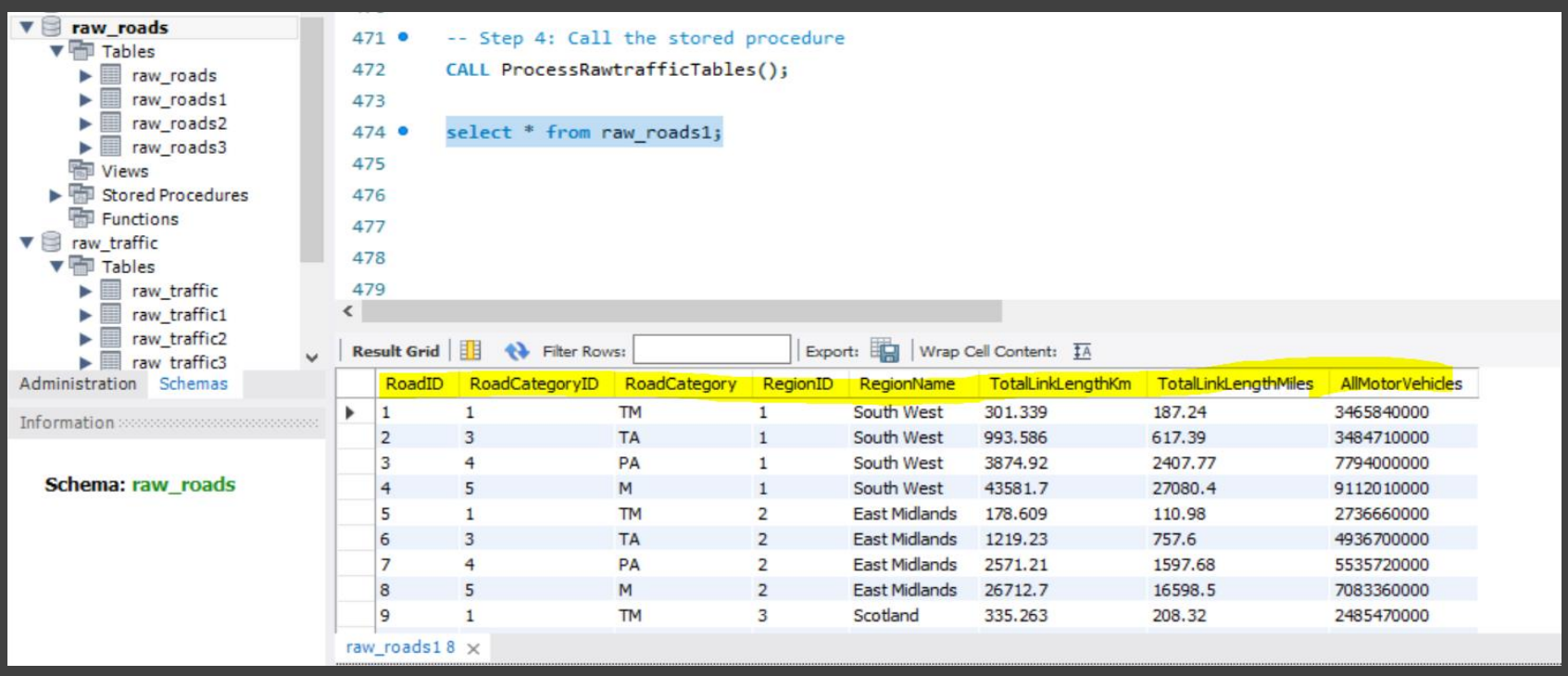
In this proposal, I designed a solution using Databricks live Delta tables and Spark Structured Streaming. I processed continuously streaming data from IoT devices and sensors and stored this data in Delta tables. This data was then analyzed through transformations and aggregations. I used multiple sources 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.

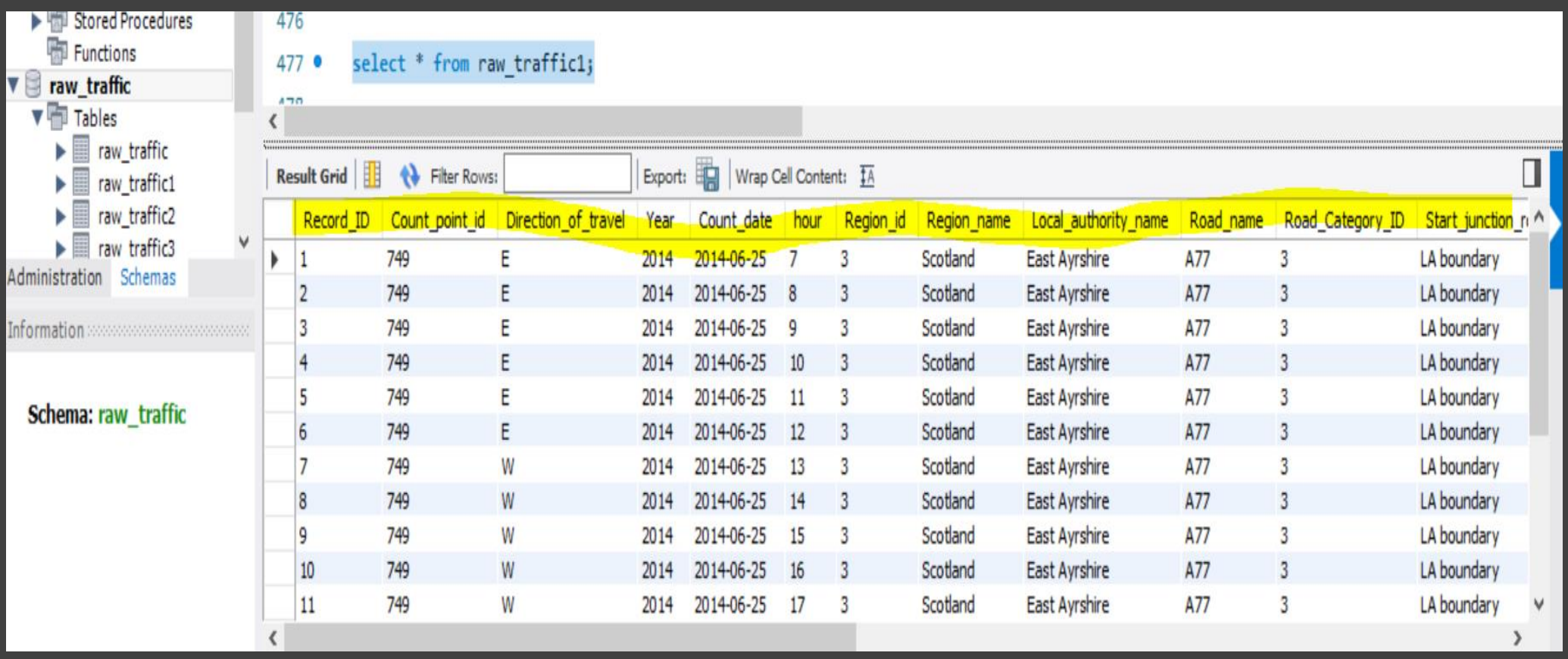
Sara Irfan (sara.data.insights@gmail.com)

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).**







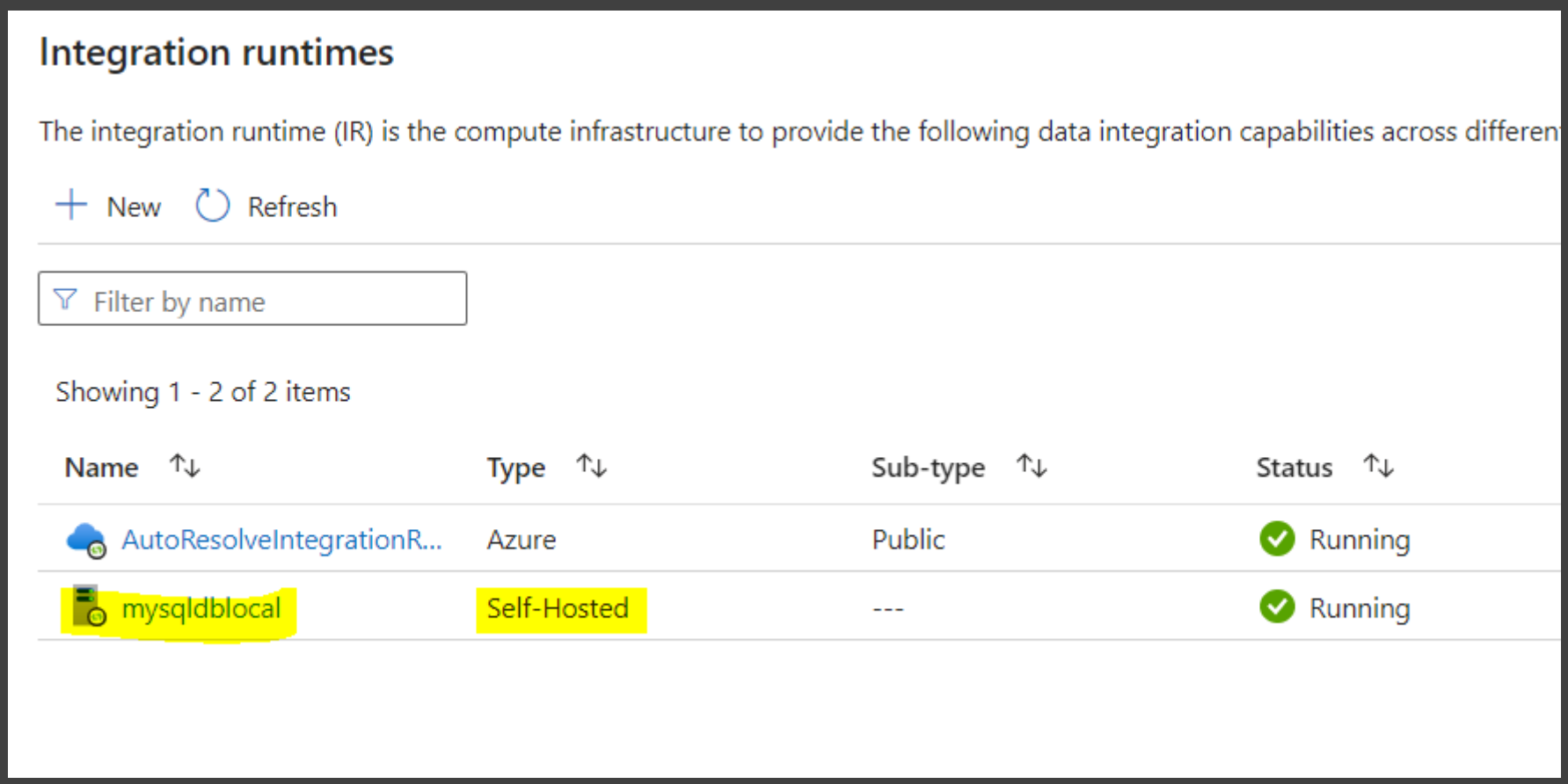
2. Set Up Control Table:

- **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;
```

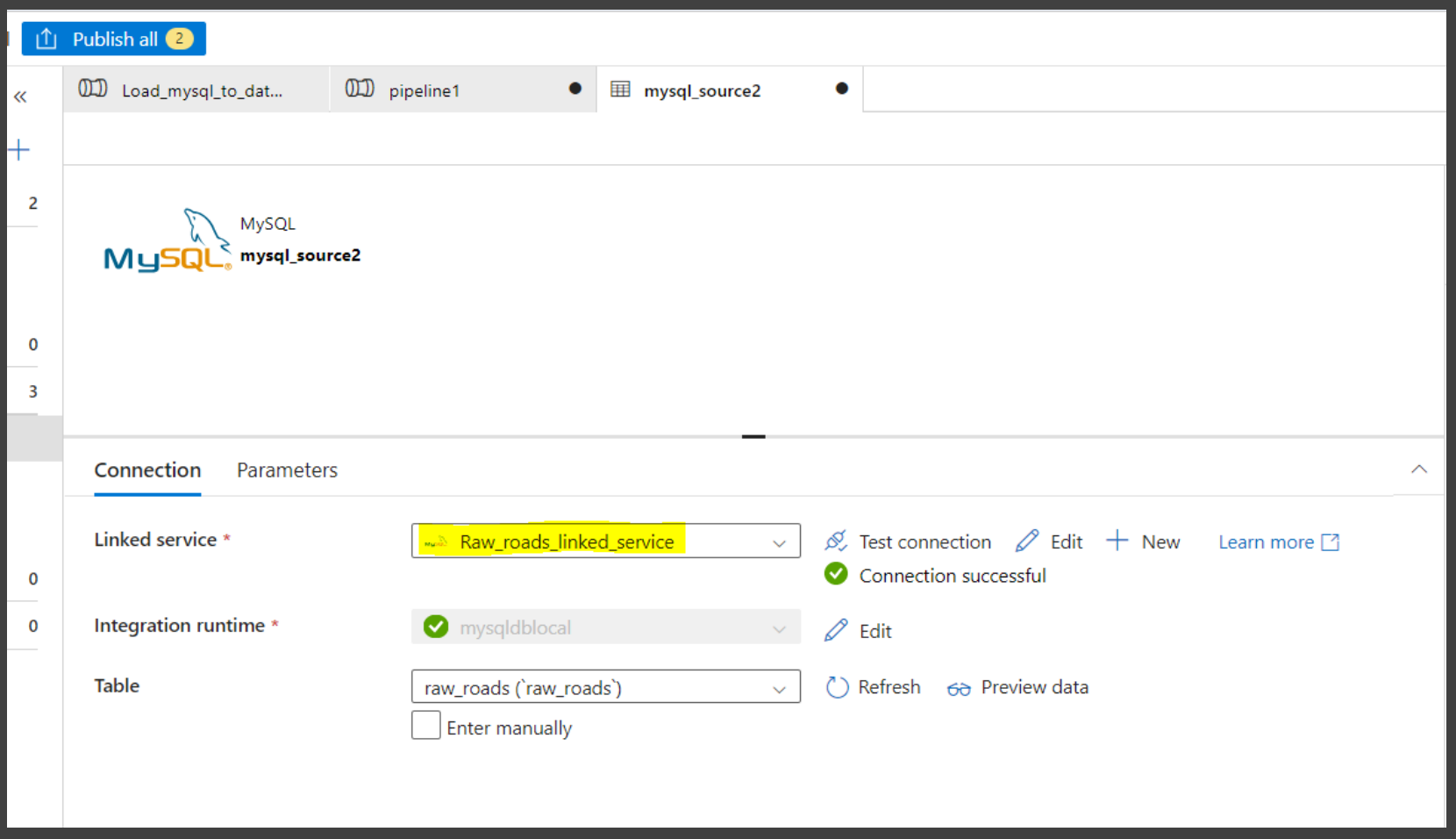
3. Configured Azure Data Factory:

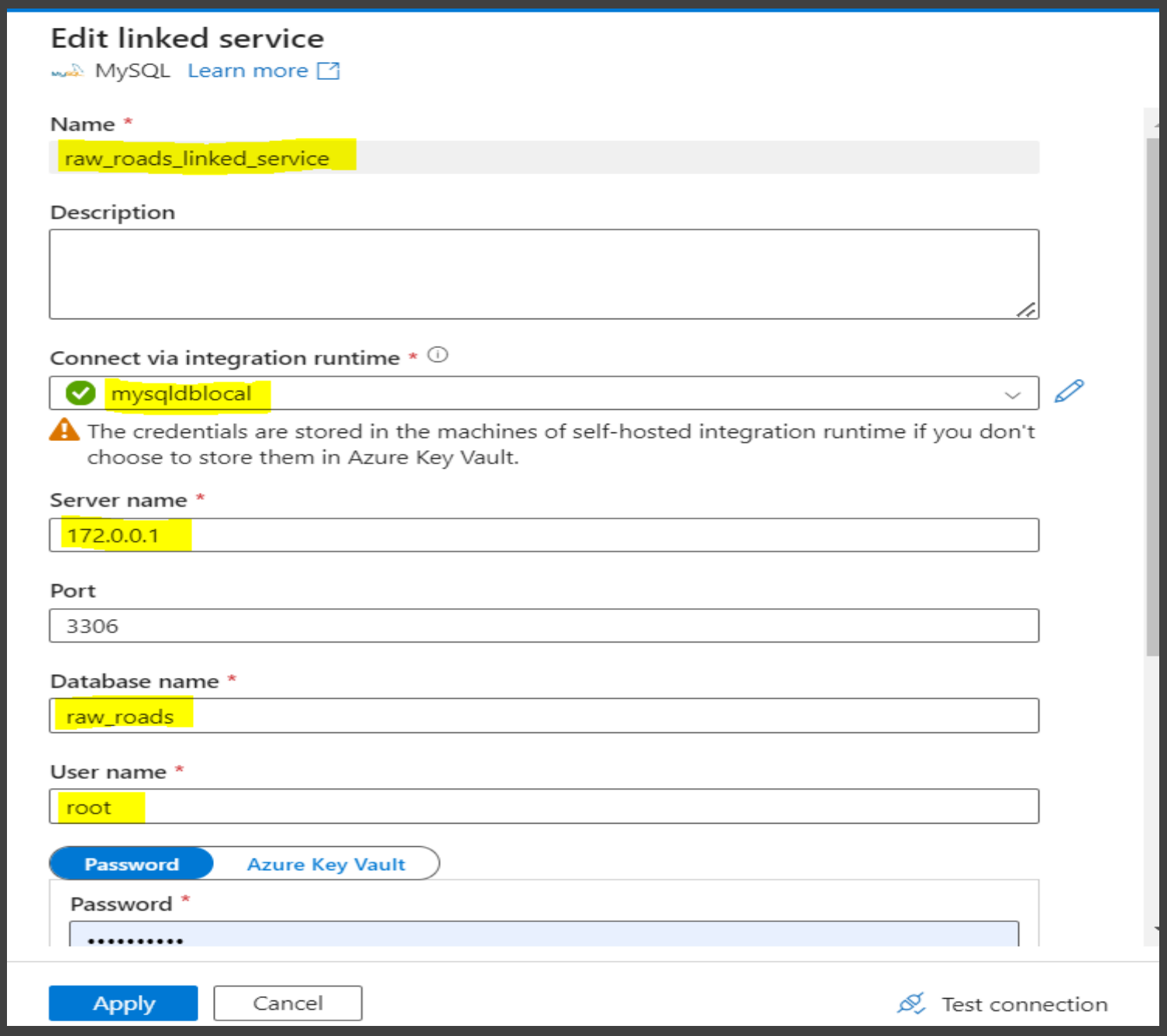
- **To connect to the data, I needed a self-hosted integration runtime and set up a linked service in Azure Data Factory.**



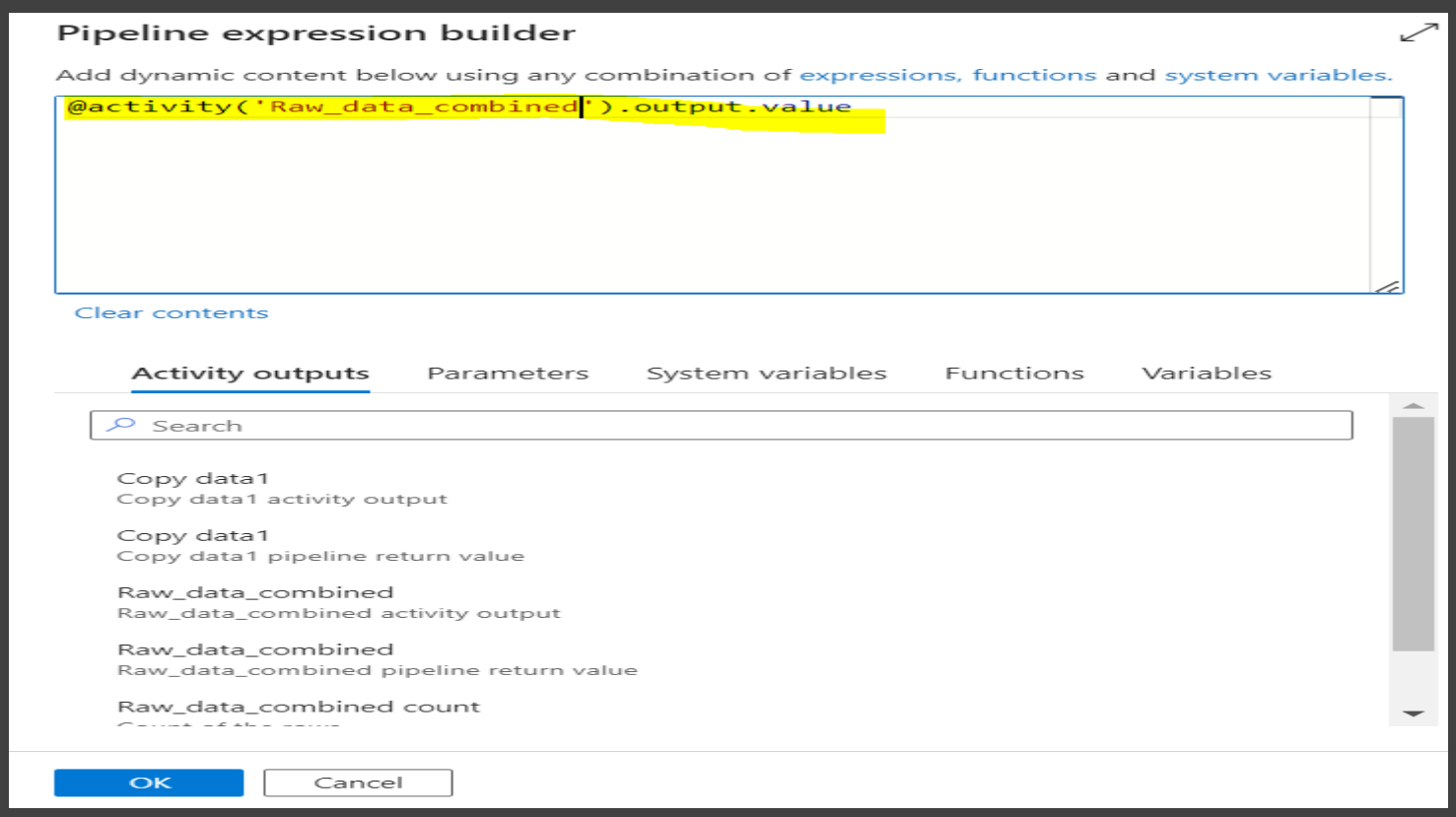
4. Verified Connections:

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

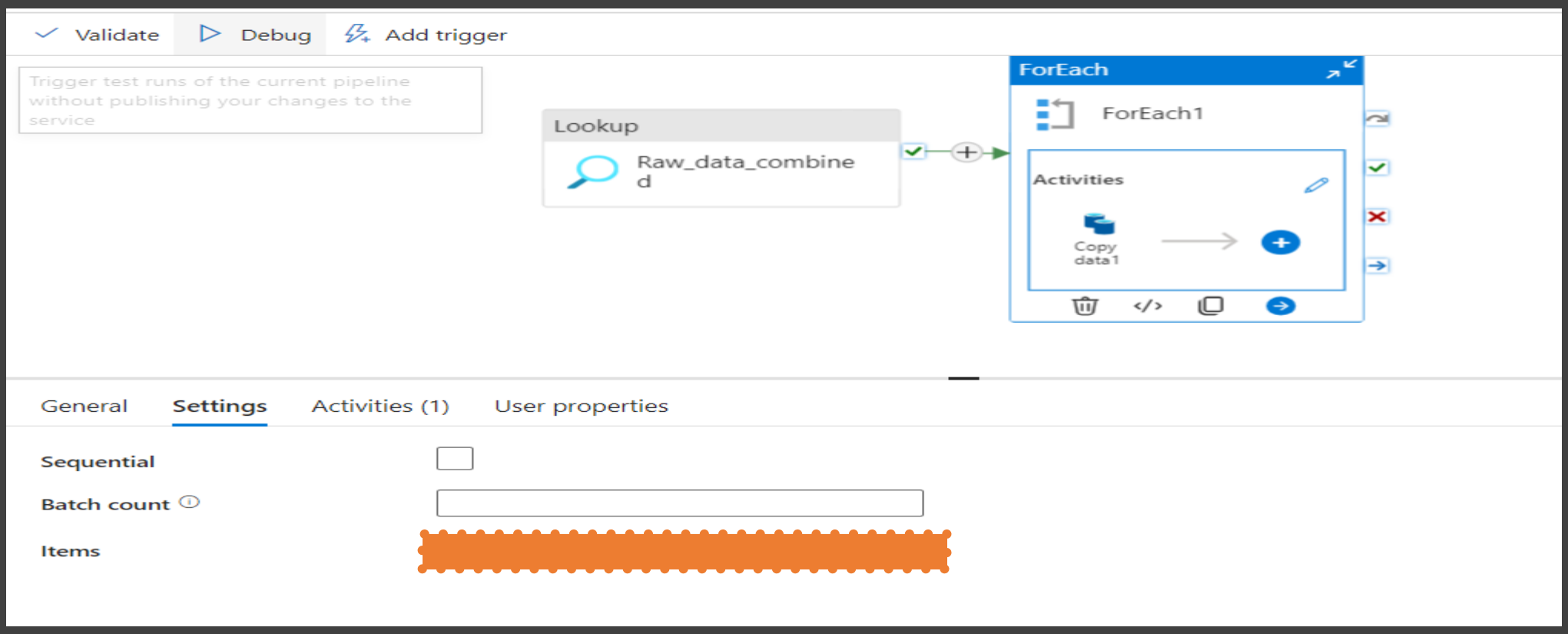


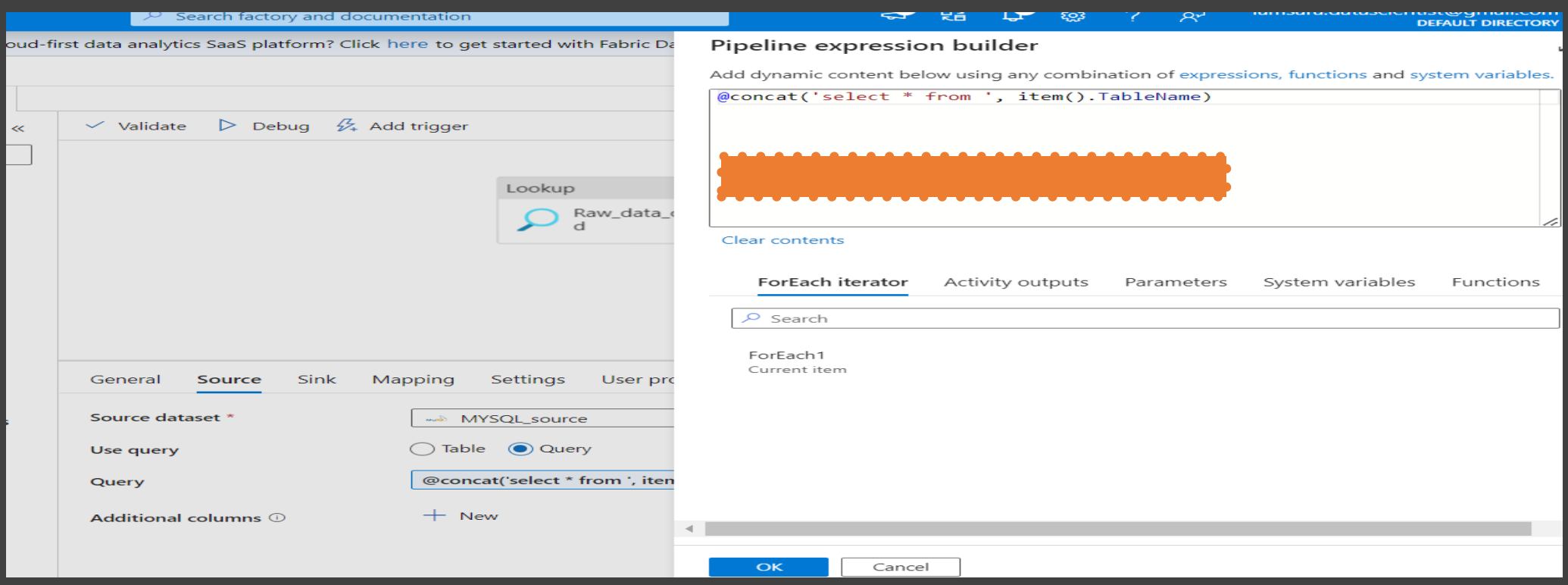


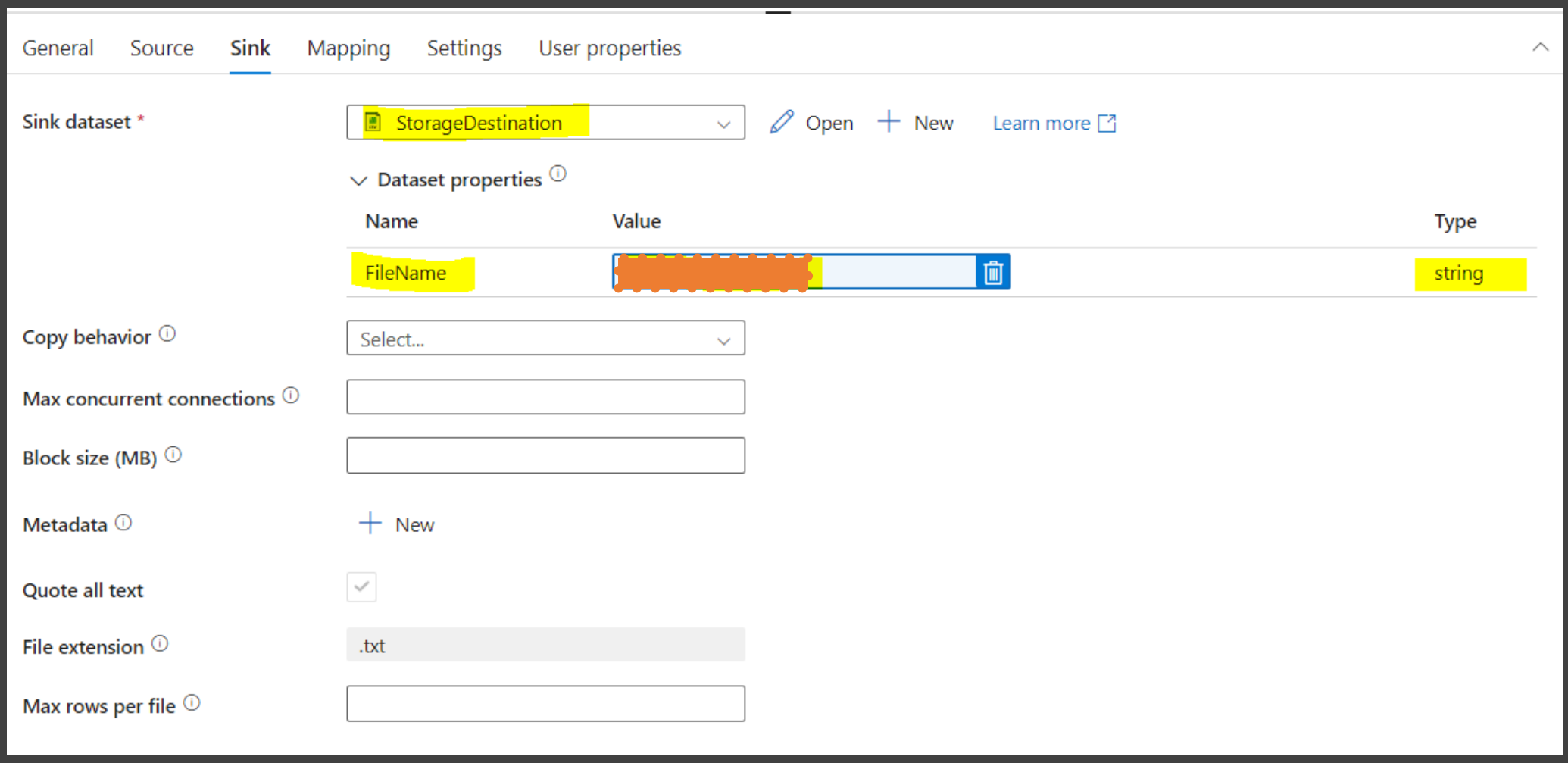
5. **Configured Data Pipeline:**
   - In Azure Data Factory, I used a **Lookup activity** to select the appropriate source from the control table. This involved dragging and dropping the activity into the pipeline.
6. **Used ForEach Activity:**
   - I utilized a **ForEach activity** to iterate over the 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.



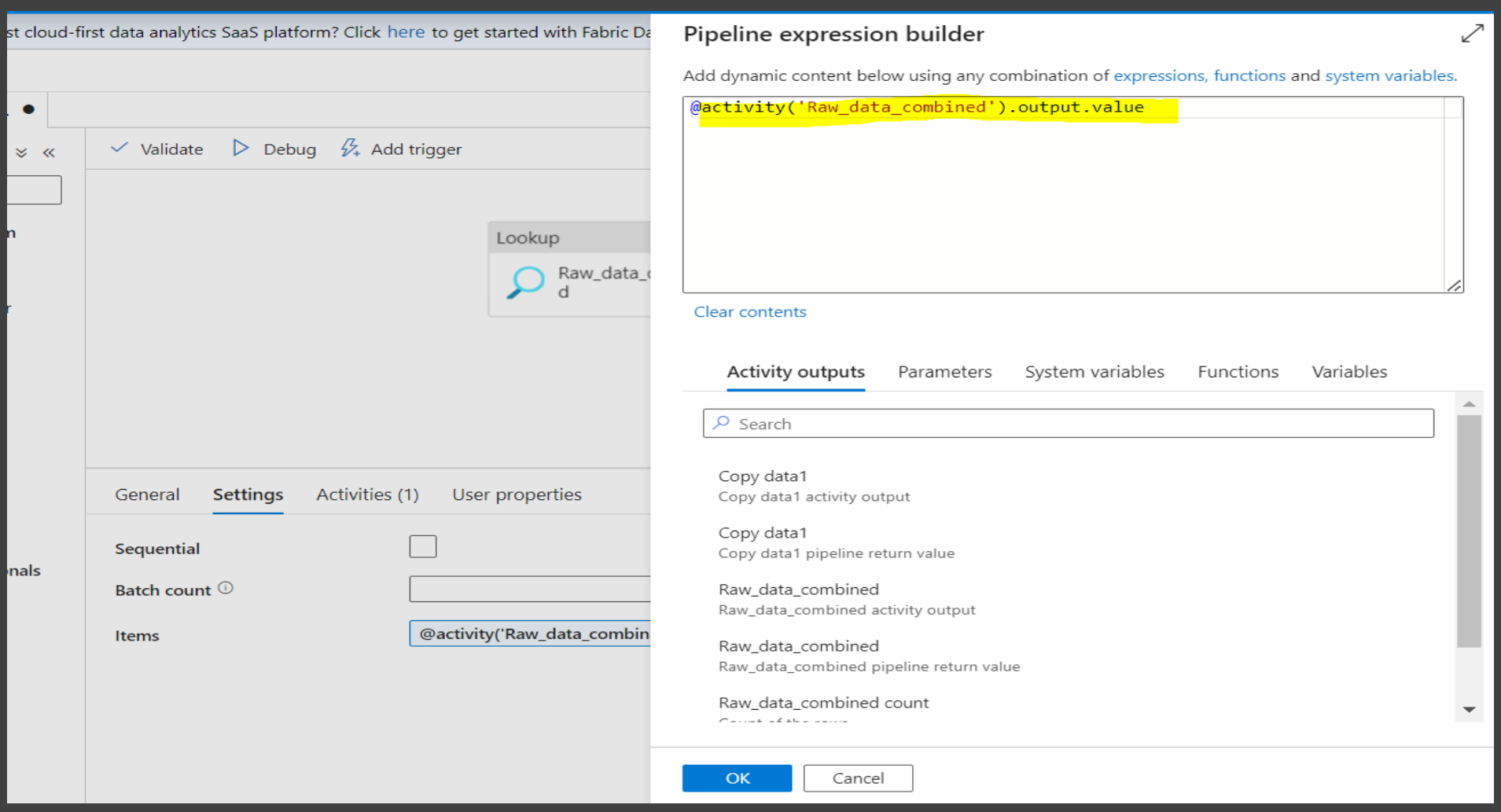
8. **Debugged and Tested:**
   - I debugged the pipeline to ensure it was working correctly and that the data was being processed as expected. I provided the path to the landing container, which I had set up in my **Azure storage account**.
9. **Loaded CSV Files:**
   - In the Azure storage account, 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

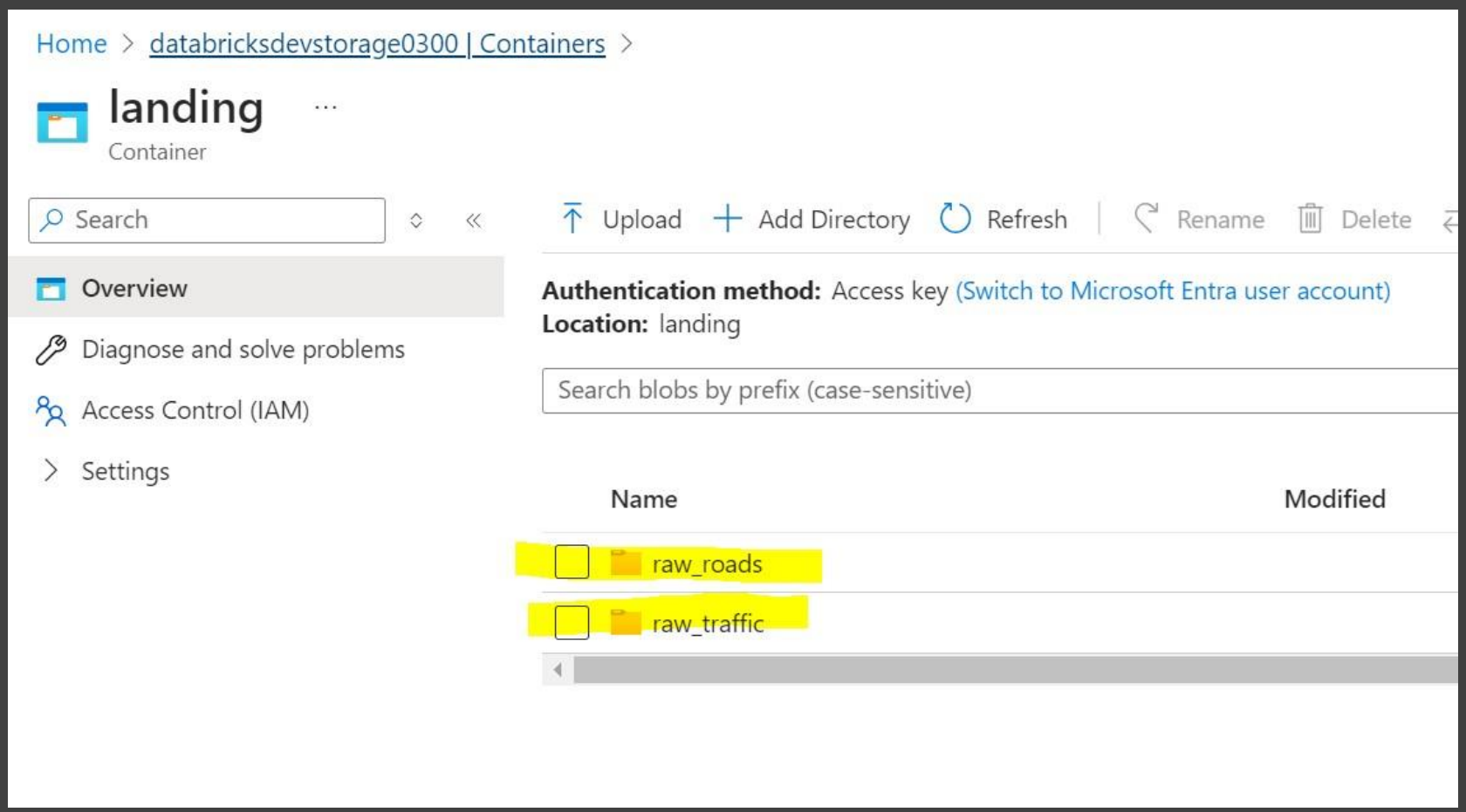

Validate
Debug
Add trigger
Trigger test runs of the current pipeline without publishing your changes to the service
Lookup
Raw_data_combined
ForEach
ForEach1
Activities
Copy data1
General
Settings
Activities (1)
User properties
Sequential
Batch count
Items


Search factory and documentation
DEFAULT DIRECTORY
oud-first data analytics SaaS platform? Click here to get started with Fabric Da
Validate
Debug
Add trigger
Lookup
General
Source
Sink
Mapping
Settings
Source dataset *
MYSQL_source
Use query
Table
Query
Query
@concat('select * from ', item
Additional columns
New
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



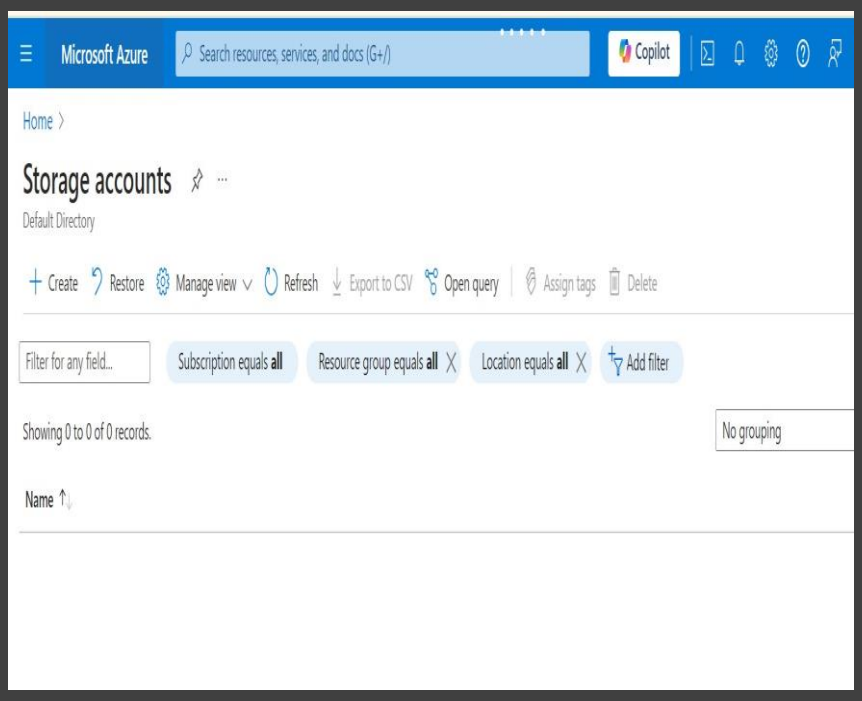
Foreach working

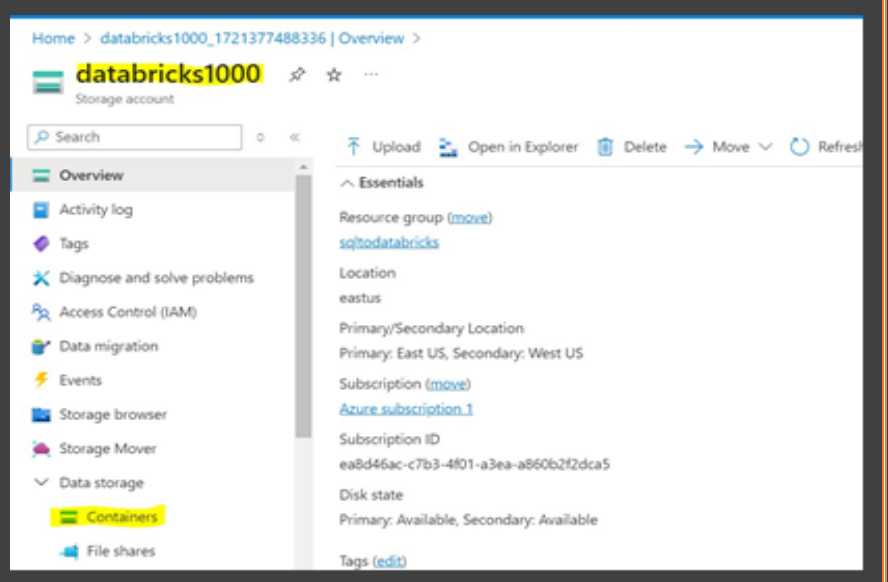


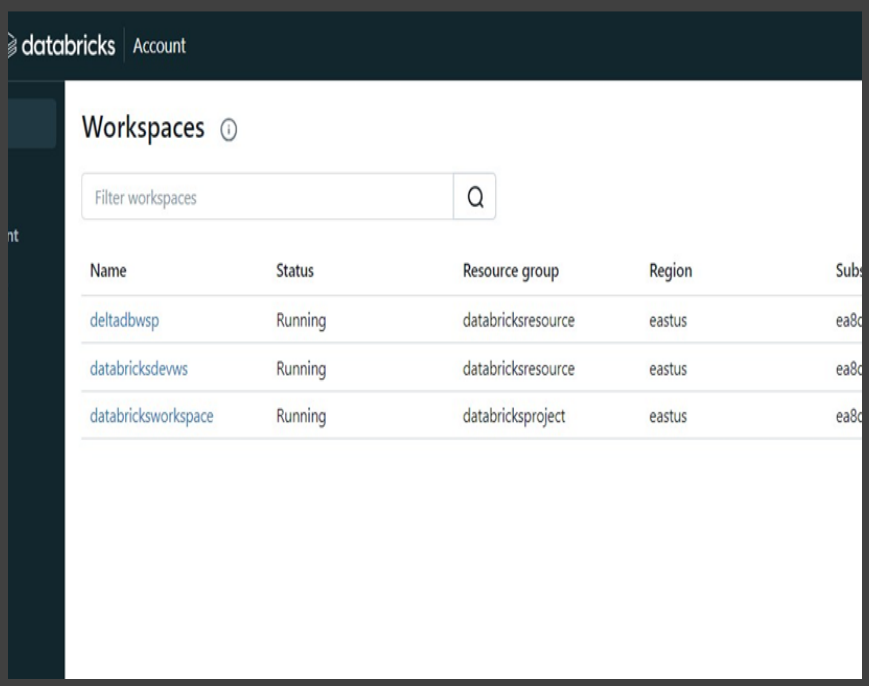


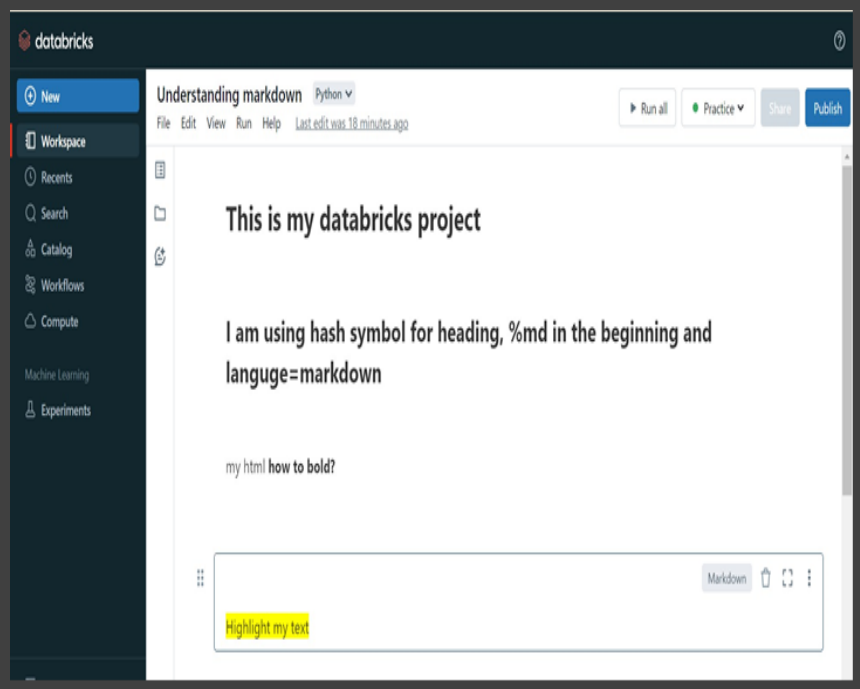
Working:

- 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 herairichal namespace because it will create simple bolb 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.



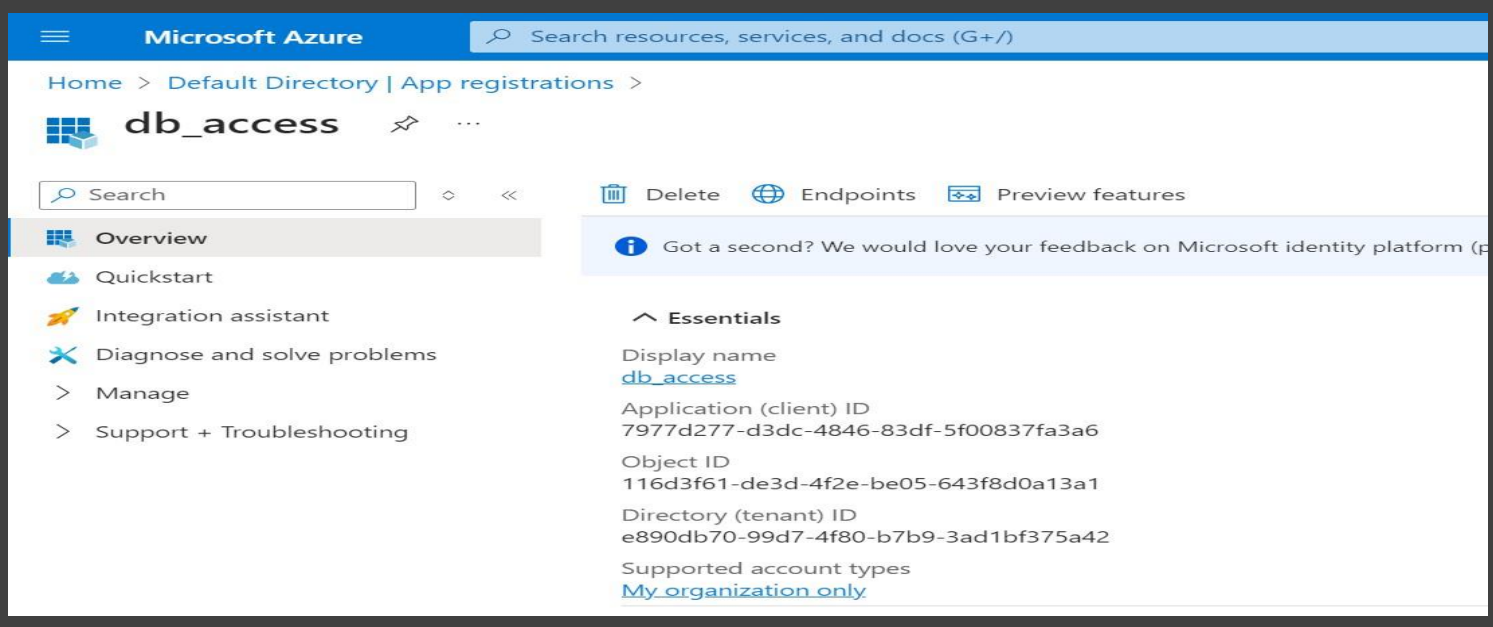



databricks
Account
Workspaces
Filter workspaces
Name
Status
Resource group
Region
deltadbwsp
Running
databricksresource
eastus
databricksdevws
Running
databricksresource
eastus
databricksworkspace
Running
databricksproject
eastus


databricks
New
Workspace
Recents
Search
Catalog
Workflows
Compute
Machine Learning
Experiments
Understanding markdown
Python
File Edit View Run Help
Last edit was 18 minutes ago
Run all
Practice
Share
Publish
This is my databricks project
I am using hash symbol for heading, %md in the beginning and languge=markdown
my html how to bold?
Markdown
Highlight my text

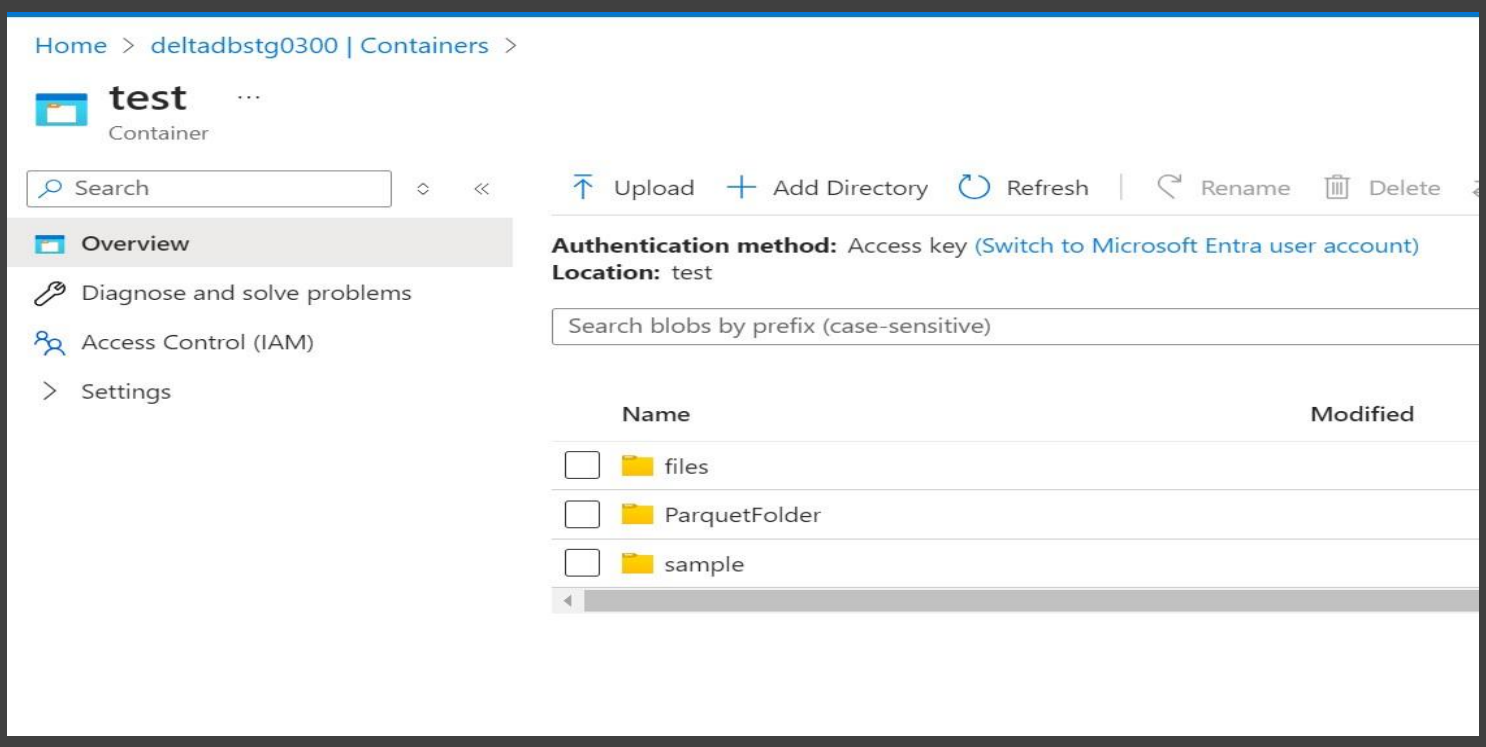
Q. How to configure connection between databricks and azure lake storage?

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 choosen 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"



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



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

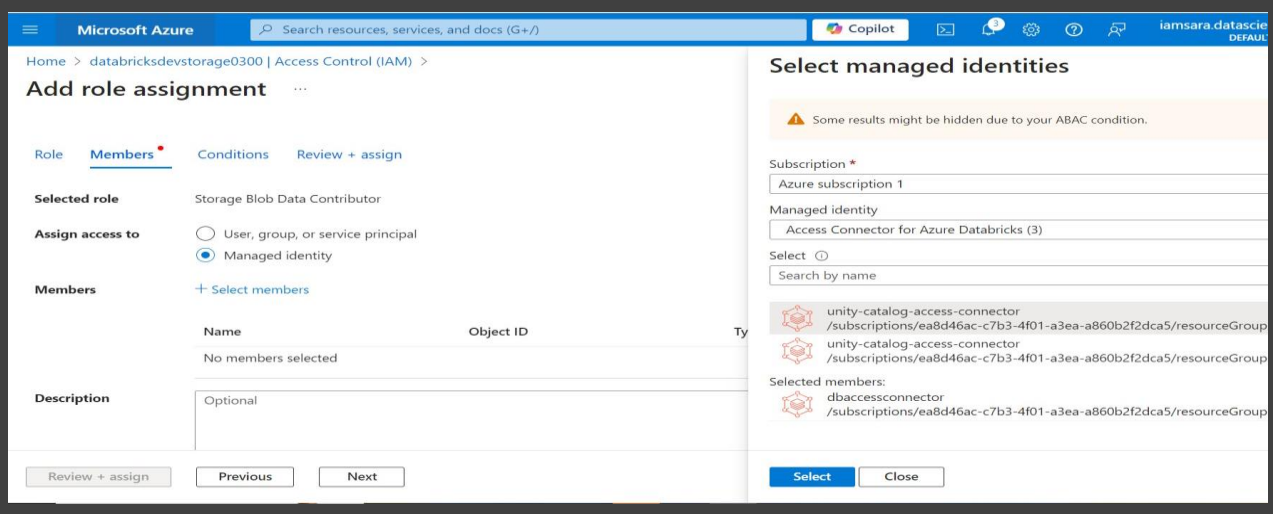
Sample:

```
service_credential = dbutils.secrets.get(scope="<secret-scope>",key="<service-credential-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@deltadbstg0300.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?

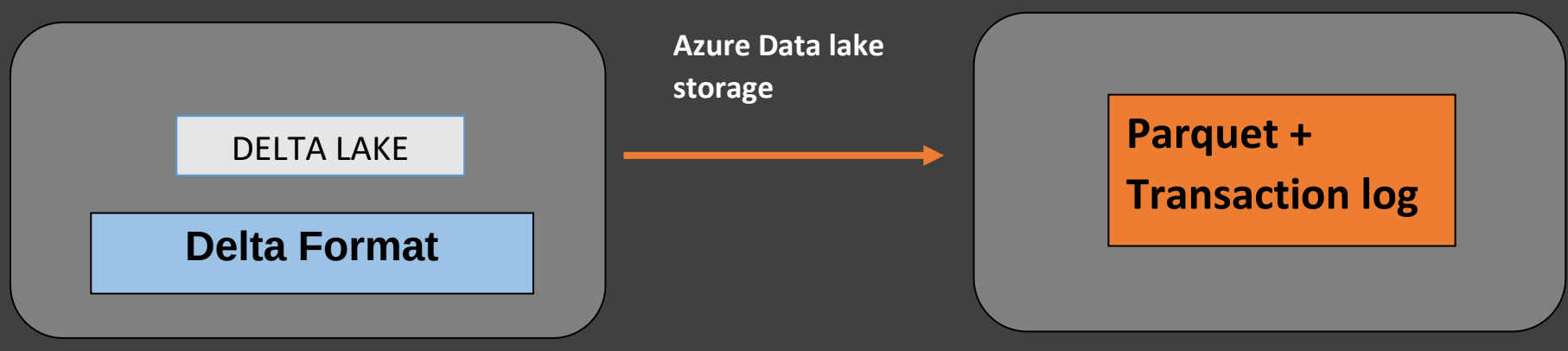
Databricks Utilities (dbutils) are a set of functions provided by Databricks to help users interact with Databricks notebooks and clusters more efficiently. They facilitate tasks such as managing files, creating widgets, and running notebooks programmatically.

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.**

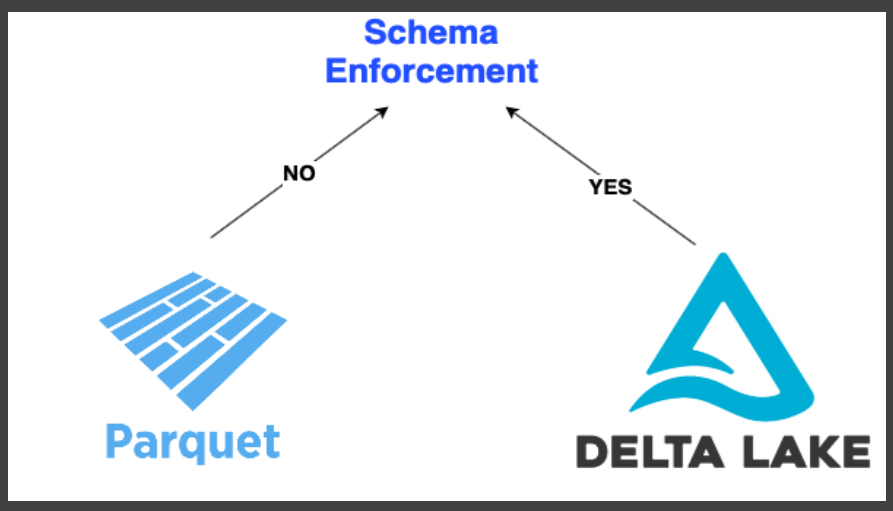
Q. How to create delta tables using SQL?

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 etastore 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.

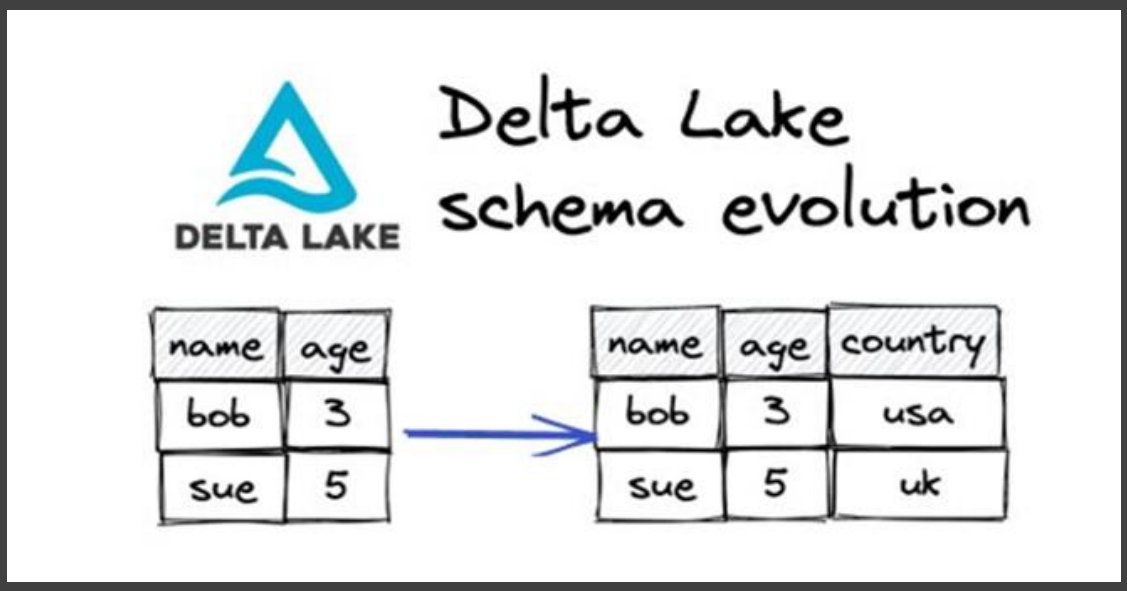
Errors:

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)

Sara Irfan (sara.data.insights@gmail.com)

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

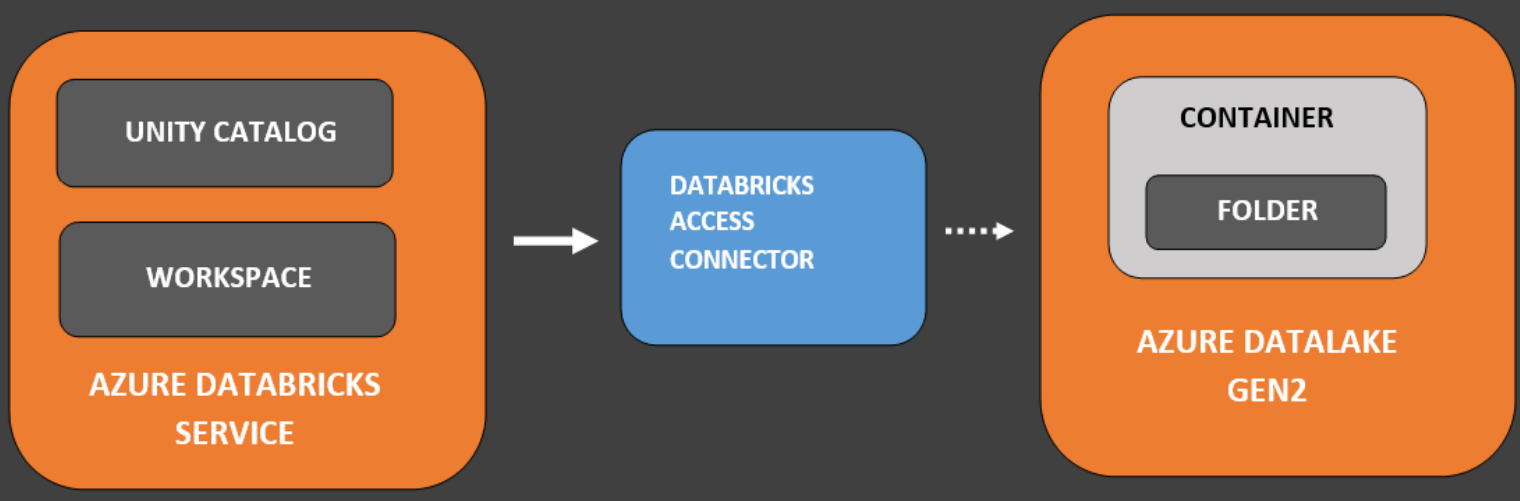
Q. Why need of Switching from Hive metastore to Unity Catalog?

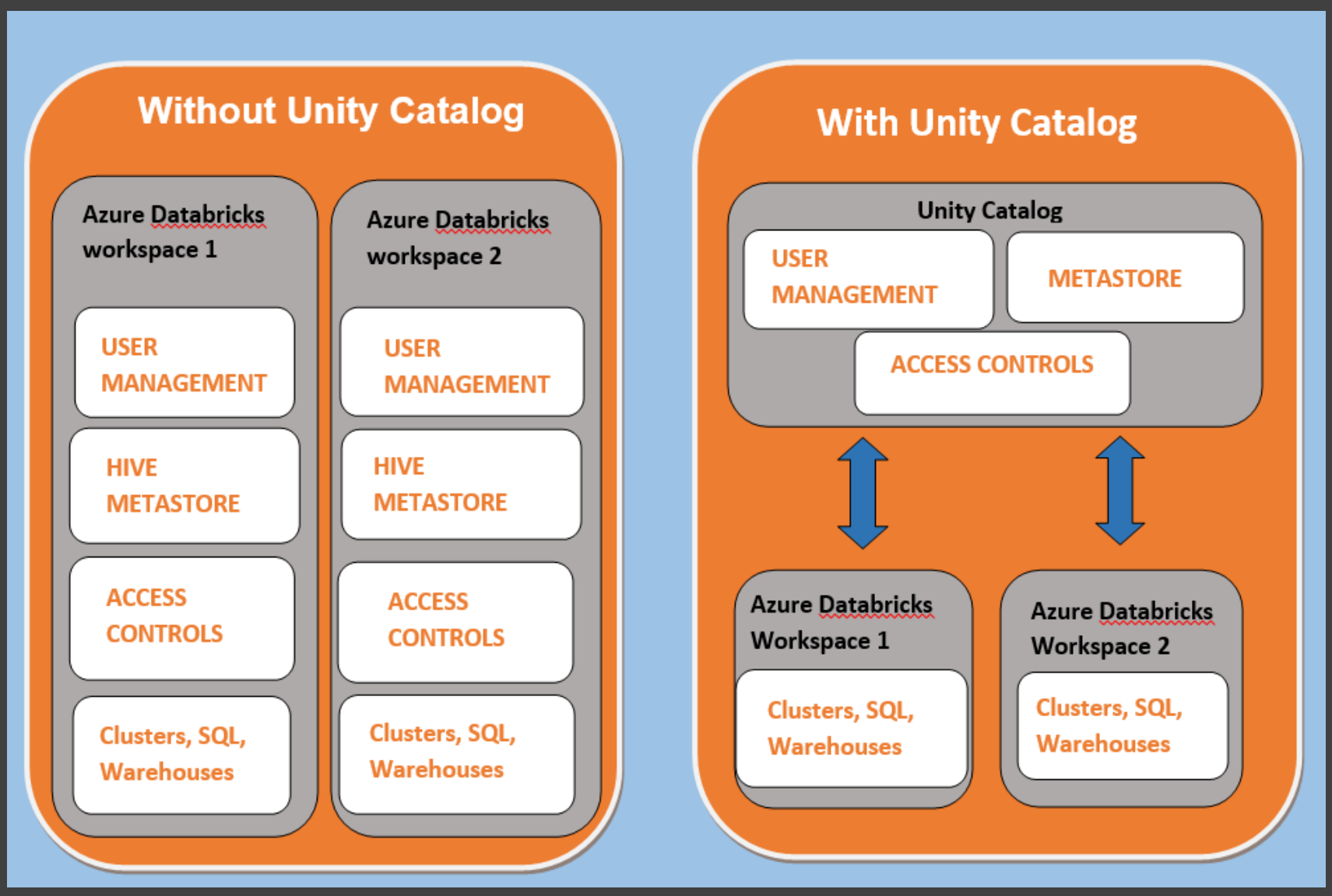
Unity catalog is overcoming the limitations of Traditional Hive metastore and providing more control over metadata and access management.

1- **Unity Catalog provides native support for Delta Lake, a powerful storage layer that enhances reliability, performance, and data integrity in Databricks environments.**

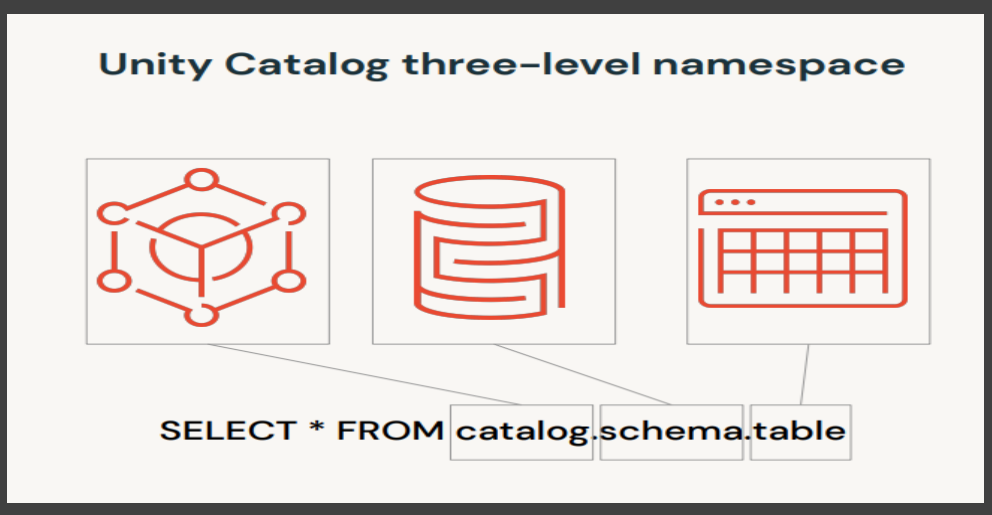
2- **Delta Lake offers features such as ACID transactions, schema enforcement, time travel queries, and data versioning, which are not available in traditional Hive Metastore setups.**

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.**

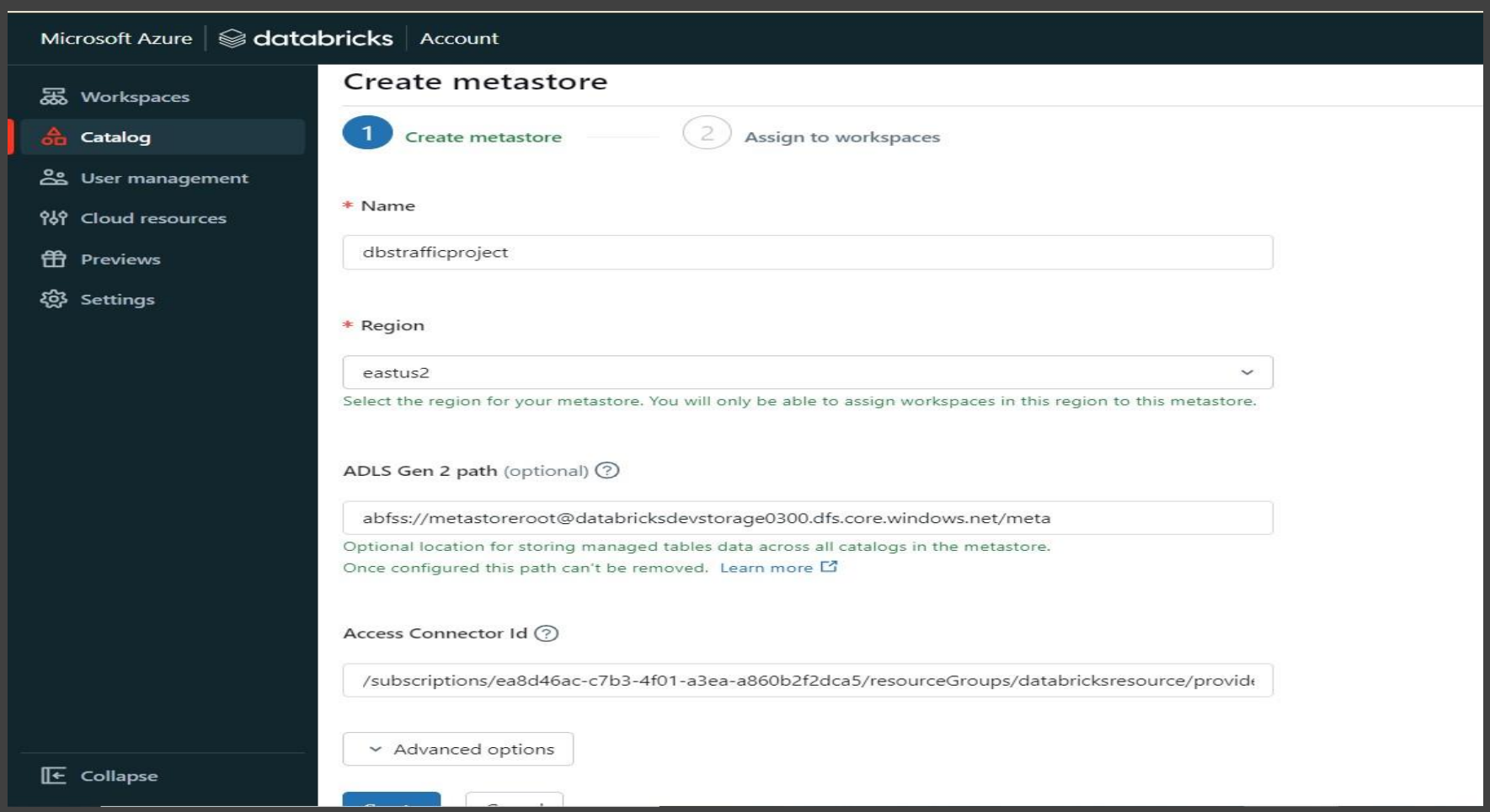




Create **databricksdevws**, search access connector for databricks. Its **resource id** is used to manage **metadata**. The access connector has given to this storage account



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**

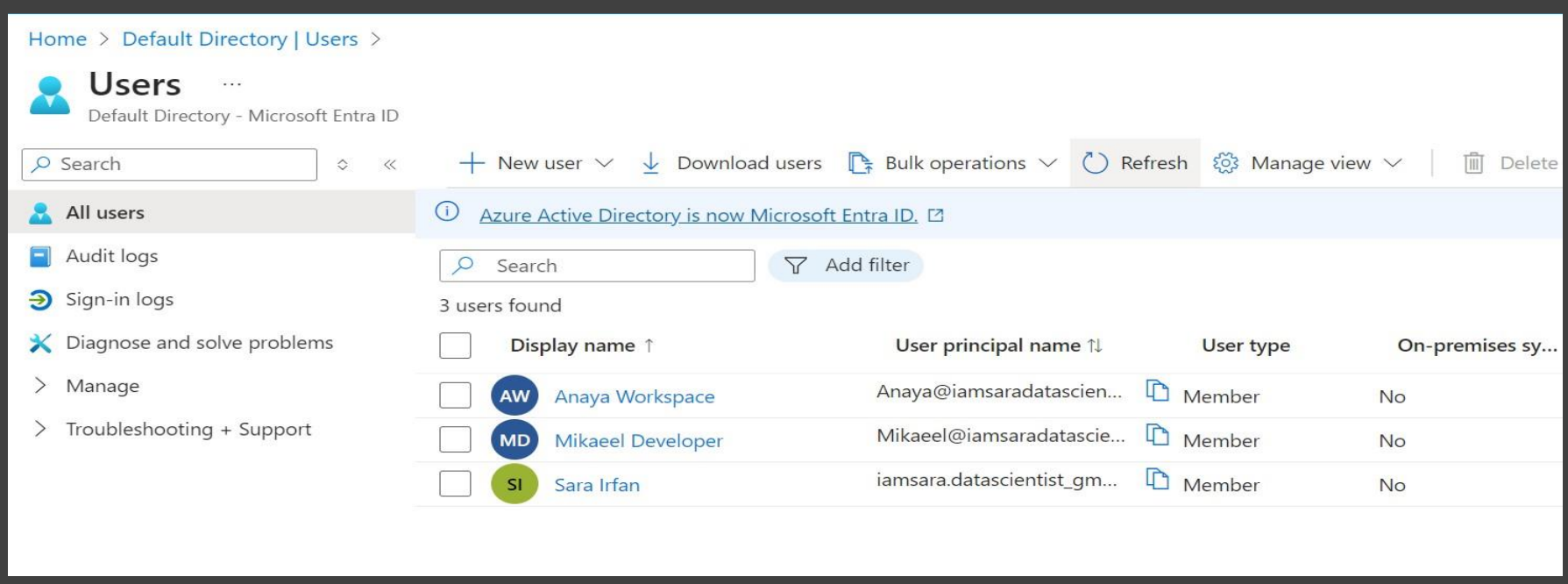


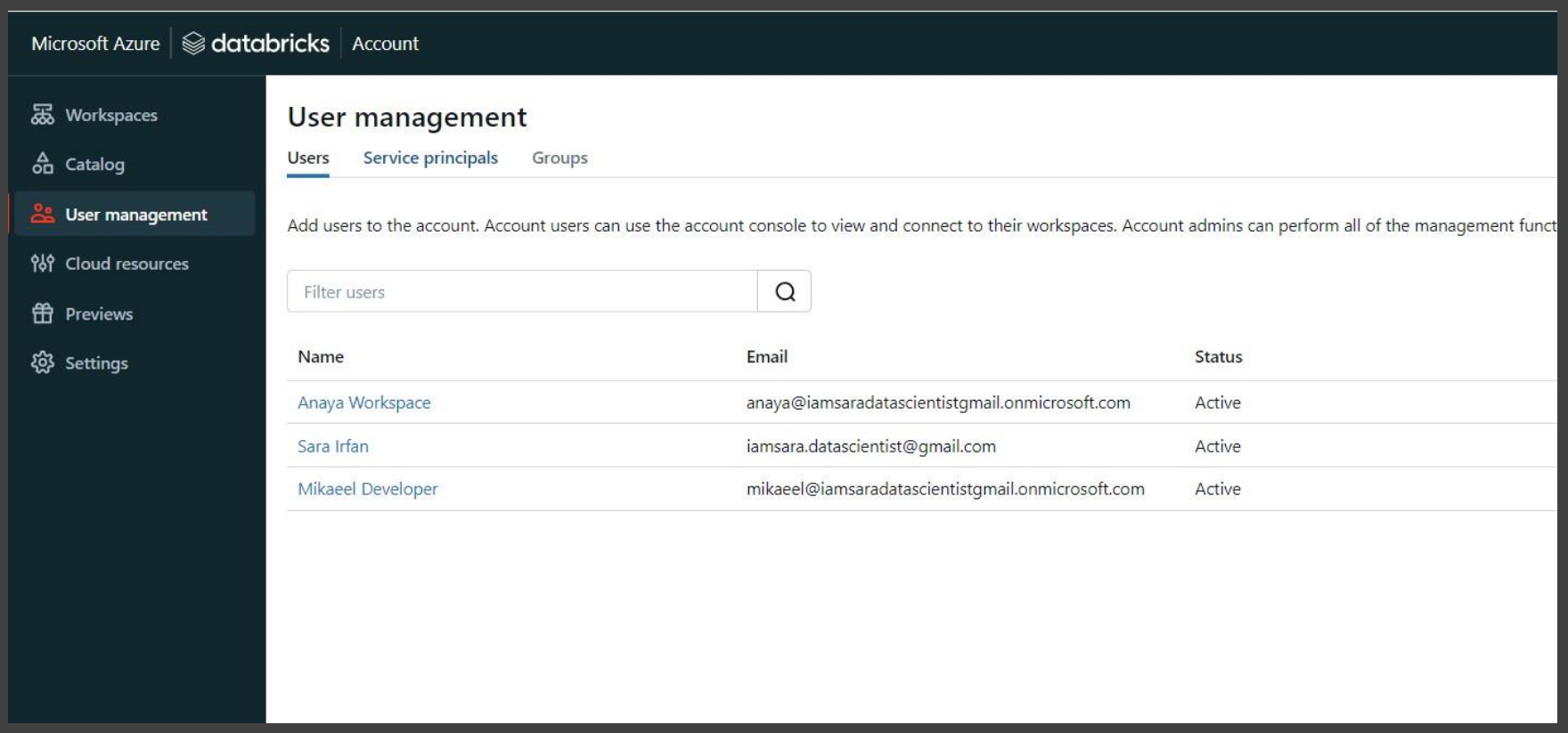
User management:

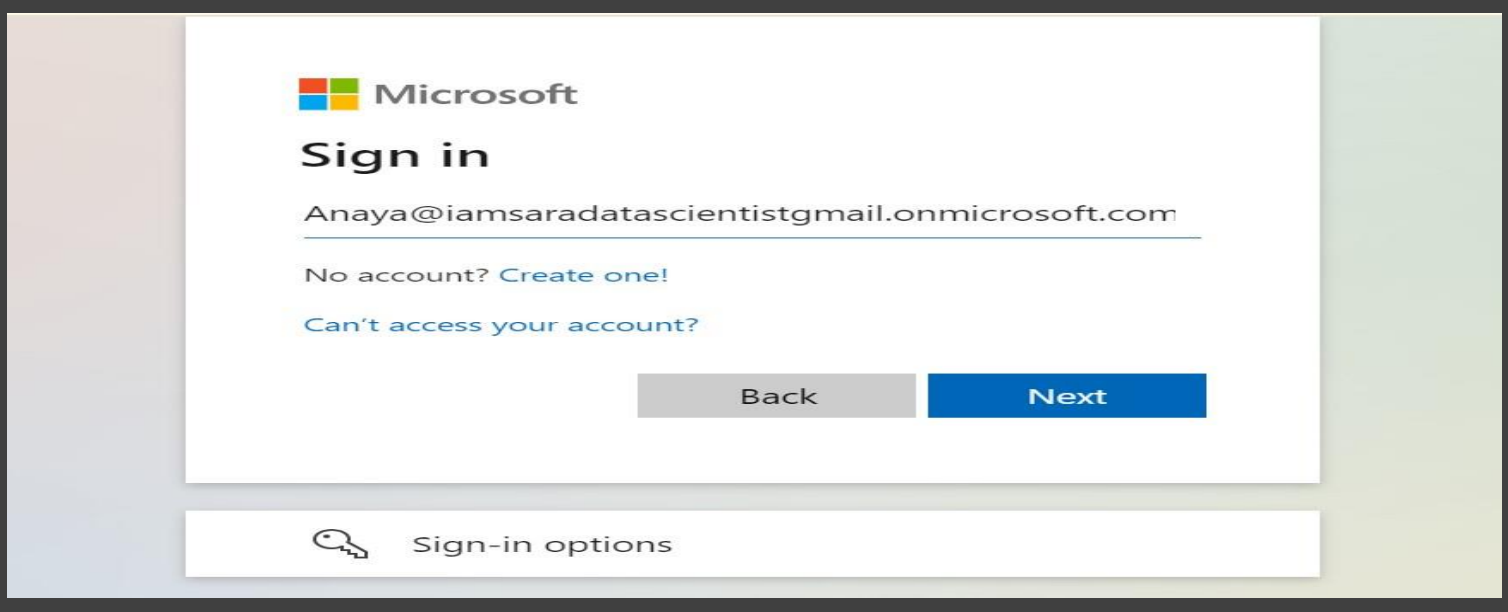
1) Invite and add users to Unity Catalog
2) Create groups
- **Workspace admins**
- **Developers**

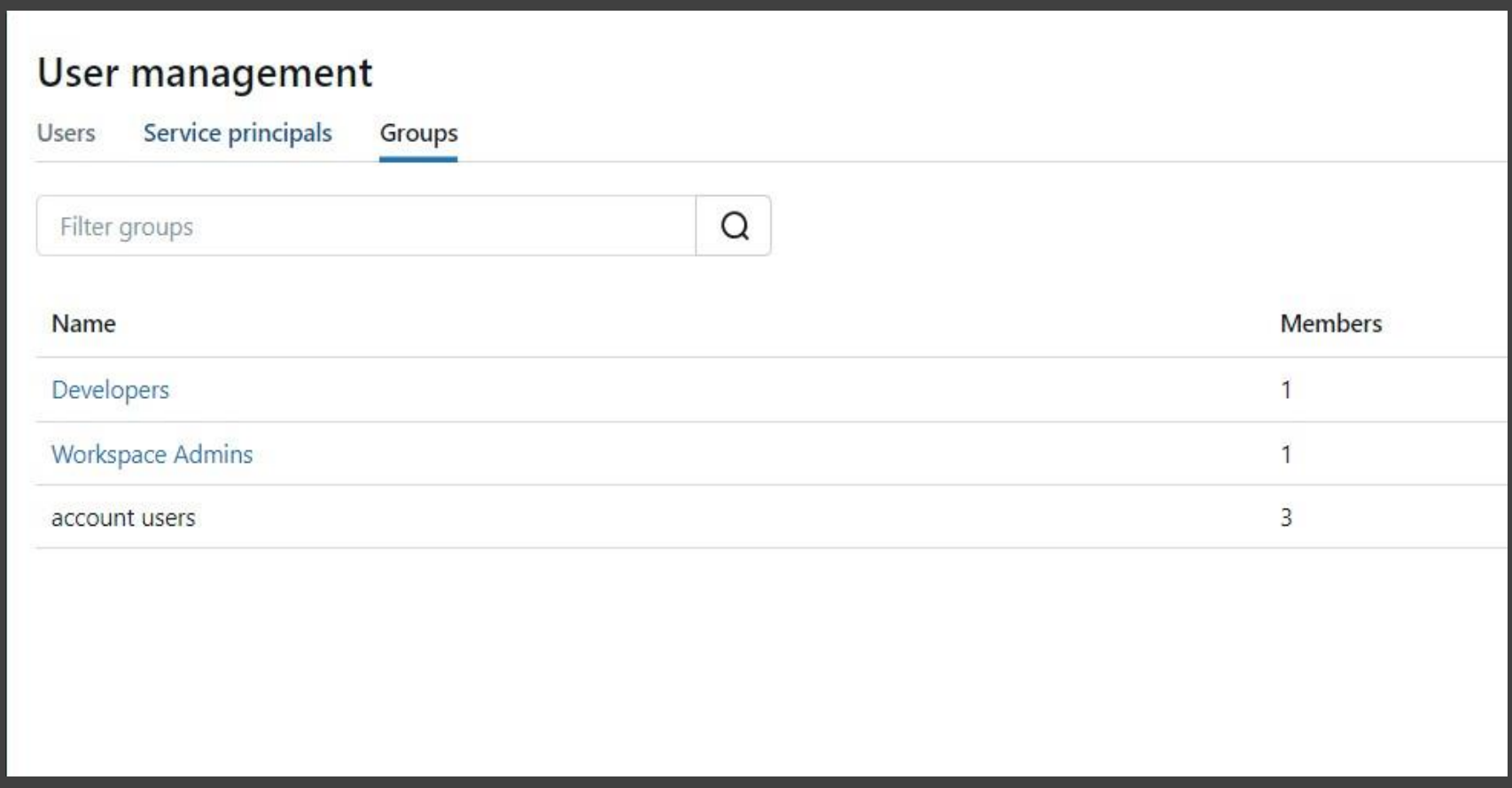
3) Assign groups to users
- **Workspace admins – Anaya**
- **Developers - Mikaeel**

4) Assign roles to groups
- **Workspace Admin – Workspace Admins Group**
- **Workspace User – Developers Group**



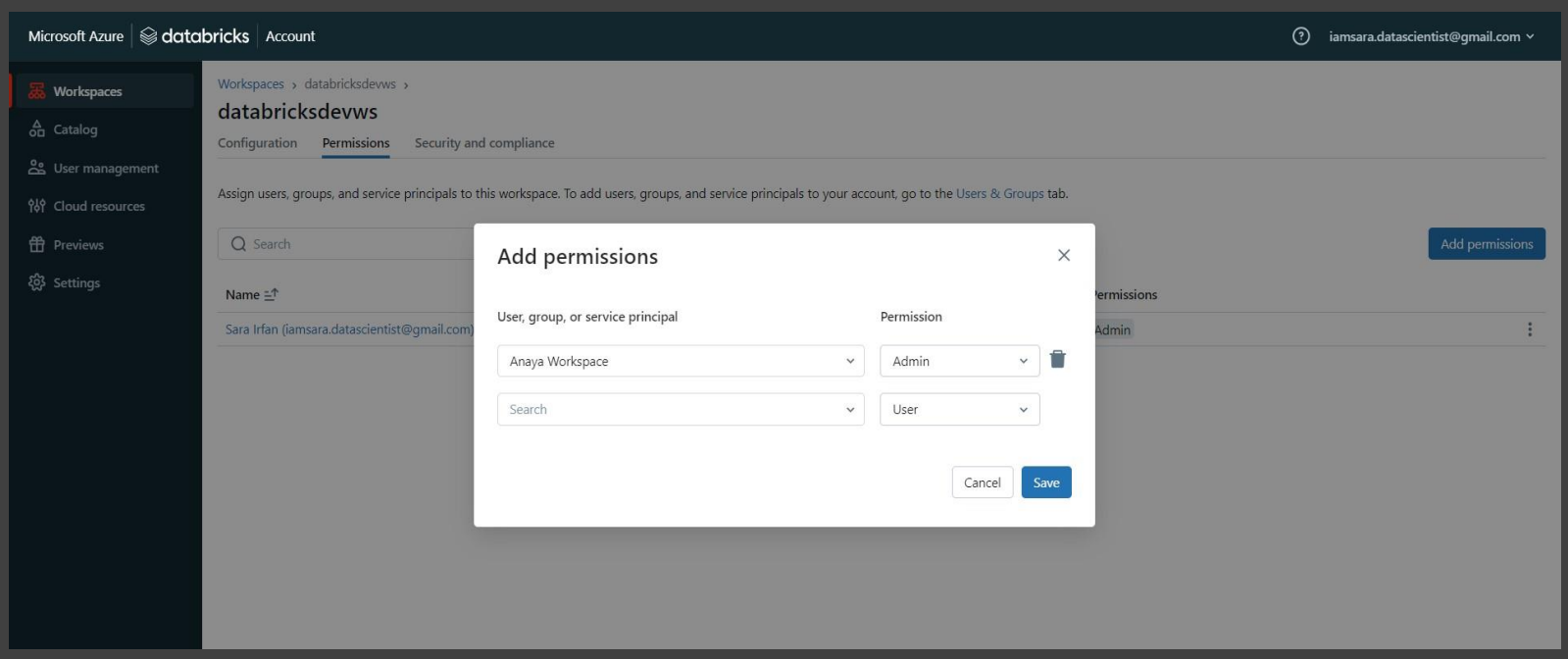

Microsoft Azure
databricks
Account
Workspaces
Catalog
User management
Cloud resources
Previews
Settings
User management
Users
Service principals
Groups
Add users to the account. Account users can use the account console to view and connect to their workspaces. Account admins can perform all of the management funct
Filter users
Name
Email
Status
Anaya Workspace
anaya@iamsaradatascientistgmail.onmicrosoft.com
Active
Sara Irfan
iamsara.datascientist@gmail.com
Active
Mikaeel Developer
mikaeel@iamsaradatascientistgmail.onmicrosoft.com
Active

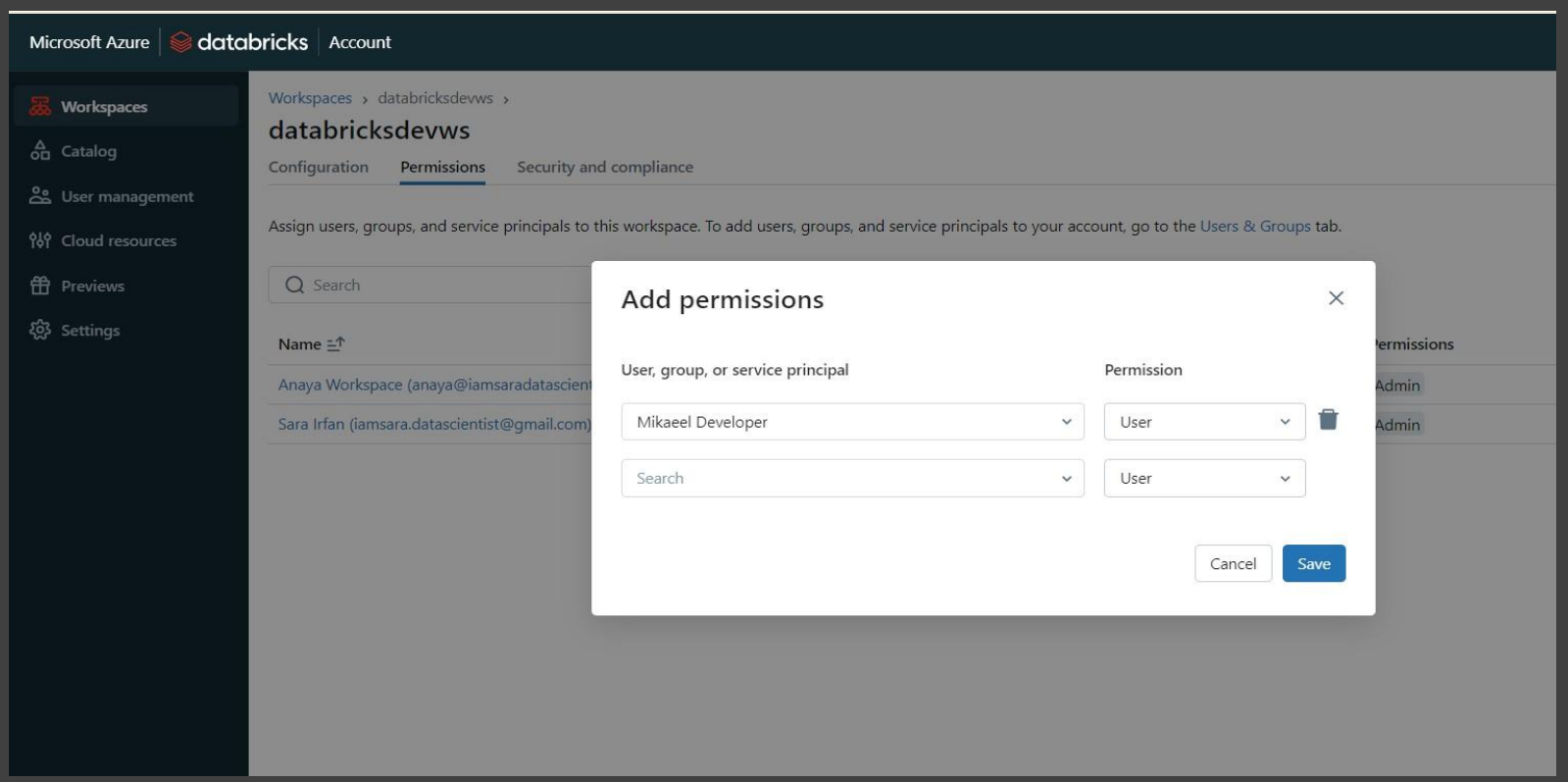

Microsoft
Sign in
Anaya@iamsaradatascientistgmail.onmicrosoft.com
No account? Create one!
Can't access your account?
Back
Next
Sign-in options

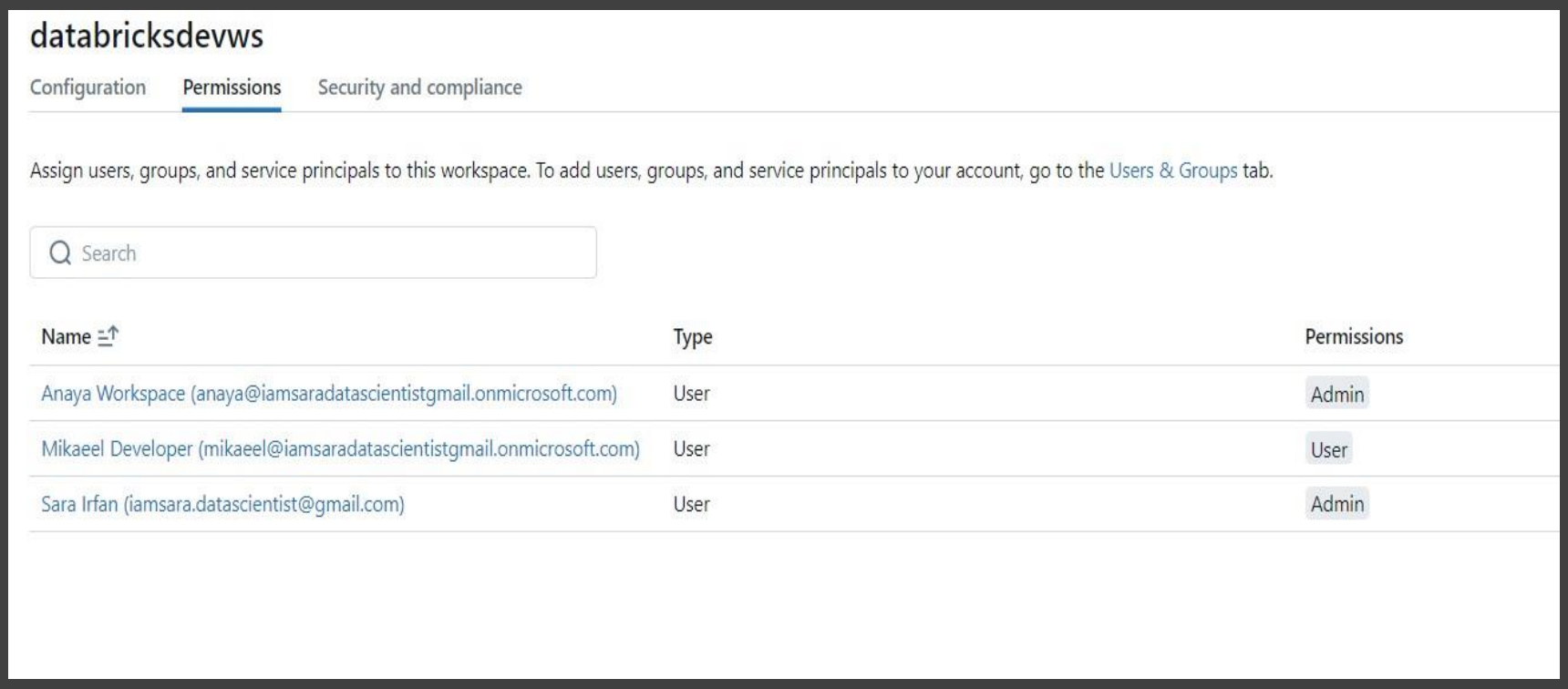

User management
Users
Service principals
Groups
Filter groups
Name
Members
Developers
1
Workspace Admins
1
account users
3

Q. How to give permissions to specific users?

By clicking on workspaces, permissions, add permissions

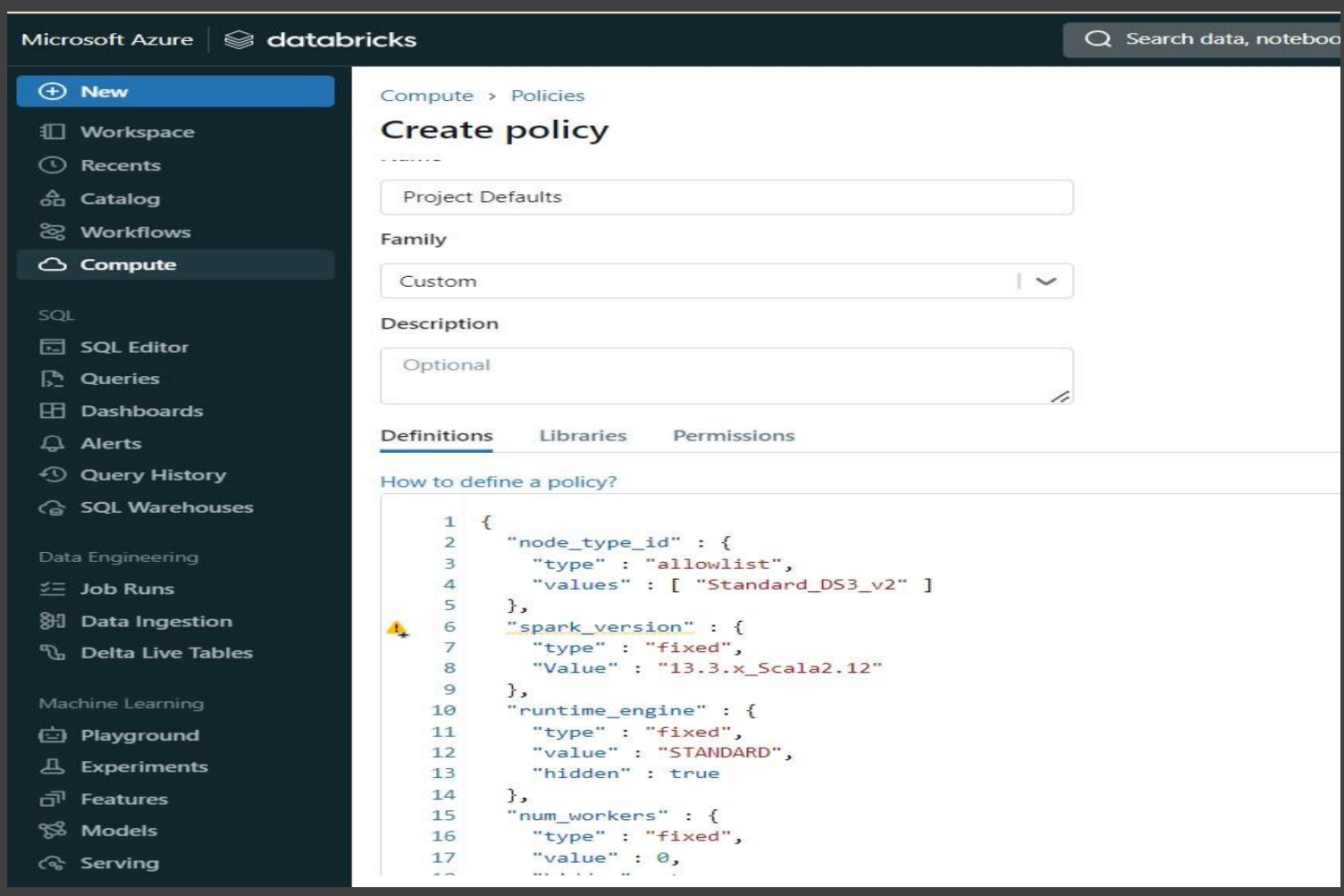






Q. How to create Cluster Pools and Policy?

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



Compute > Policies

Create policy

Definitions Libraries Permissions

How to define a policy?

```
      "value" : 0,
      "hidden" : true
    },
    "data_security_mode" : {
      "type" : "fixed",
      "values" : "SINGLE_USER"
    },
    "cluster_type" : {
      "type" : "fixed",
      "value" : "all-purpose"
    },
    "instance_pool_id" : {
      "type" : "forbidden",
      "hidden" : true
    },
    "azure_attributes.availability" : {
      "type" : "fixed",
      "value" : "ON_DEMAND_AZURE",
      "hidden" : true
    },
    "spark_conf.spark.databricks.cluster.profile" : {
      "type" : "fixed",
      "value" : "singleNode",
      "hidden" : true
    },
    "autotermination_minutes" : {
      "type" : "fixed",
      "Value" : 4000
    }
}
```

Microsoft Azure | databricks | Account

- Workspaces
- Catalog
- User management
- Cloud resources
- Previews
- Settings

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.

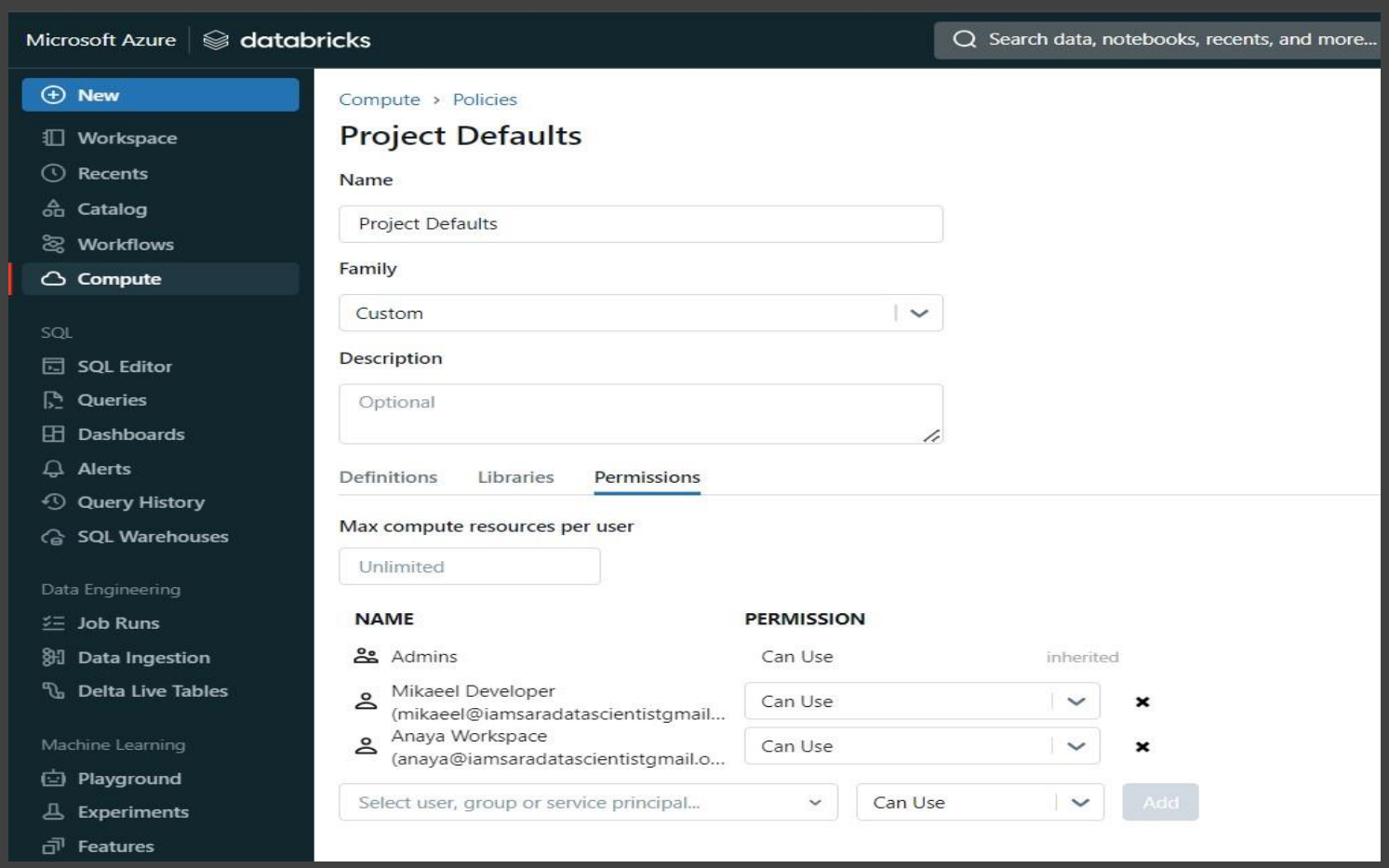
Documentation

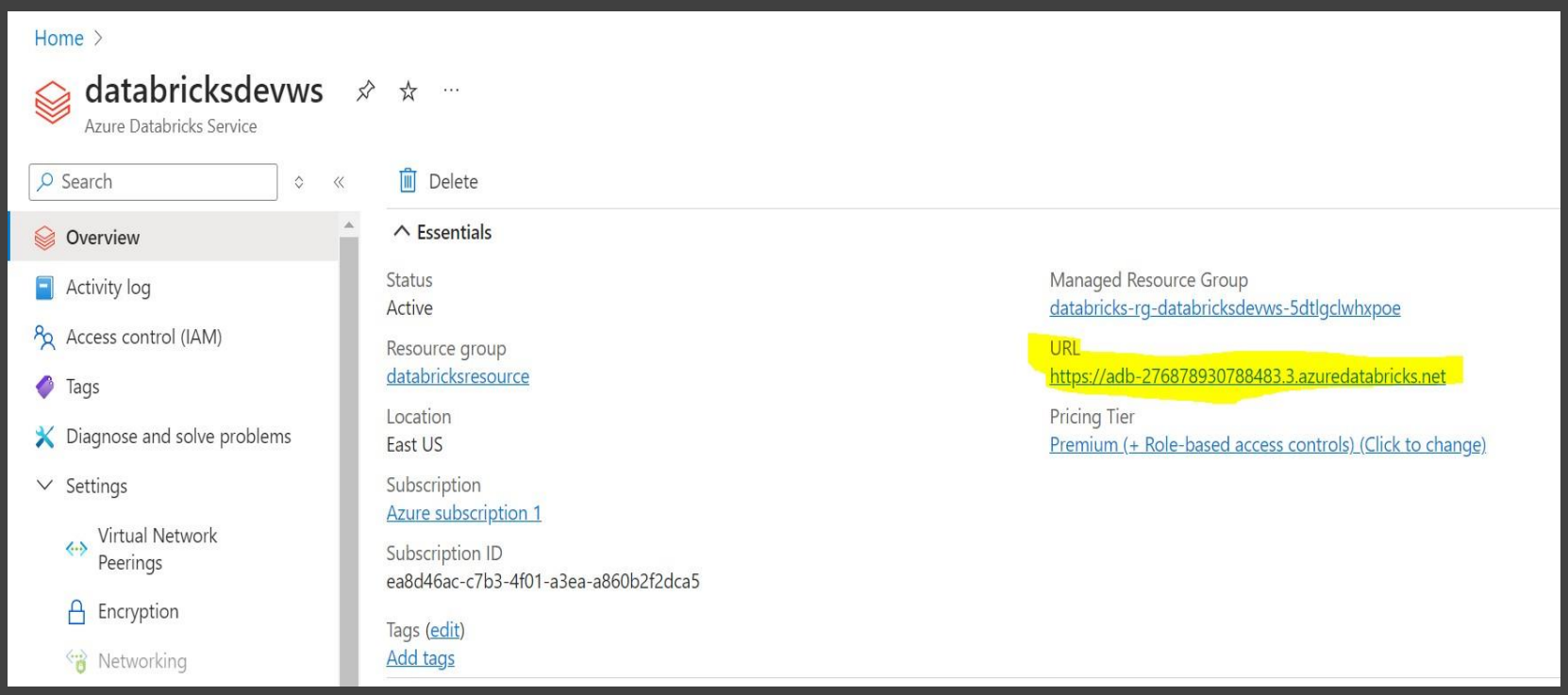
Personal Compute — Enable for all

Enables all users to create single-machine compute using the Personal Compute default policy. Learn more

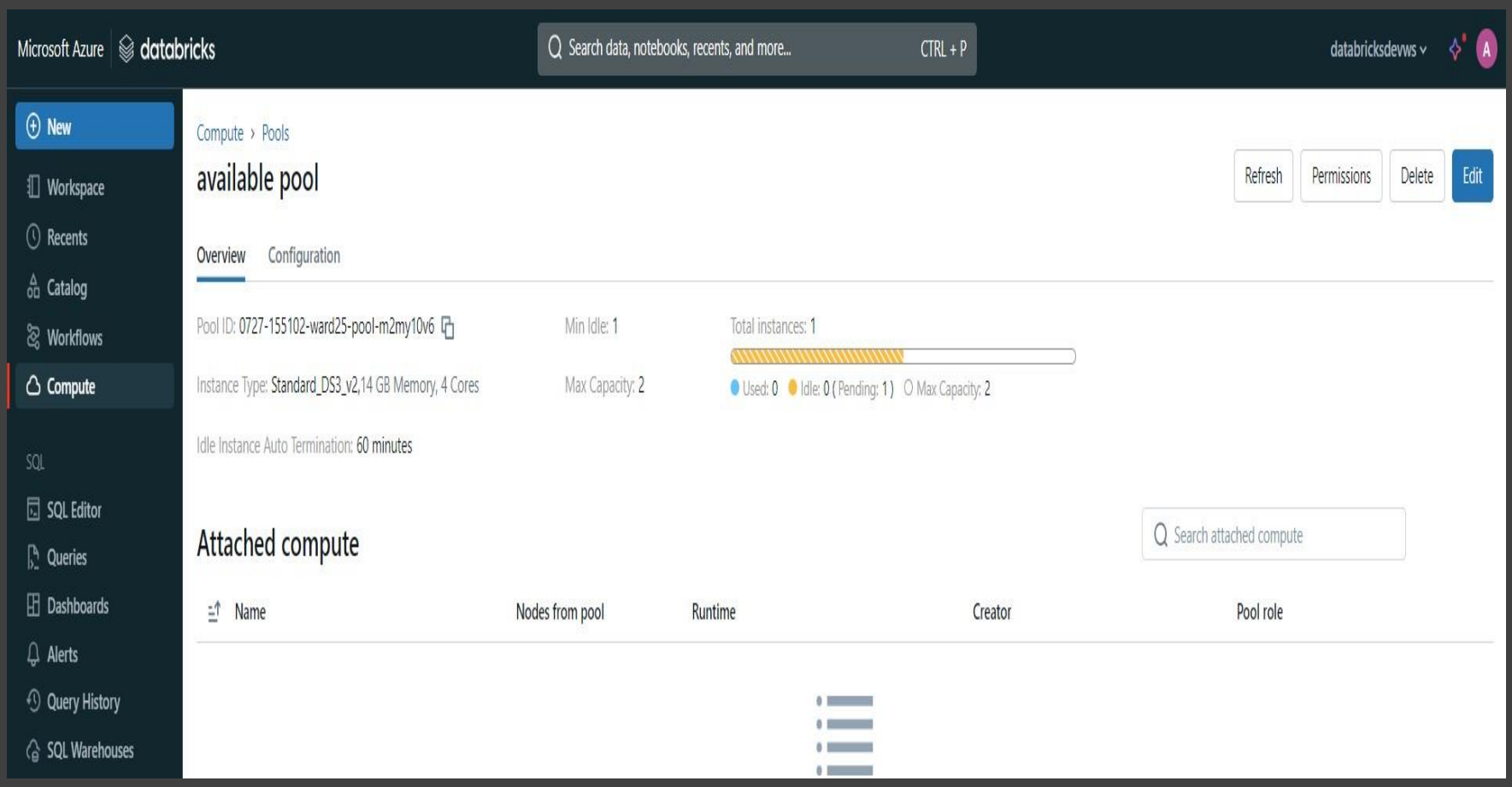
Web Terminal — Enforce On

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
Search data, notebooks, recents, and more...
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
Compute > Policies
Project Defaults
Name
Project Defaults
Family
Custom
Description
Optional
Definitions
Libraries
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
Add


Home
databricksdevws
Azure Databricks Service
Search
Delete
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Settings
Virtual Network Peerings
Encryption
Networking
Essentials
Status
Active
Resource group
databricksresource
Location
East US
Subscription
Azure subscription 1
Subscription ID
ea8d46ac-c7b3-4f01-a3ea-a860b2f2dca5
Tags (edit)
Add tags
Managed Resource Group
databricks-rg-databricksdevws-5dtlgclwhxpoe
URL
https://adb-276878930788483.3.azuredatabricks.net
Pricing Tier
Premium (+ Role-based access controls) (Click to change)

Microsoft Azure
databricks
Search data, notebooks, recents, and more...
CTRL + P
databricksdevws
New
Workspace
Recents
Catalog
Workflows
Compute
SQL
SQL Editor
Queries
Dashboards
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Compute
All-purpose compute
Job compute
SQL warehouses
Vector Search
Pools
Policies
Create pool
Name
Instance type
Min idle
Max capacity
Idle instances
Used instances
Create an instance pool
Your workspace has no pools. Learn more about instance pools
Create pool

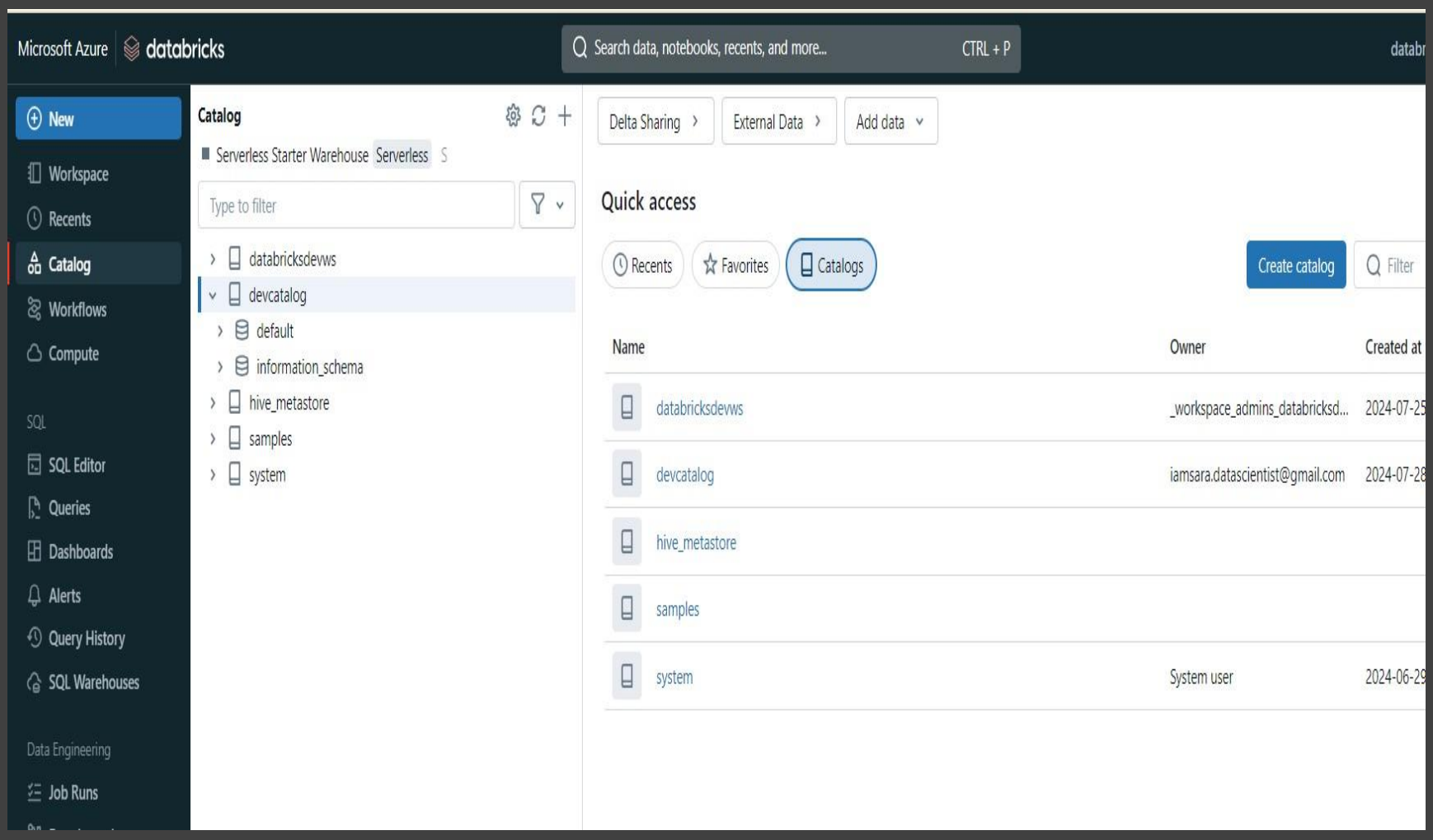


Q. How to create catalog and how it can be viewable by all other users?

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



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.

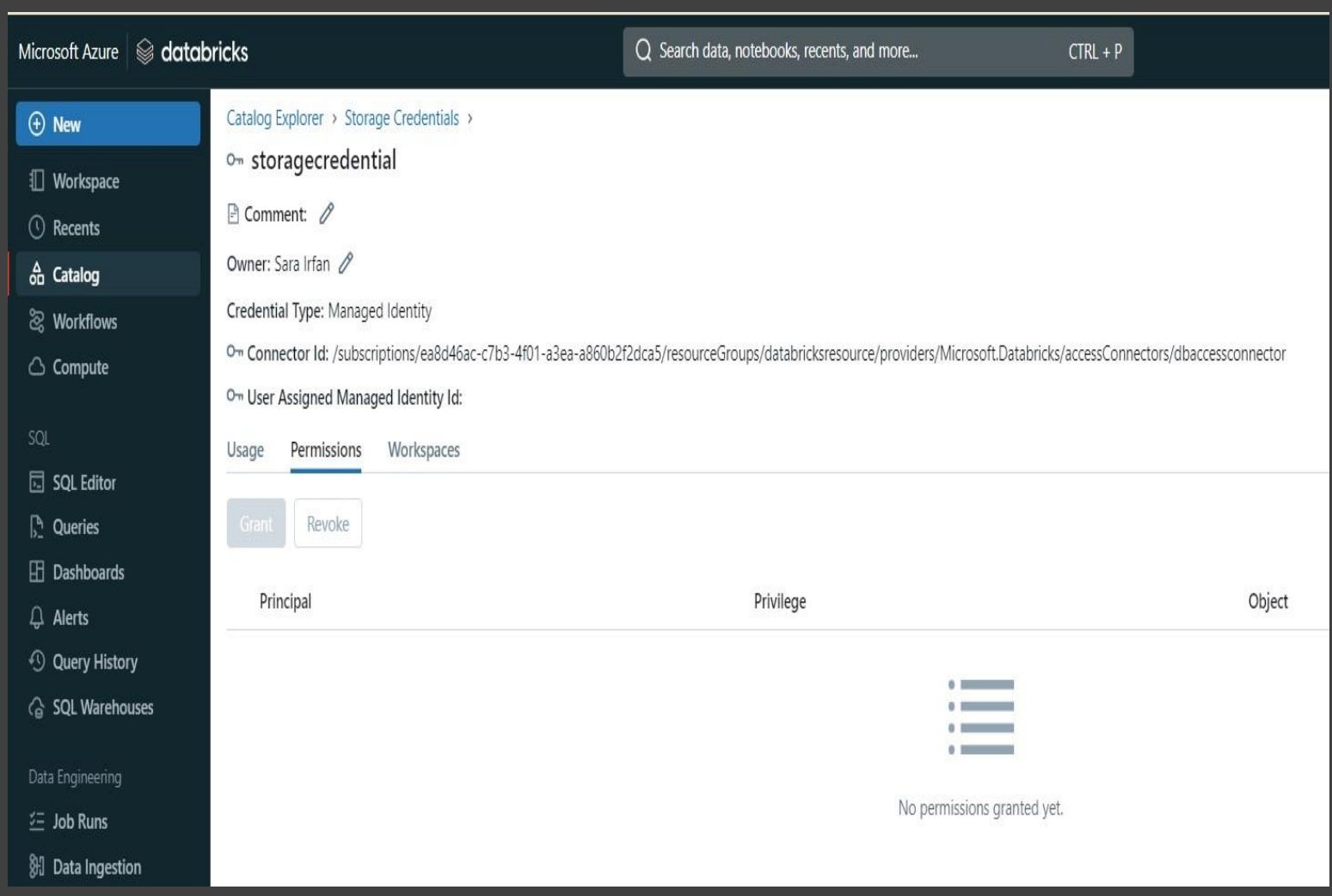
External Tables

- 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.

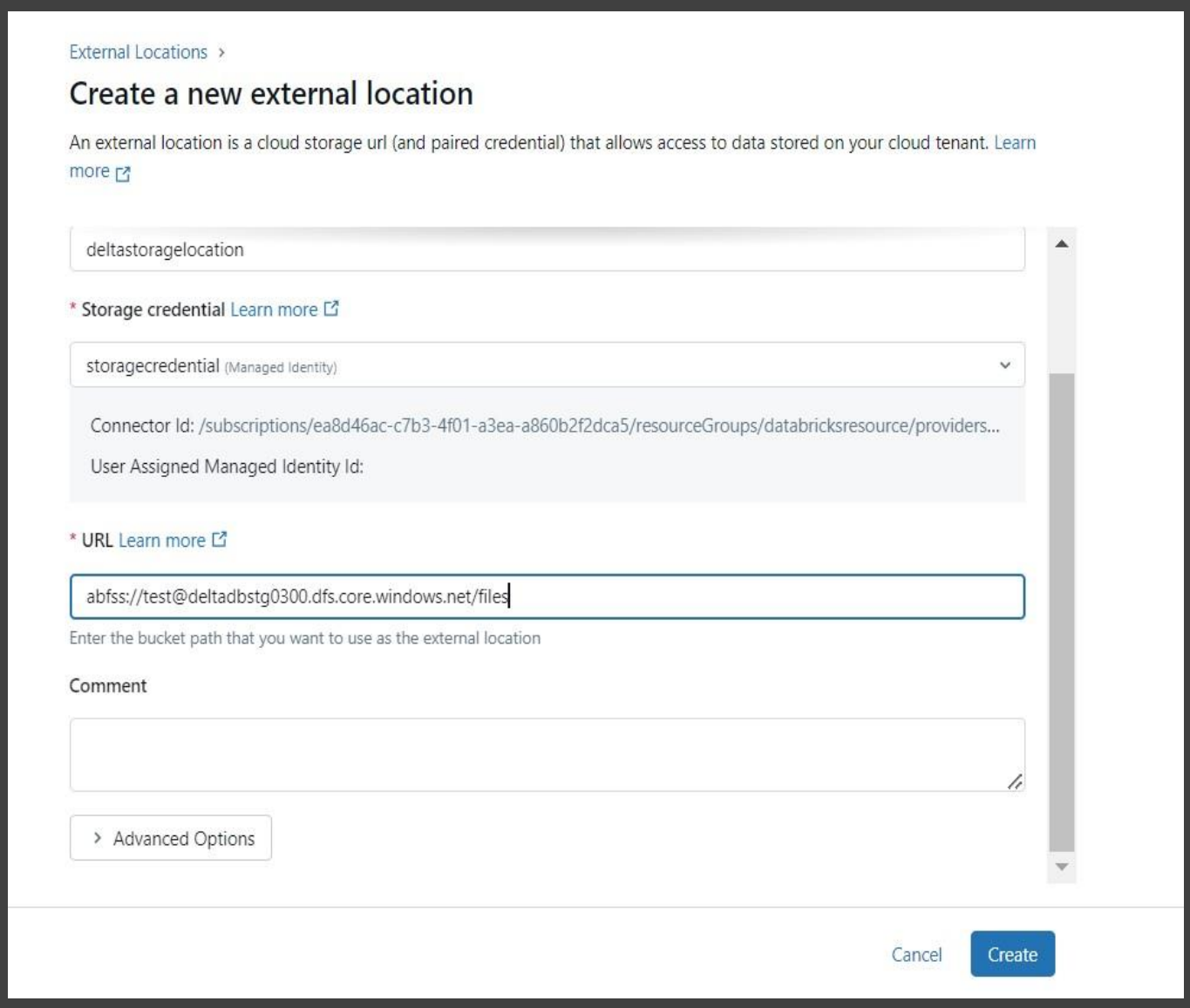
SELECT * FROM `catalog`.schema.table
But we need permission for schema too

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

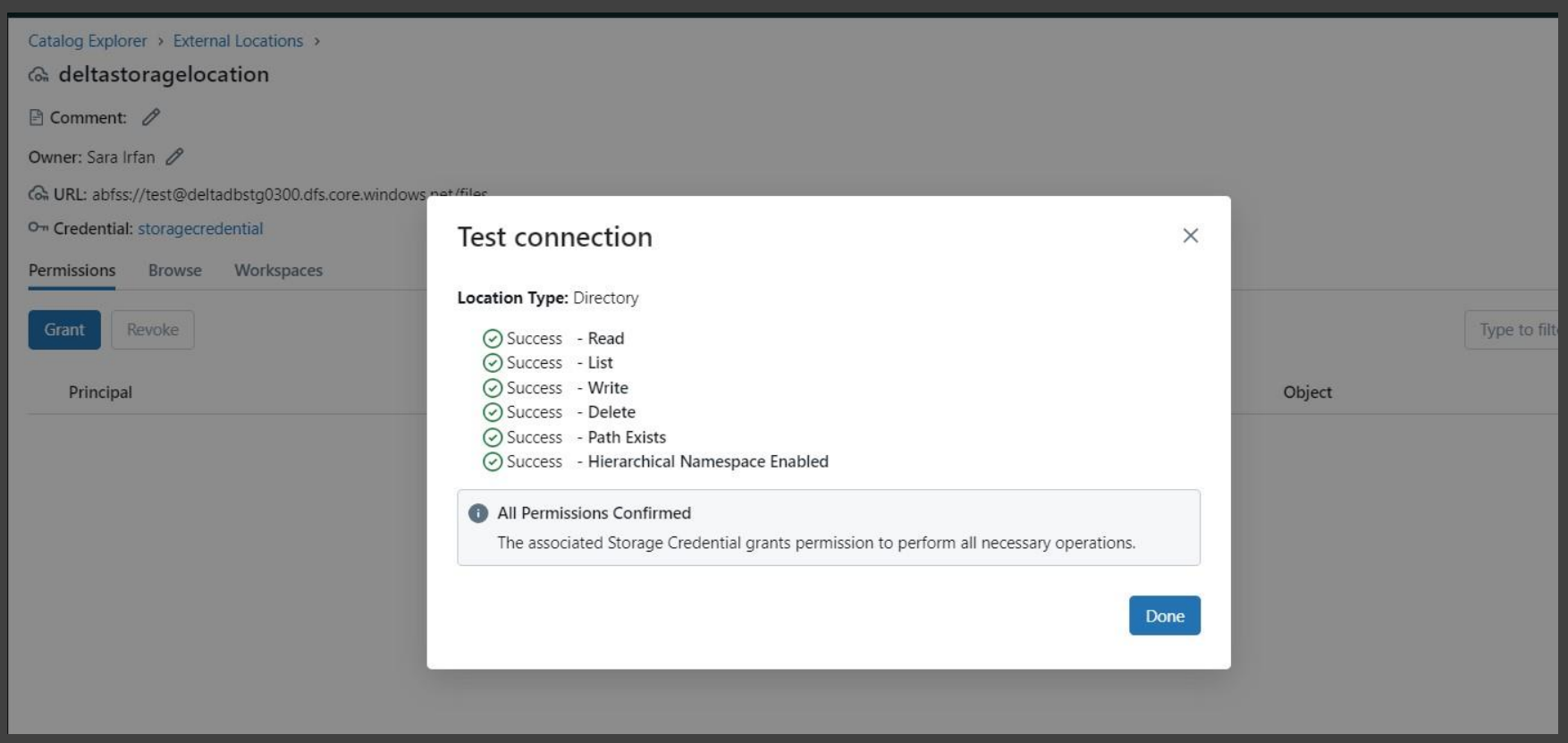
1. **Creating storage credential and external location:**
   - Performed by a workspace admin through the **Anaya Portal**.
   - Go to the catalog section to find external data.
   - Note: External data options might be missing; this is managed at the metastore level.
   - Need **Metastore admin** or 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.



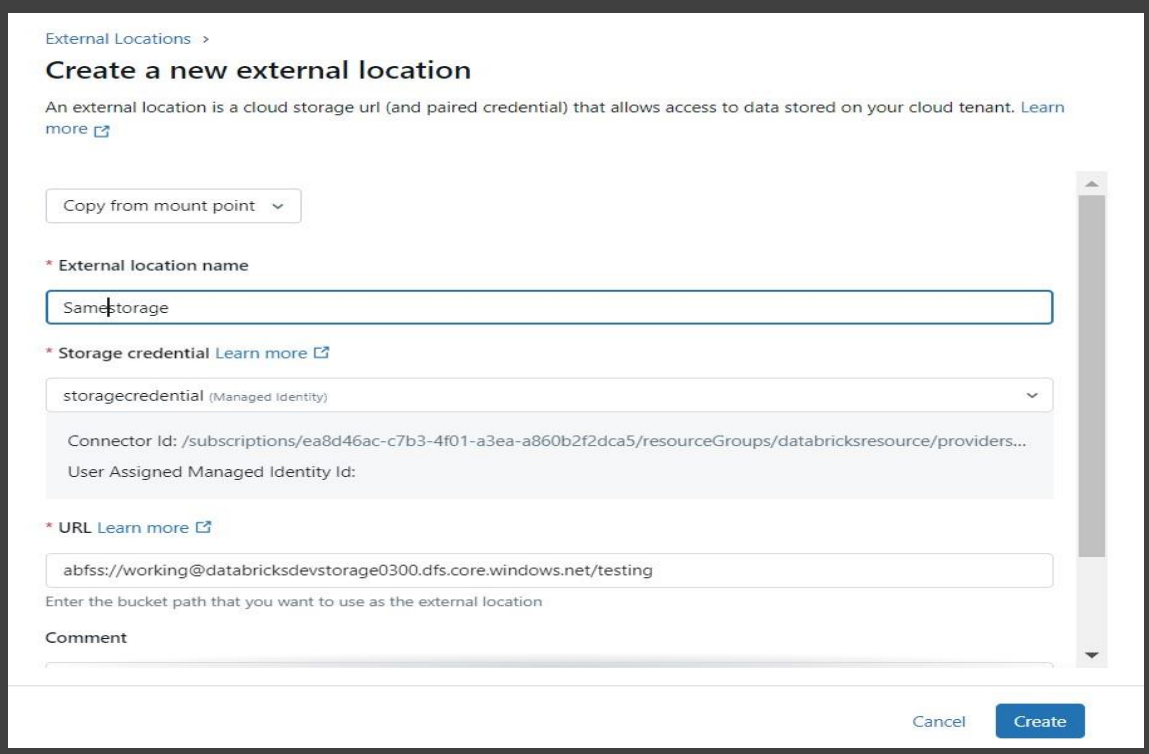
Creating an external location:



Testing a connection:



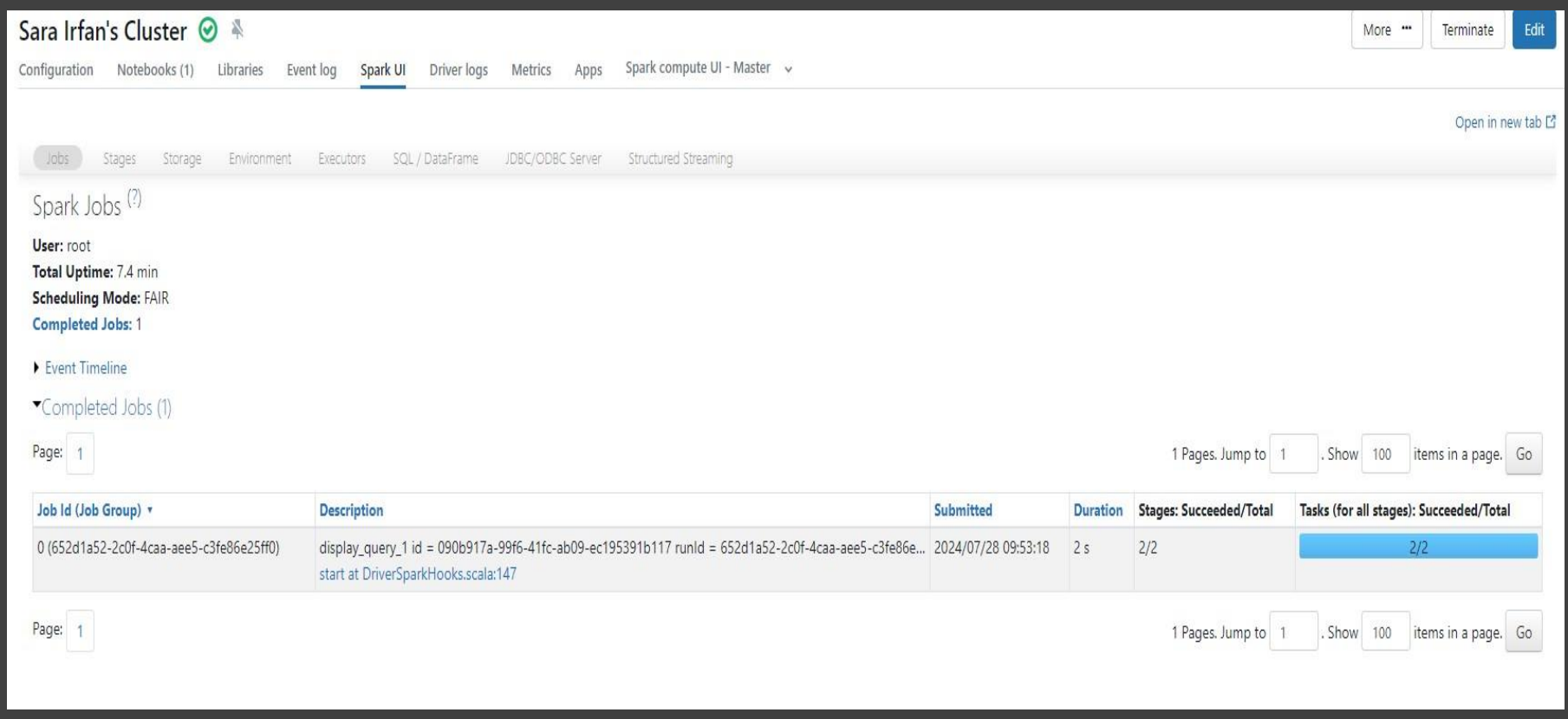
Creating another external location:



Q. How to handle Spark Structured Streaming?

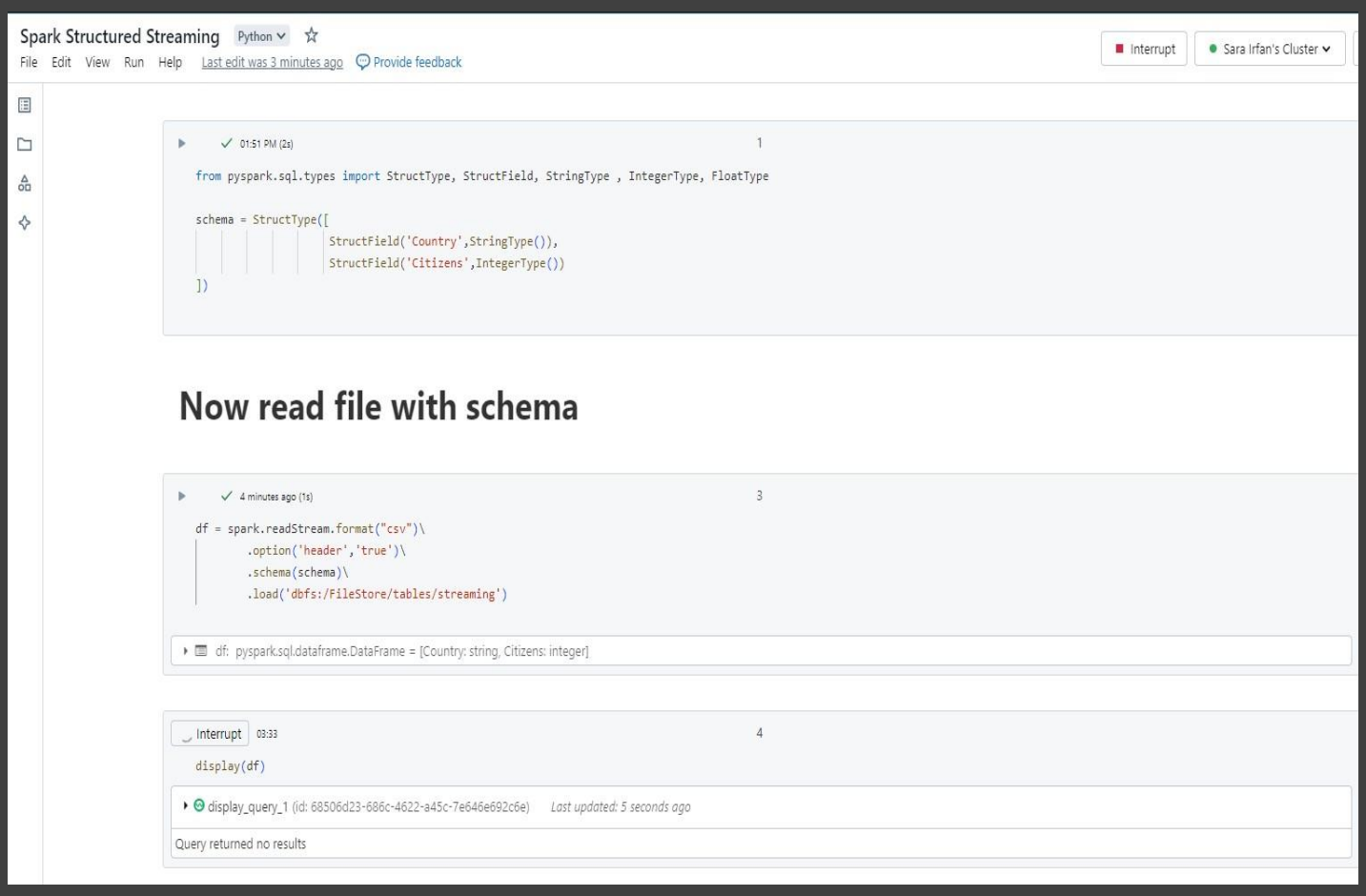
1. What is Streaming Data?
   - Streaming data refers to data that is continuously generated and never completely stops because it keeps coming in.
2. Conceptualization:
   - Understand how structured streaming works:
- Source: Imagine an IoT device collecting car details.
- Data: Data flows from thousands 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."

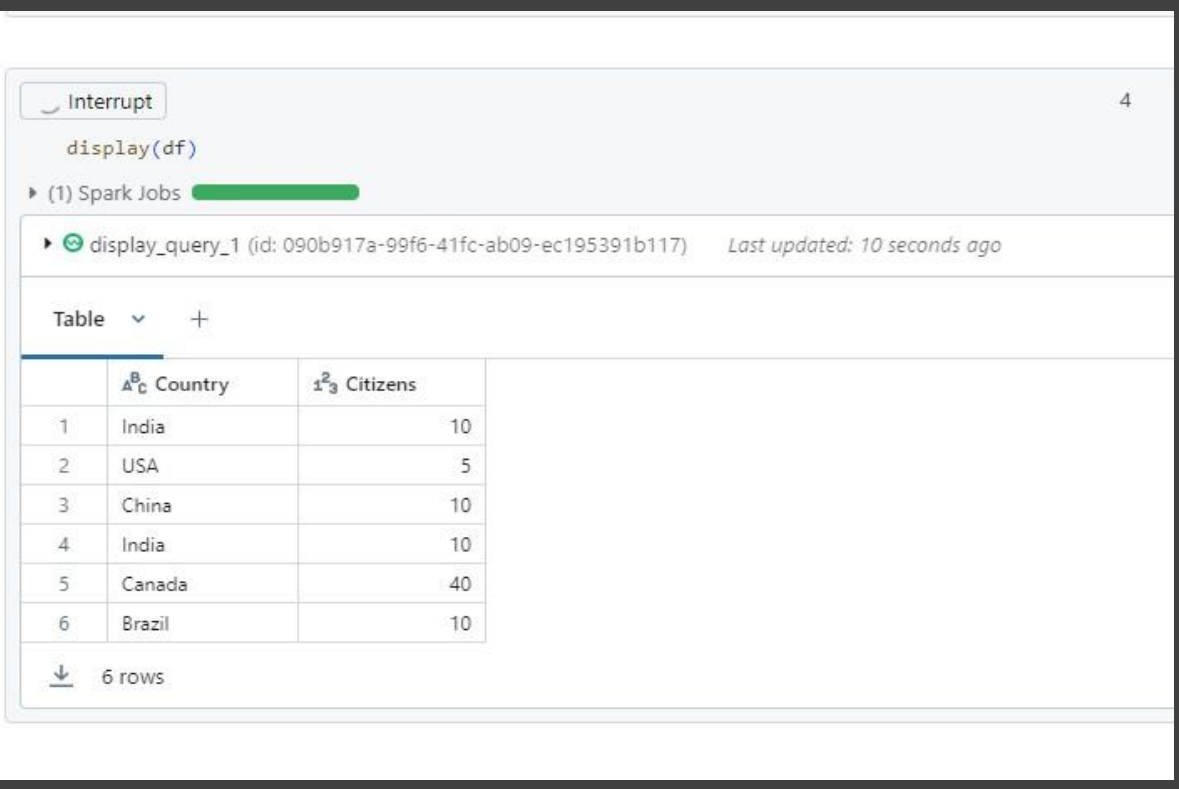


5. **File Source Setup:**
    - **File Upload:**
- **Click on "Data," then "DBFS," and then "File Store."**
- **Create a new folder named "streaming."**
- **Upload files like "countries1.csv."**
    - **countries1.csv:**
- **This file contains two columns: Country Name and Number of Citizens.**

6. **Data Reading:**
    - **Read Stream:**
- **Copy the path and use spark.readStream to read the data.**



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



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

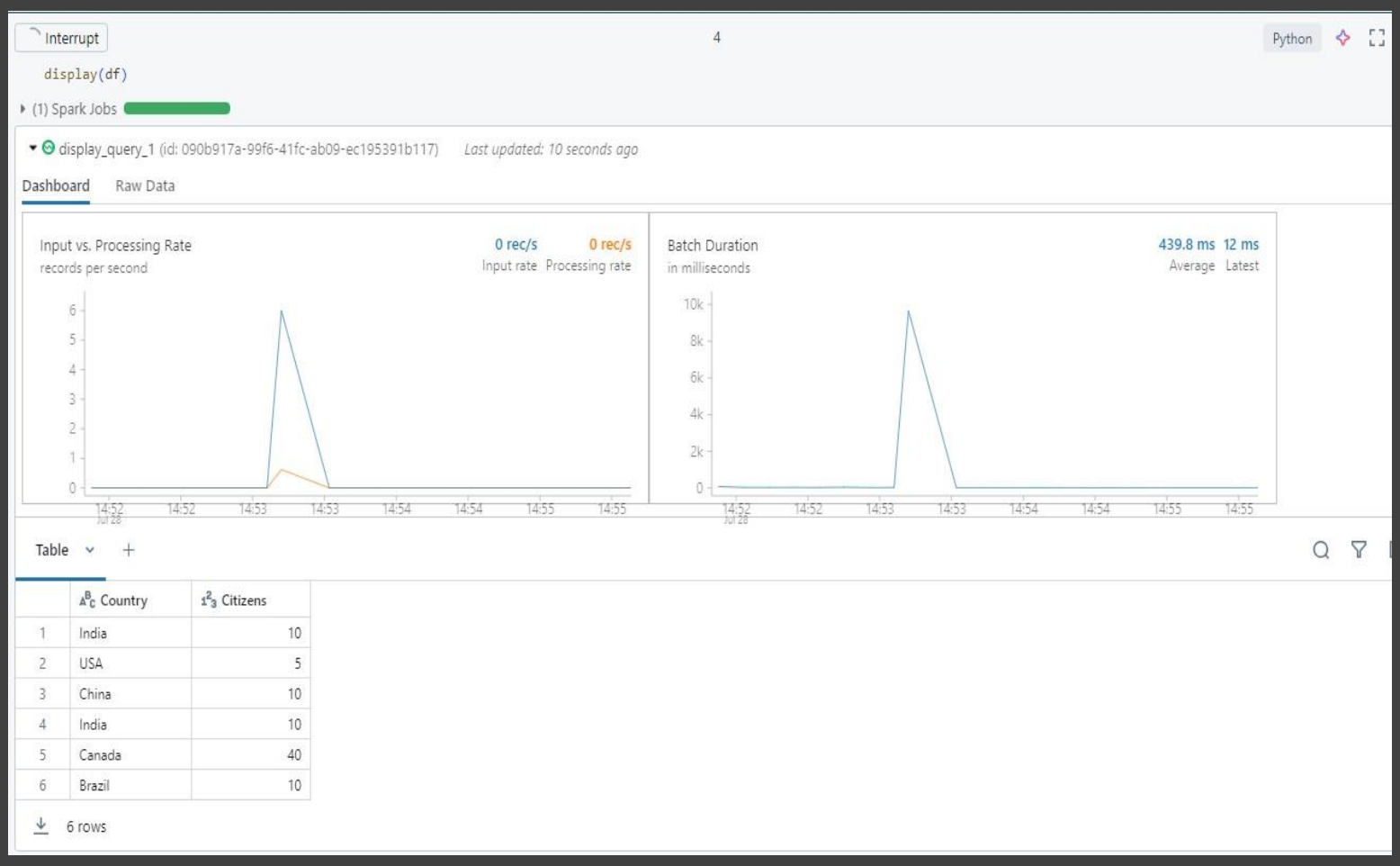
7. Streaming Query Monitoring:

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

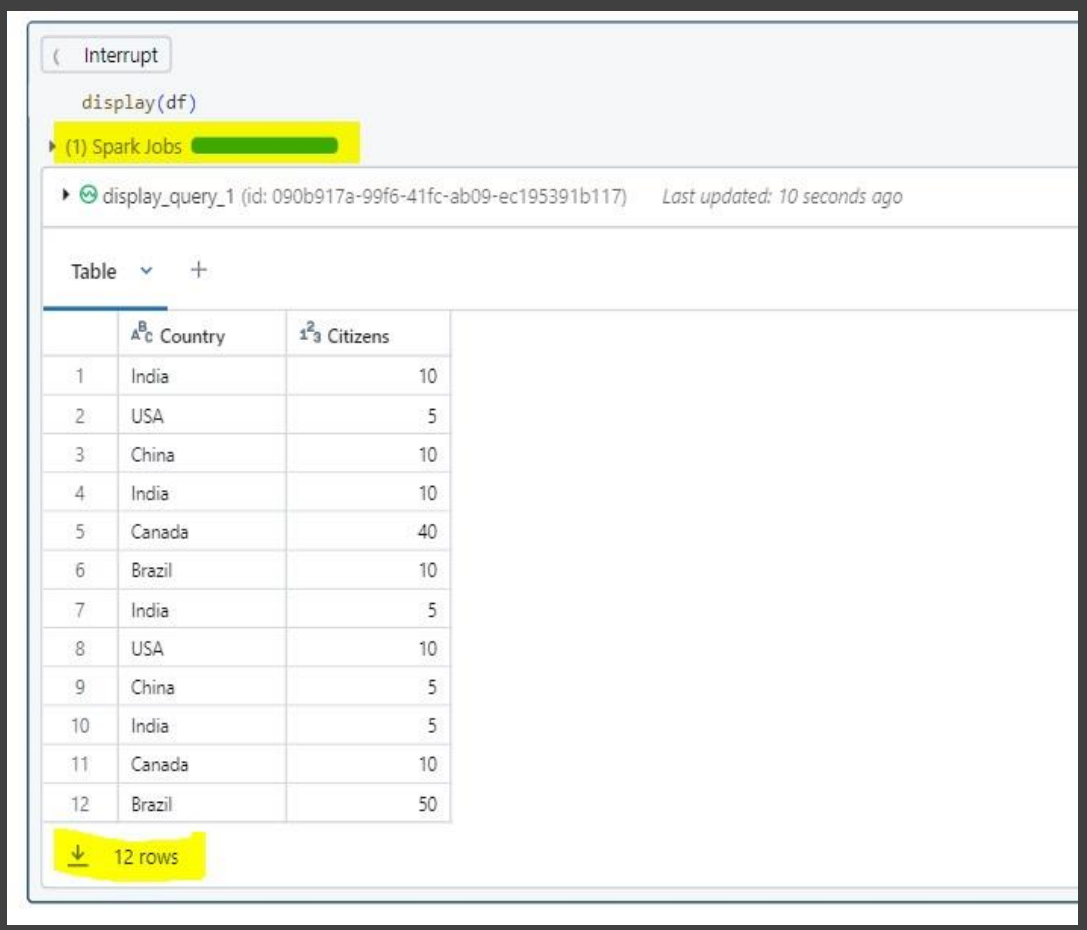
8. Streaming Query Acting as a Watcher:

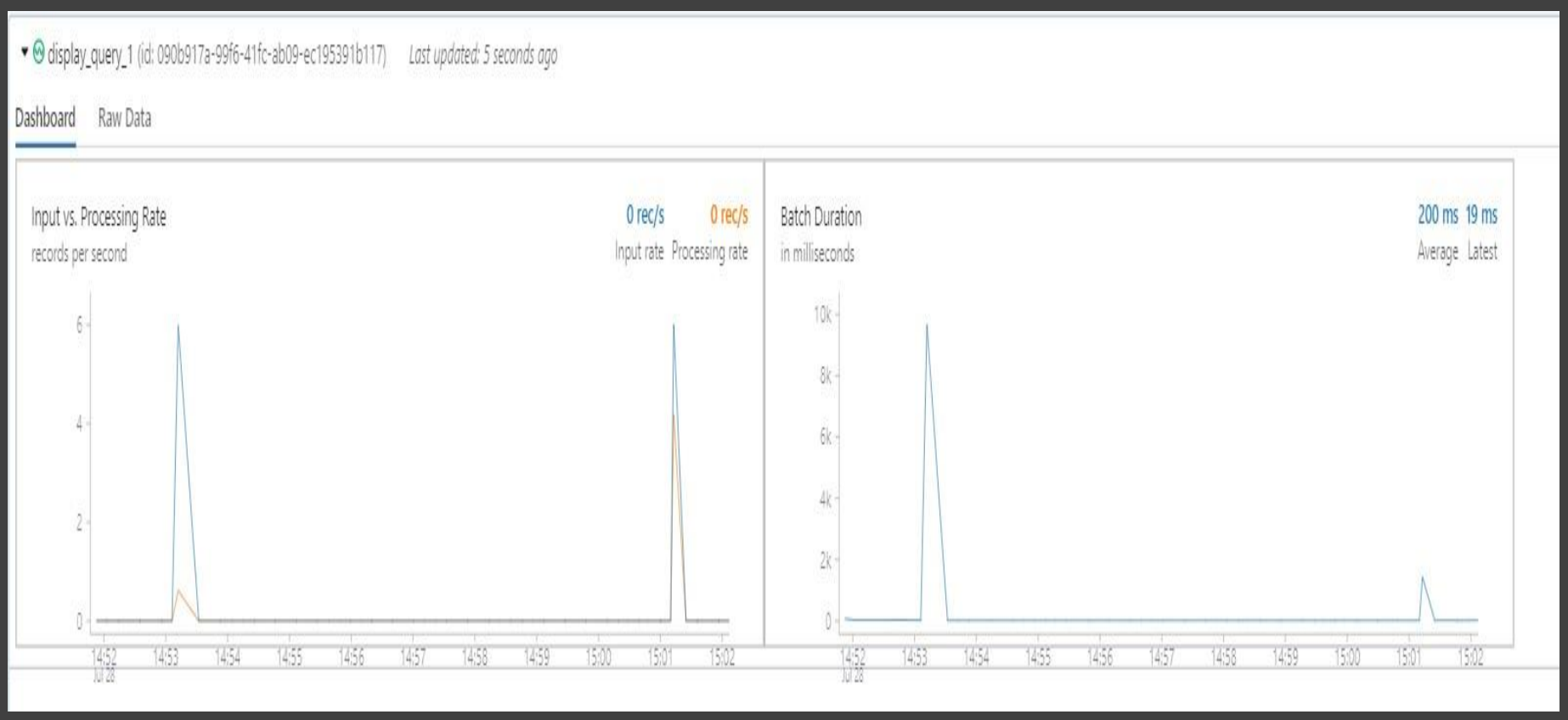
- **The streaming query acts as a watcher, continuously checking for new data.**
- **The streaming query runs continuously in the background.**

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

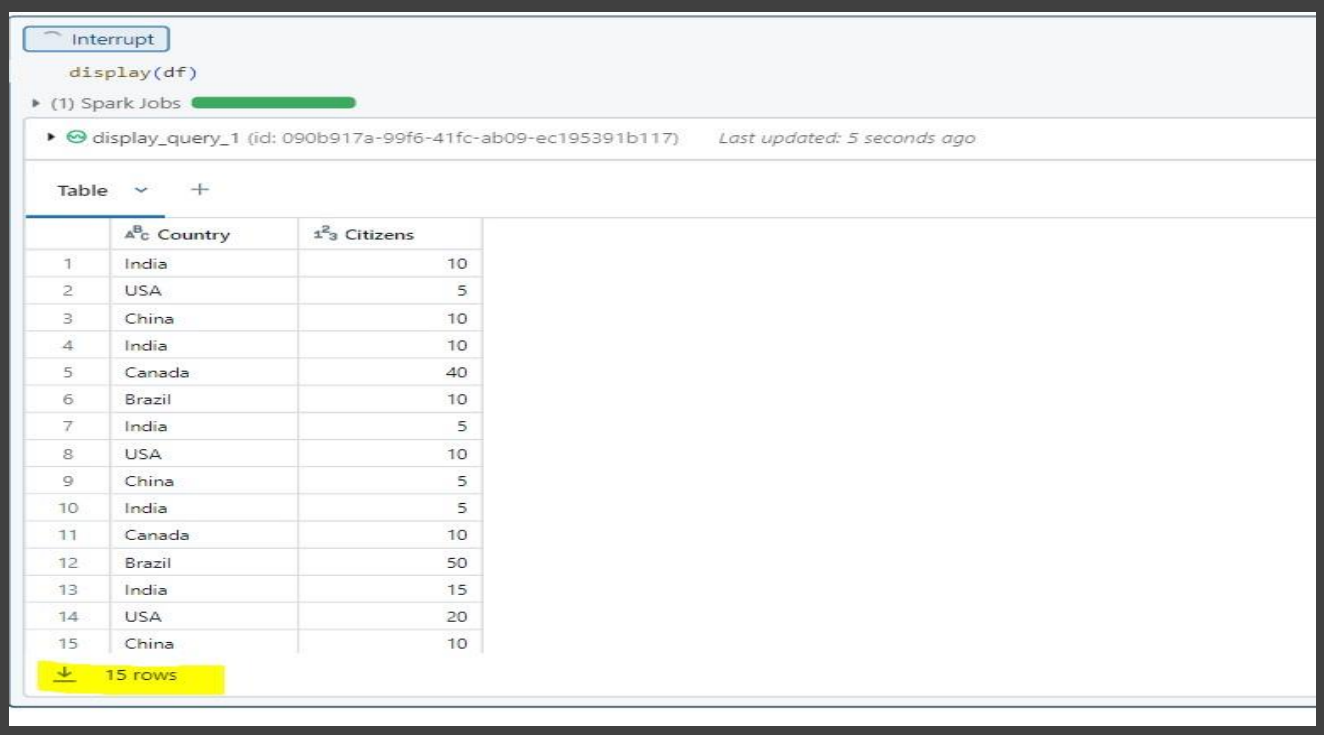


Adding more data in the container of data lake storage, the number of rows are increased





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



Check out the spark jobs from databricks workspace enviornment

Spark Jobs (?)

User: root
Total Uptime: 14 min
Scheduling Mode: FAIR
Completed Jobs: 3

▸ Event Timeline

▾Completed Jobs (3)

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

Sara Irfan's Cluster

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

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

Streaming Query

▾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

Page: 1

1 Pages. Jump to 1 . Show 100 items in a page. Go

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

1 Pages. Jump to 1 . Show 100 items in a page. Go

Writing the data with writeStream

We need specific schema to write this

Interrupt

```
1  WriteStream = ( df.writeStream
2          .option('checkpointLocation',f'{source_dir}/AppendCheckpoint')
3          .outputMode("append")
4          .queryName('AppendQuery')
5          .toTable("stream.AppendTable"))
```

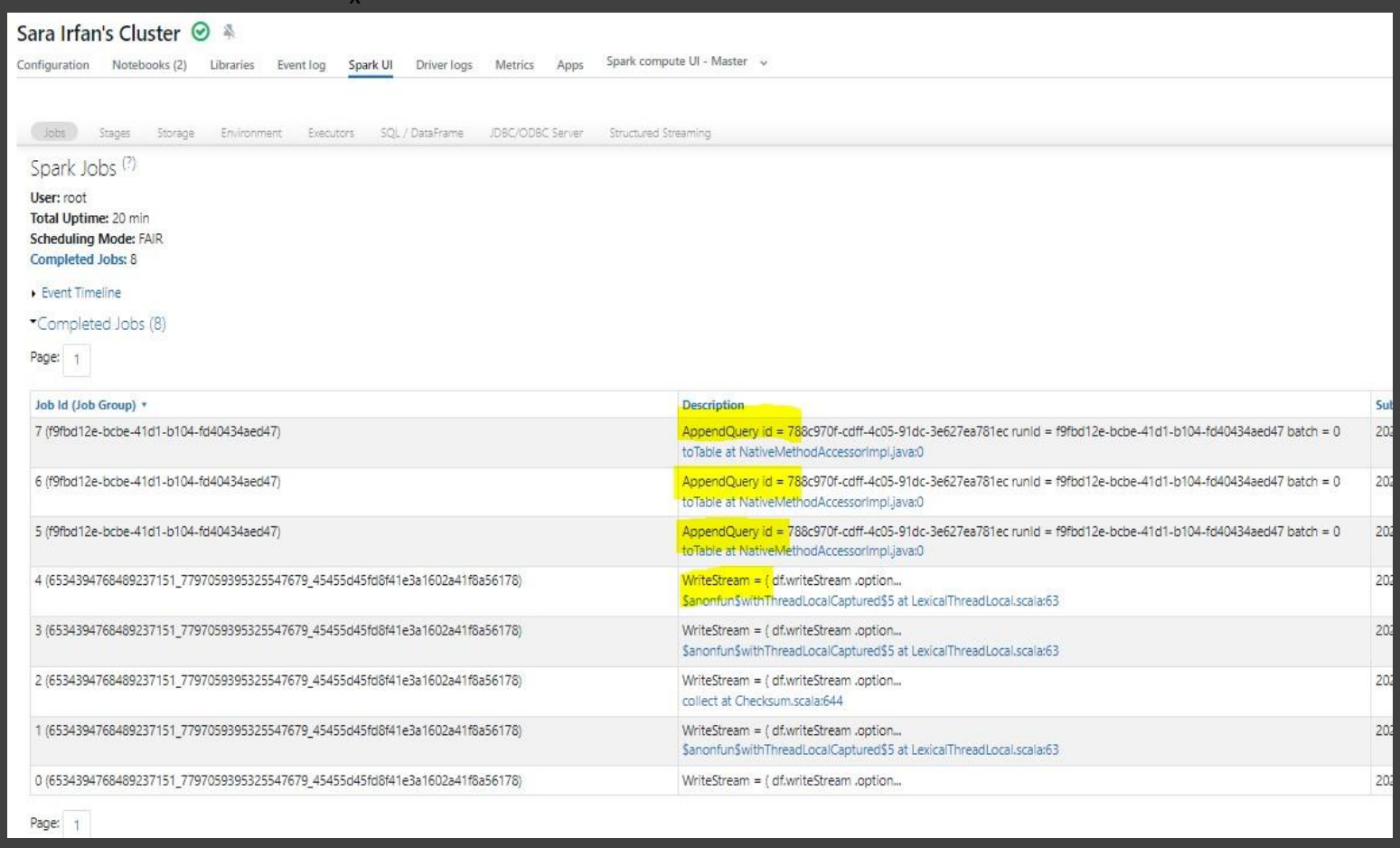
(2) Spark Jobs

AppendQuery (id: 788c970f-cdff-4c05-91dc-3e627ea781ec) *Last updated: 5 seconds ago*

Dashboard Raw Data

8

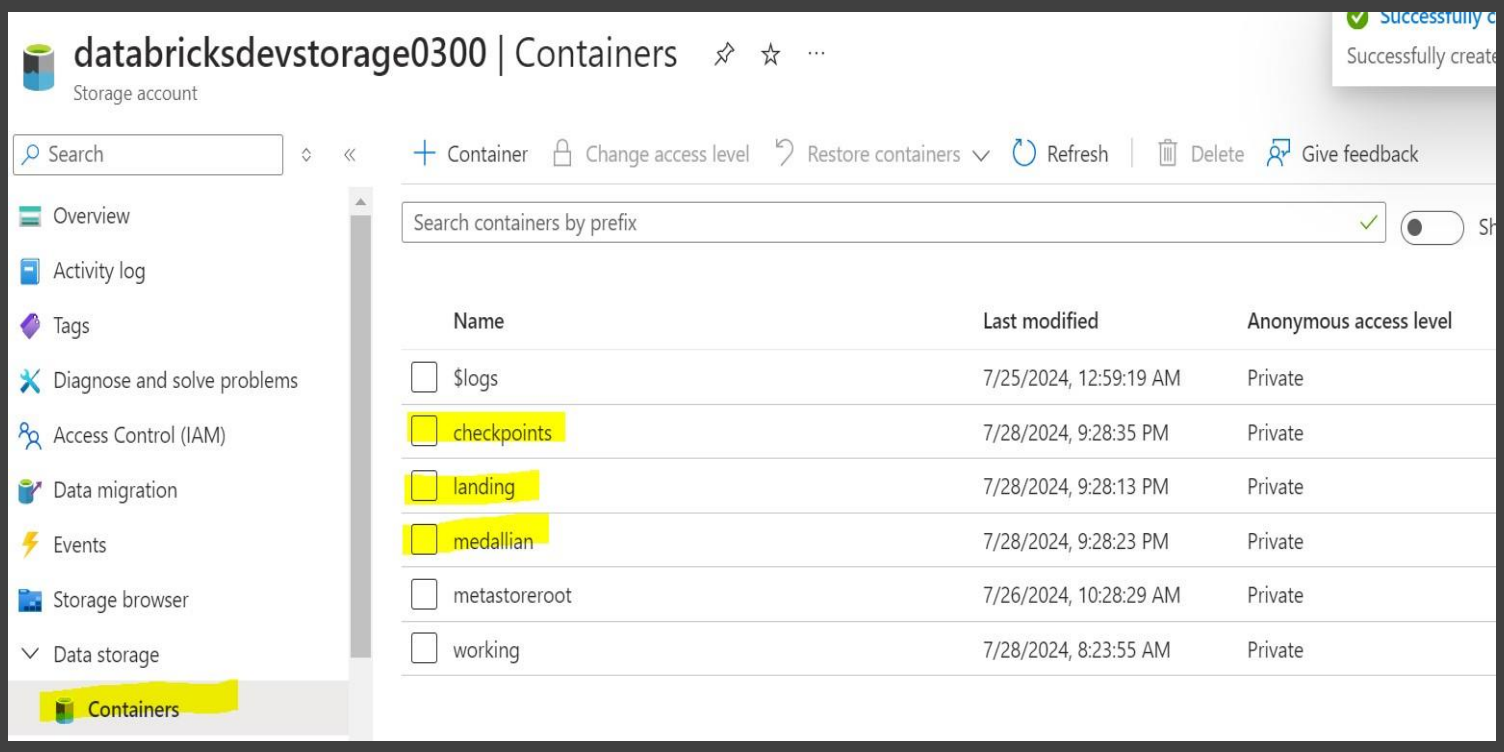
x



Sara Irfan (sara.data.insights@gmail.com)

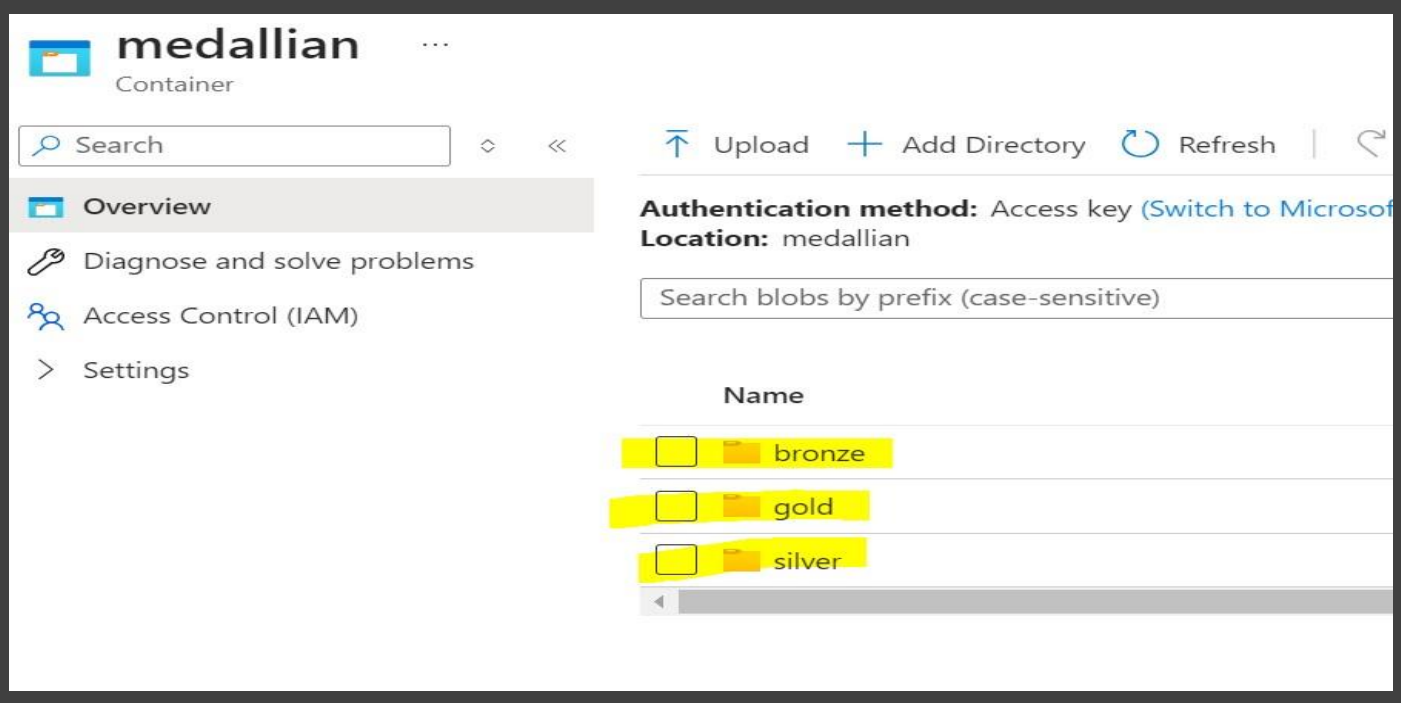
Q. What are the steps for Data Engineering Project in Azure Databricks using three layer architecture?

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.



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.



Project Workflow

1. **Create Schemas**
   - **Bronze Schema**
   - **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

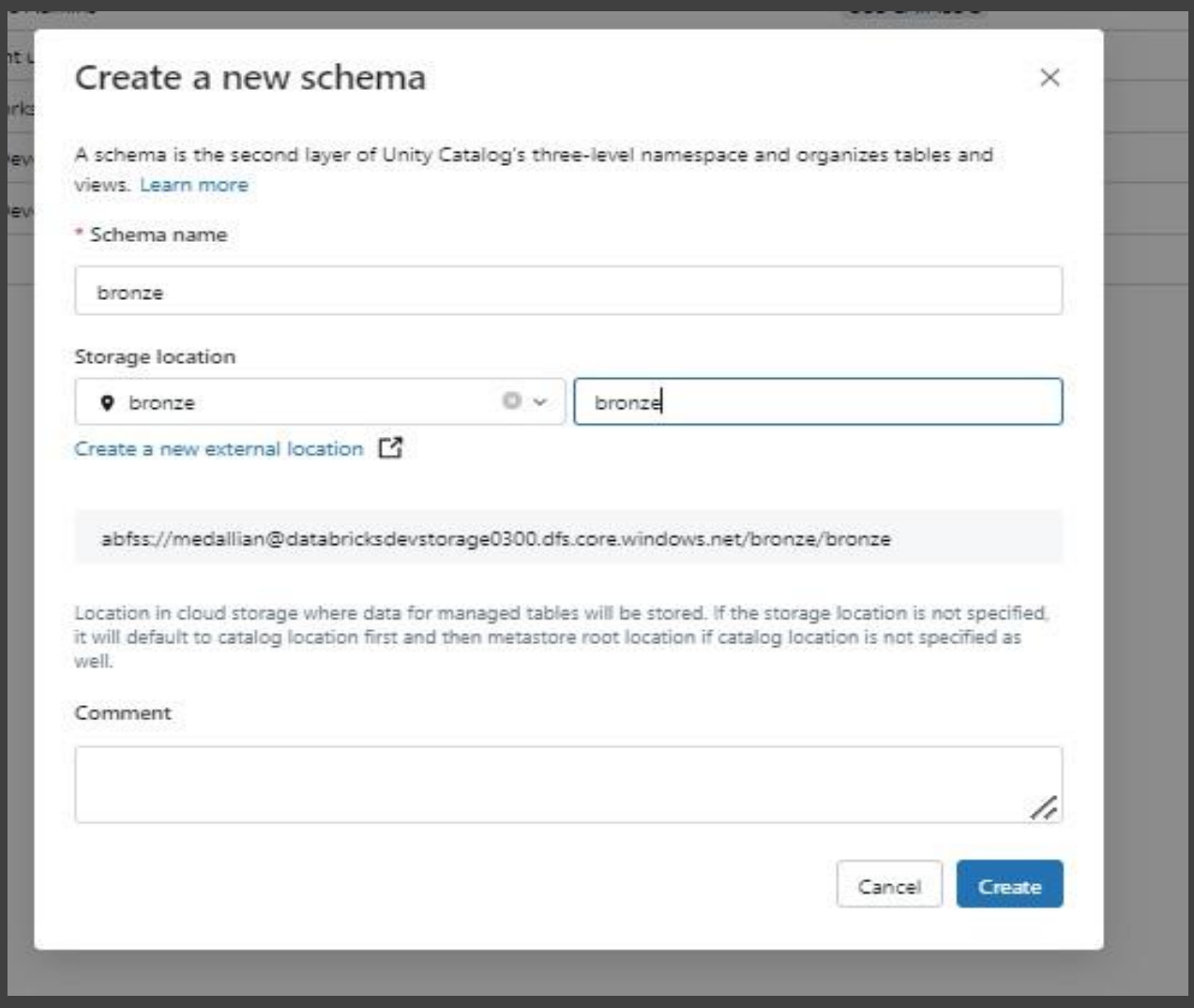
Ensure to test the connection for each external location to confirm the correct setup.

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:

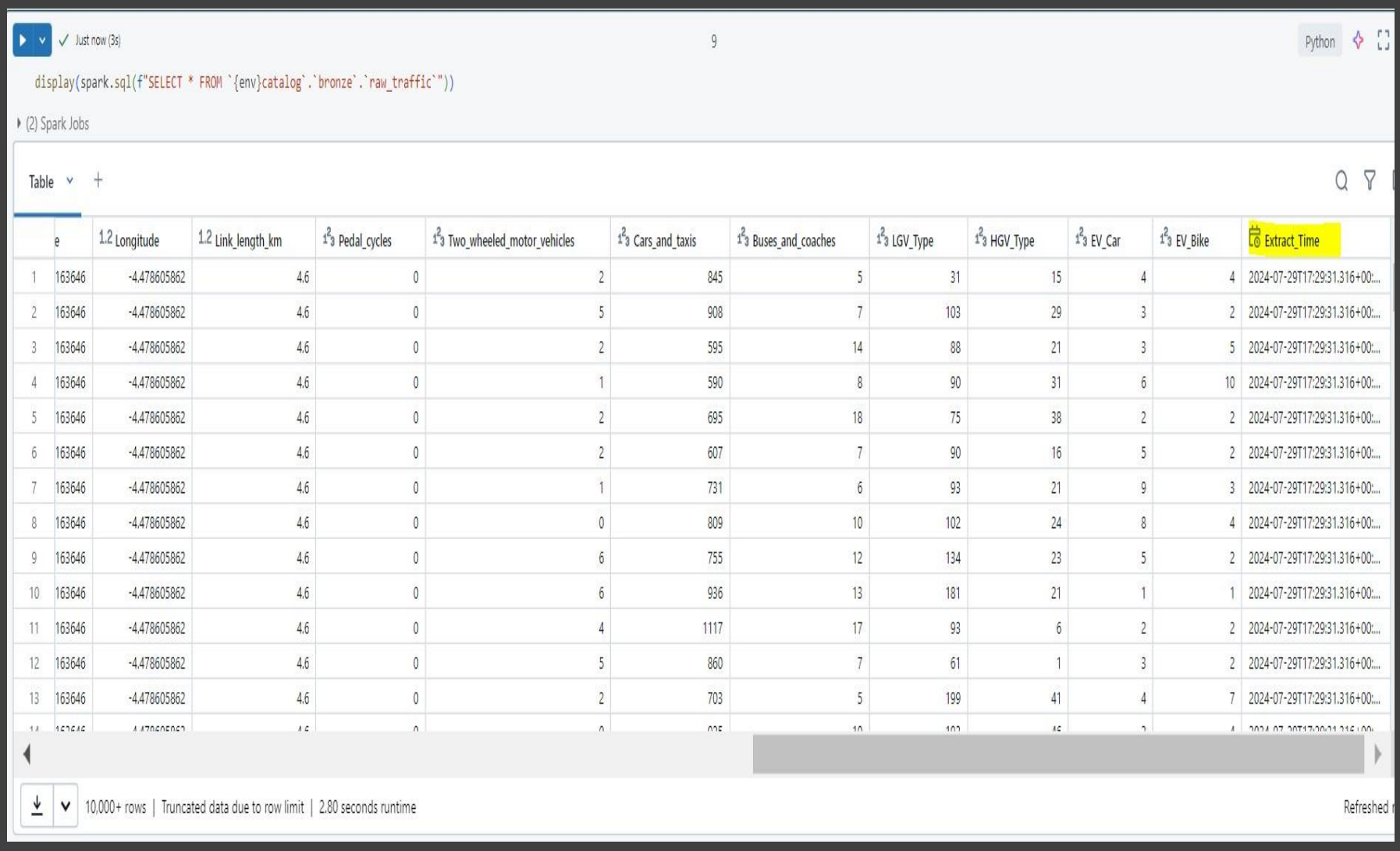
Raw Roads Table

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.



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

(2) Spark Jobs

Table

	e	Longitude	Link_length_km	Pedal_cycles	Two_wheeled_motor_vehicles	Cars_and_taxis	Buses_and_coaches	LGV_Type	HGV_Type	EV_Car	EV_Bike	Extract_Time
1	163646	-4.478605862	4.6	0	2	845	5	31	15	4	4	2024-07-29T17:29:31.316+00:...
2	163646	-4.478605862	4.6	0	5	908	7	103	29	3	2	2024-07-29T17:29:31.316+00:...
3	163646	-4.478605862	4.6	0	2	595	14	88	21	3	5	2024-07-29T17:29:31.316+00:...
4	163646	-4.478605862	4.6	0	1	590	8	90	31	6	10	2024-07-29T17:29:31.316+00:...
5	163646	-4.478605862	4.6	0	2	695	18	75	38	2	2	2024-07-29T17:29:31.316+00:...
6	163646	-4.478605862	4.6	0	2	607	7	90	16	5	2	2024-07-29T17:29:31.316+00:...
7	163646	-4.478605862	4.6	0	1	731	6	93	21	9	3	2024-07-29T17:29:31.316+00:...
8	163646	-4.478605862	4.6	0	0	809	10	102	24	8	4	2024-07-29T17:29:31.316+00:...
9	163646	-4.478605862	4.6	0	6	755	12	134	23	5	2	2024-07-29T17:29:31.316+00:...
10	163646	-4.478605862	4.6	0	6	936	13	181	21	1	1	2024-07-29T17:29:31.316+00:...
11	163646	-4.478605862	4.6	0	4	1117	17	93	6	2	2	2024-07-29T17:29:31.316+00:...
12	163646	-4.478605862	4.6	0	5	860	7	61	1	3	2	2024-07-29T17:29:31.316+00:...
13	163646	-4.478605862	4.6	0	2	703	5	199	41	4	7	2024-07-29T17:29:31.316+00:...

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:

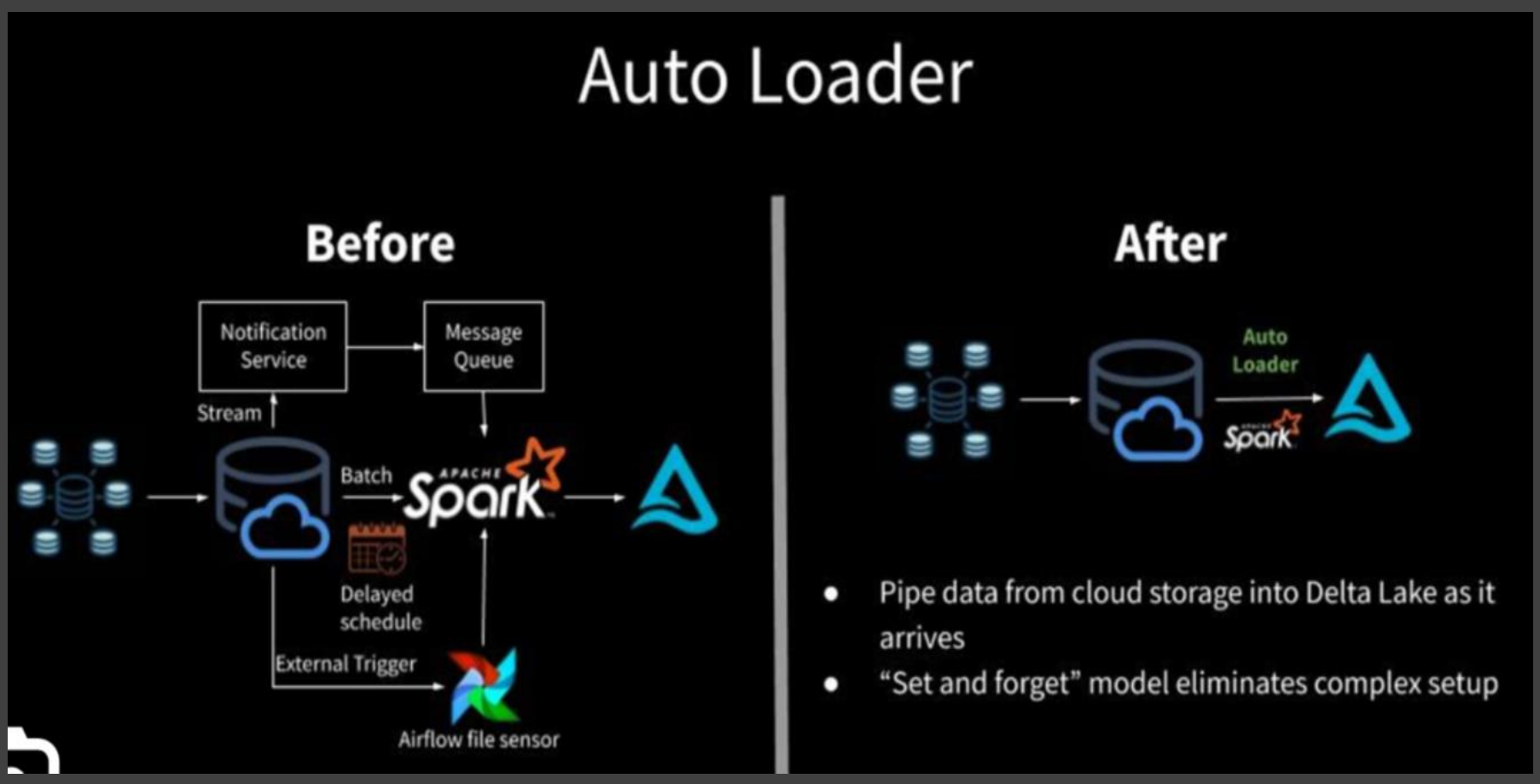
- Example: If 10,000 files are ingested today and 20,000 more files are ingested tomorrow, only the additional 20,000 files should be processed.

Table Structure:

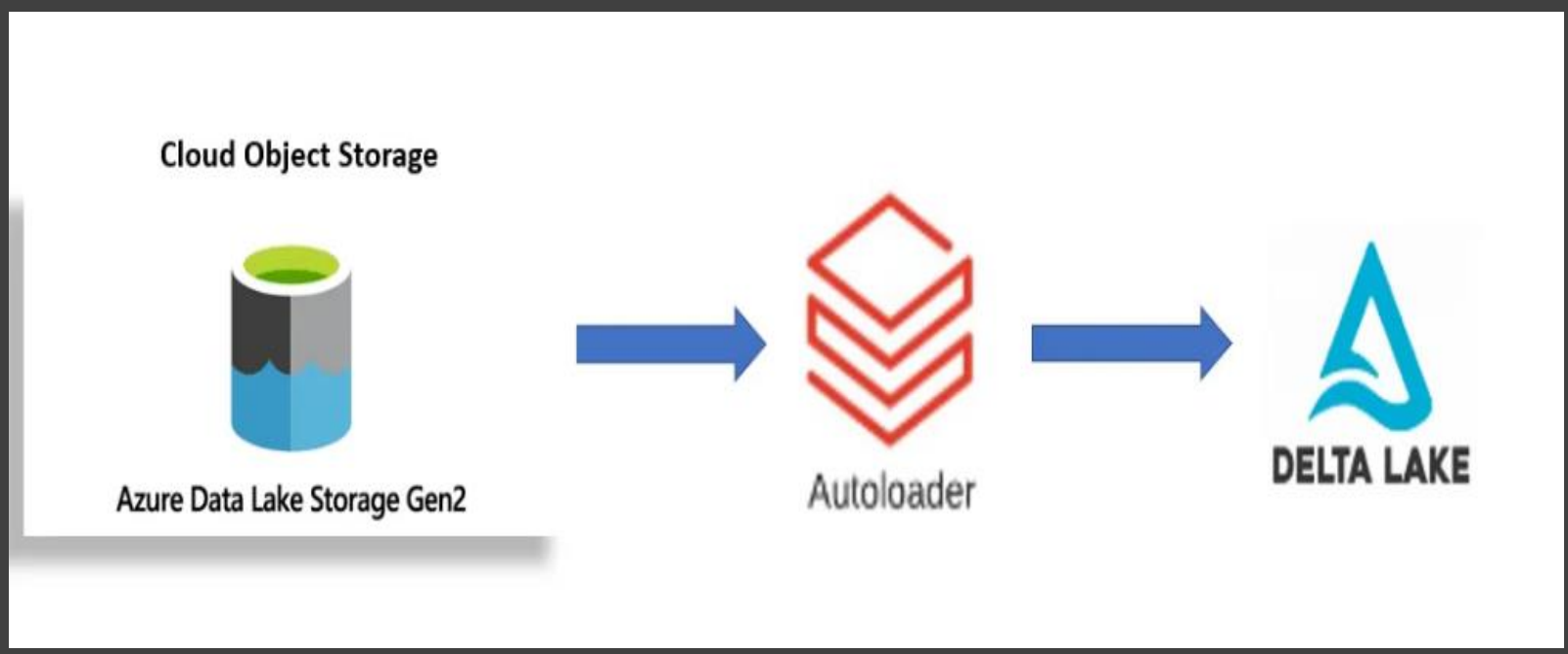
- Data should be loaded as-is with an additional column for extract_time.
- No extra fields or transformations are applied.

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

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@databricksdevstorage0300.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

Test connection

Location Type: Directory

- Success - Read
- Success - List
- Success - Write
- Success - Delete
- Success - Path Exists
- Success - Hierarchical Namespace Enabled

All Permissions Confirmed

The associated Storage Credential grants permission to perform all necessary operations.

Done

Catalog Explorer ›

External Data

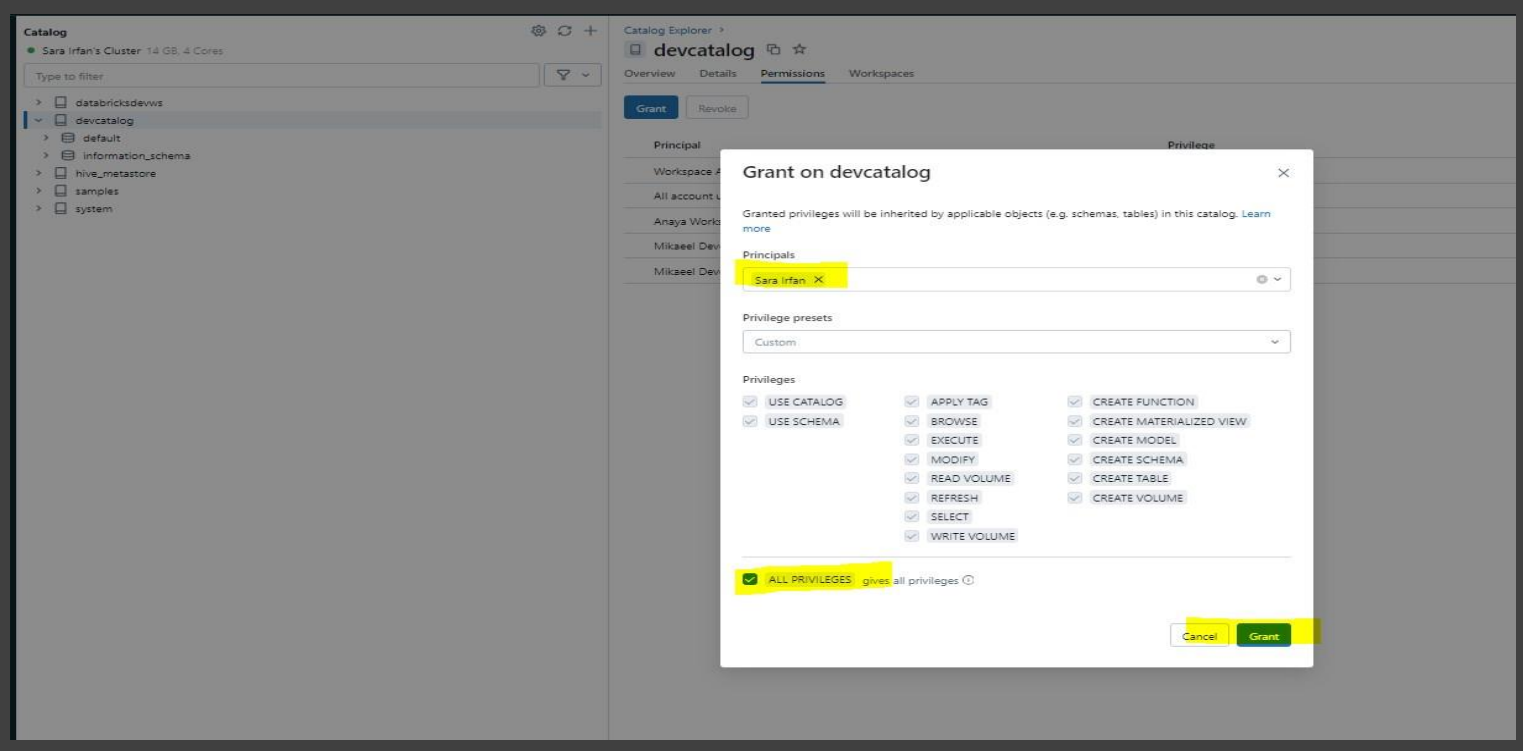
External Locations | Storage Credentials

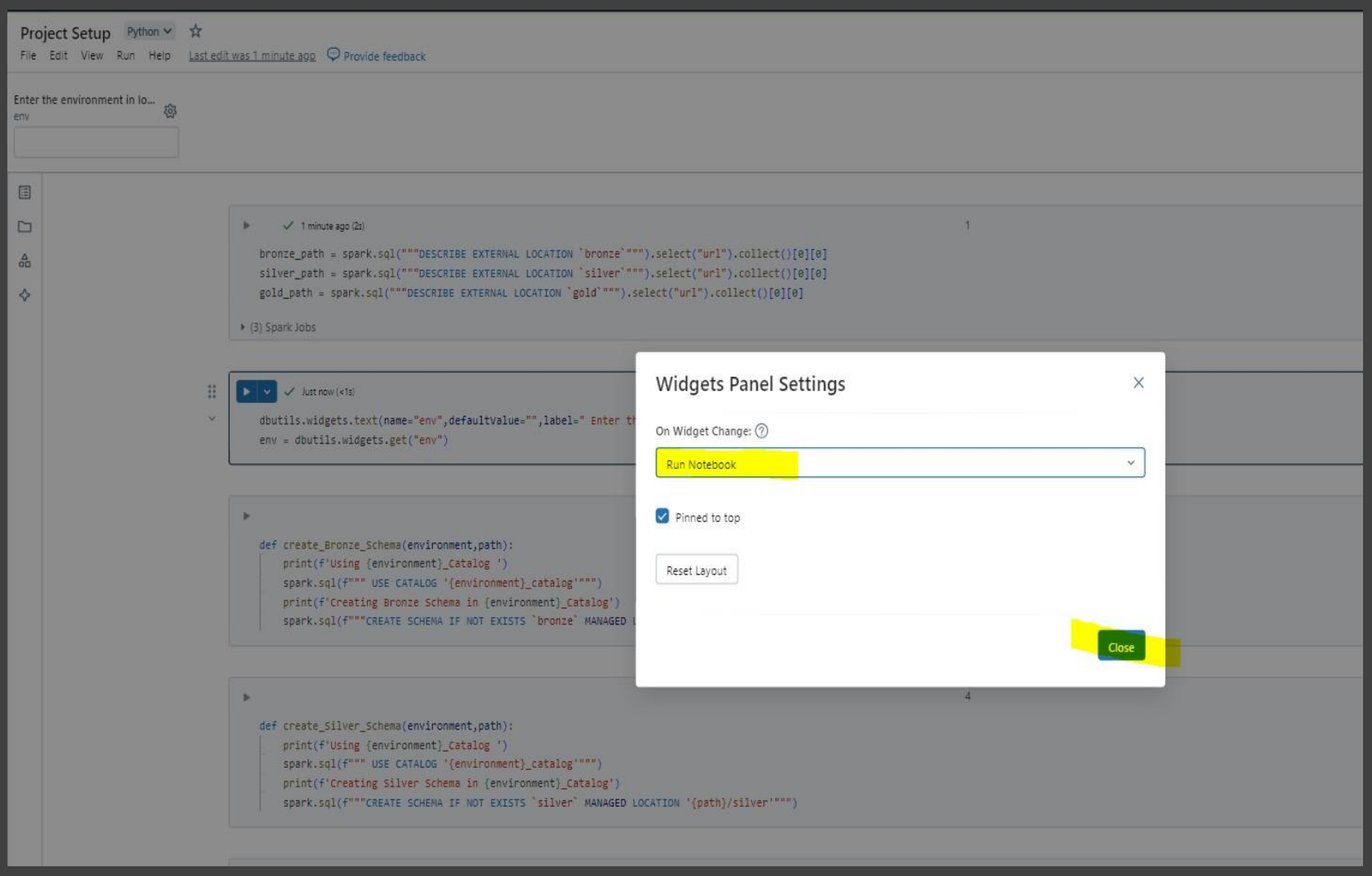
Filter locations — 8 locations

Name	Credential	URL
bronze	storagecredential	abfss://medallian@databricksdevstorage0300.dfs.core.windows.net/bronze
checkpoints	storagecredential	abfss://checkpoints@databricksdevstorage0300.dfs.core.windows.net/
databricksdevws	databricksdevws	abfss://unity-catalog-storage@dbstoragepixmslryuphzk.dfs.core.windows.net/276878930788483
deltastoragelocation	storagecredential	abfss://test@deltadbstg0300.dfs.core.windows.net/files
gold	storagecredential	abfss://medallian@databricksdevstorage0300.dfs.core.windows.net/gold
landing	storagecredential	abfss://landing@databricksdevstorage0300.dfs.core.windows.net/
samestorage	storagecredential	abfss://working@databricksdevstorage0300.dfs.core.windows.net/testing
silver	storagecredential	abfss://medallian@databricksdevstorage0300.dfs.core.windows.net/silver

Q. How to ingest data into the brozne 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.**

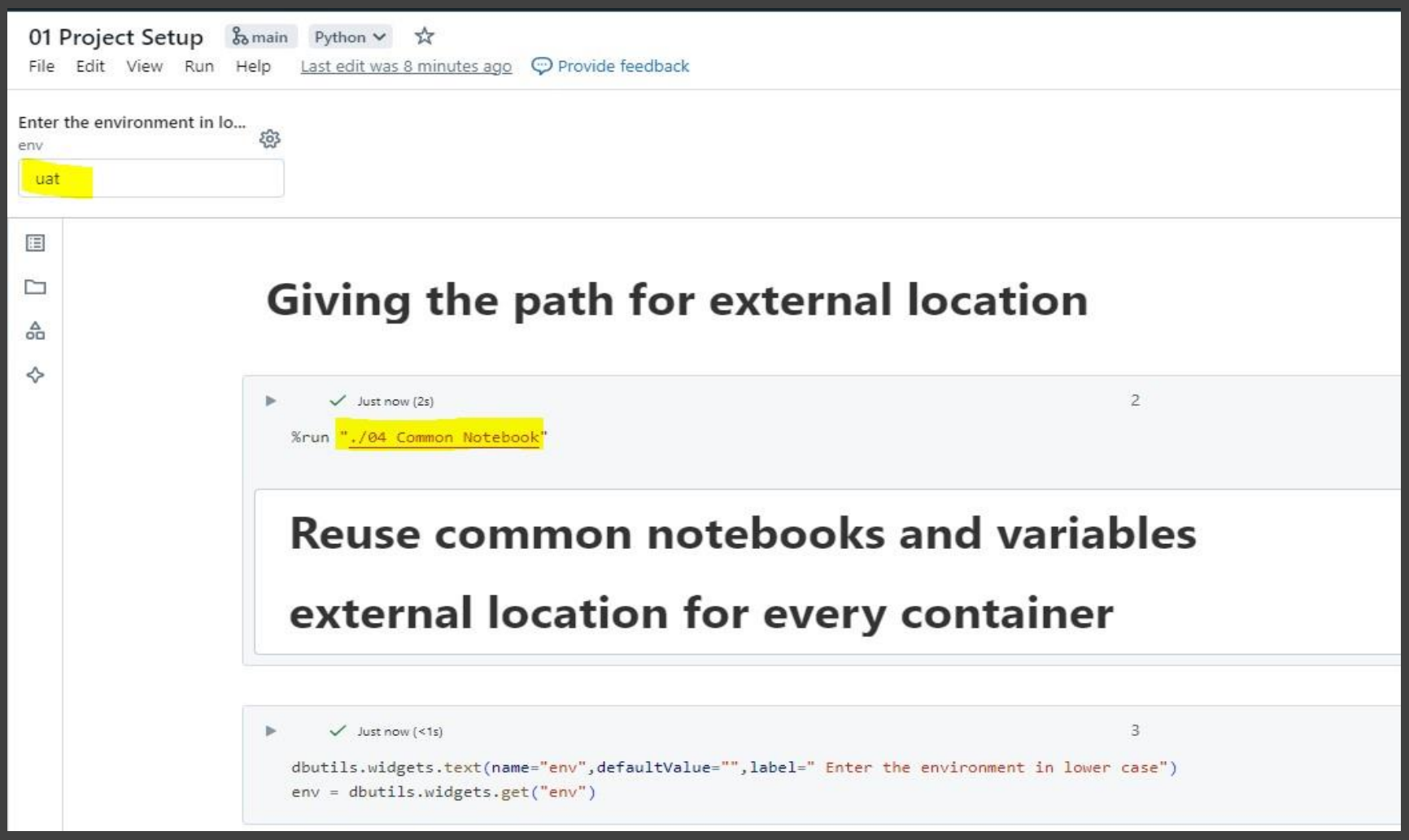
```
def extract_url(df):
    # Example function for URL extraction
    return df.withColumn('extracted_url', extract_from_column(df['column_name']))

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.

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.

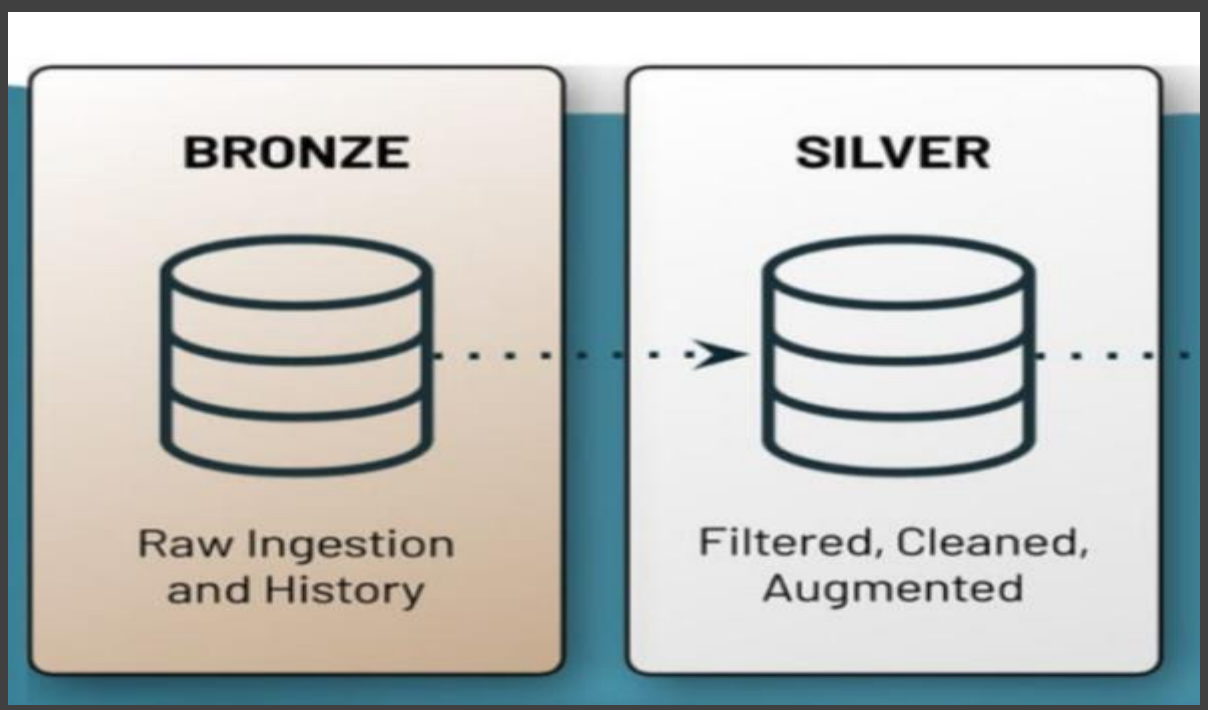
Q. How to transform Data from Bronze to Silver Layer?

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")
```

Waiting

4

```
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'""")
```

Waiting

5

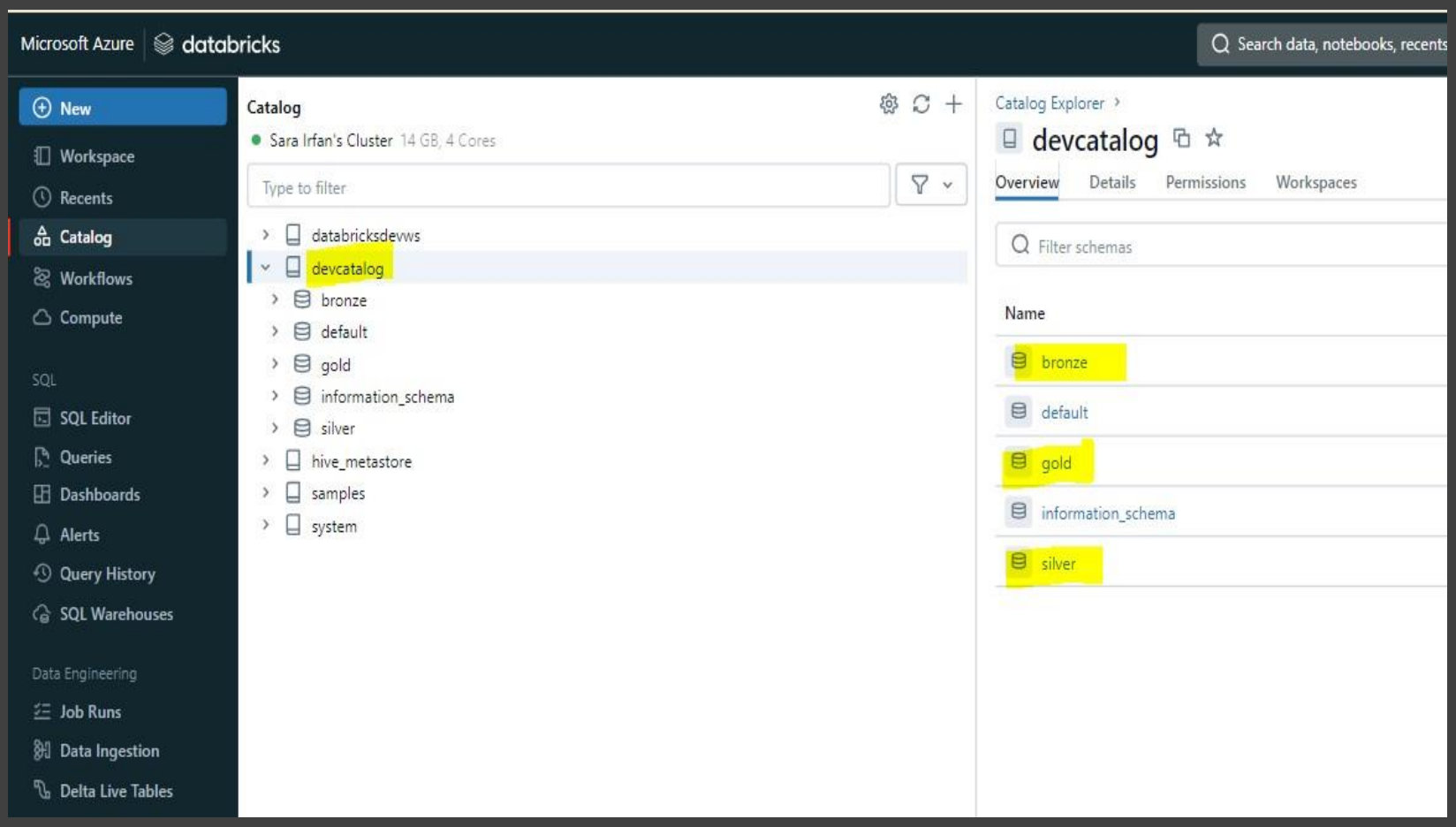
```
def create_Gold_Schema(environment,path):
    print(f'Using {environment}Catalog ')
    spark.sql(f""" USE CATALOG '{environment}catalog'""")
    print(f'Creating Gold Schema in {environment}Catalog')
    spark.sql(f"""CREATE SCHEMA IF NOT EXISTS `gold` MANAGED LOCATION '{path}/gold'""")
```

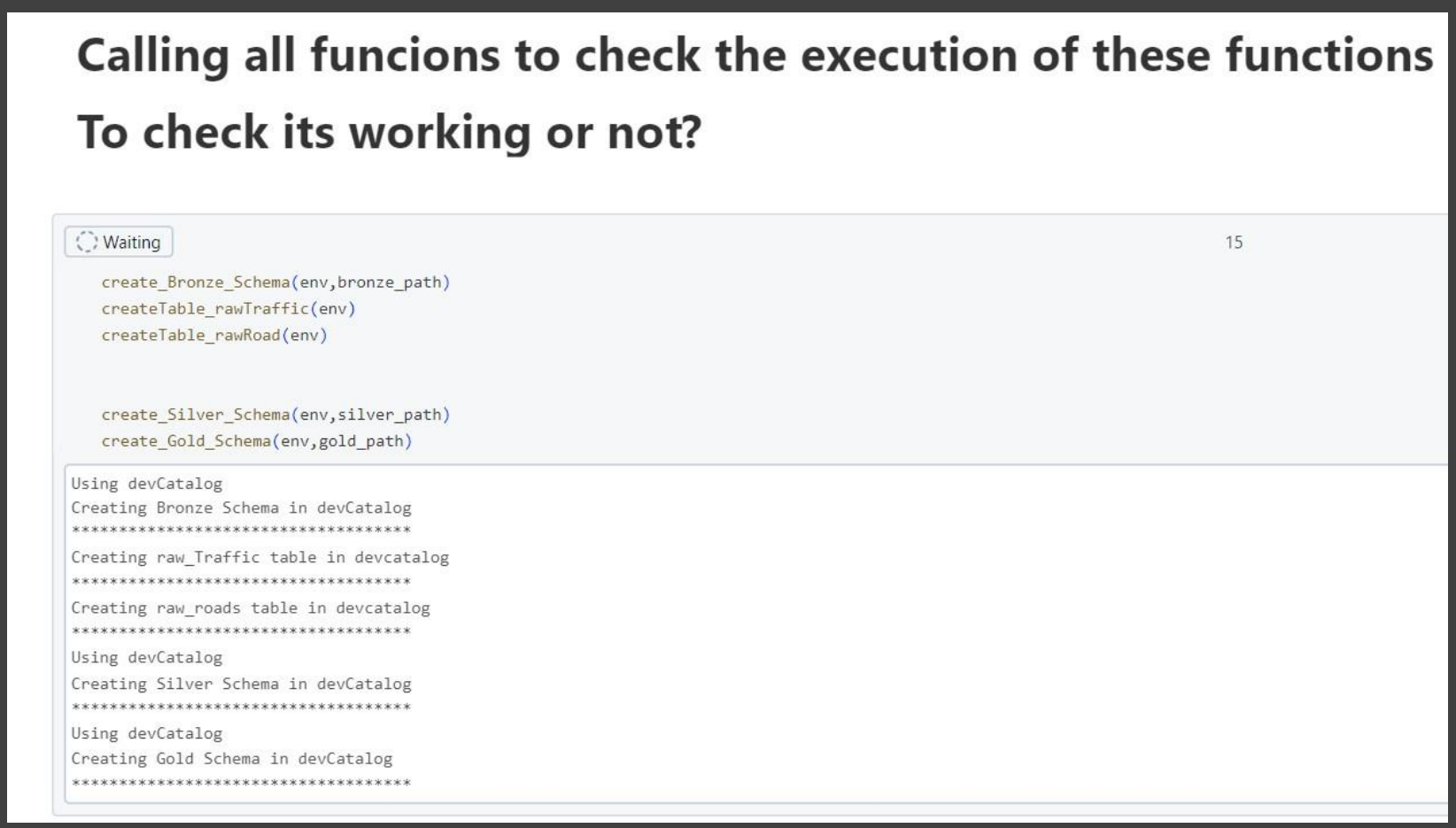
Waiting

6

```
create_Bronze_Schema(env,bronze_path)
```

```
Using devCatalog
Creating Bronze Schema in devCatalog
```





Calling all funcions to check the execution of these functions

To check its working or not?

Waiting 15

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


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

```
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())
```

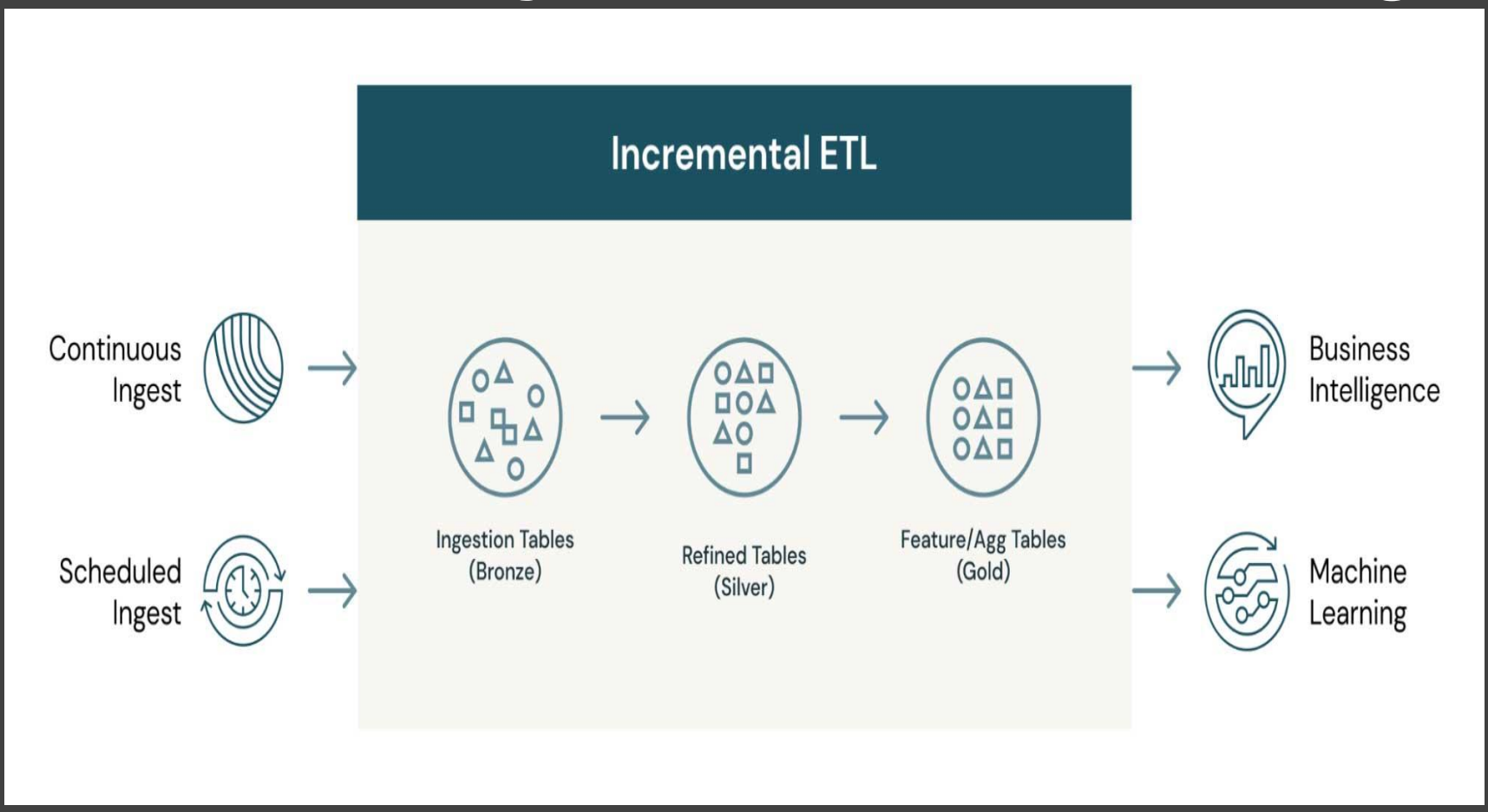
8. Write Transformed Data to Silver Layer:

```
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:

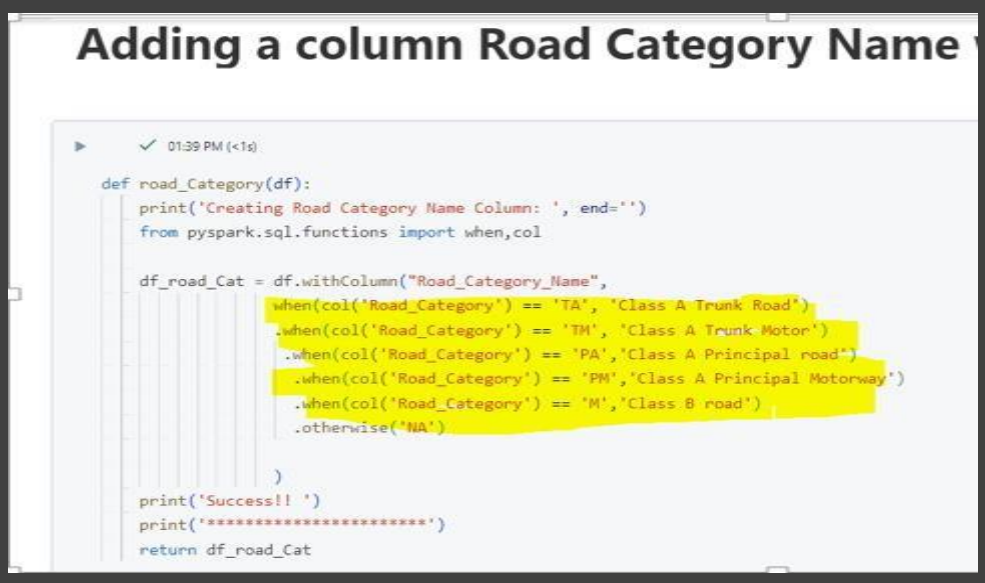
1. **Read Data:**
   - We will read the roads data from the bronze table named "raw_roads."

     ```
     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:**



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 Motor	Major
Class A Trunk Road	Major
Class A Principal Motorway	Major
Class A Trunk Road	Major
Class A Trunk Motor	Major

Refreshed

```
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()
```

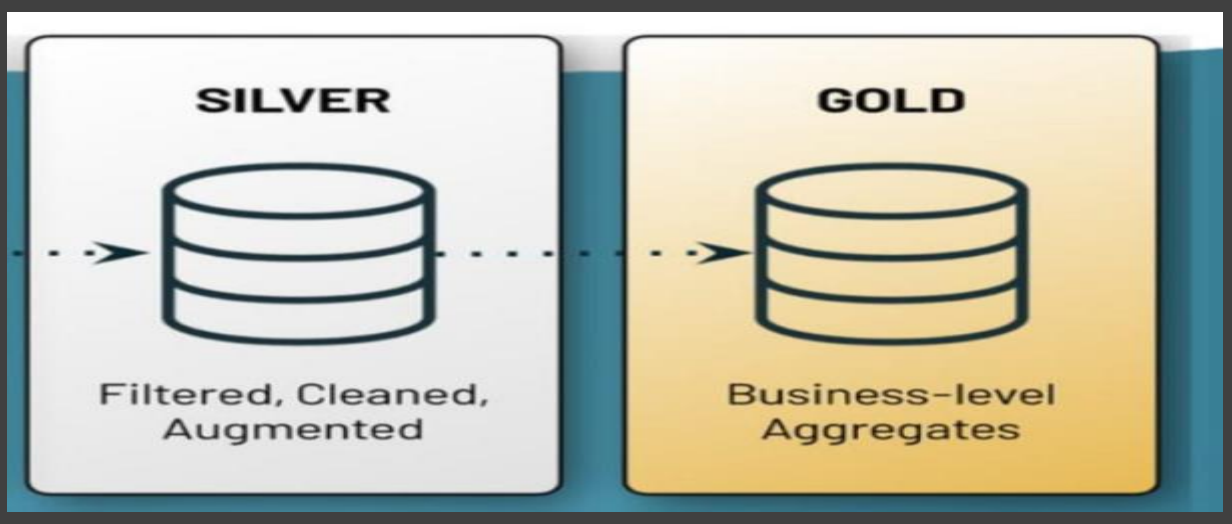
6. Write to Silver Roads Table:

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

7. Verification:

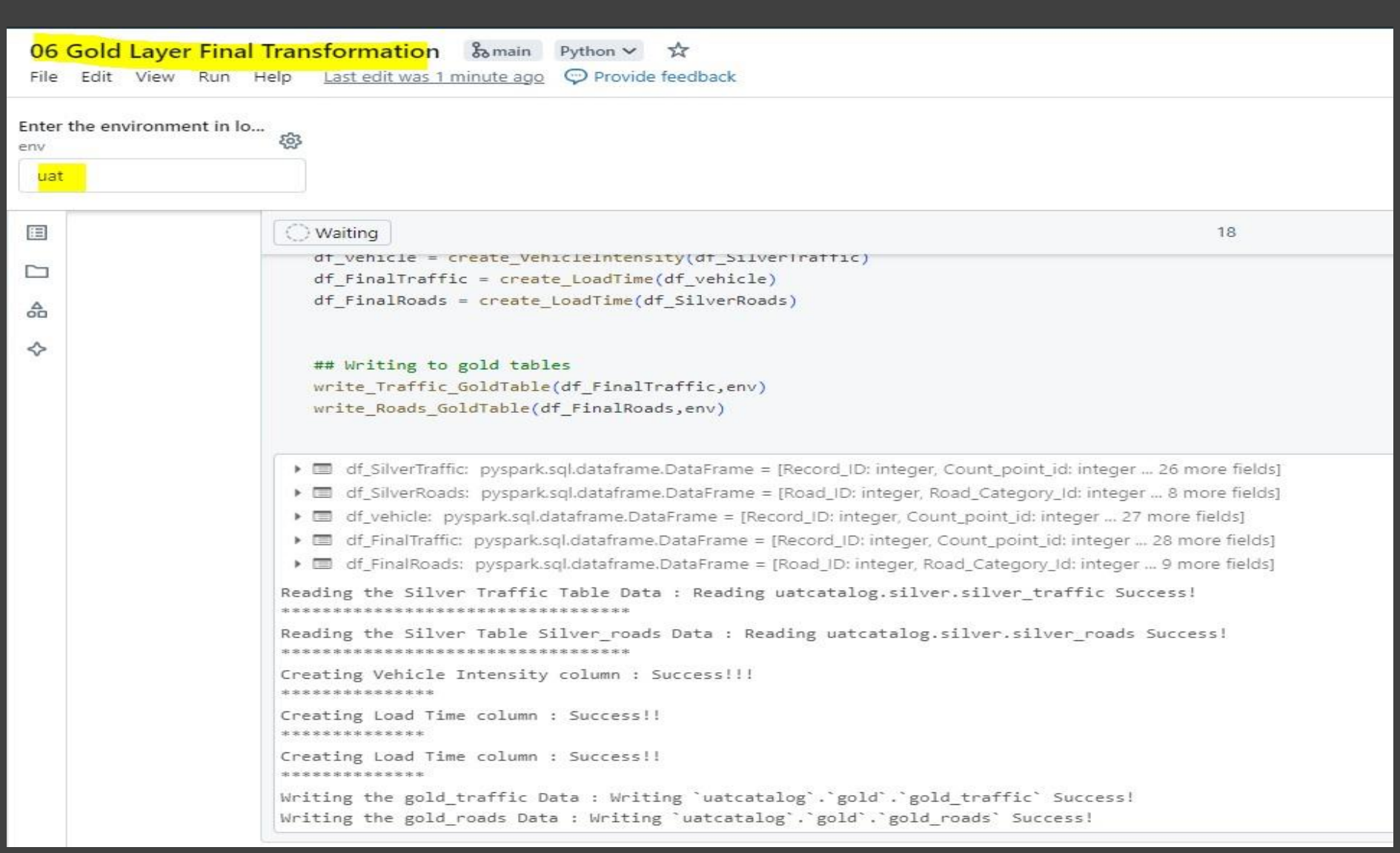
- **Query the Silver layer tables to verify that the transformations have been applied correctly**

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.
   -

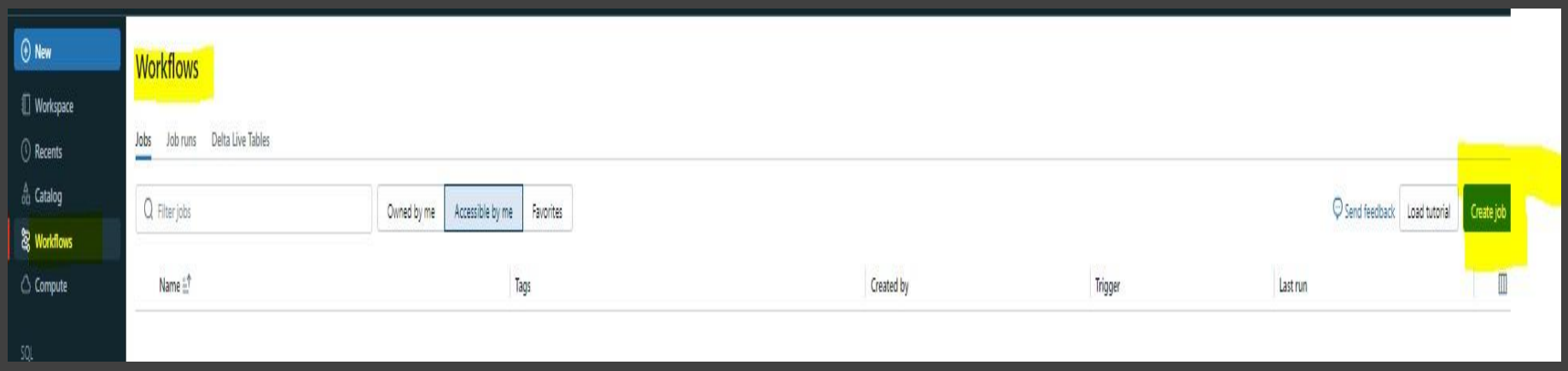


5. **Run Functions:**
   - Define and apply necessary functions in a notebook cell.
6. **Validate and Verify:**
   - Run the notebook to ensure correctness.
   - Refresh and query the Gold tables in the Catalog Explorer to verify the data.

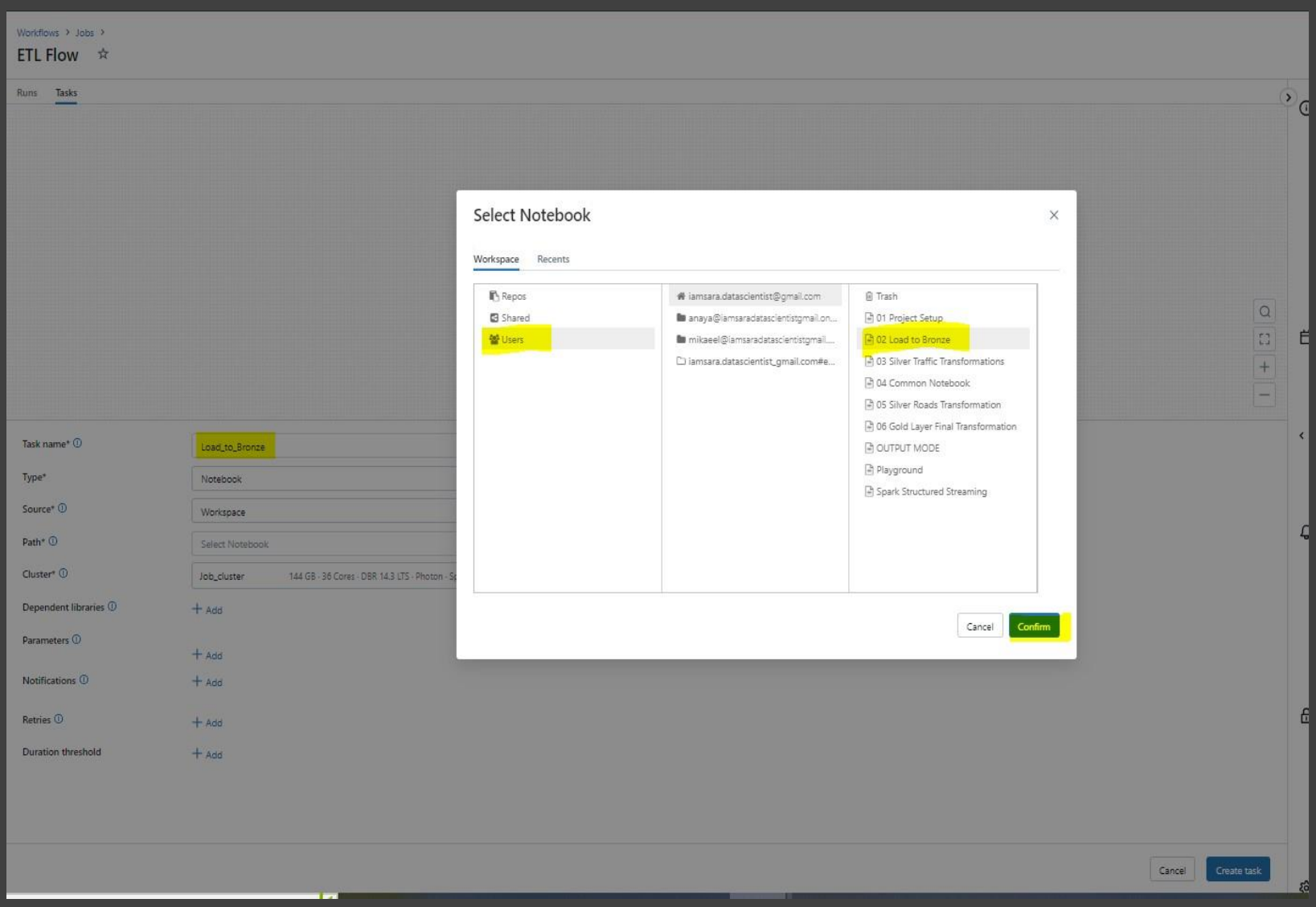
Sara Irfan (sara.data.insights@gmail.com)

Q. How to orchestrate workflows in Azure Databricks?

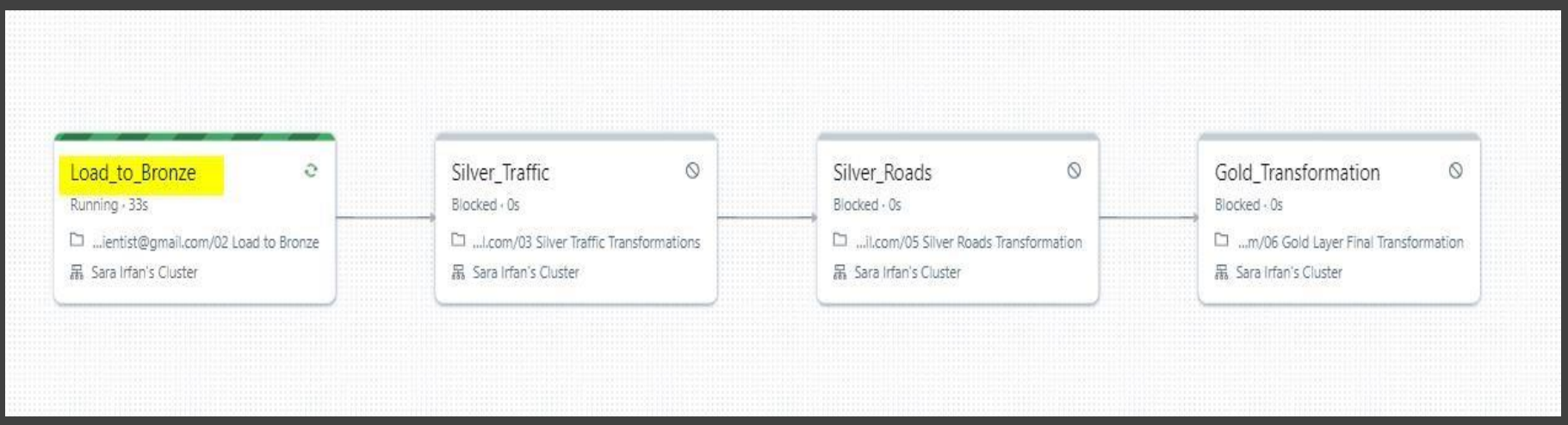
Orchestrating Workflows in Azure Databricks



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:

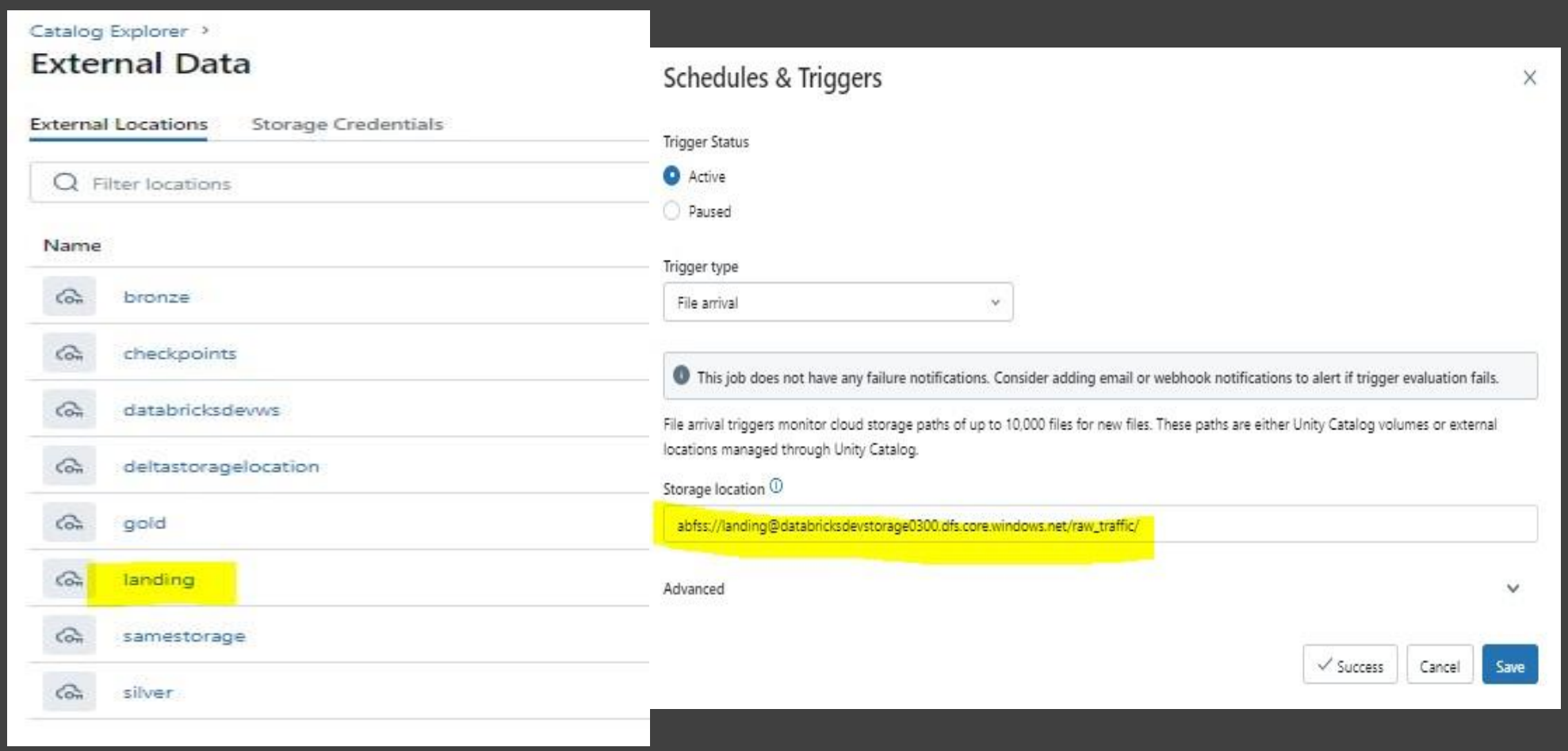
- **Execute the entire workflow.**
- **Query tables to verify data.**
- **Compare record counts in different layers.**

6. Adjust and Optimize:

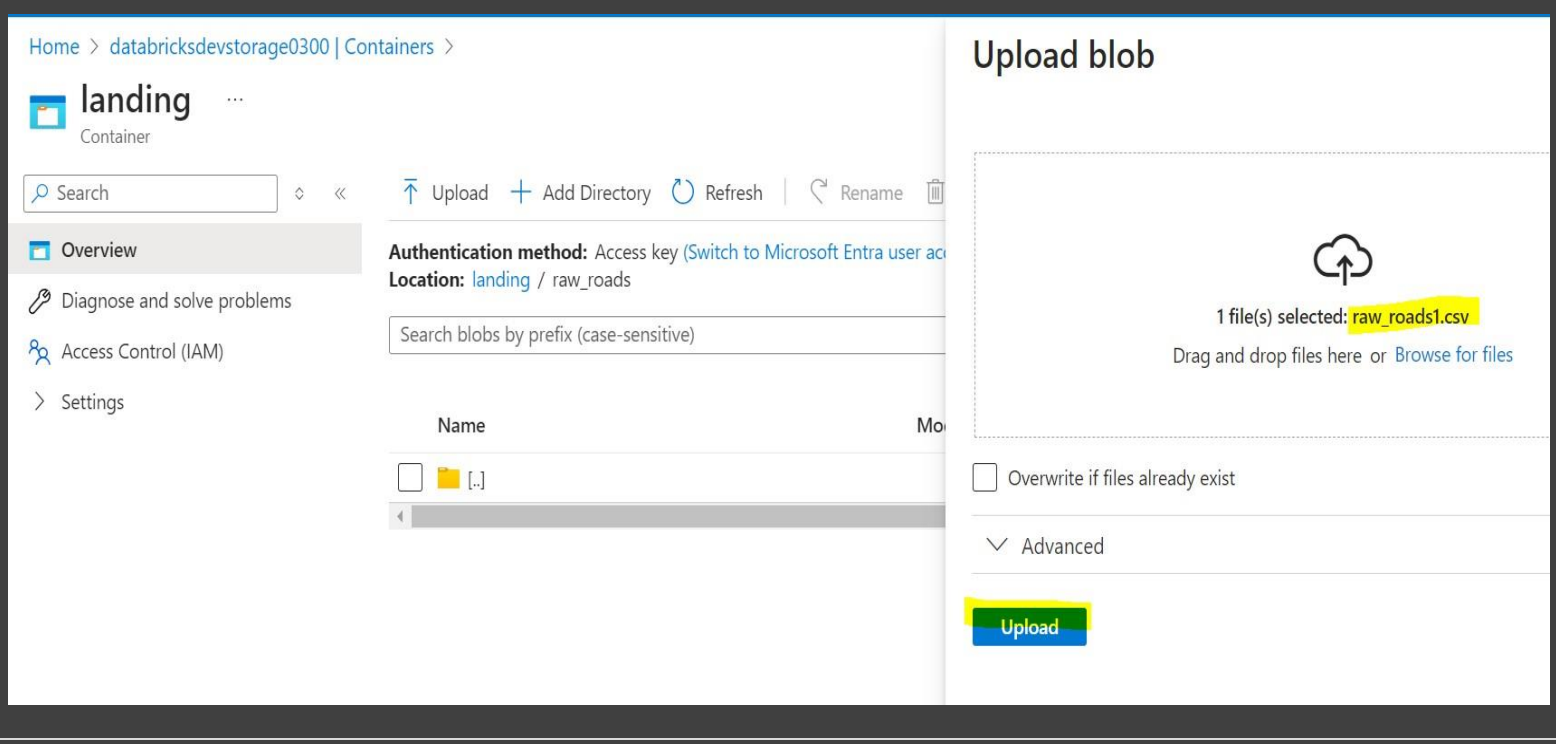
- **Objective: Improve and troubleshoot.**
- **Steps:**
  - **Review performance.**
  - **Adjust clusters or parameters.**
  - **Set up triggers for automatic execution.**

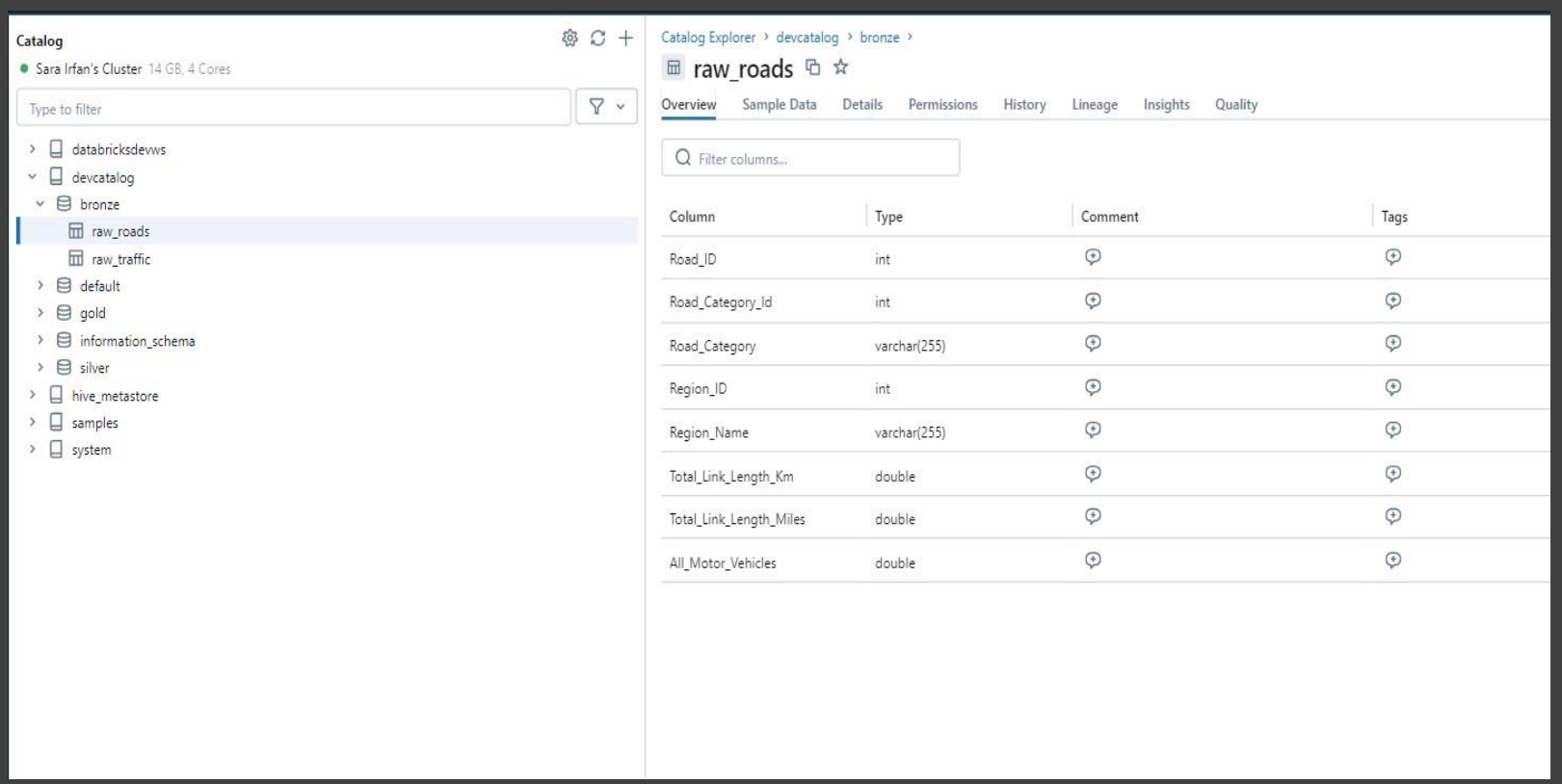
Q. How to set Up Triggers for Azure Databricks Jobs?

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.

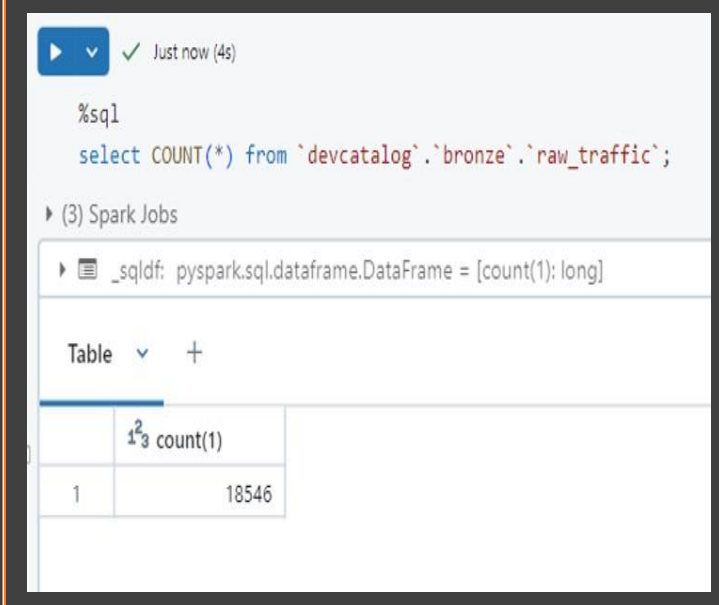


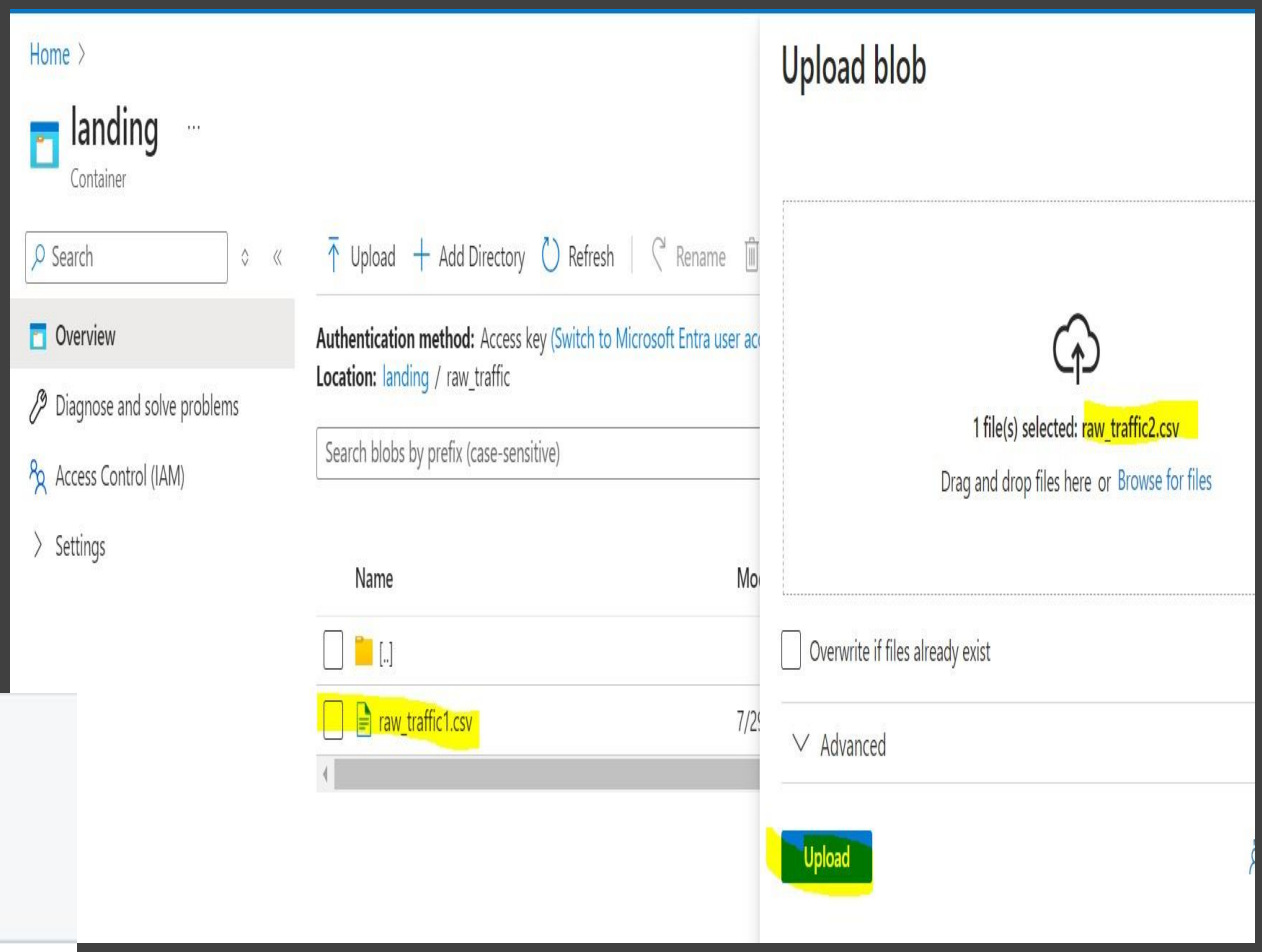
- Ensure the cluster is running or swap to a new cluster if needed.
- Open the job workflow in Azure Databricks.
- Click on "Add Trigger."
- Choose Trigger Type:
- Scheduled Trigger: For periodic execution.
- File Arrival Trigger: Executes when new files appear.
- Continuous Trigger: For real-time processing.
- 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.

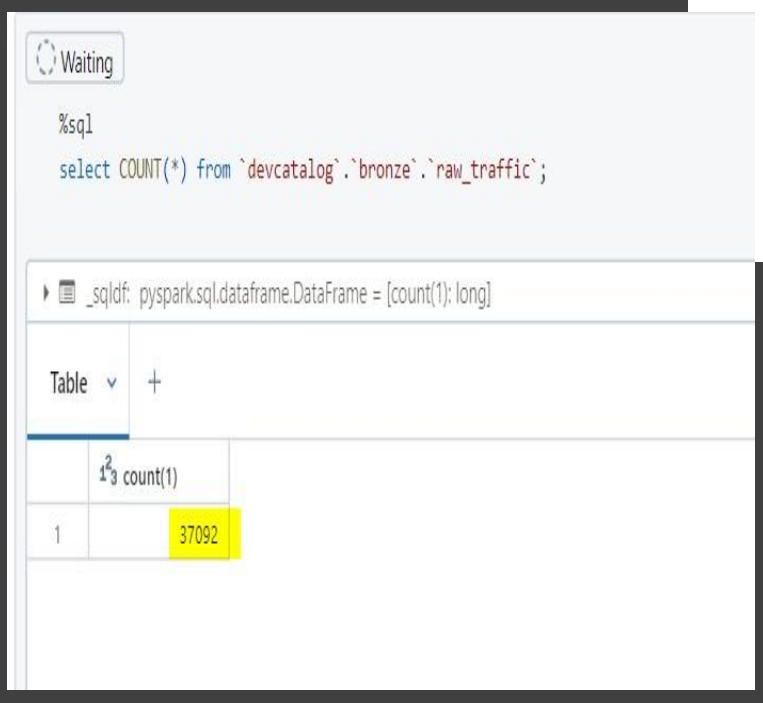




Please analyze the increase in records after inserting a file in a container







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 +

	count(1)
1	37092

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

(4) Spark Jobs

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

Table +

	count(1)
1	76

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

(4) Spark Jobs

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

Table +

	count(1)
1	76

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

(5) Spark Jobs

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

Table +

	count(1)
1	37092

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

(4) Spark Jobs

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

Table +

	count(1)
1	37092

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

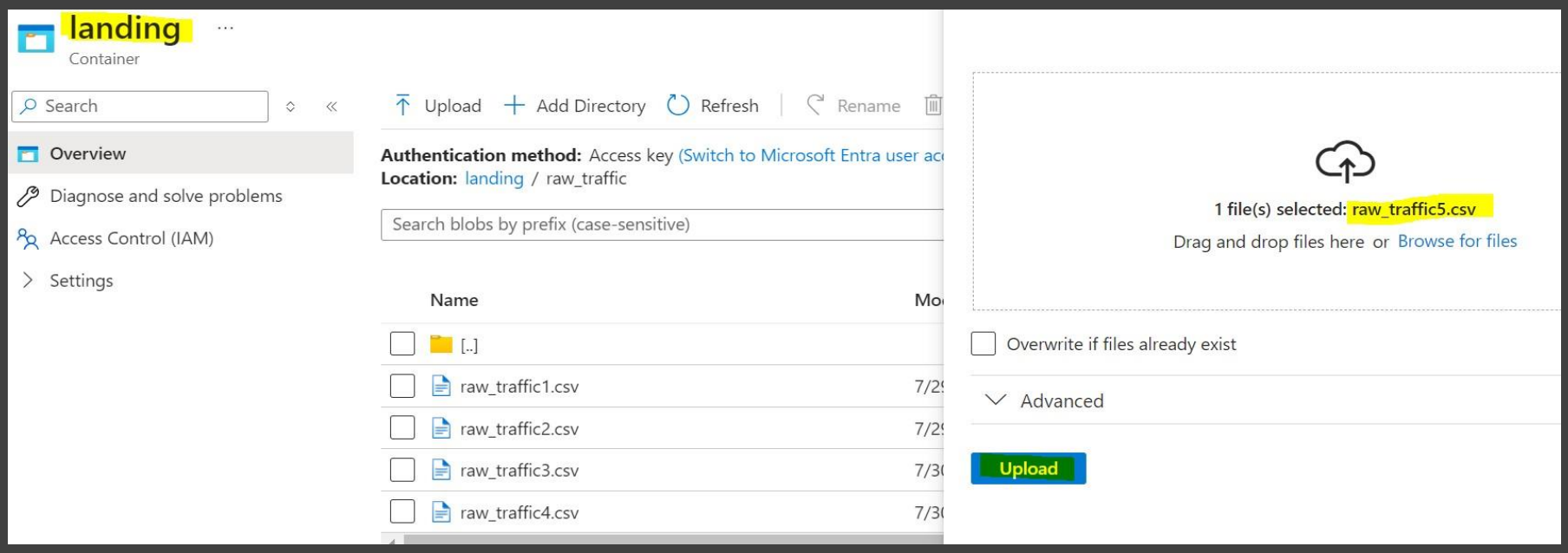
(5) Spark Jobs

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

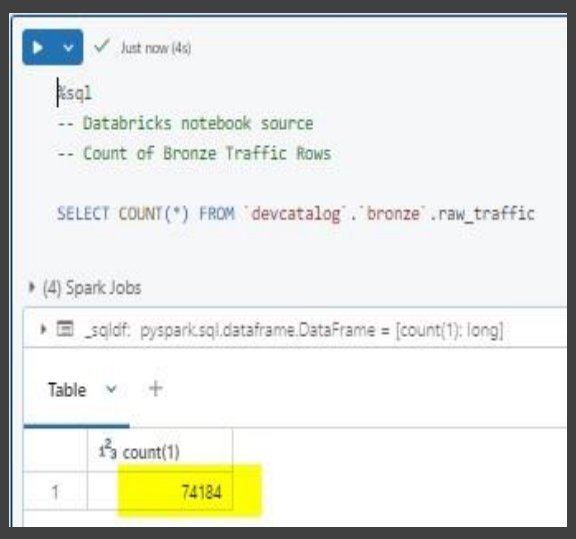
Table +

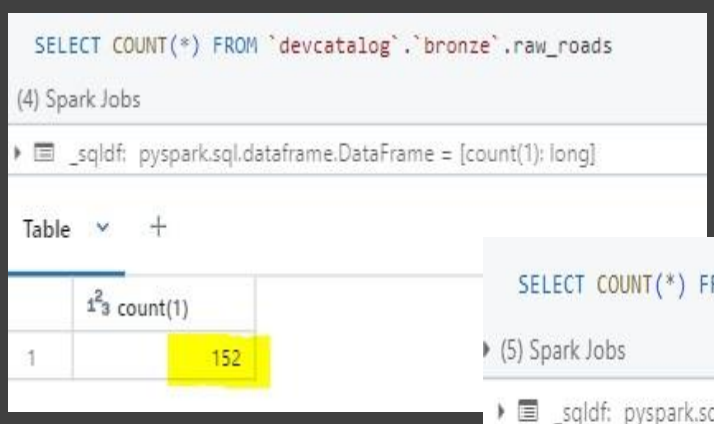
	count(1)
1	76

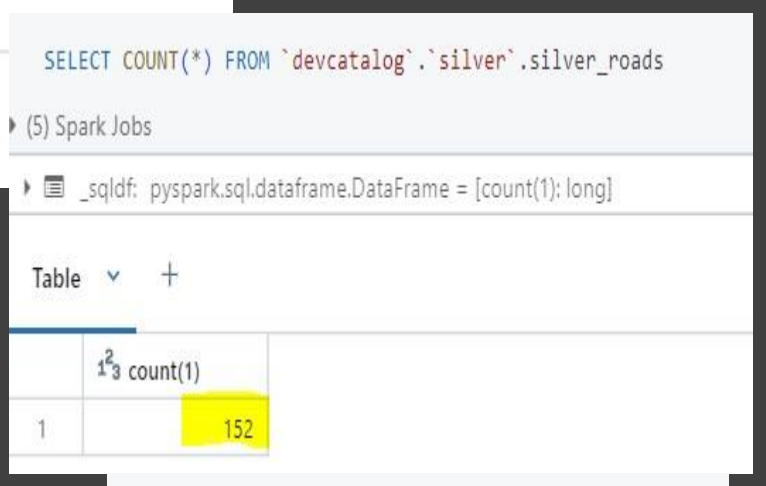
Name	Type	Owner
01 Project Setup	Notebook	Sara Irfan
02 Load to Bronze	Notebook	Sara Irfan
03 Silver Traffic Transformations	Notebook	Sara Irfan
04 Common Notebook	Notebook	Sara Irfan
05 Silver Roads Transformation	Notebook	Sara Irfan
06 Gold Layer Final Transformation	Notebook	Sara Irfan
OUTPUT MODE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

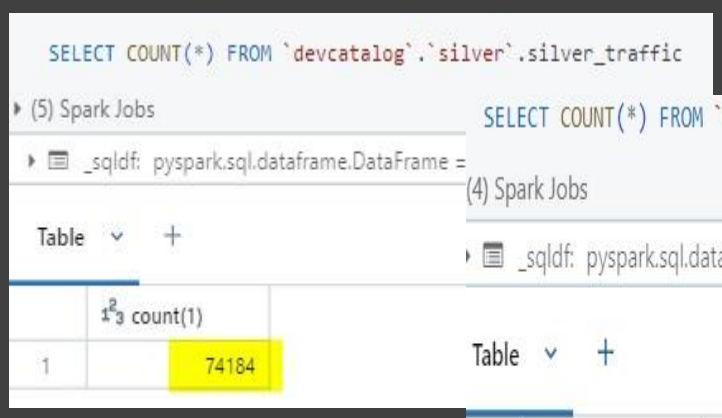


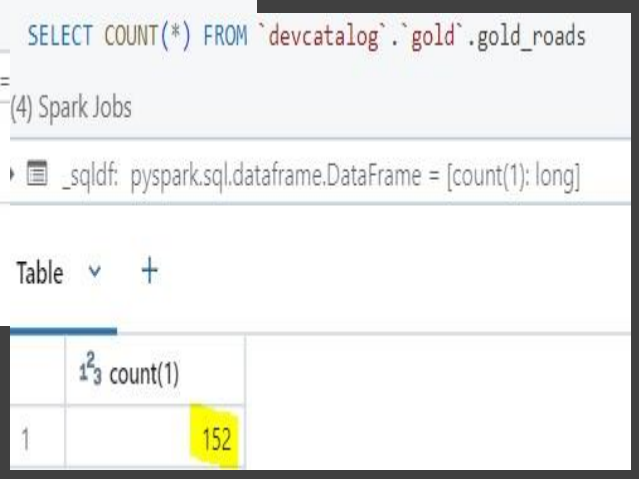
Please analyze the change in values, with insertion of files

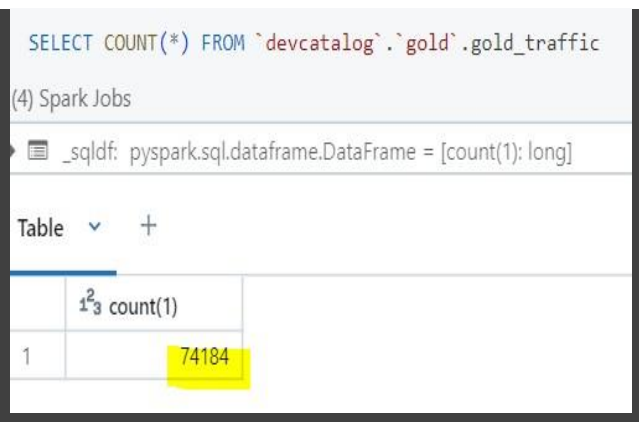










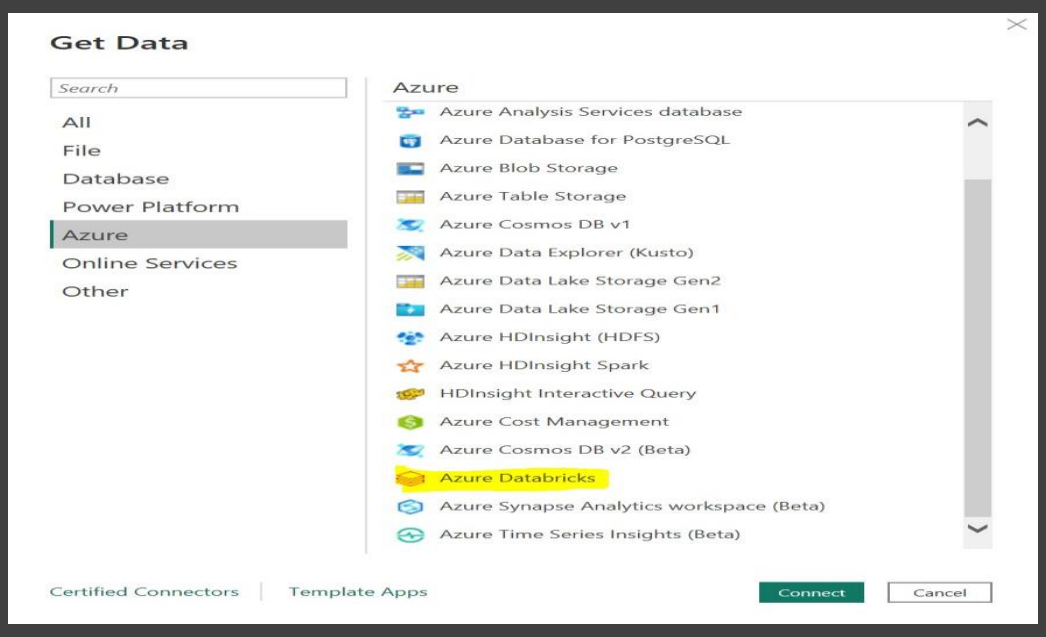


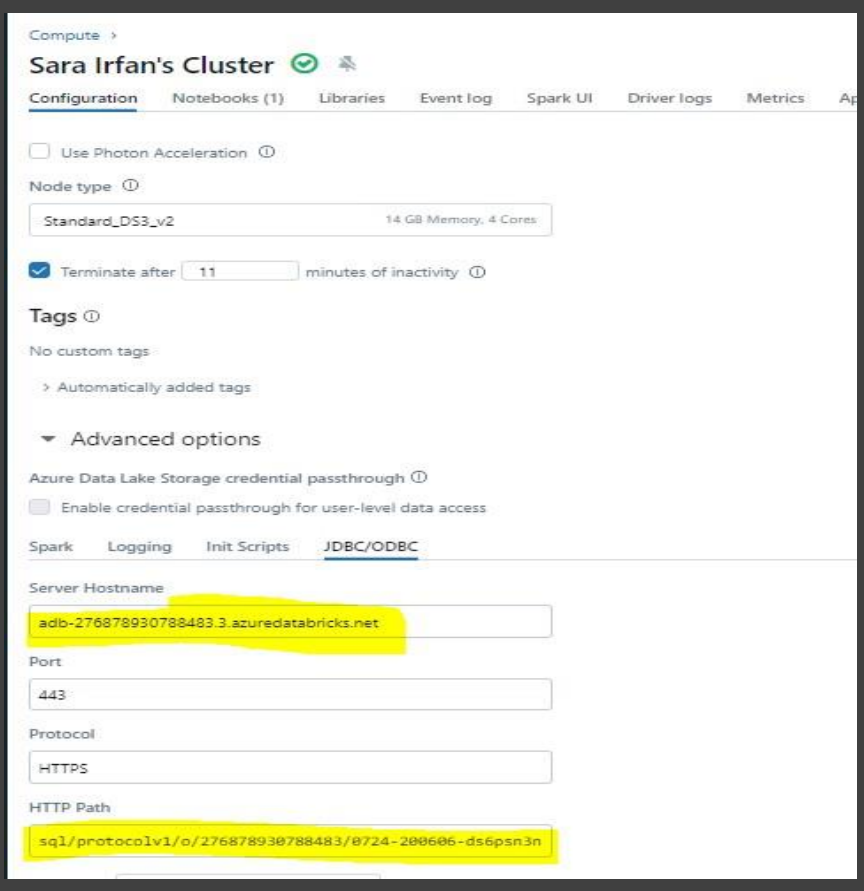
Start time	Run ID	Launched	Duration	Status
Jul 30, 2024, 08:01 PM	299860350075303	By file arrival	1m 5s	Running
Jul 30, 2024, 05:19 PM	965390140979113	Manually	2m 59s	Succeeded

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.**



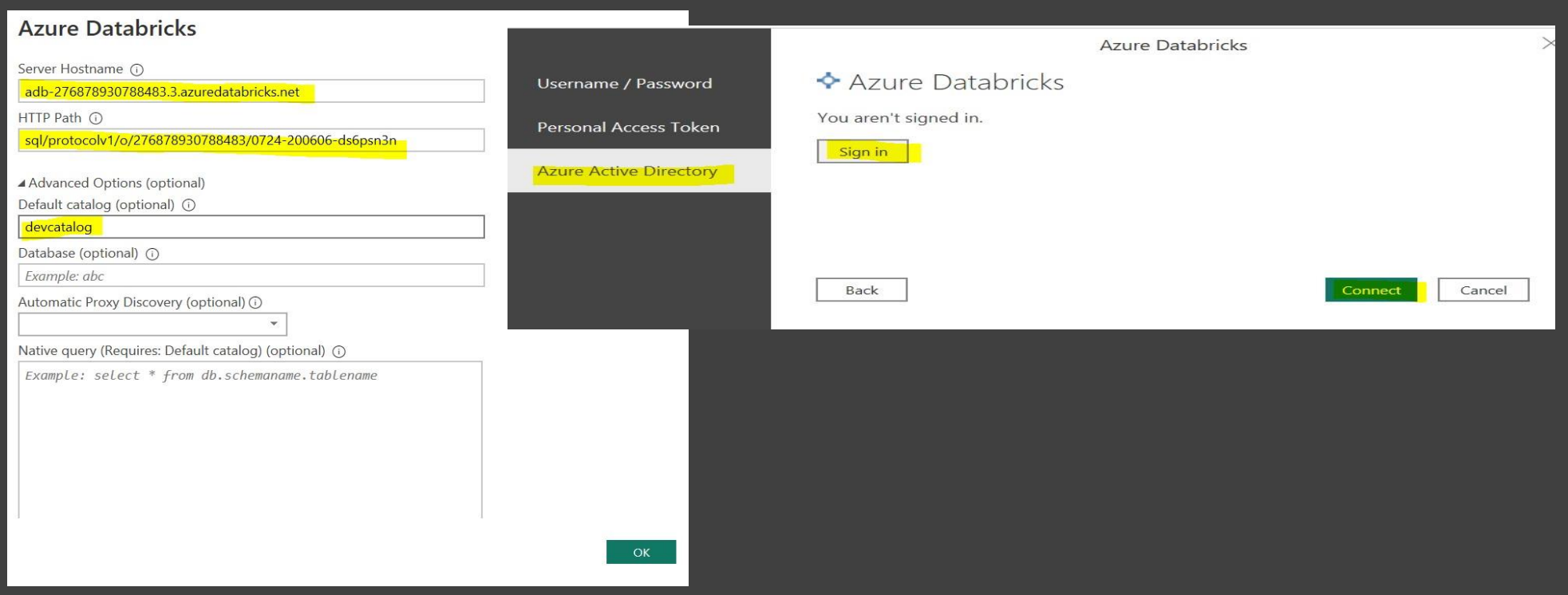


3. Log In to Azure Databricks:

- **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.

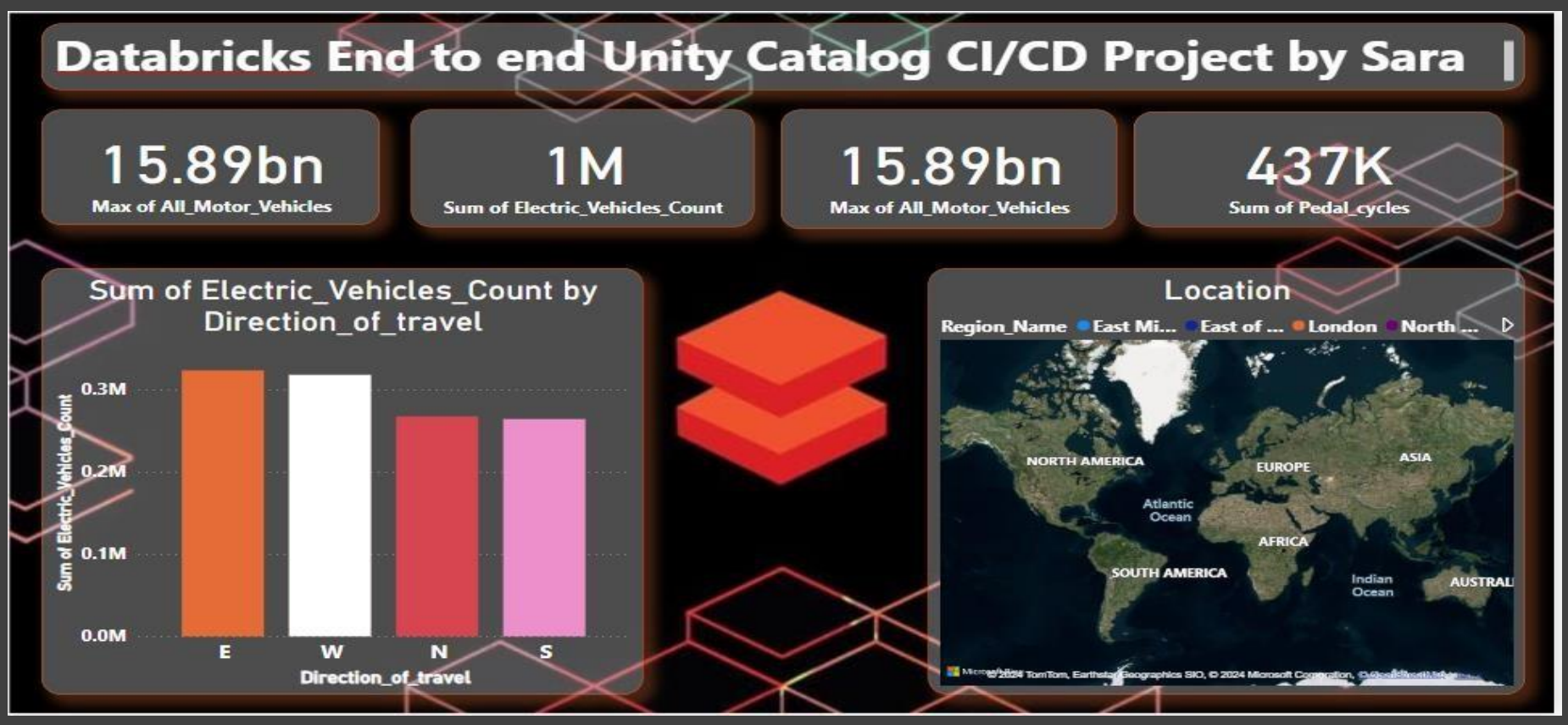
6. Update Data:

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

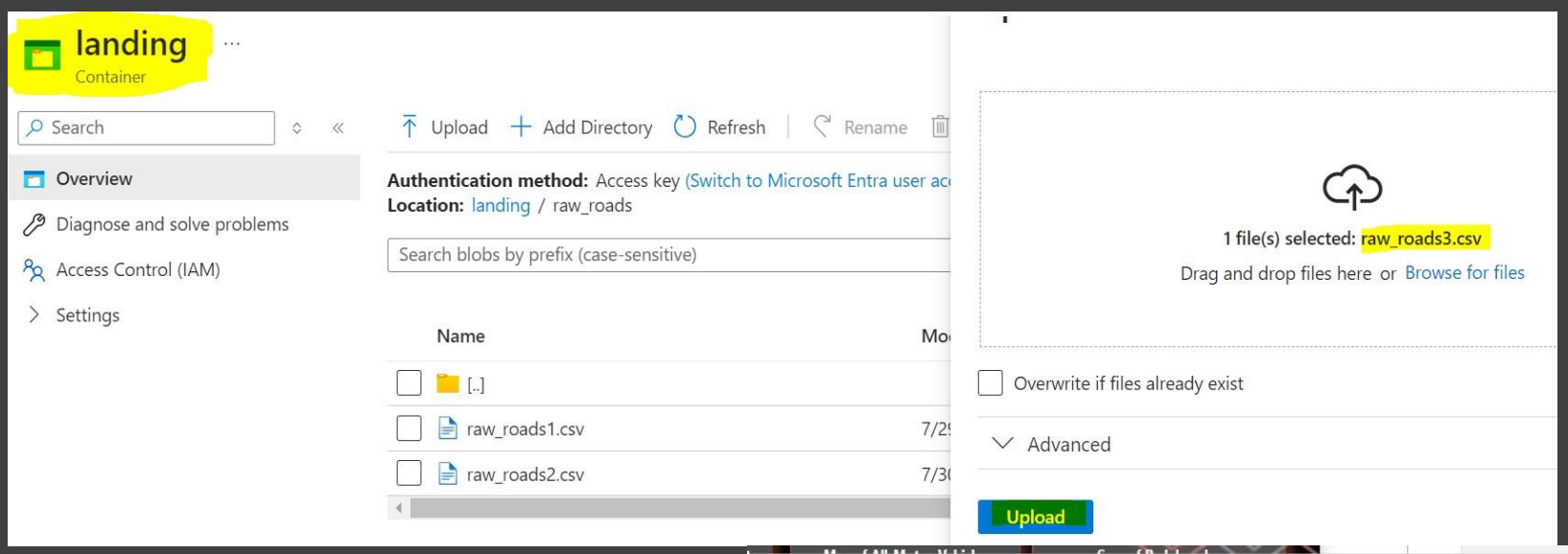
7. Publish Power BI Report:

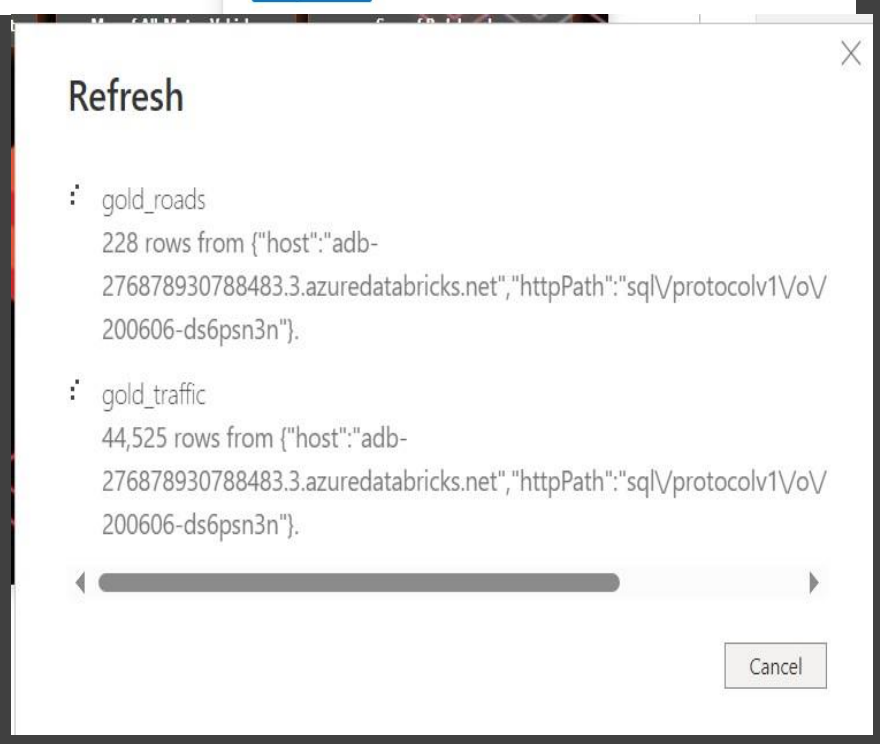
- Publish to Power BI Service
- Configure scheduled refresh for automatic updates

Power BI Dashboard

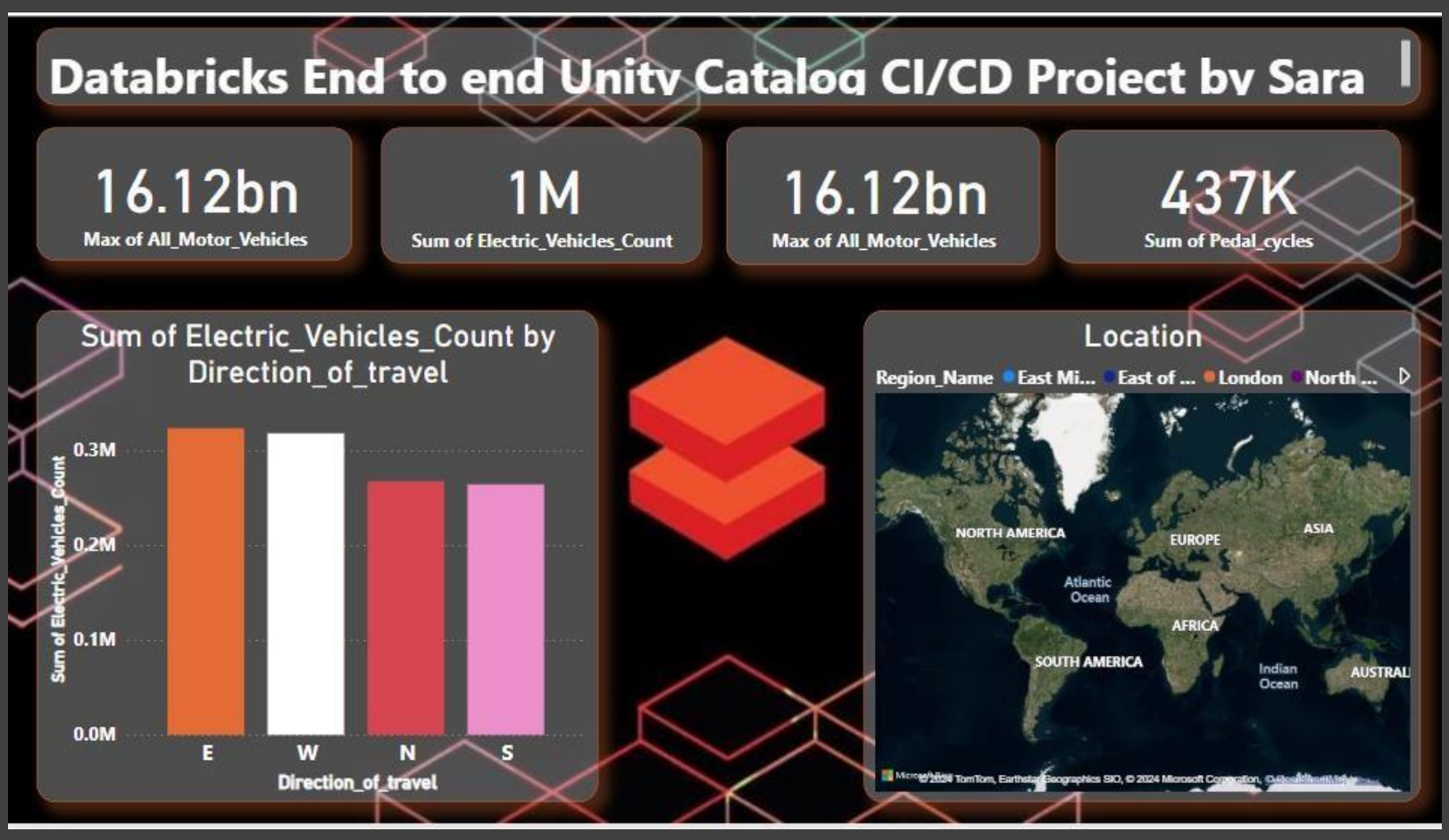


Uploading raw_roads3.csv in landing container, the values of power BI dashboard has been changed





Updated Power BI Dashboard

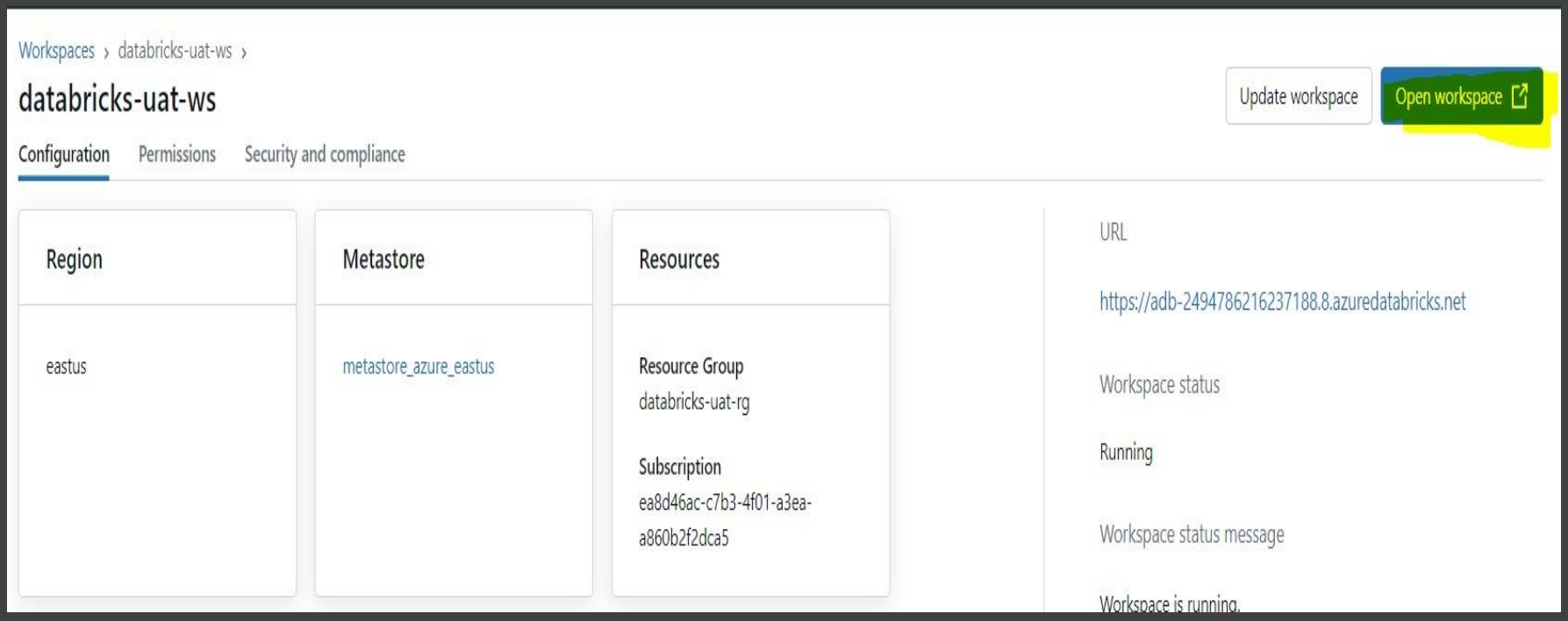


Q. How do you configure a UAT environment in Azure, and what are the essential steps to integrate it with Azure DevOps?

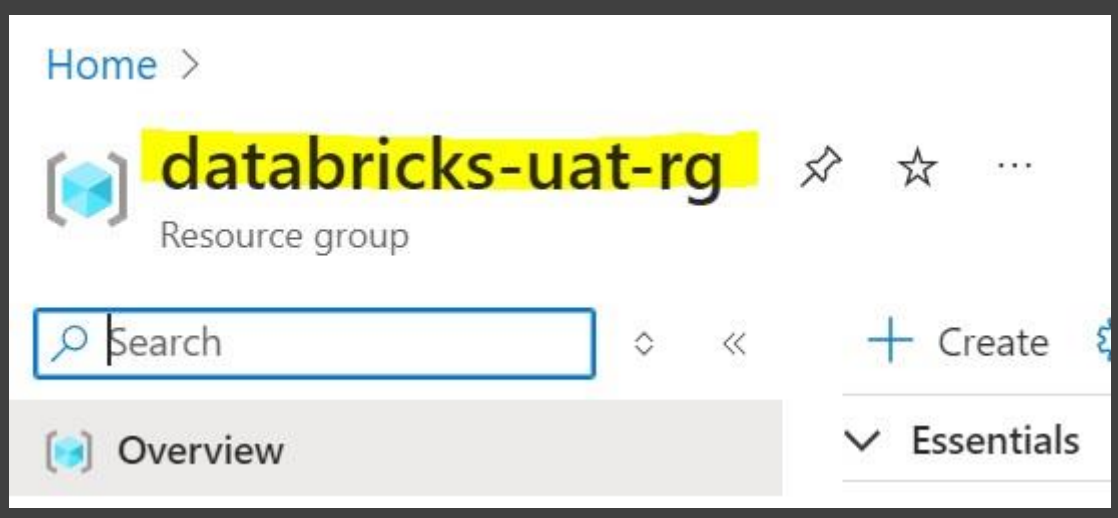
To configure a UAT environment in Azure and integrate it with Azure DevOps, follow these essential steps:

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.**

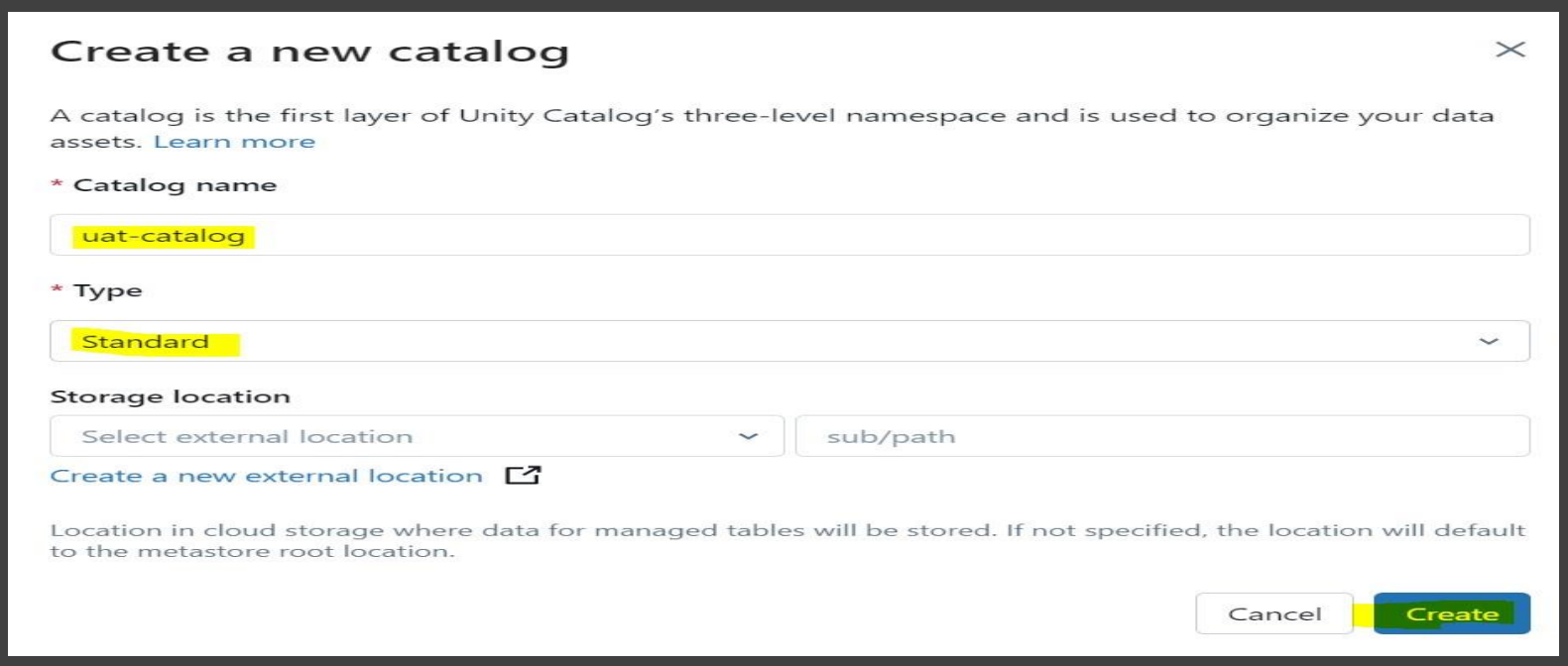


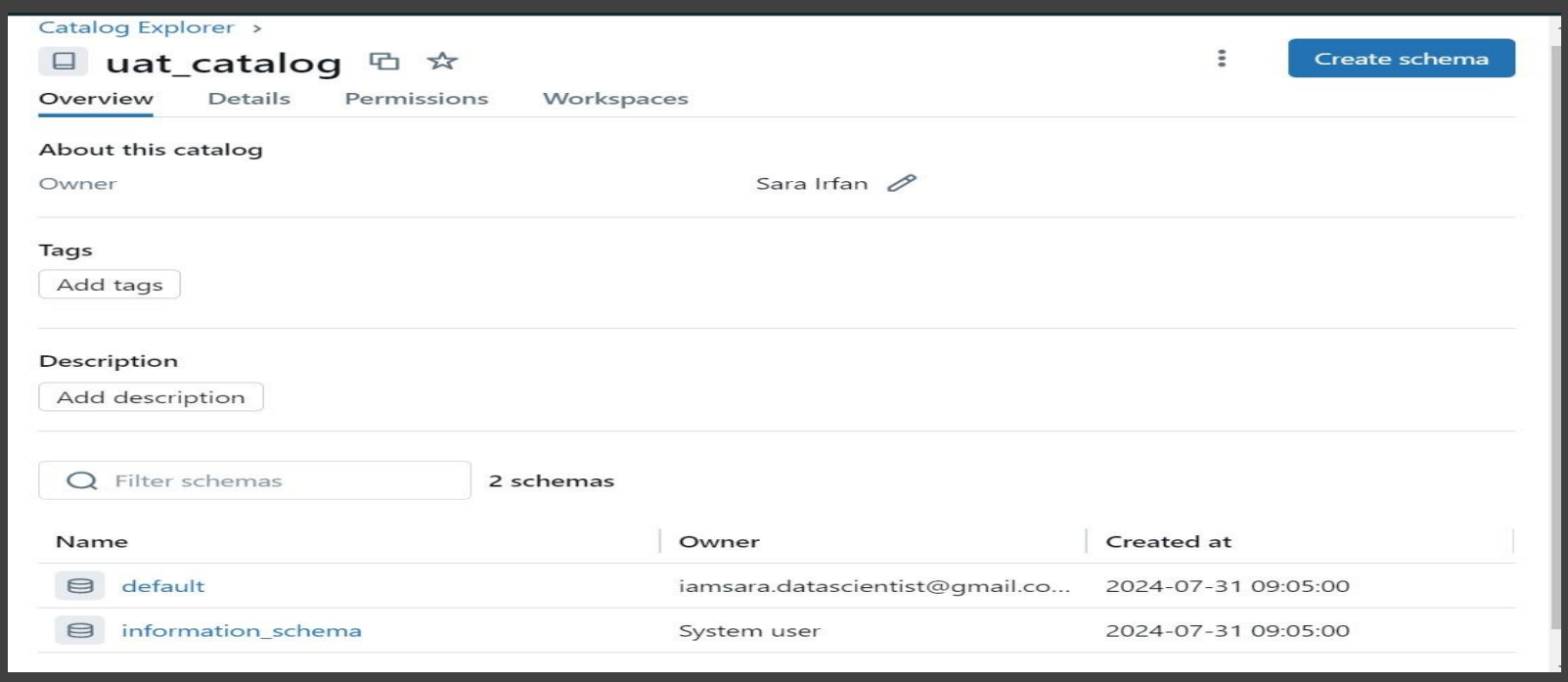
- **Storage Account: Set up a storage account (e.g., uatstorageaccount) with hierarchical namespace enabled for Data Lake Storage Gen2.**



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.**



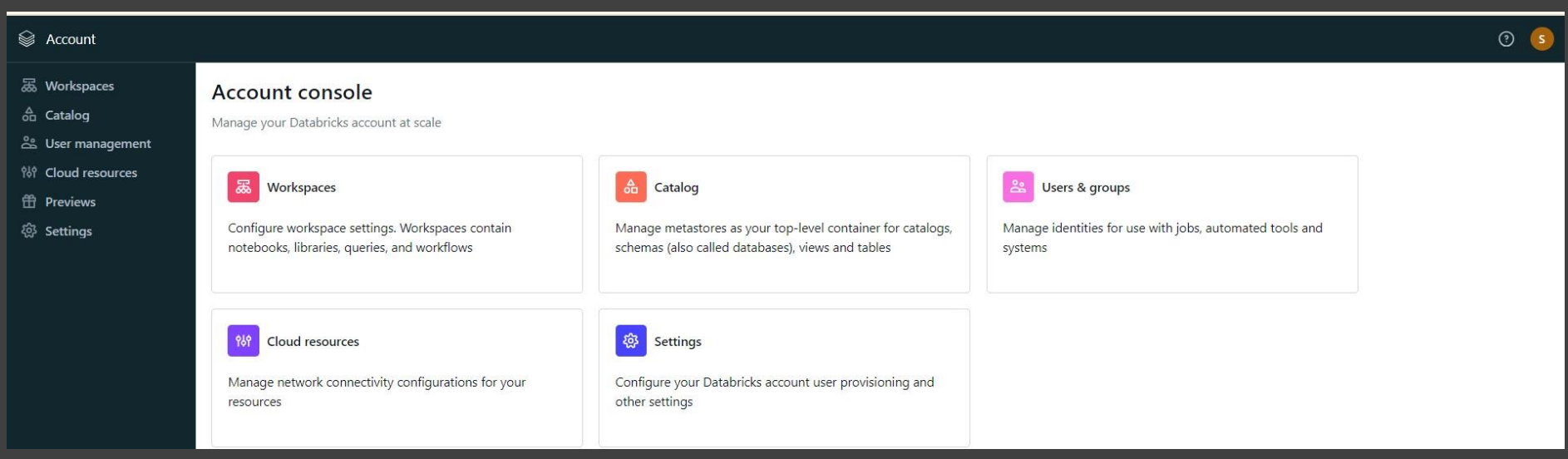


3. Create External Locations:

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

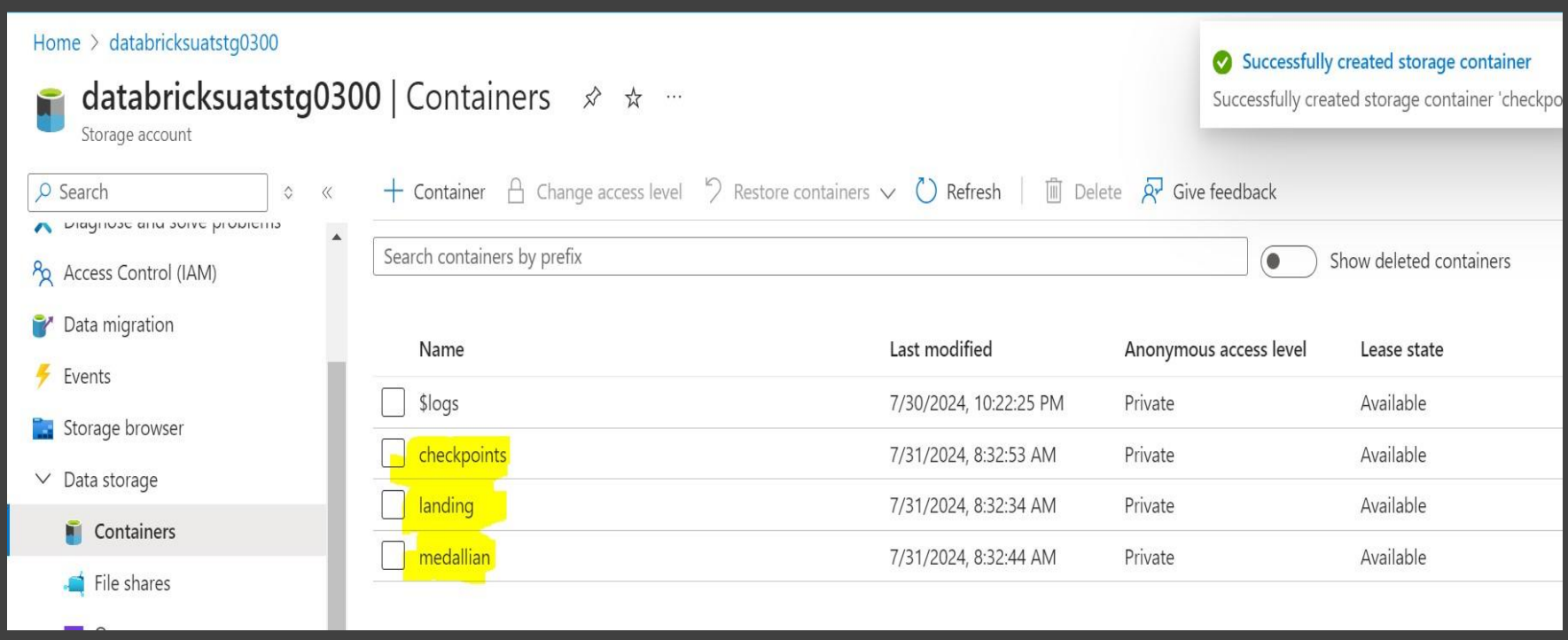
4. Implement Continuous Deployment:

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



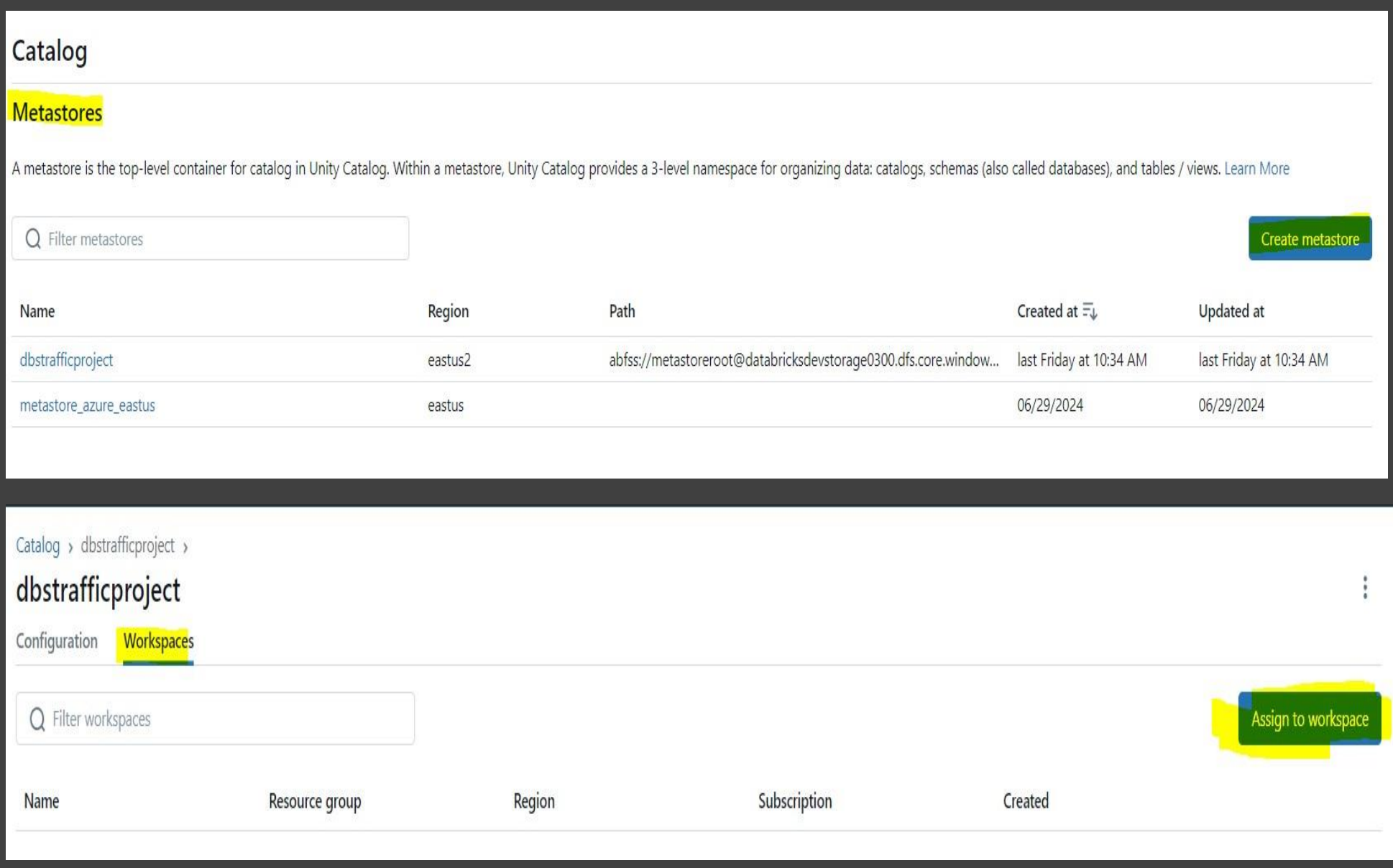
5. Validate and Test:

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



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.

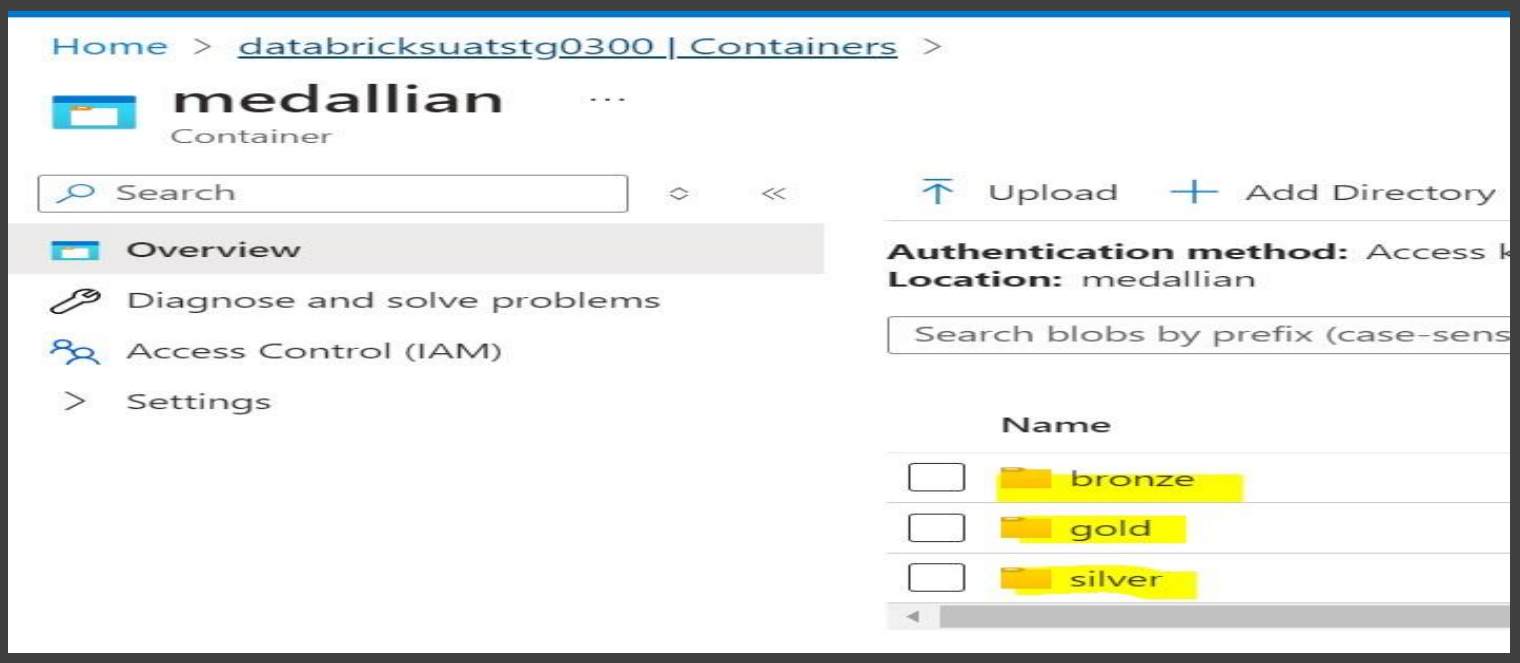


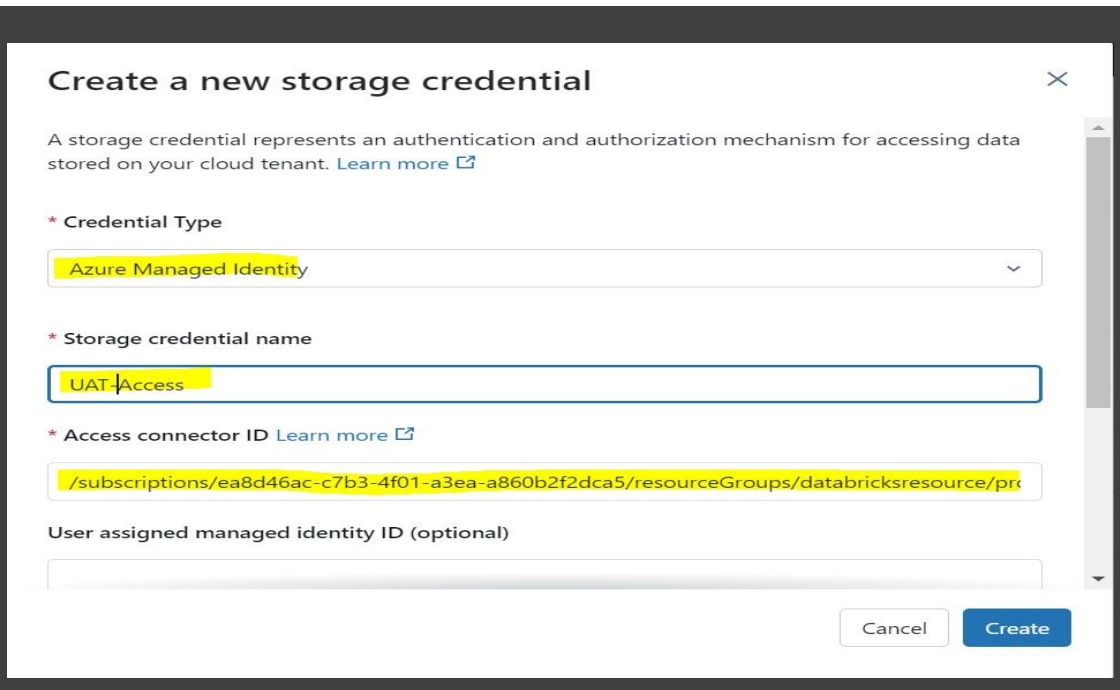
7. Branch Management and Pull Requests:

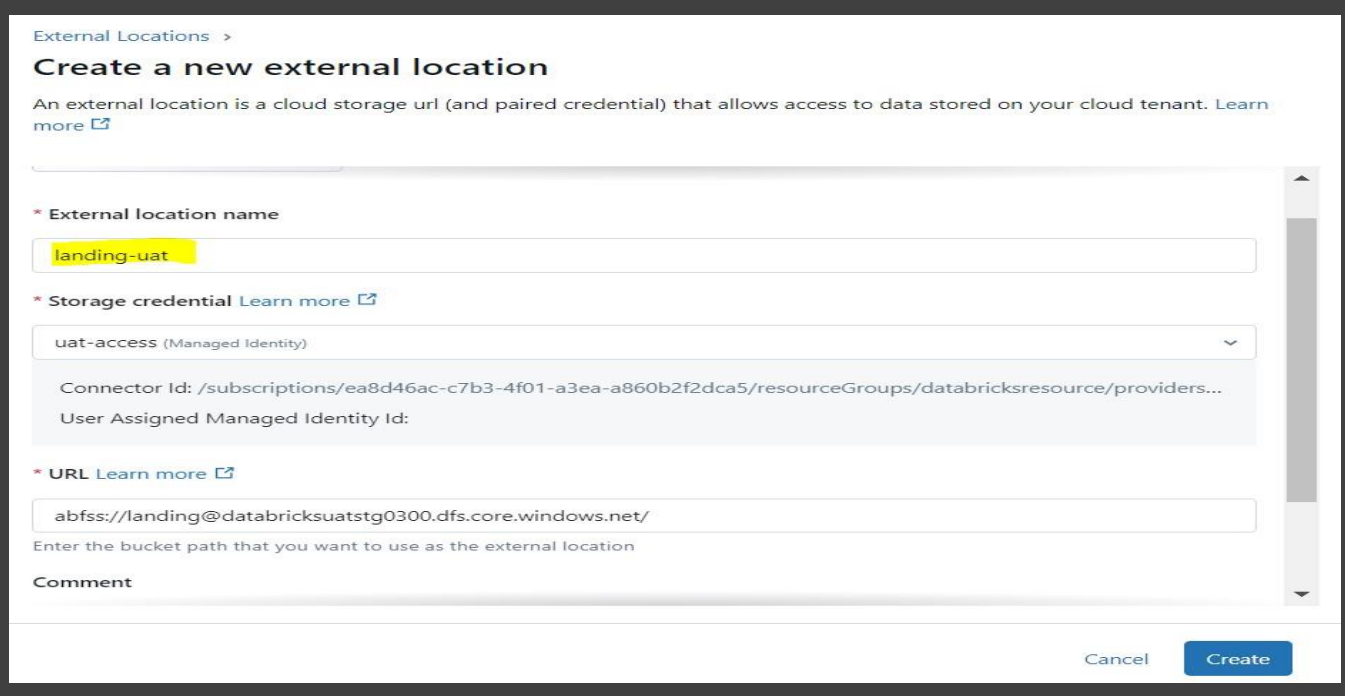
- Establish branch policies for code reviews and access management.
- Create a feature branch, add notebooks, and commit changes.
- Open a pull request, review, approve, and merge changes to the main branch.

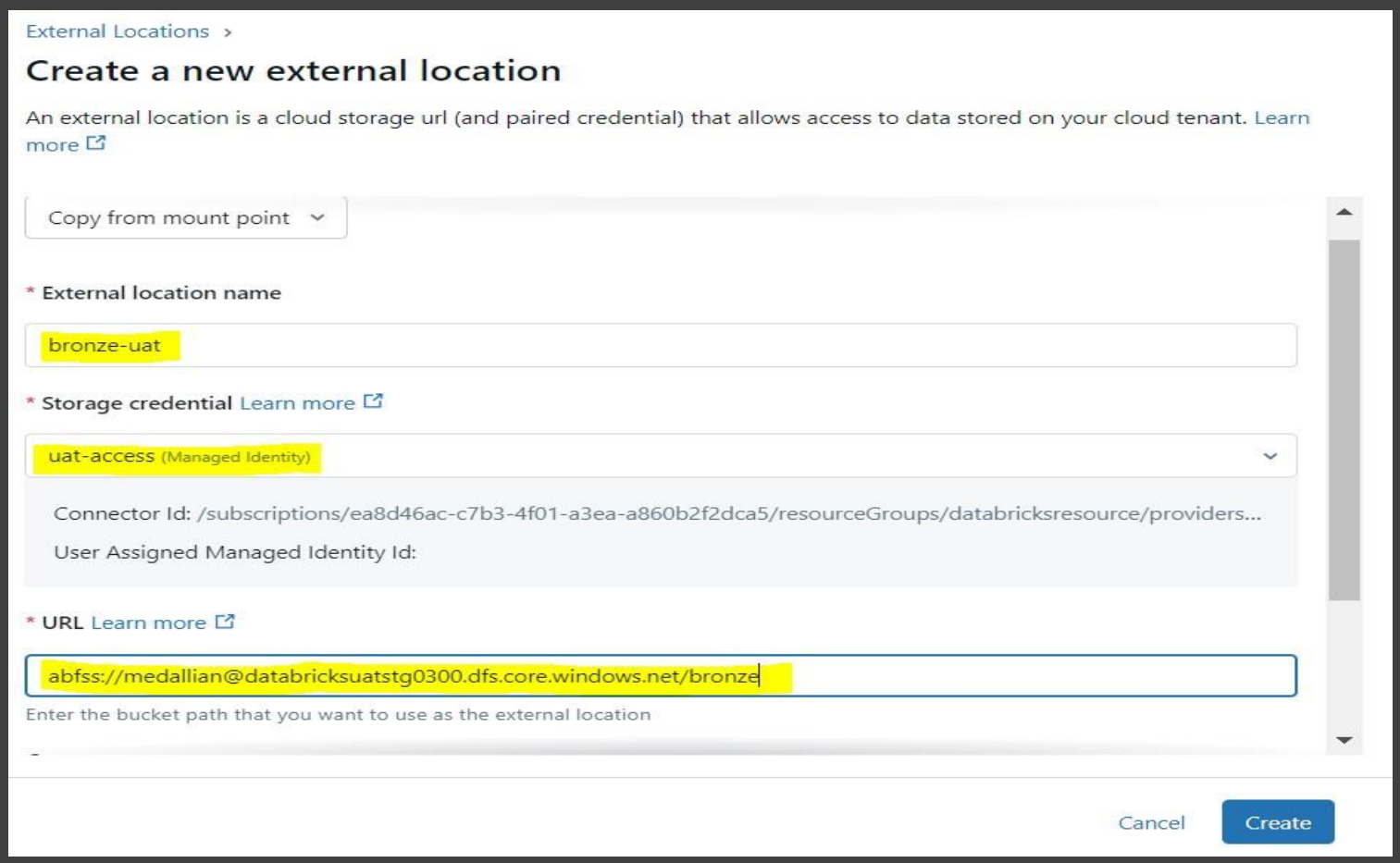
8. Mikael's Integration:

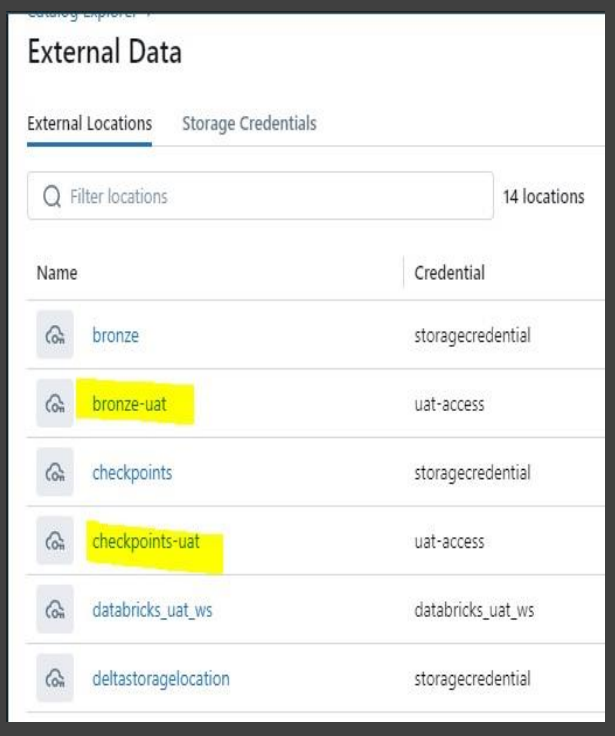
- Mikaeel configures a repository, creates a feature branch, and commits changes.
- Pull request is reviewed, approved, and merged.
- Set up a live folder for automatic updates and integration pipeline.

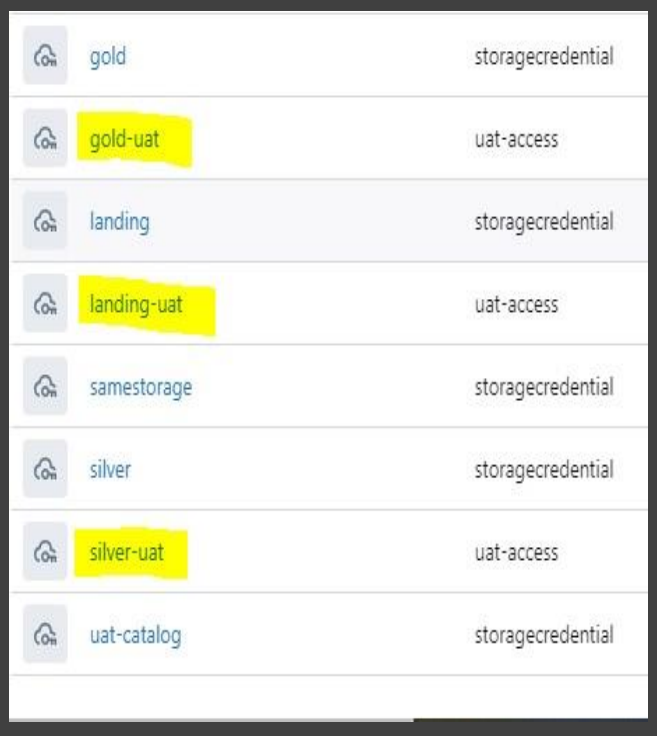

Home > databricksuatstg0300 | Containers >
medallian
Container
Search
Overview
Diagnose and solve problems
Access Control (IAM)
Settings
Upload
Add Directory
Authentication method: Access k
Location: medallian
Search blobs by prefix (case-sens
Name
bronze
gold
silver


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 Learn more
/subscriptions/ea8d46ac-c7b3-4f01-a3ea-a860b2f2dca5/resourceGroups/databricksresource/pr
User assigned managed identity ID (optional)
Cancel
Create

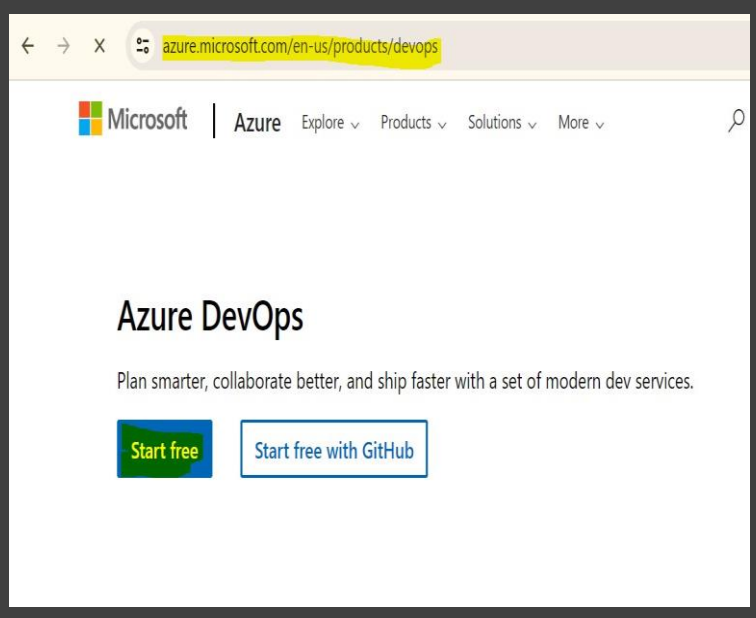

External Locations >
Create a new external location
An external location is a cloud storage url (and paired credential) that allows access to data stored on your cloud tenant. Learn more
* External location name
landing-uat
* Storage credential Learn more
uat-access (Managed Identity)
Connector Id: /subscriptions/ea8d46ac-c7b3-4f01-a3ea-a860b2f2dca5/resourceGroups/databricksresource/providers...
User Assigned Managed Identity Id:
* URL Learn more
abfss://landing@databricksuatstg0300.dfs.core.windows.net/
Enter the bucket path that you want to use as the external location
Comment
Cancel
Create

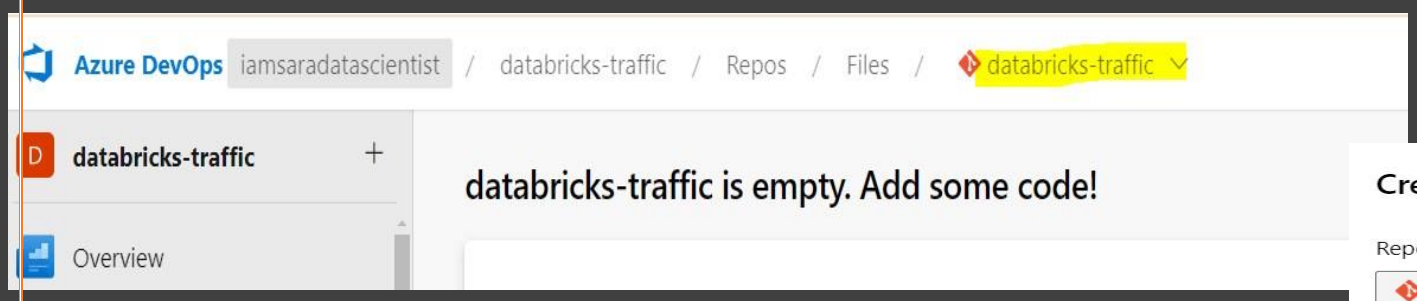


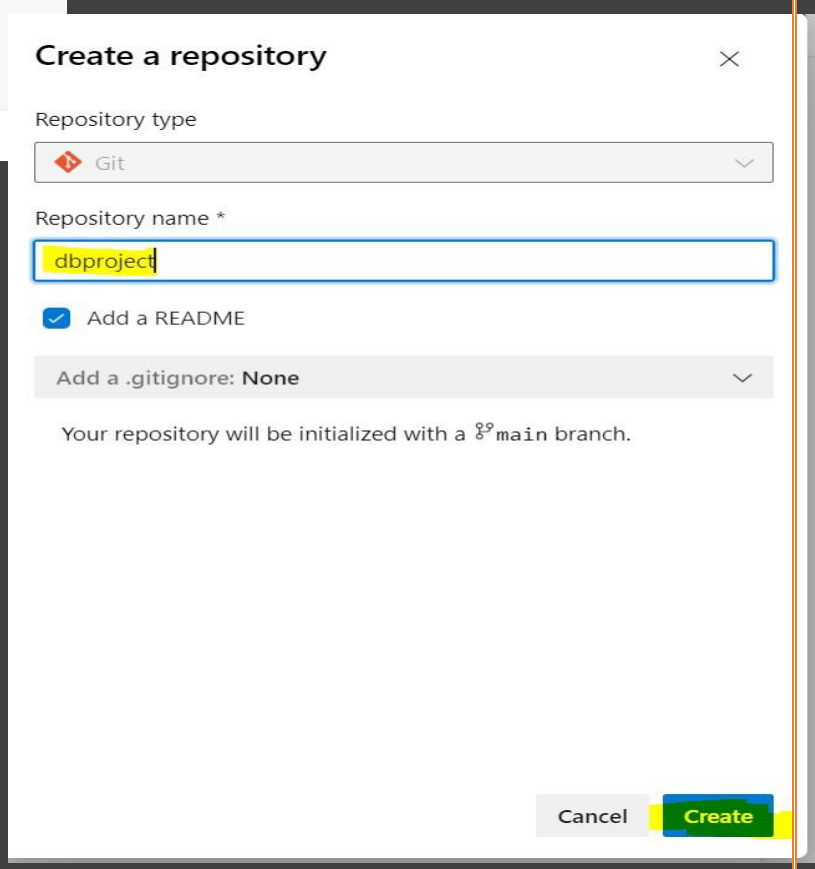


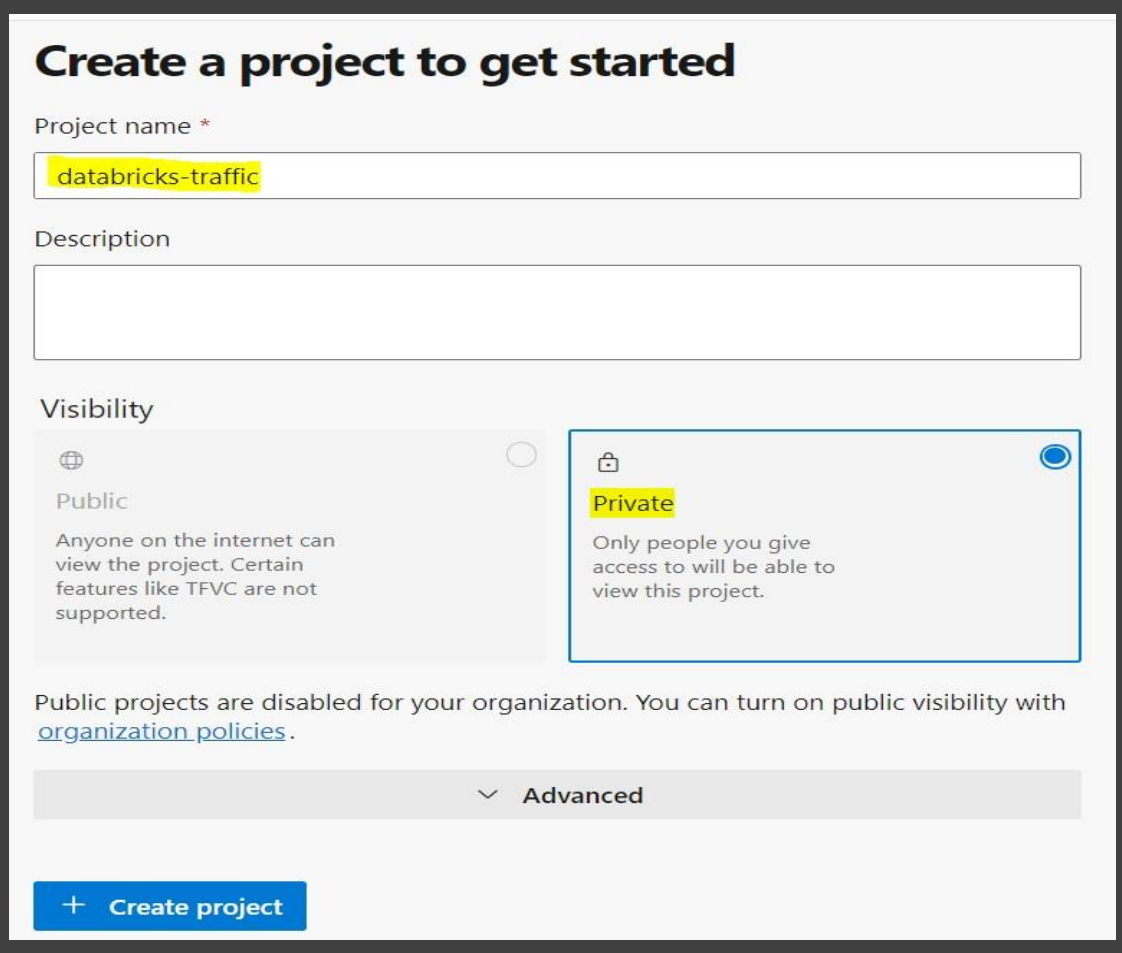


Q. How to sign in and use the Azure Devops?



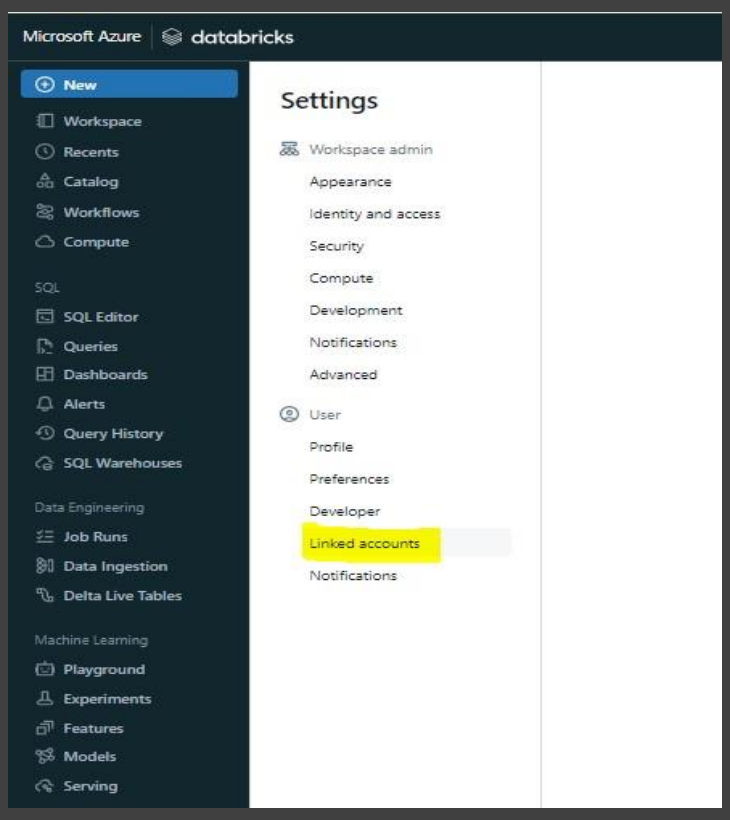

Azure DevOps
iamsaradatascientist
/
databricks-traffic
/
Repos
/
Files
/
databricks-traffic
databricks-traffic
Overview
databricks-traffic is empty. Add some code!

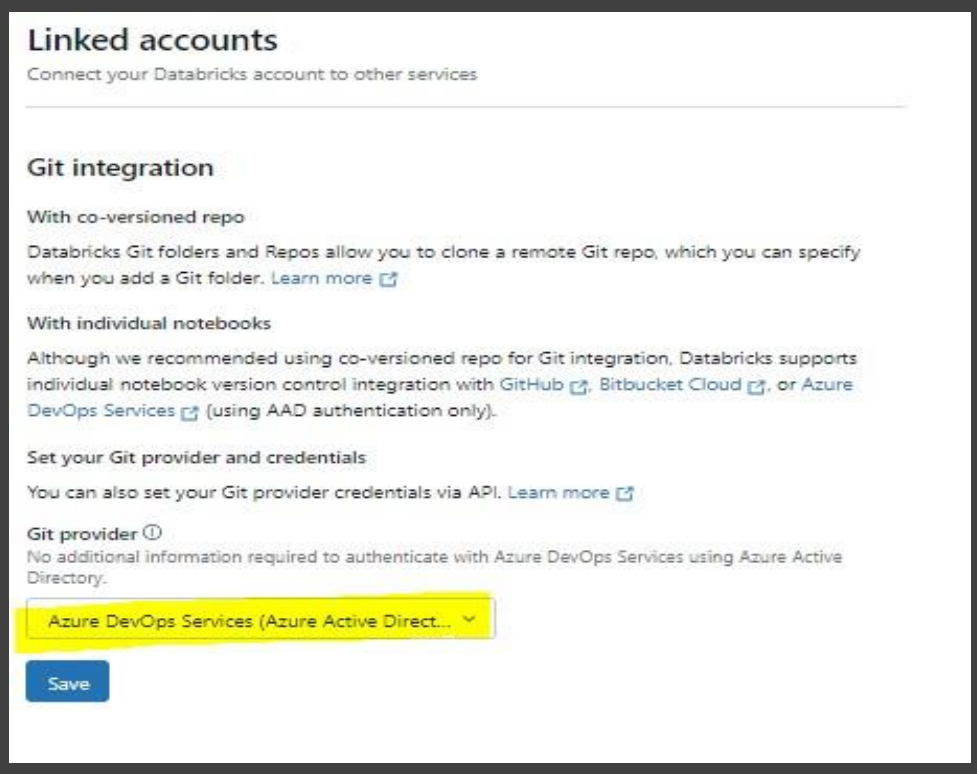

Create a repository
Repository type
Git
Repository name *
dbproject
Add a README
Add a .gitignore: None
Your repository will be initialized with a main branch.
Cancel
Create


Create a project to get started
Project name *
databricks-traffic
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.
Advanced
Create project

Q. How to link databricks and azure devops?

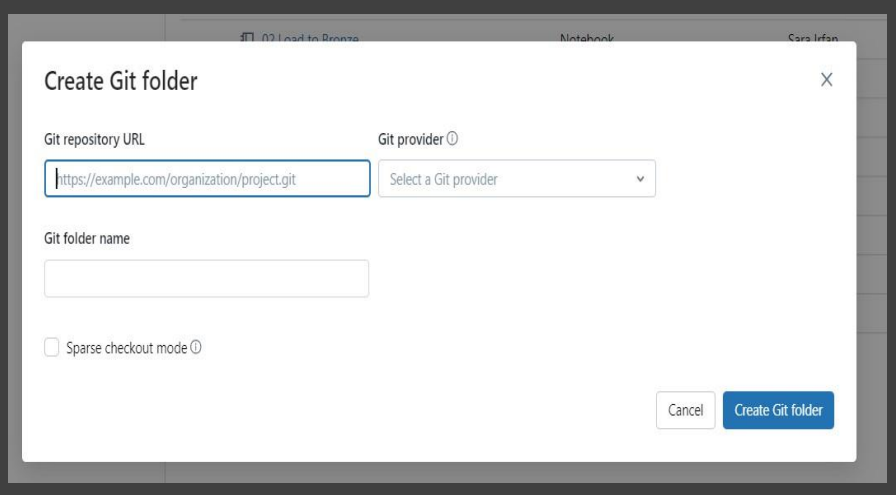
To link Databricks and Azure DevOps, use the Azure DevOps services to create a new pipeline. Configure the pipeline to use Databricks CLI or REST API for deploying and managing Databricks resources. Secure the connection with service principals and store secrets in Azure Key Vault

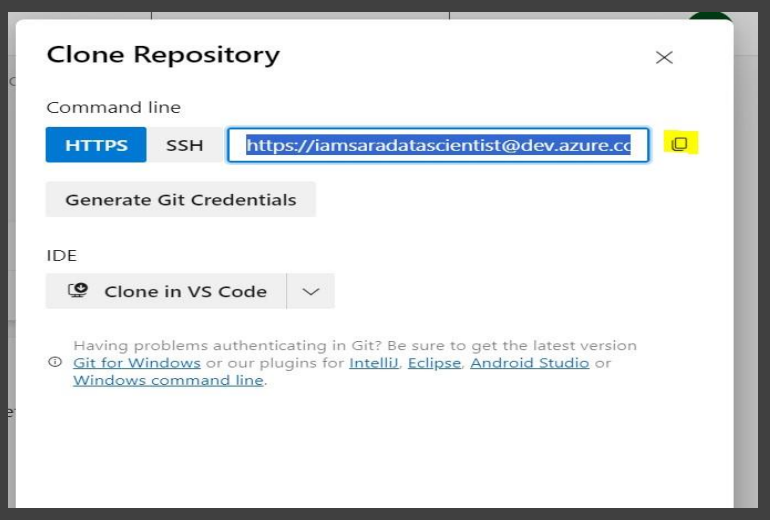


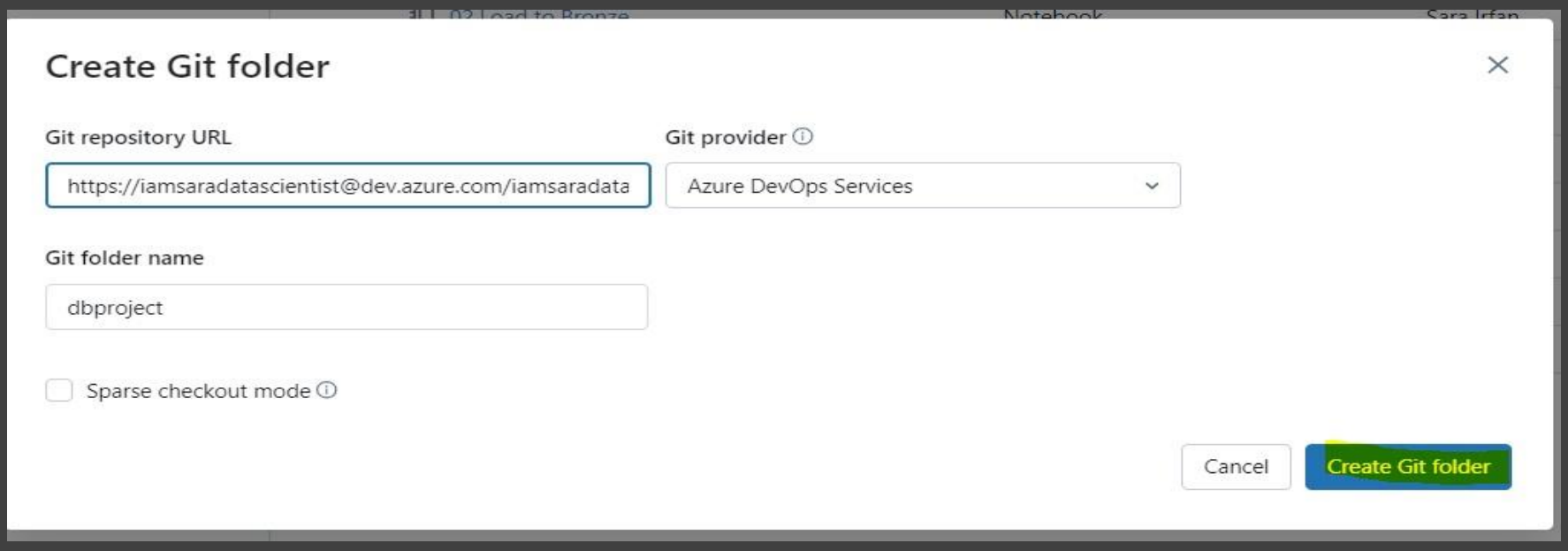


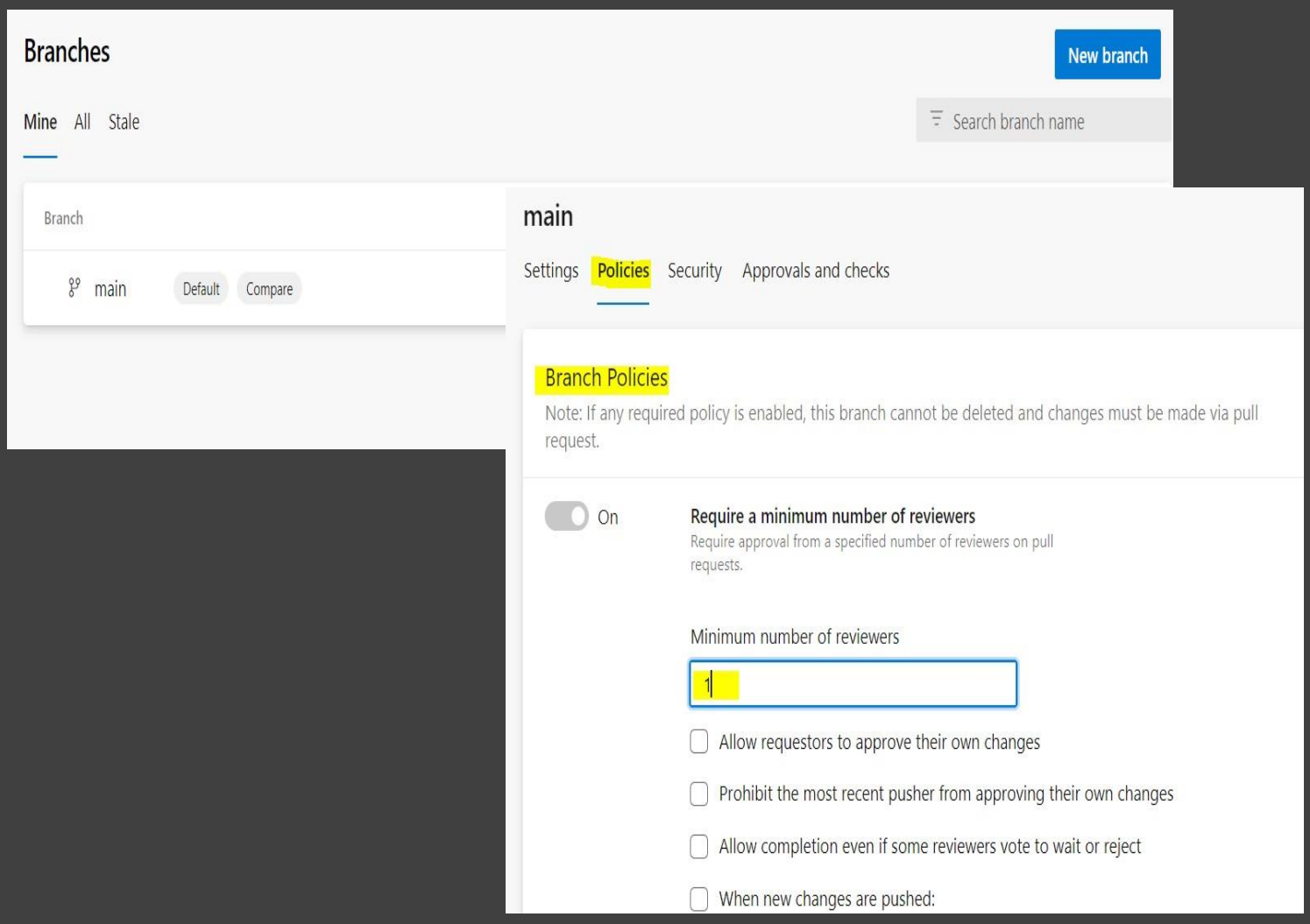
Q. How to add repo in Databricks?

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









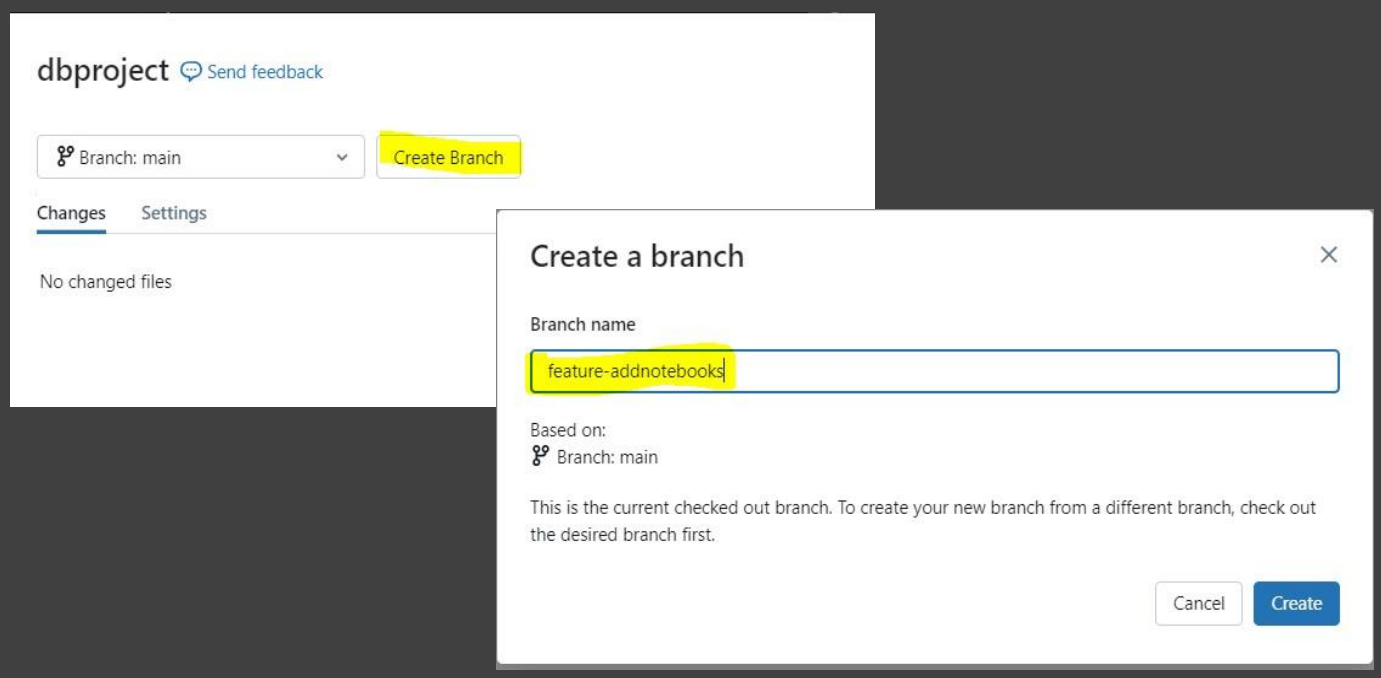
Q. How can we effectively set up and manage a CI/CD pipeline for deploying code changes from a feature branch to a live folder, and ensure smooth integration with the UAT environment using Azure DevOps and Azure Databricks?

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

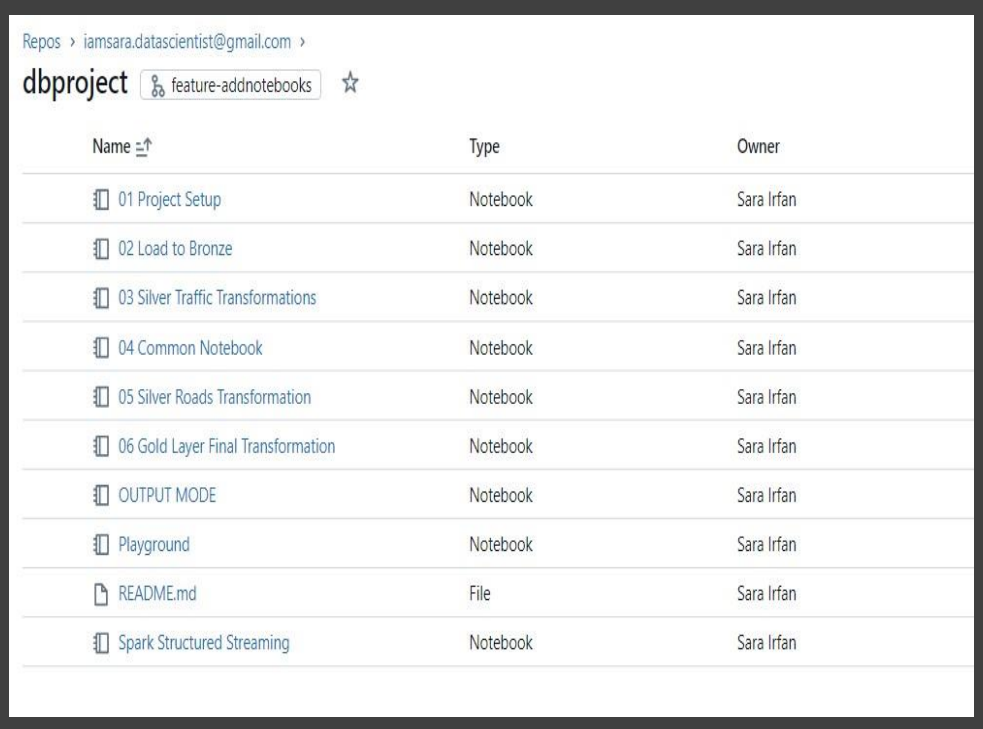
approach allows for automated and reliable updates, promoting seamless development and integration workflows.

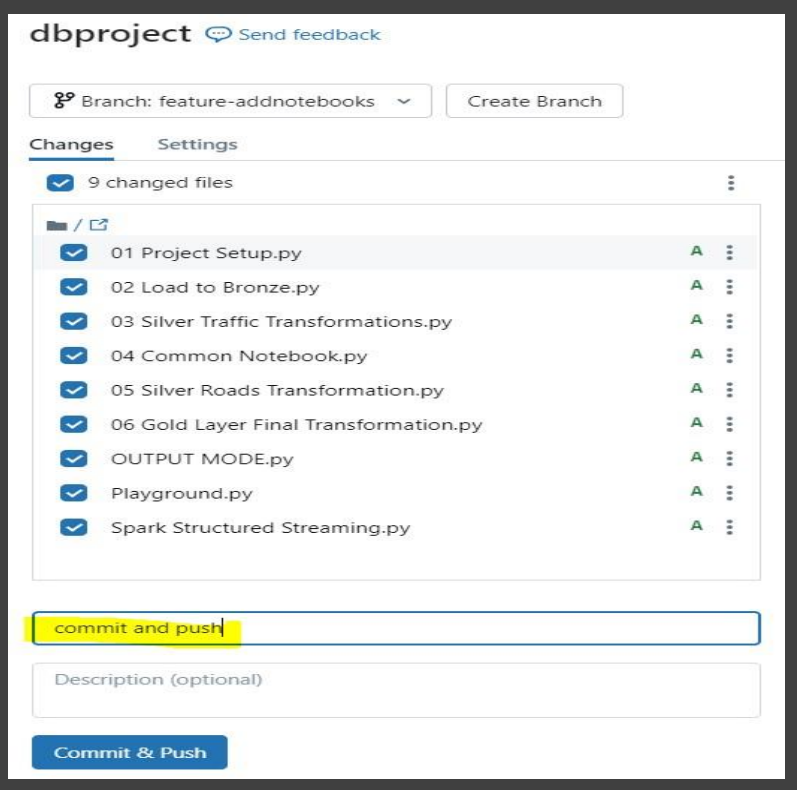
1. Feature Branch Creation and Pull Request:

- **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.

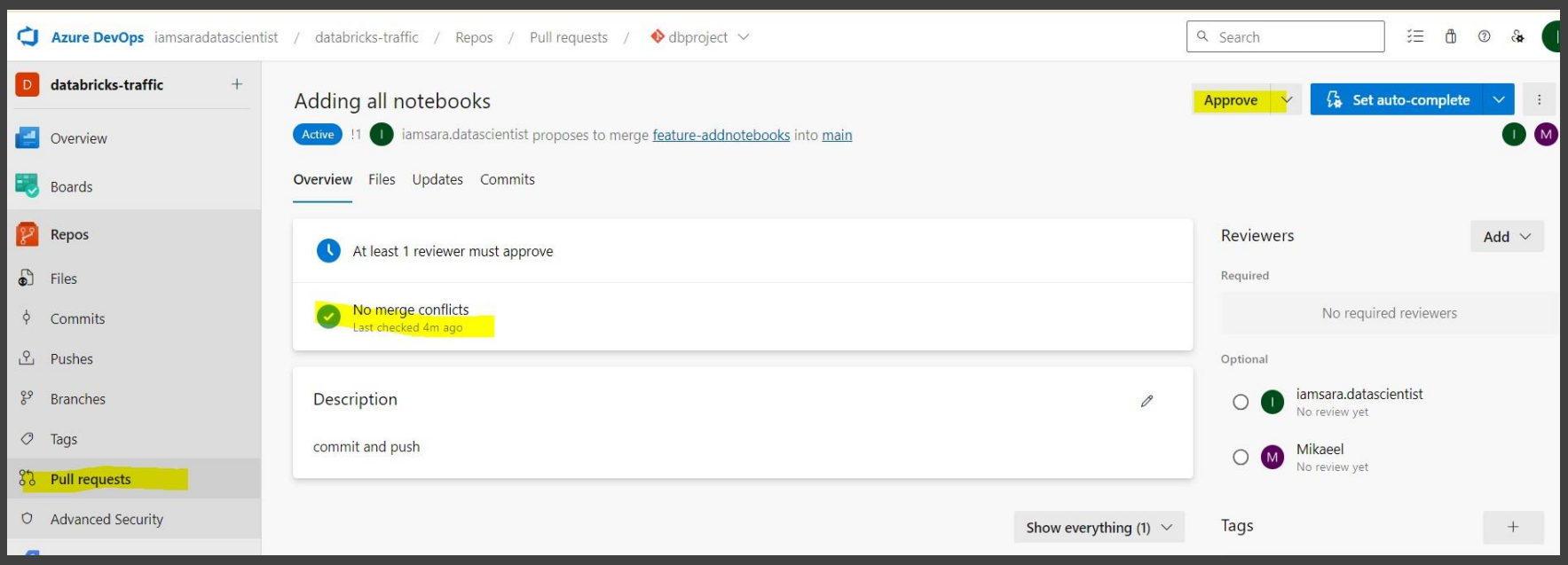


- **Commit Changes:**
  - Make necessary changes or additions to your code in the feature branch.
  - Commit these changes with a clear message.

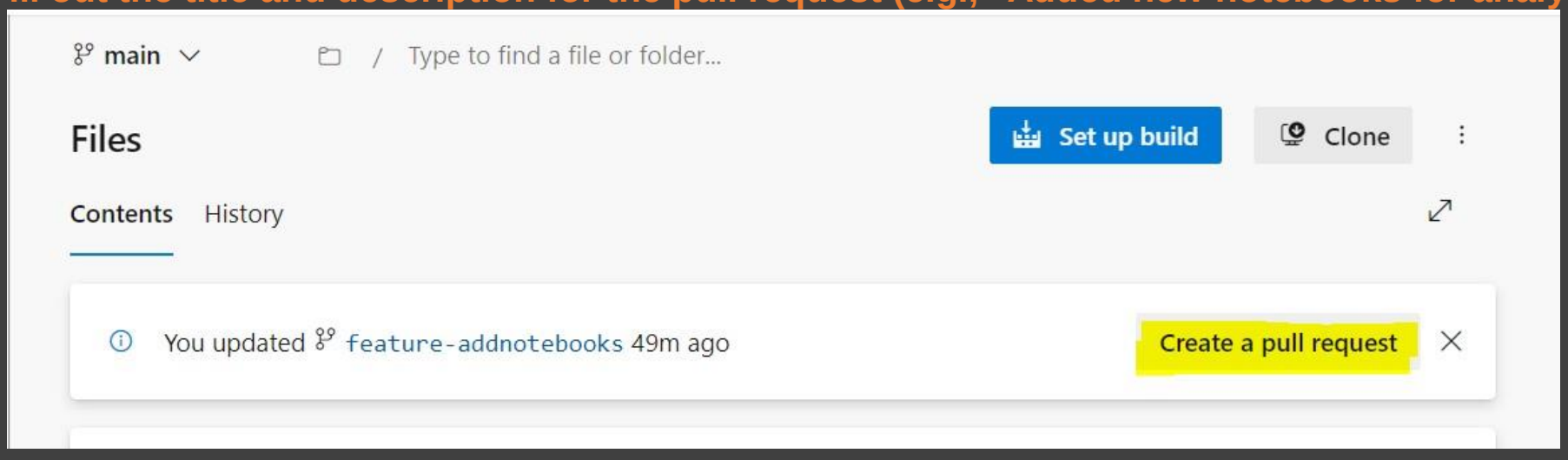




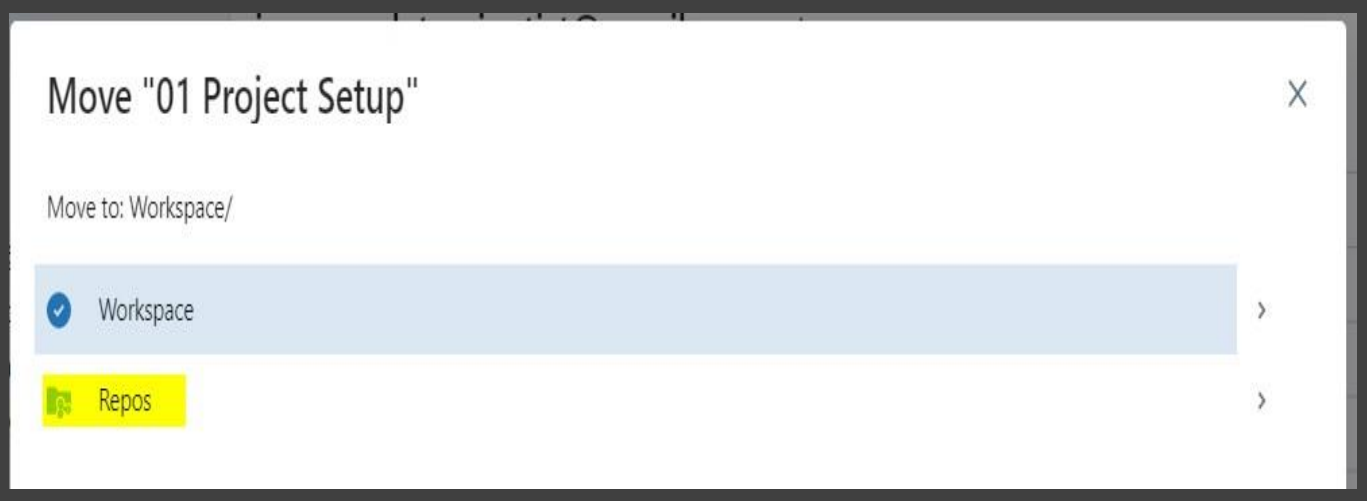
- **Create a Pull Request:**
  - Go to the "Pull Requests" section and click on "New Pull Request."
  - Set the source branch to your feature branch and the target branch to main.



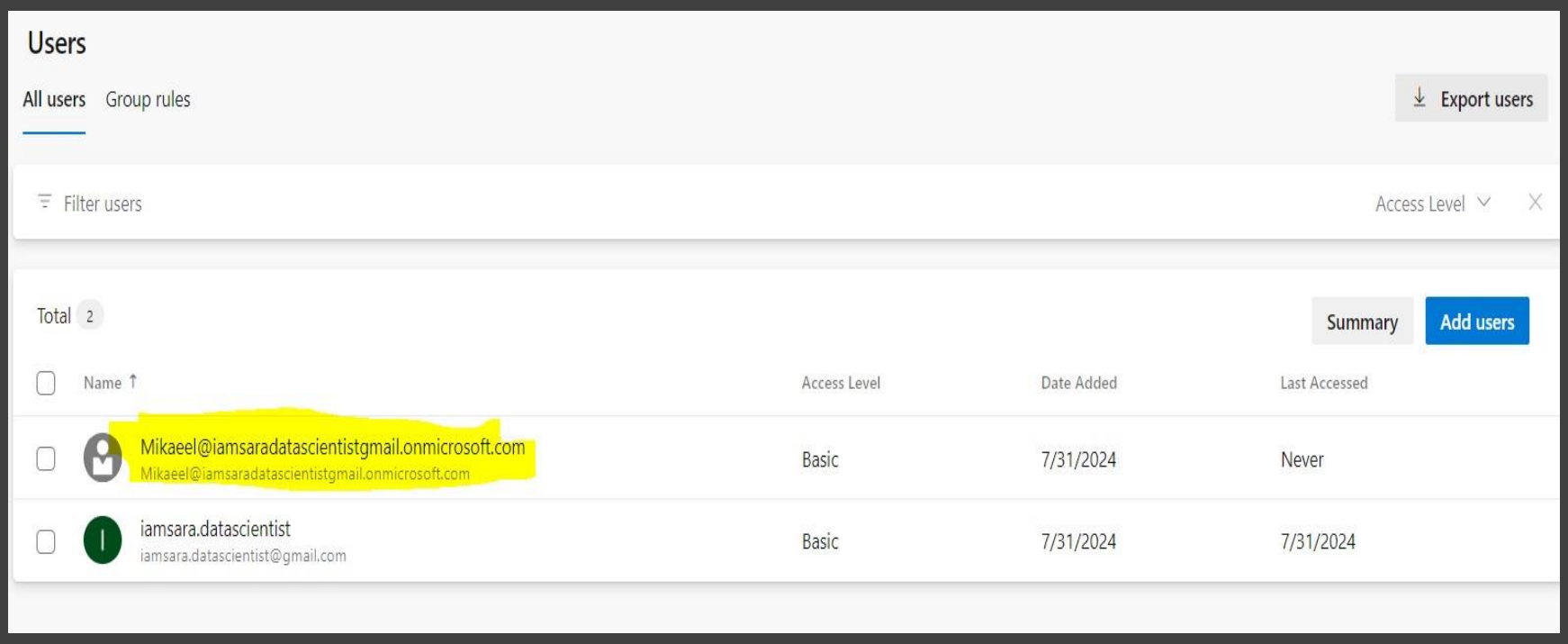
- **Fill out the title and description for the pull request (e.g., "Added new notebooks for analysis").**

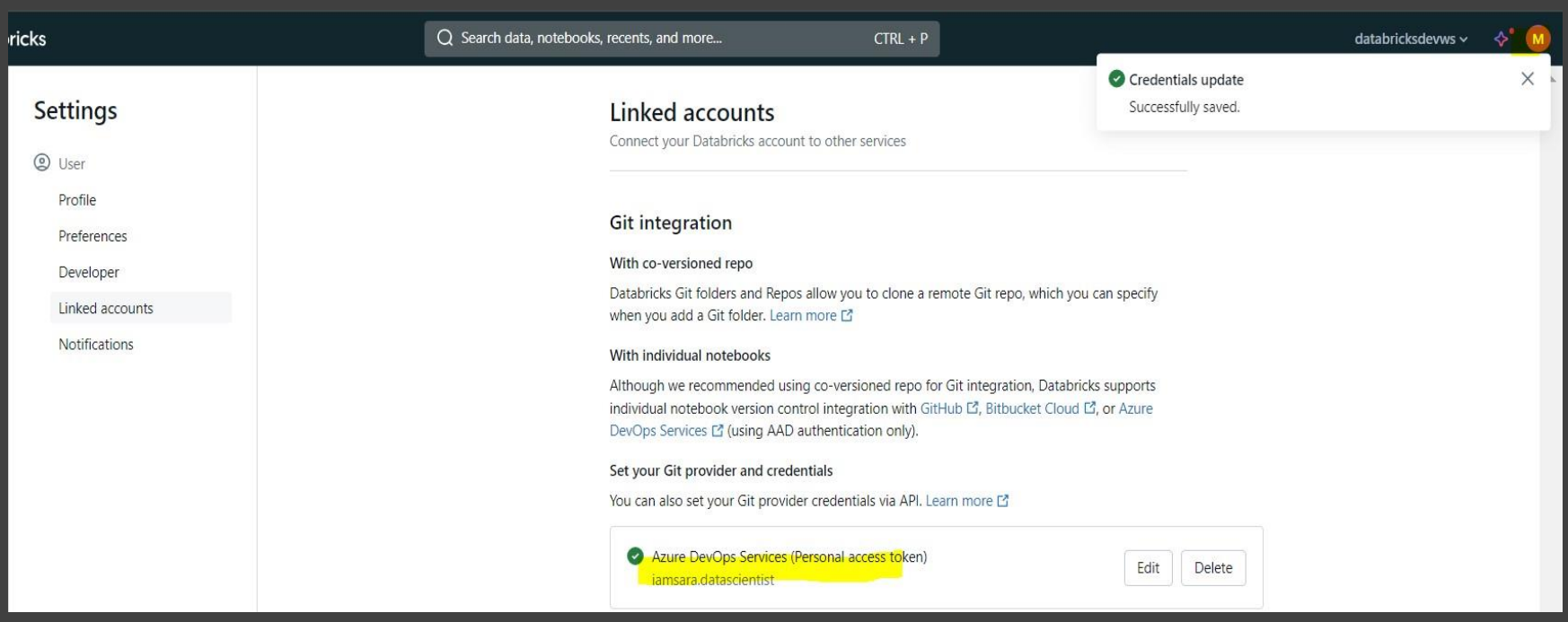


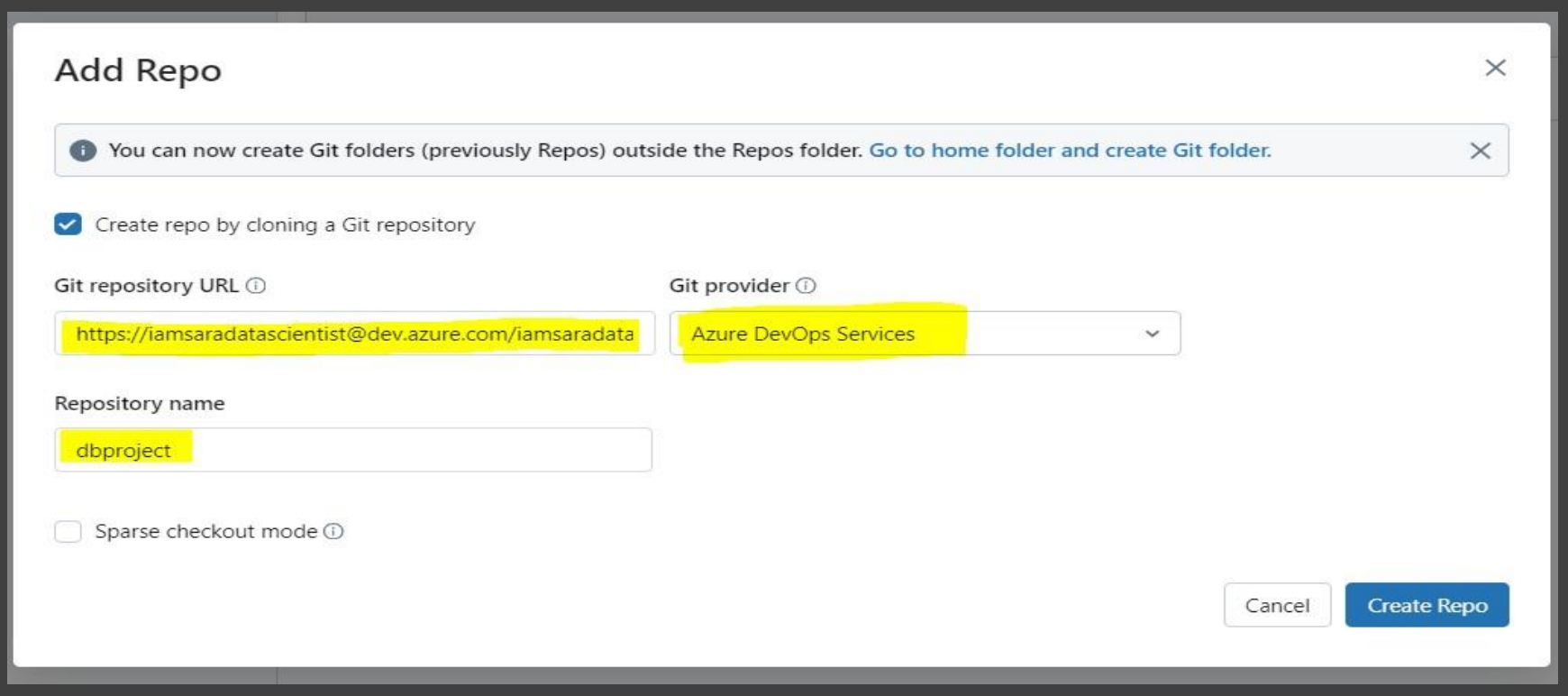
- **Reviewer and Approvals:**
- **Assign reviewers who are responsible for approving changes (e.g., a senior developer or team lead).**
- **Reviewers will assess the changes and provide feedback. The pull request must be approved before it can be merged.**

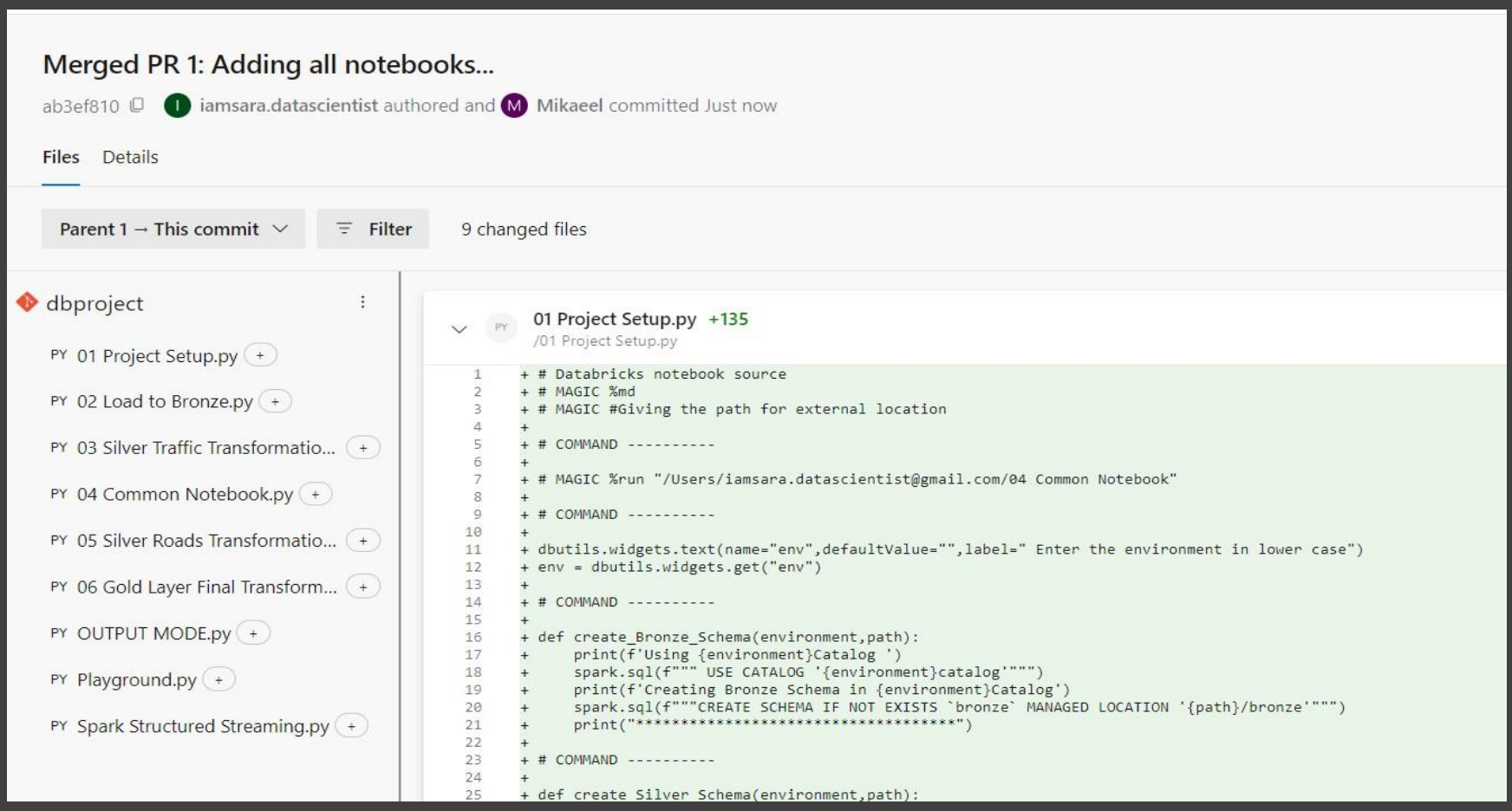


Q. How to add another user with same Working?

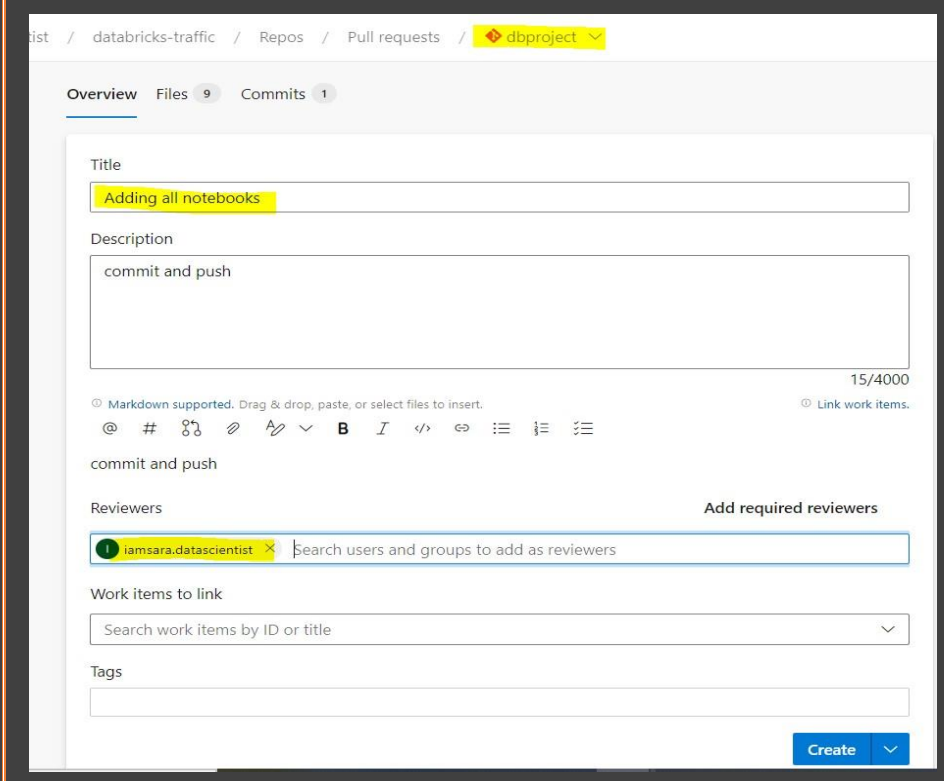

Users
All users
Group rules
Export users
Filter users
Access Level
Total 2
Summary
Add users
Name
Access Level
Date Added
Last Accessed
Mikaeel@iamsaradatascientistgmail.onmicrosoft.com
Mikaeel@iamsaradatascientistgmail.onmicrosoft.com
Basic
7/31/2024
Never
iamsara.datascientist
iamsara.datascientist@gmail.com
Basic
7/31/2024
7/31/2024

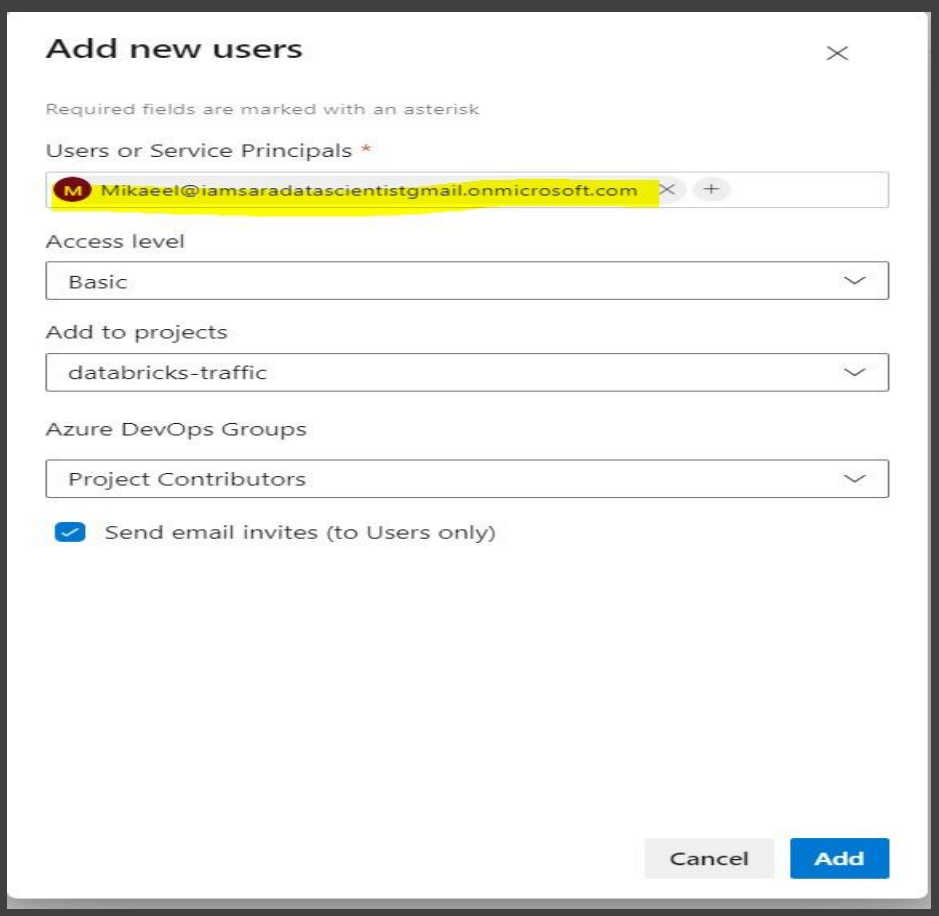

Search data, notebooks, recents, and more... CTRL + P
databricksdevws
Credentials update
Successfully saved.
Settings
User
Profile
Preferences
Developer
Linked accounts
Notifications
Linked accounts
Connect your Databricks account to other services
Git integration
With co-versioned repo
Databricks Git 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

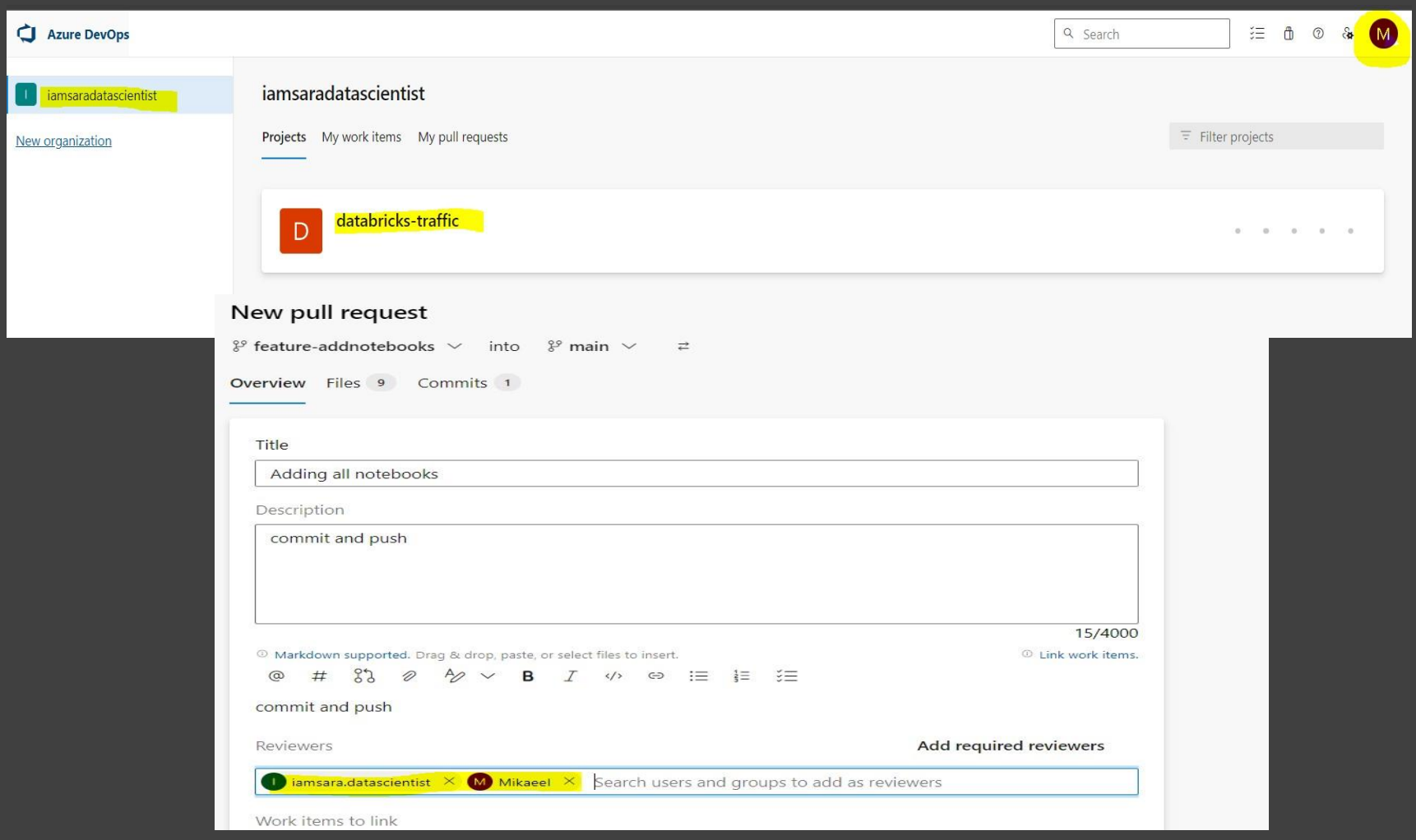

Add Repo
You can now create Git folders (previously Repos) outside the Repos folder. Go to home folder and create Git folder.
Create repo by cloning a Git repository
Git repository URL
Git provider
https://iamsaradatascientist@dev.azure.com/iamsaradata
Azure DevOps Services
Repository name
dbproject
Sparse checkout mode
Cancel
Create Repo



Creating feature branch and adding users to the service principal







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.**

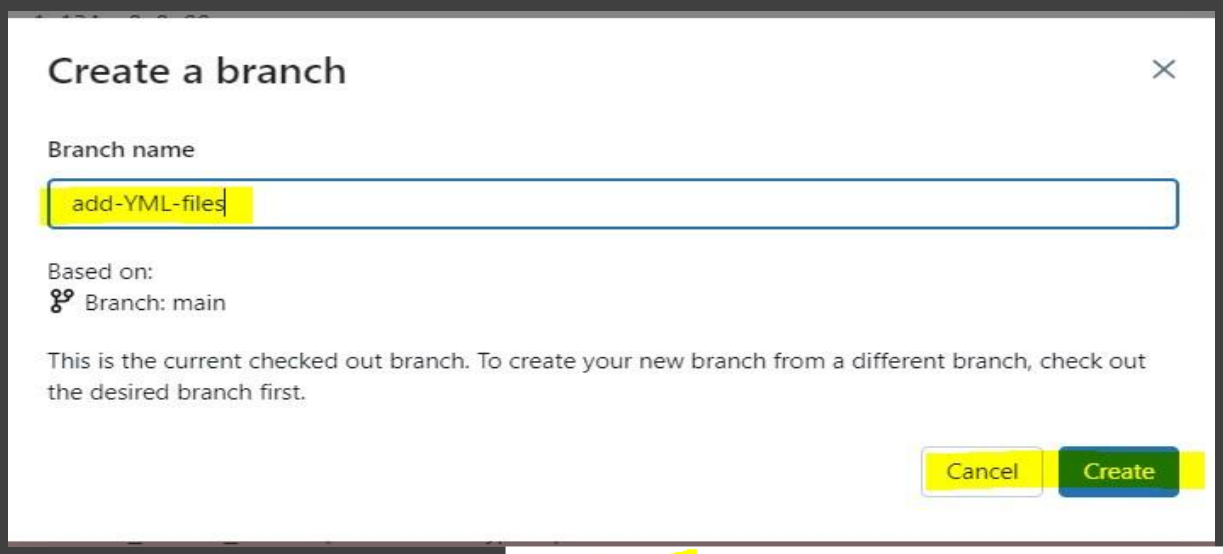
3. CI/CD Pipeline Configuration:

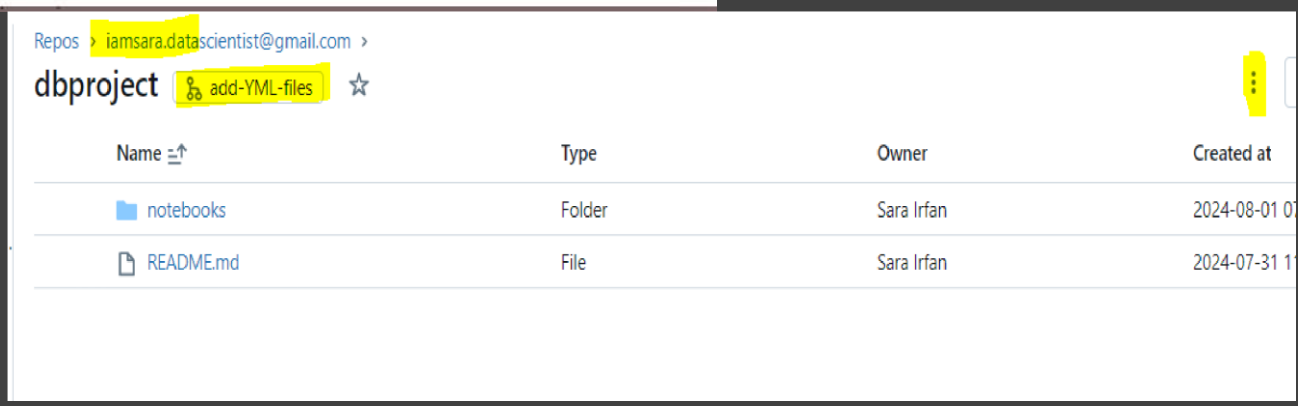
Using Yaml and PowerShell Scripts:

- **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.**

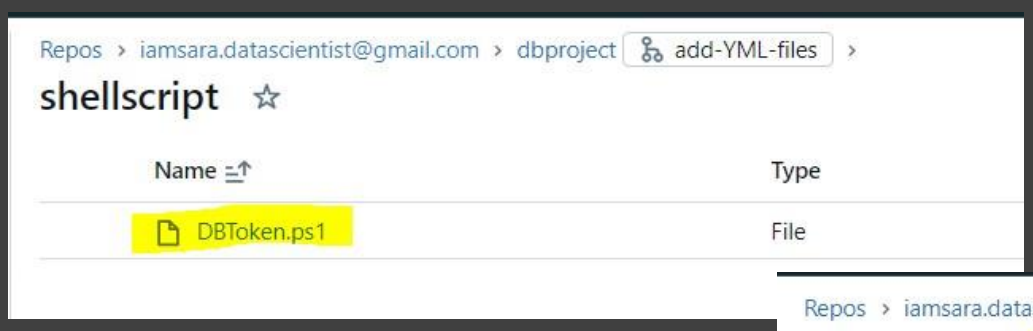
4. Integration with Databricks:

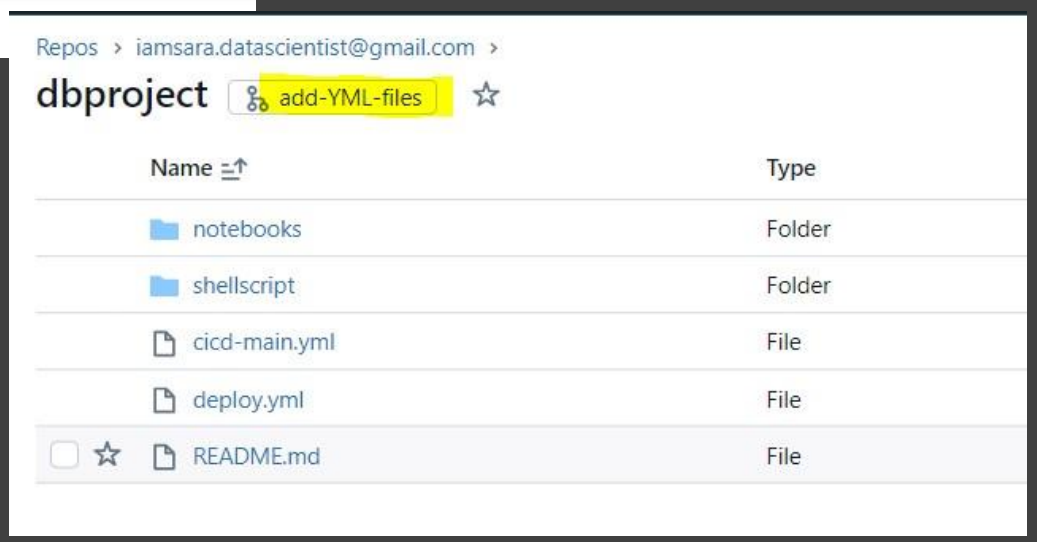
- **Uploading and Reviewing Yaml Files:**





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

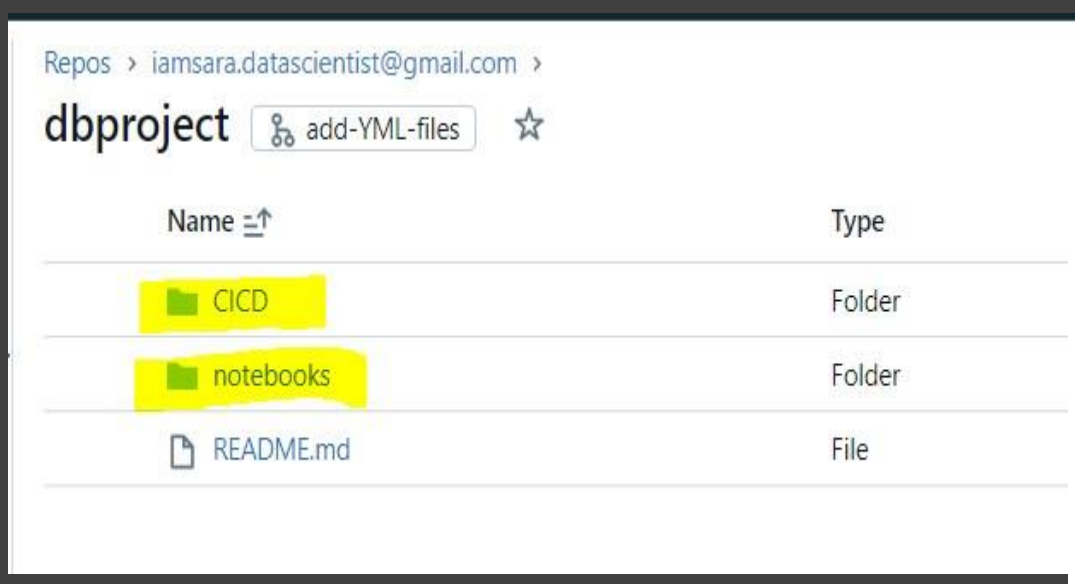


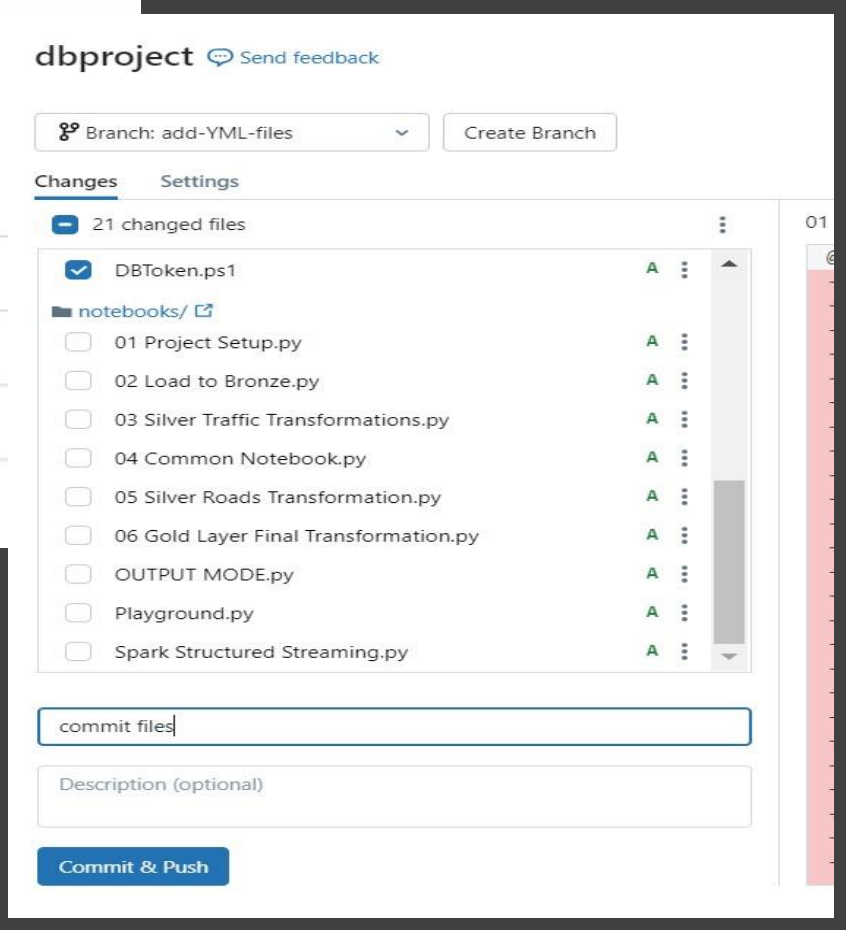


- **Configuring Notebooks and External Locations:**
  - **Set up external locations in Databricks for different stages (e.g., bronze-UAT, silver-UAT).**
  - **Ensure notebooks are configured to use these locations and handle data appropriately.**

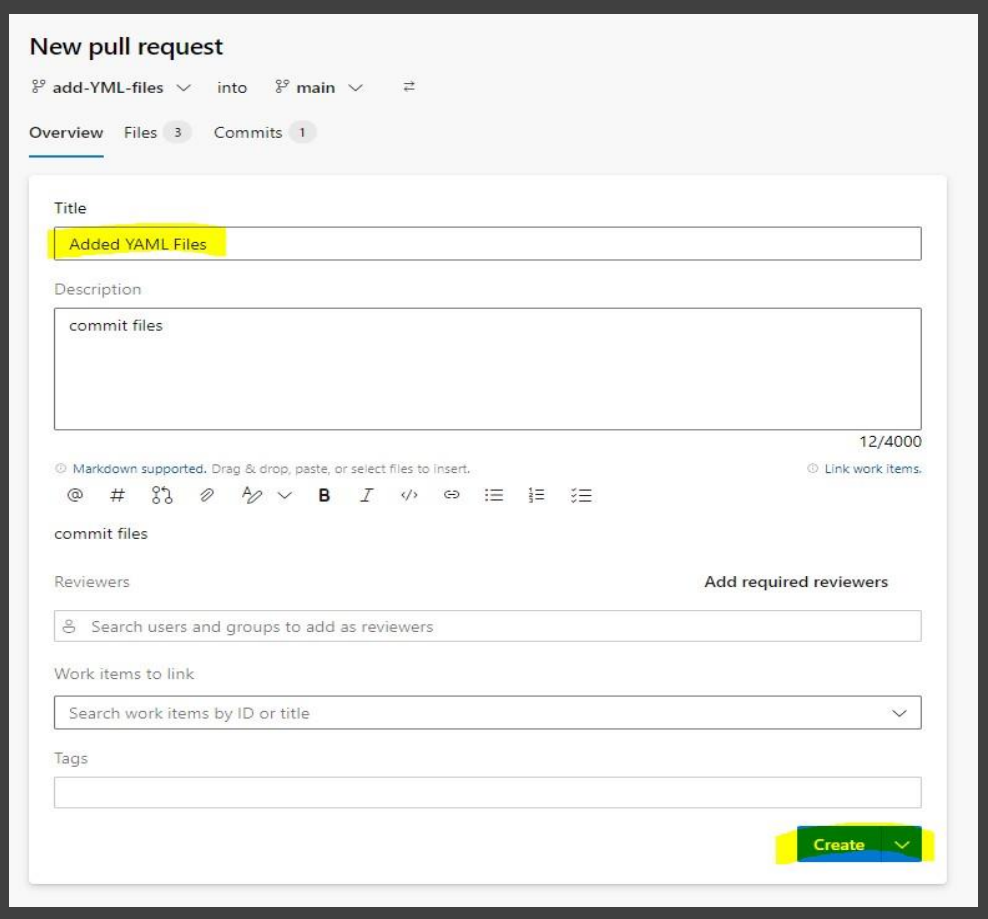
5. Testing and Verification:

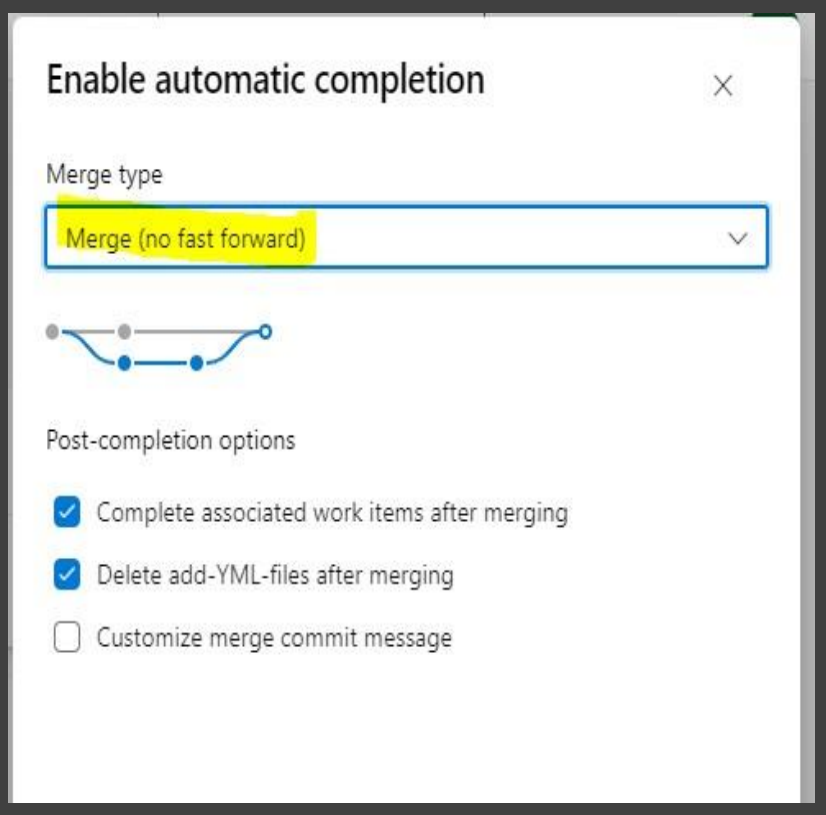
- **Create a cluster in Databricks for testing. Ensure it's configured correctly for the UAT environment.**
- **Check that UAT catalogs and external locations are set up and pointing to the correct data sources.**
- **Verify that all notebooks use the correct paths and configurations for UAT.**
- **Upload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

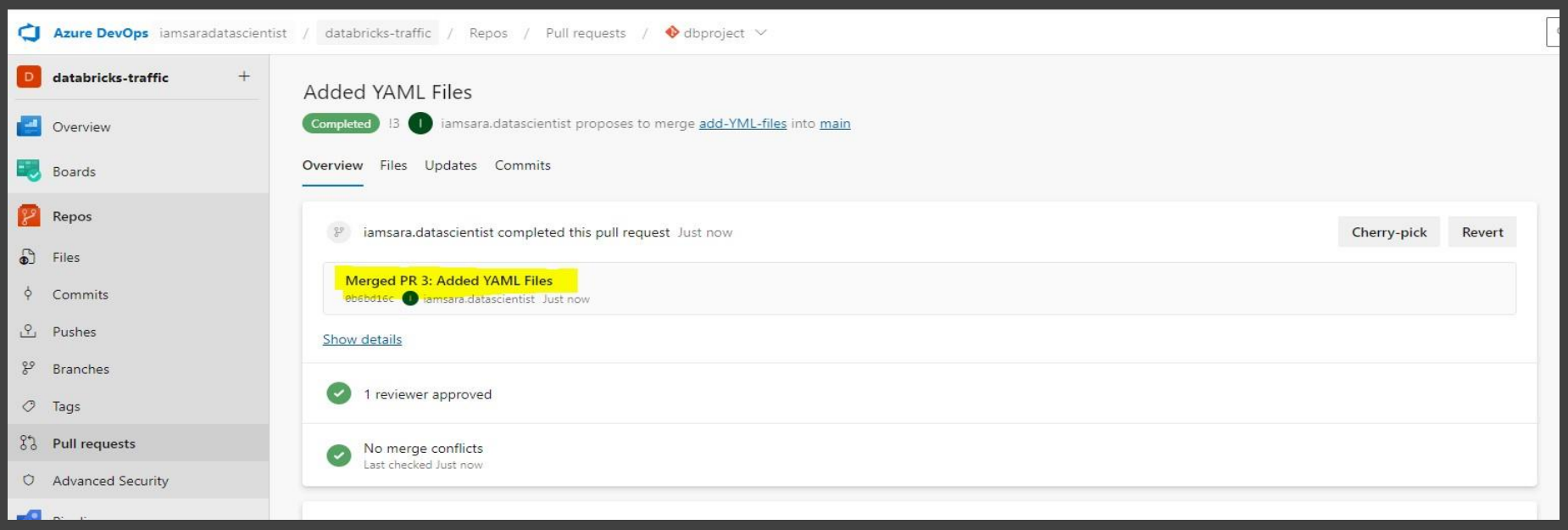




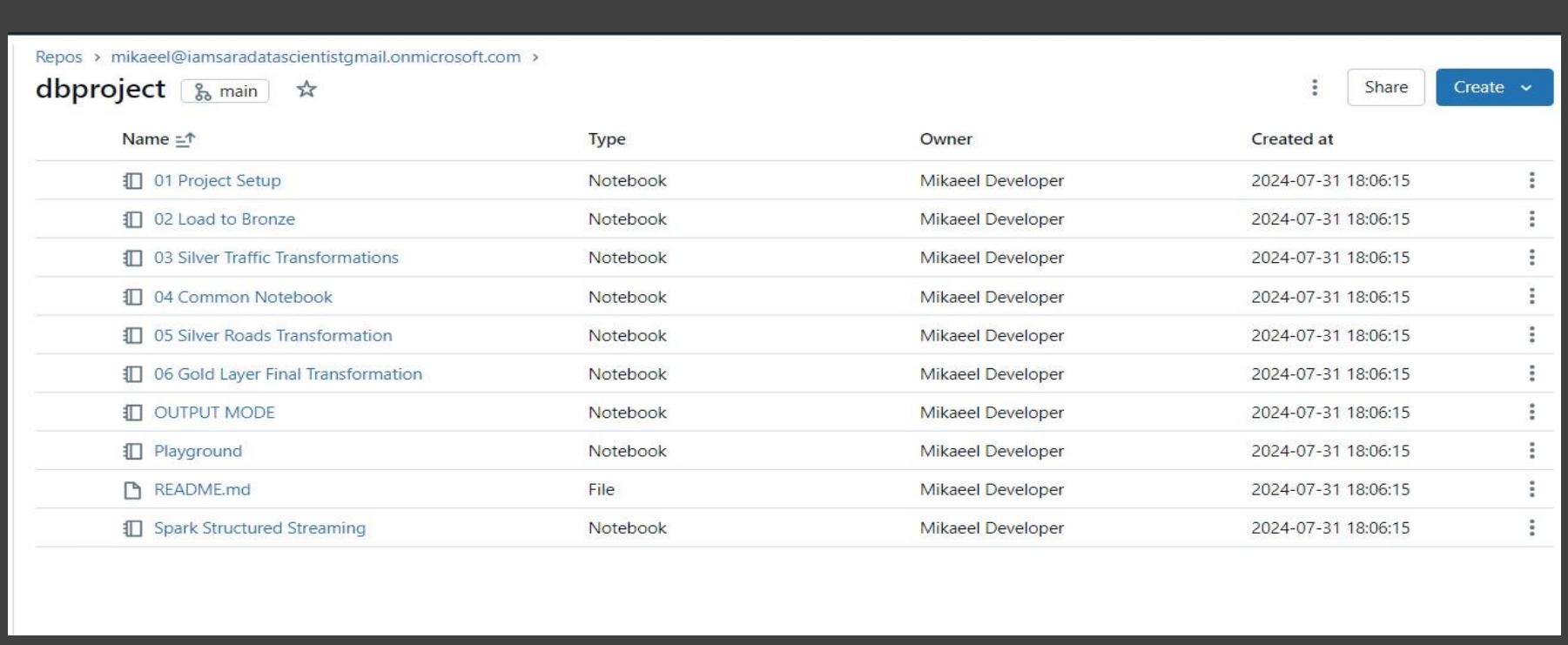
6. **Permissions and Access Control:**
   - Ensure that only authorized users have access to the live folder and Databricks workspace.
   - For users who need to manage or view the live folder, assign them appropriate permissions (e.g., admin roles).
7. **Deployment to Production:**
   - Configure the 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.



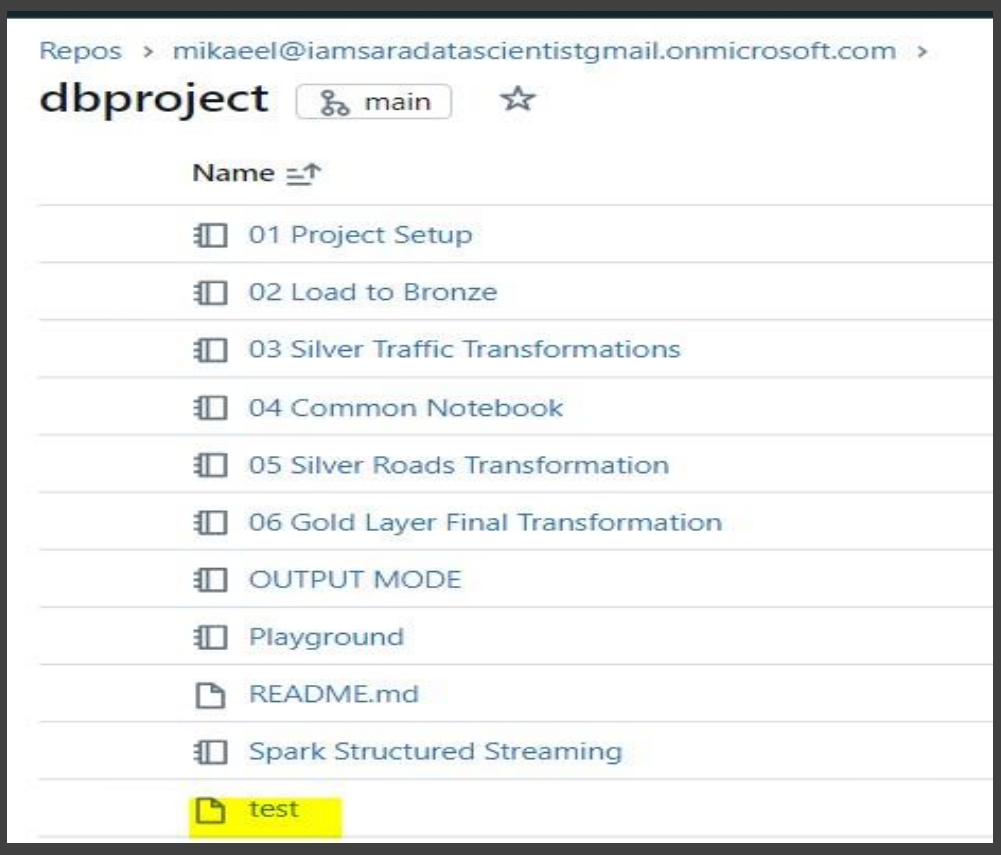




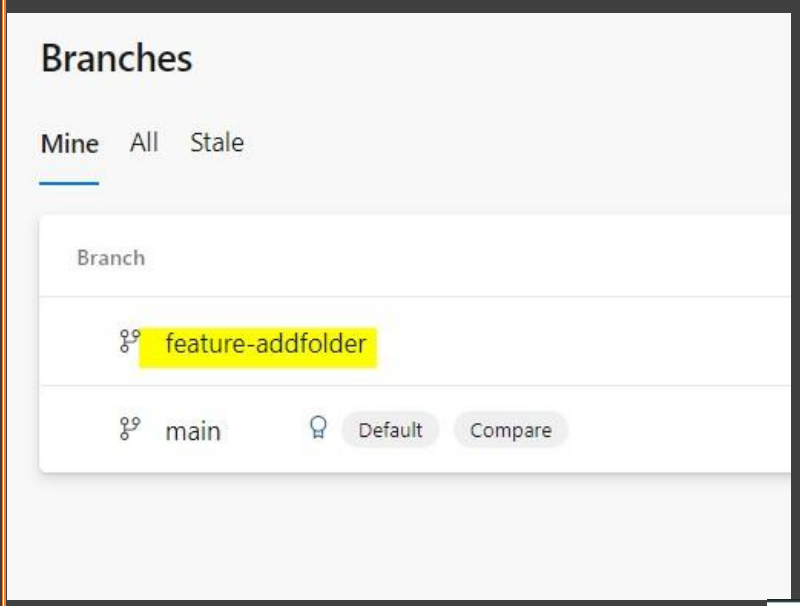
Working in Mikael's Account, creating feature branch, commit and push, then pull request to implement changes. Furthermore, pull request the main feature branch

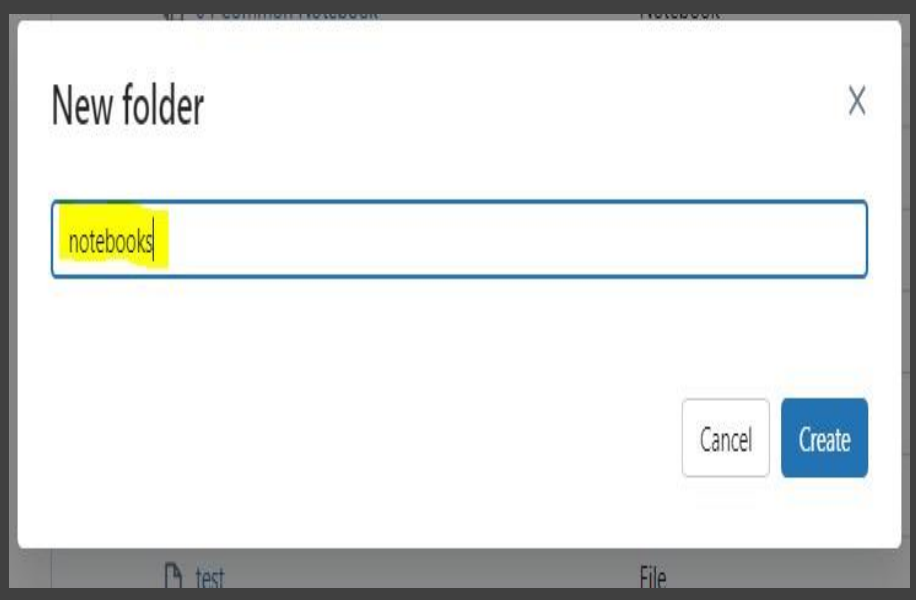


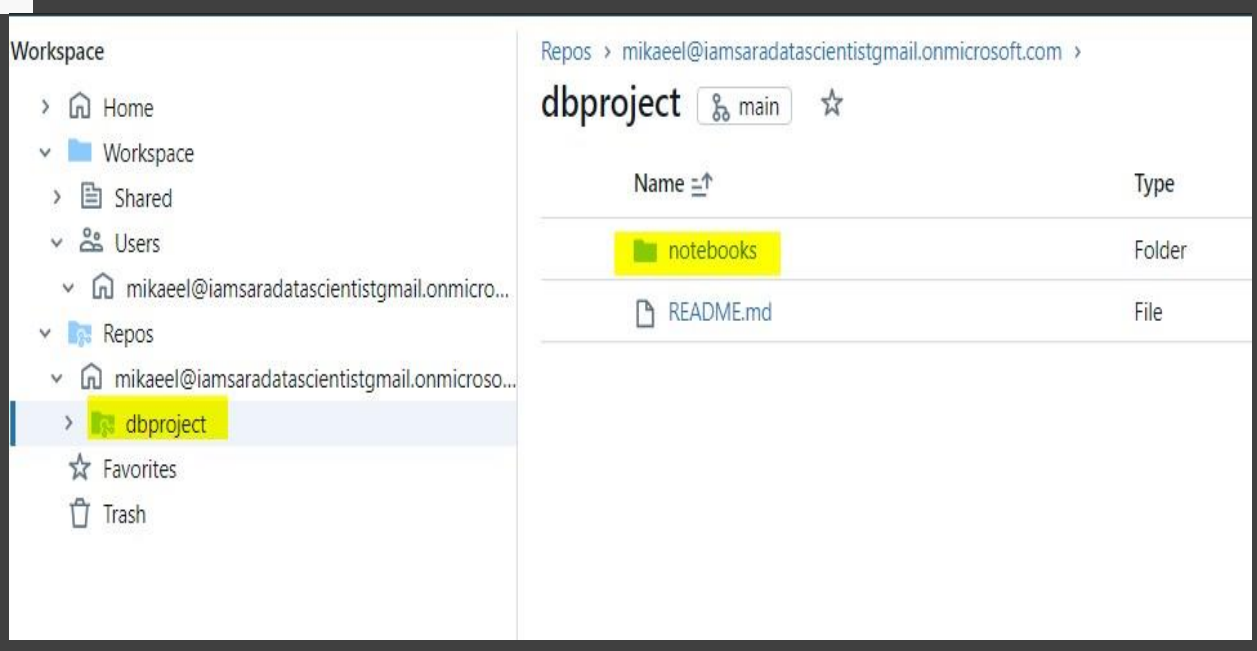
Create a folder named test



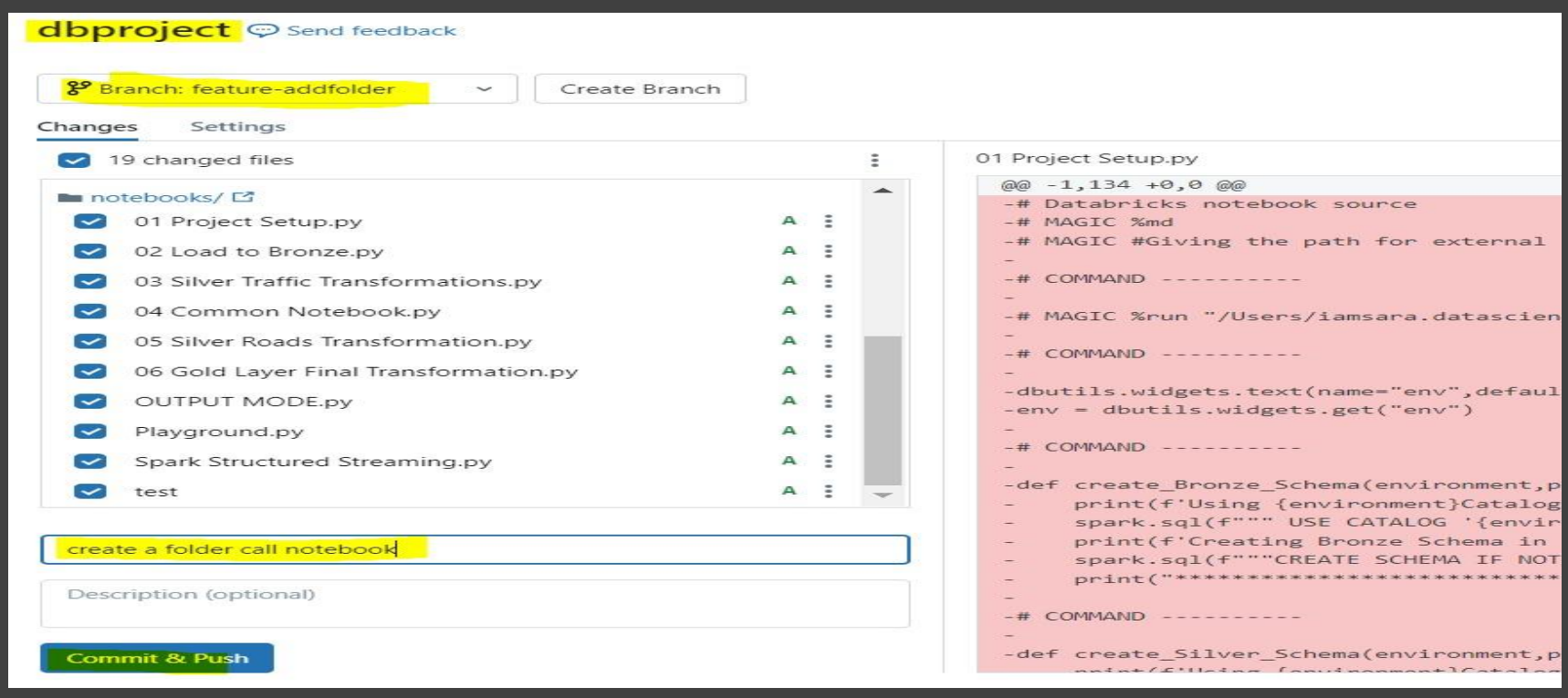
Create a feature branch, feature-addfolder, new folder named as notebooks, in the dbprojct of repos



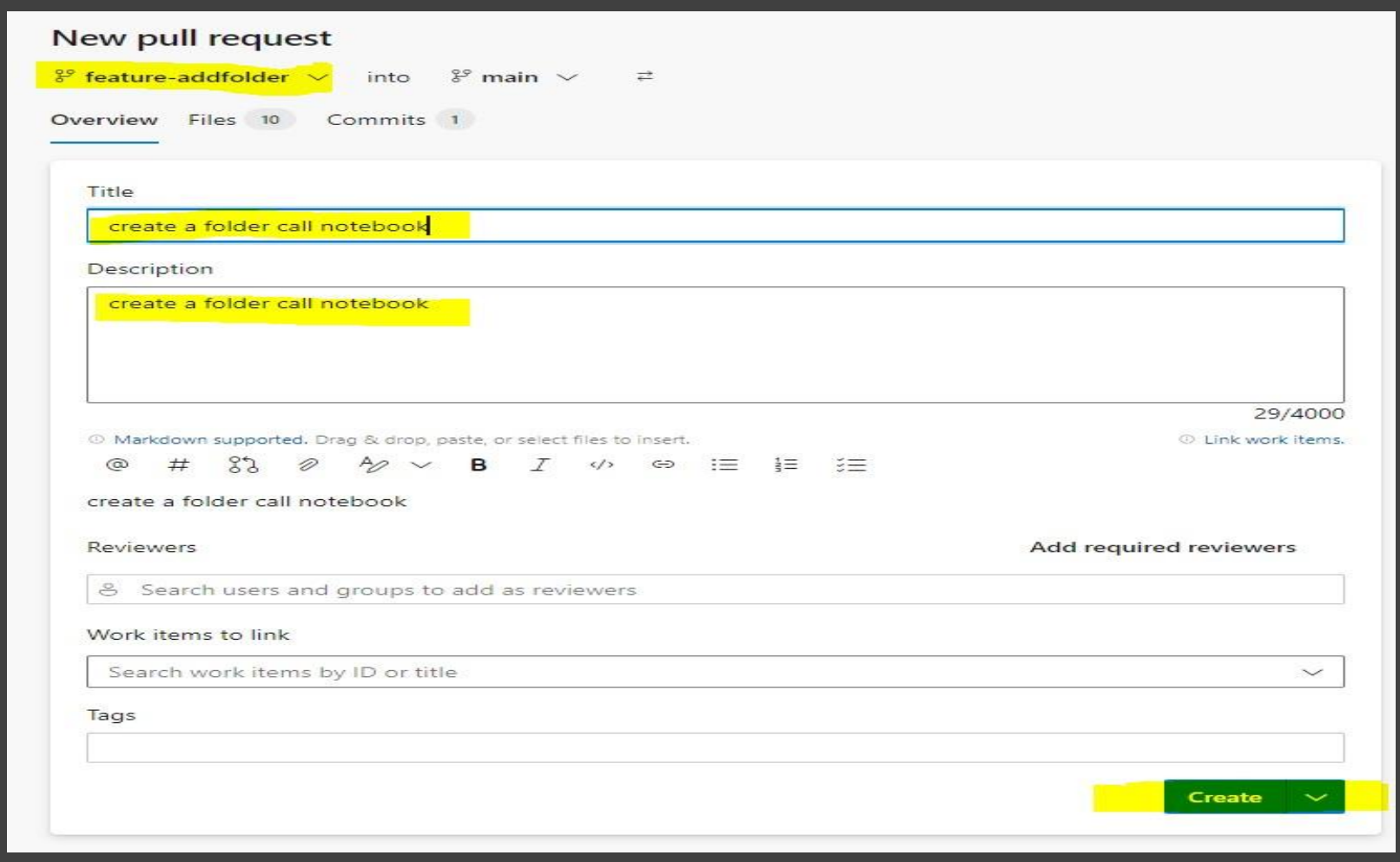




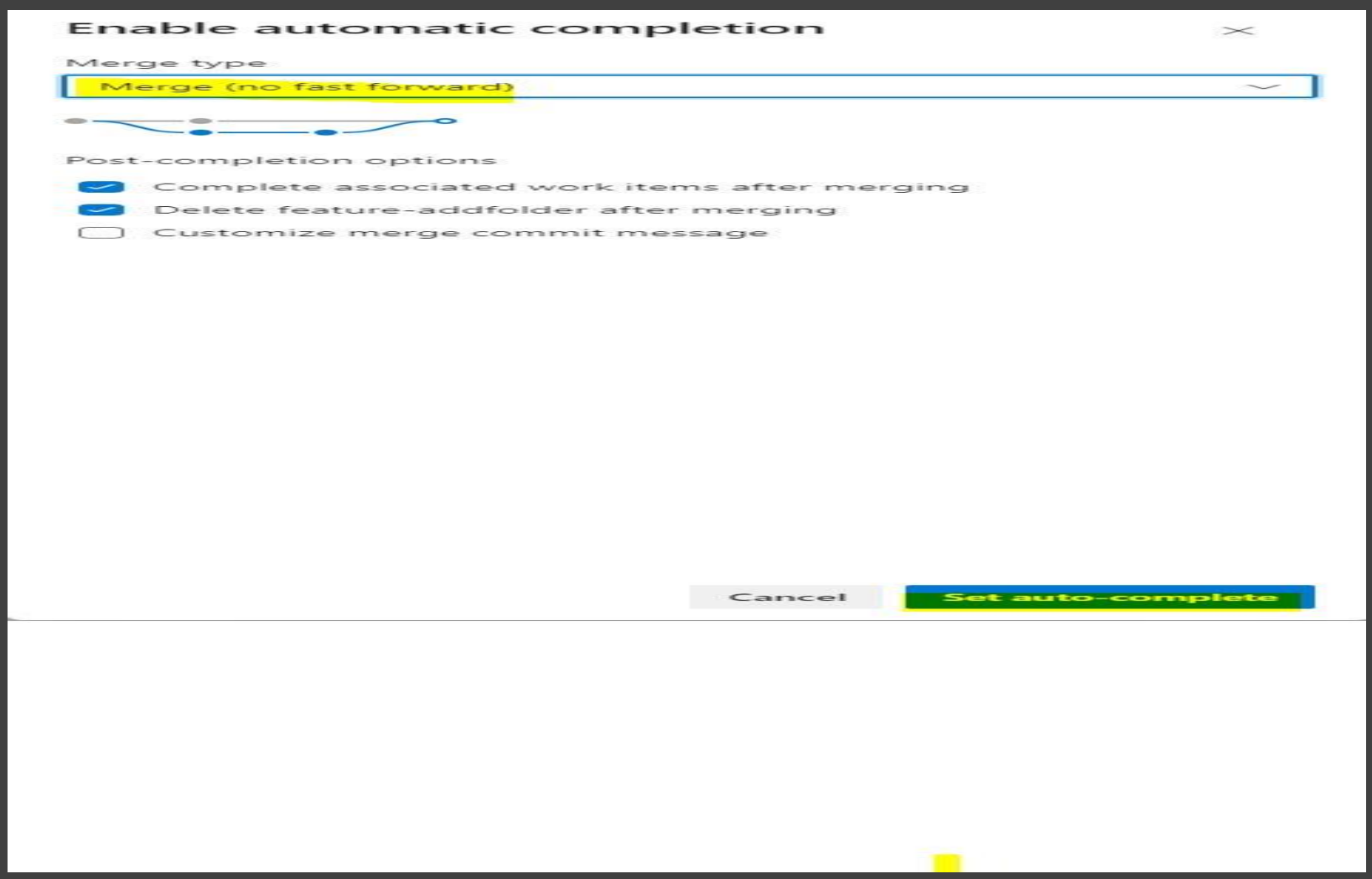
Commit and Push



Create a pull request

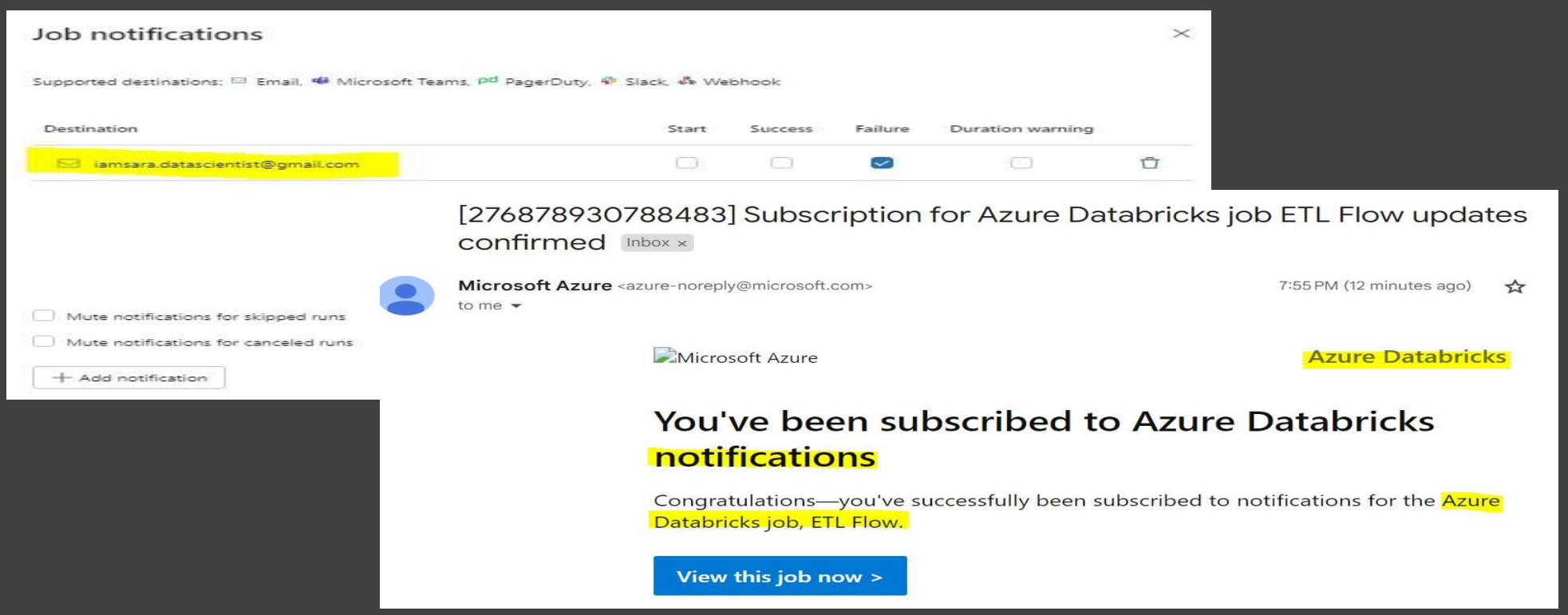


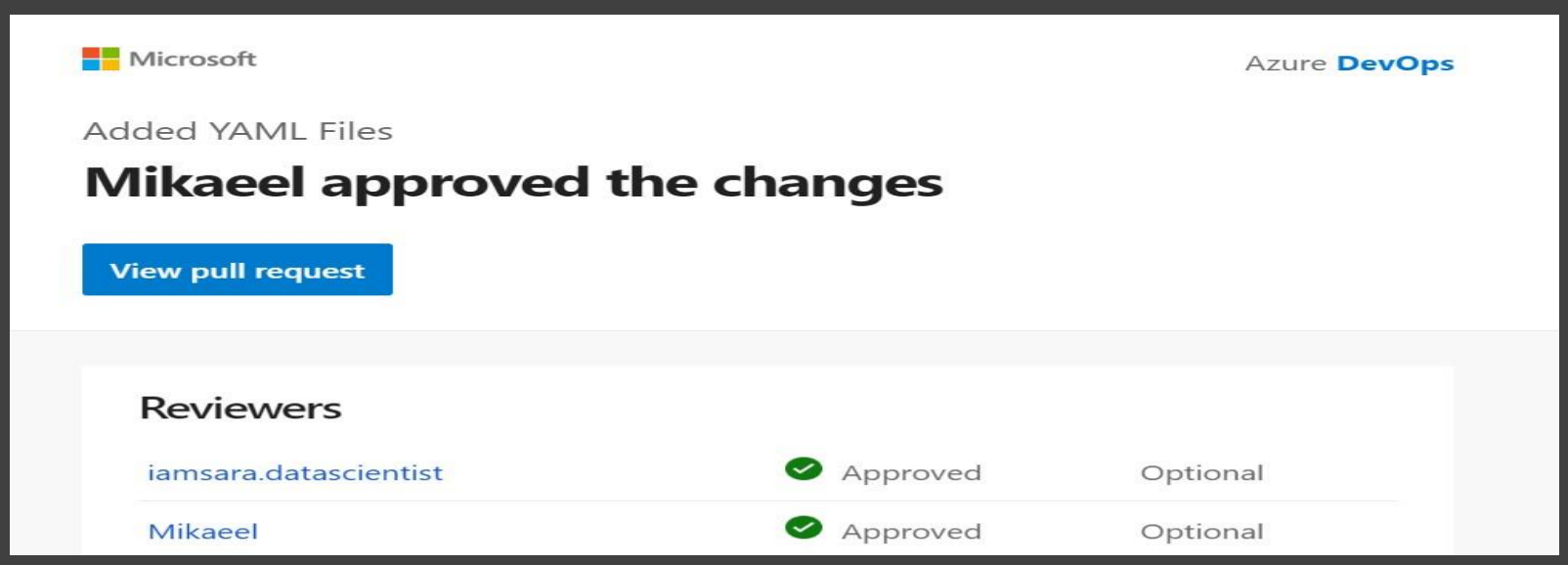
Approve from Sara's account(Workspace Admin), the click on approve and complete from Mikael's Account (a developer), then click on auto complete from Admin's account


Enable automatic completion
Merge type
Merge (no fast forward)
Post-completion options
Complete associated work items after merging
Delete feature-addfolder after merging
Customize merge commit message
Cancel
Set auto-complete

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.

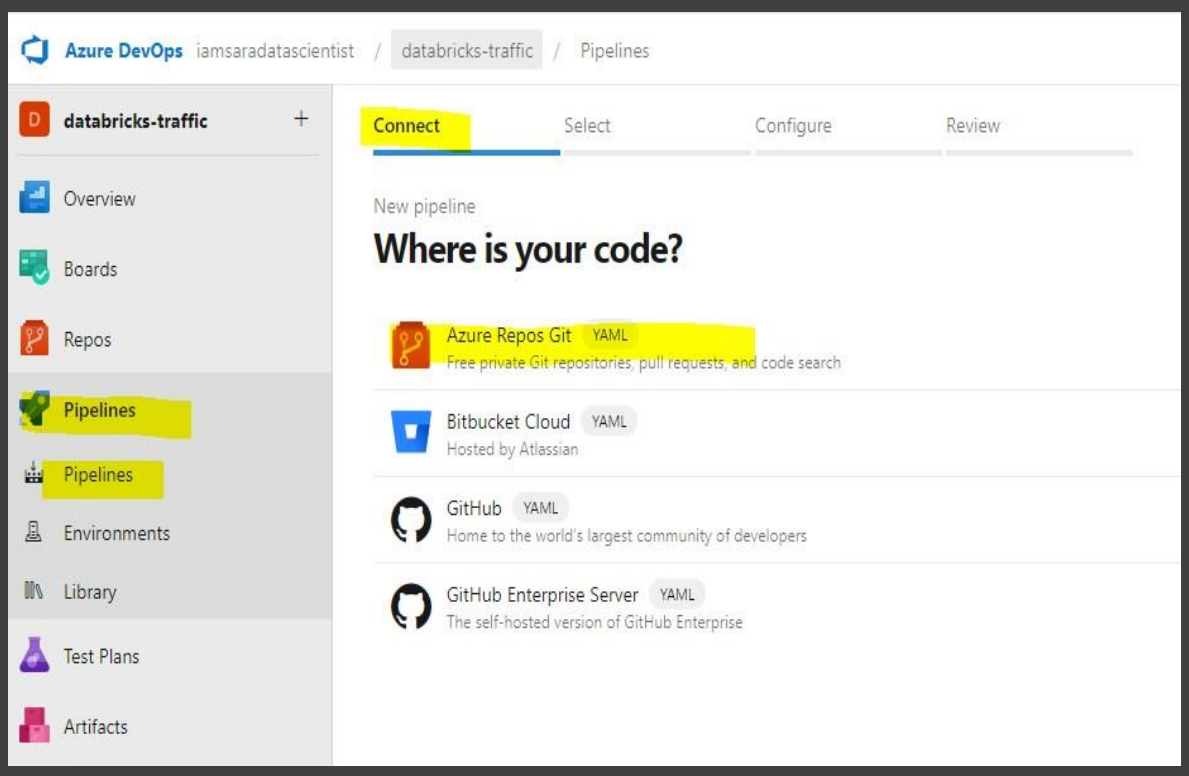


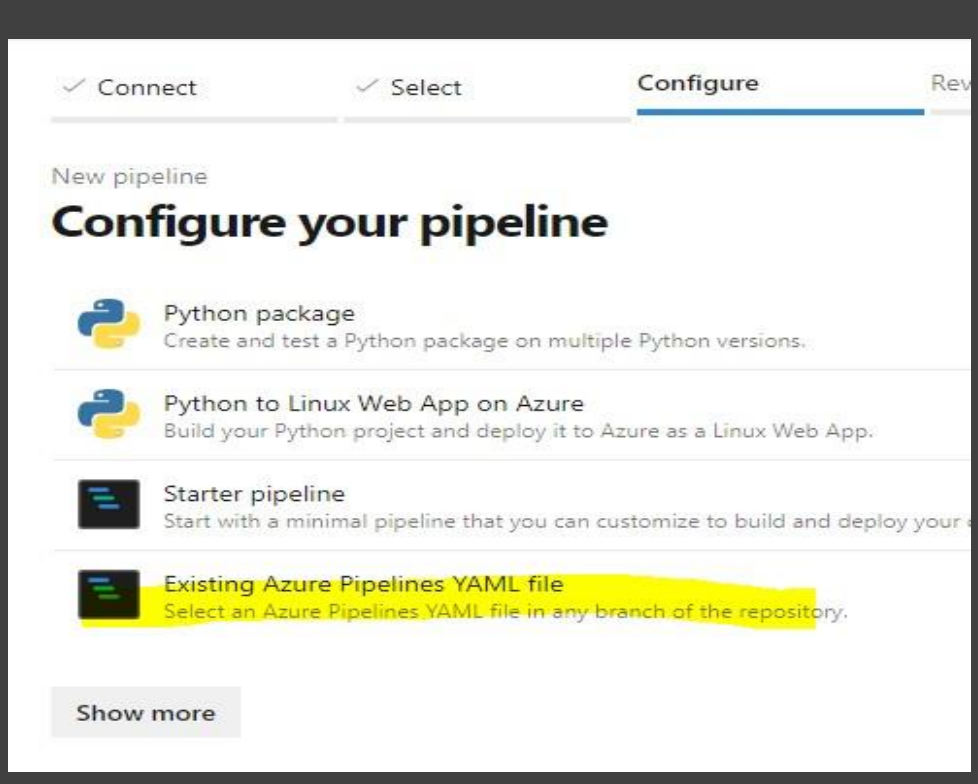


Sara Irfan (sara.data.insights@gmail.com)

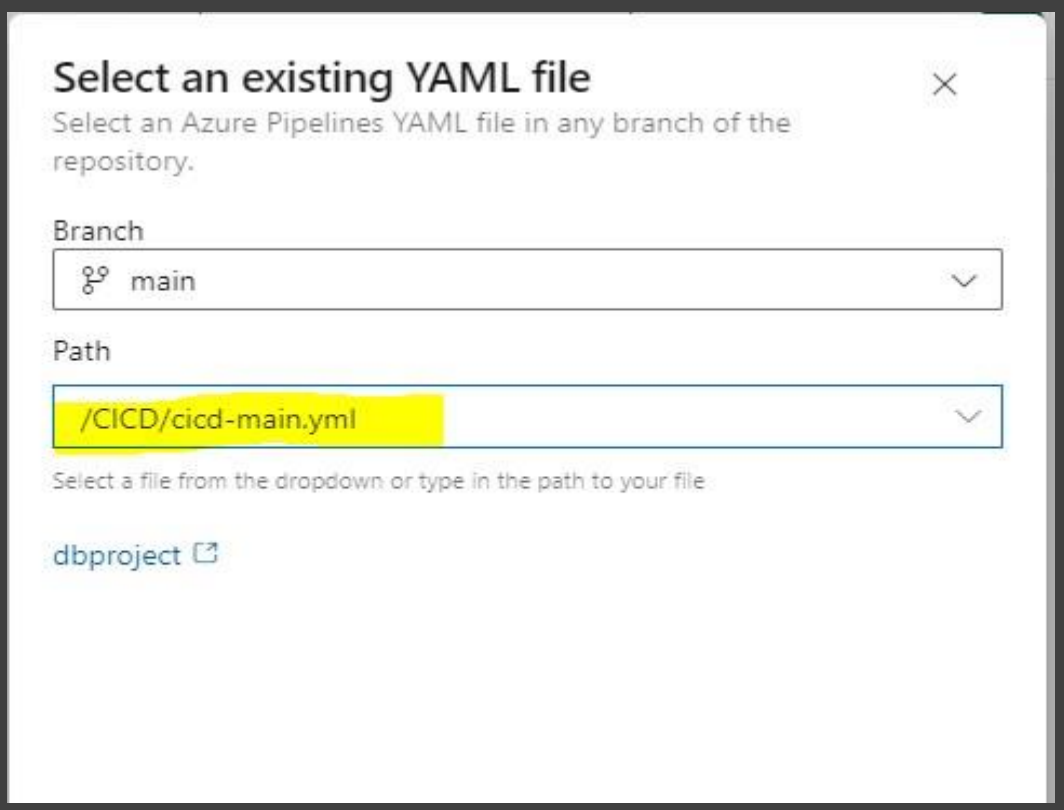
Q. How to create CI pipeline to have live folder?

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.

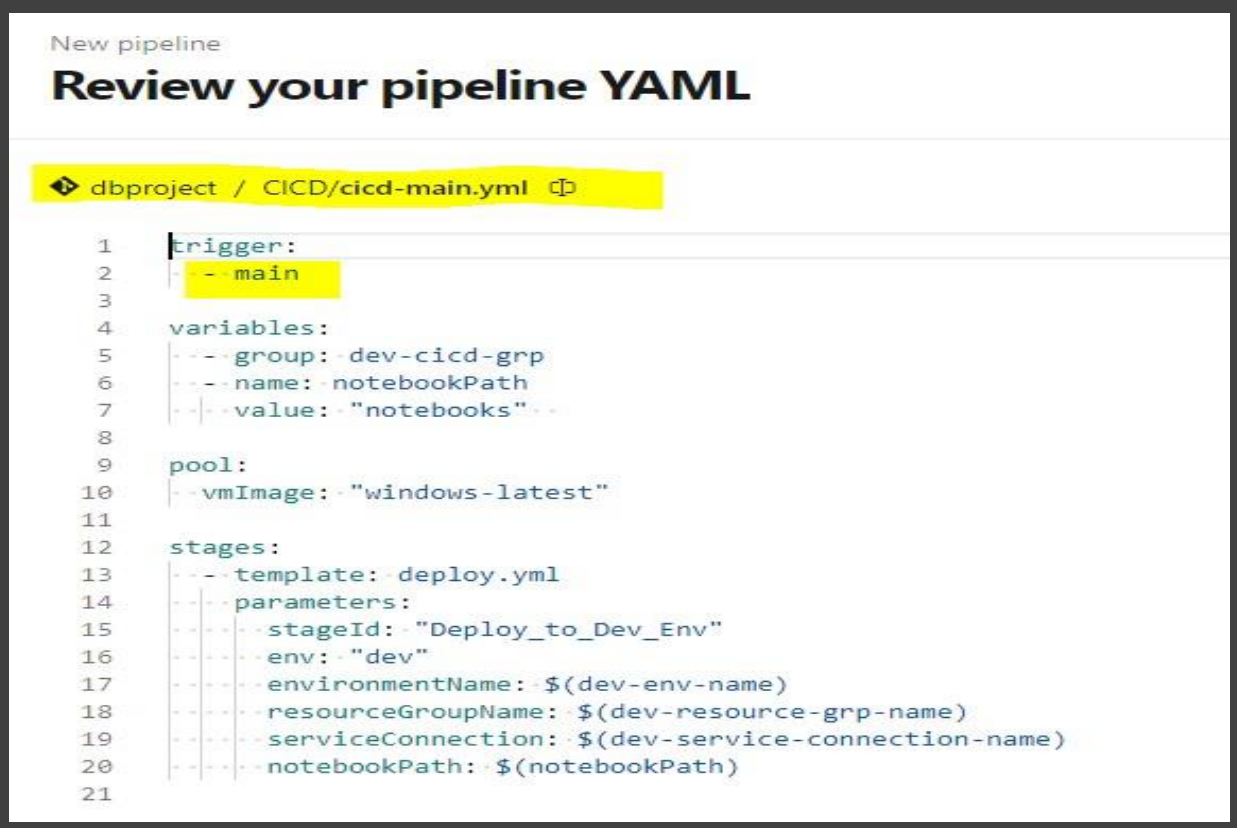




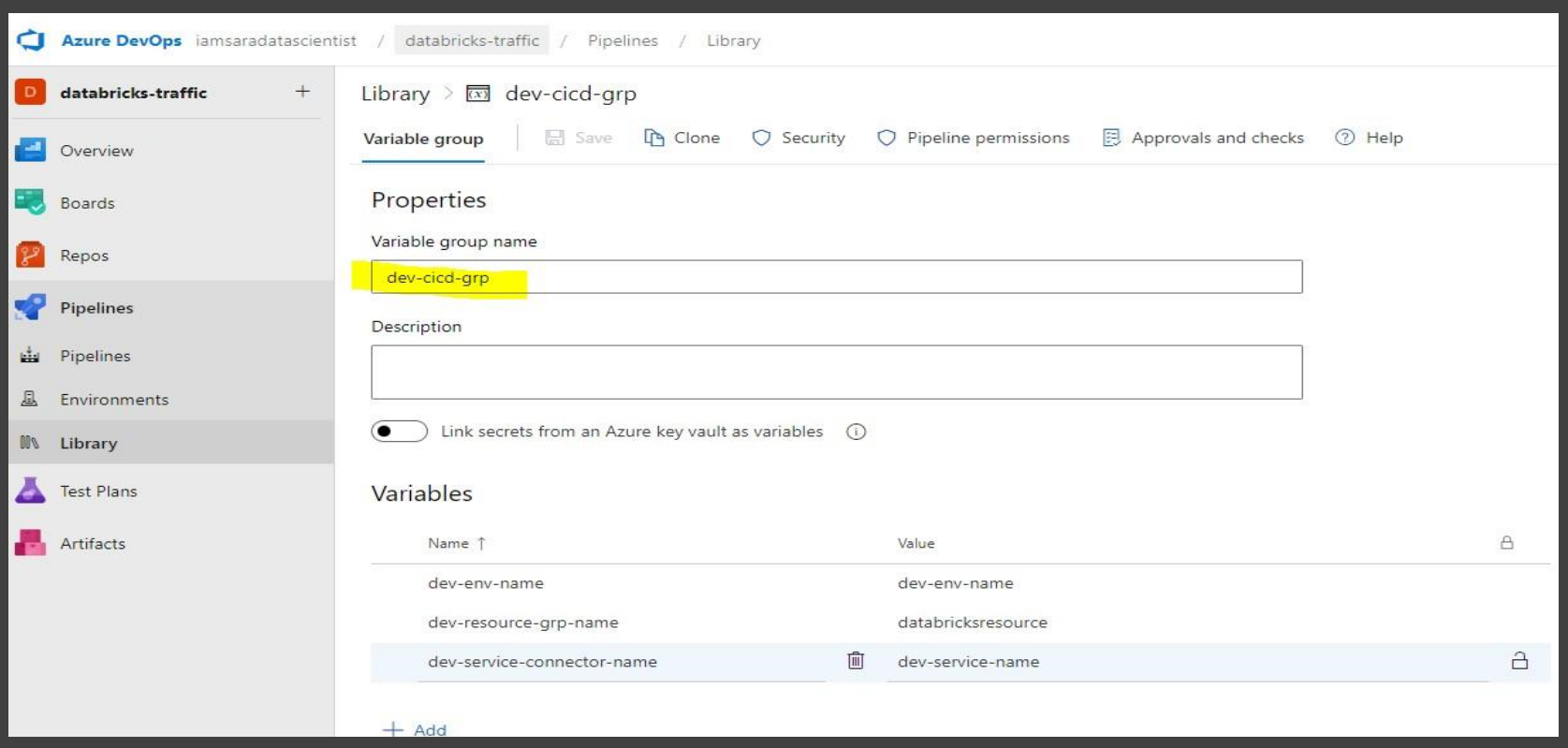
Select an existing cicd-main.yaml file



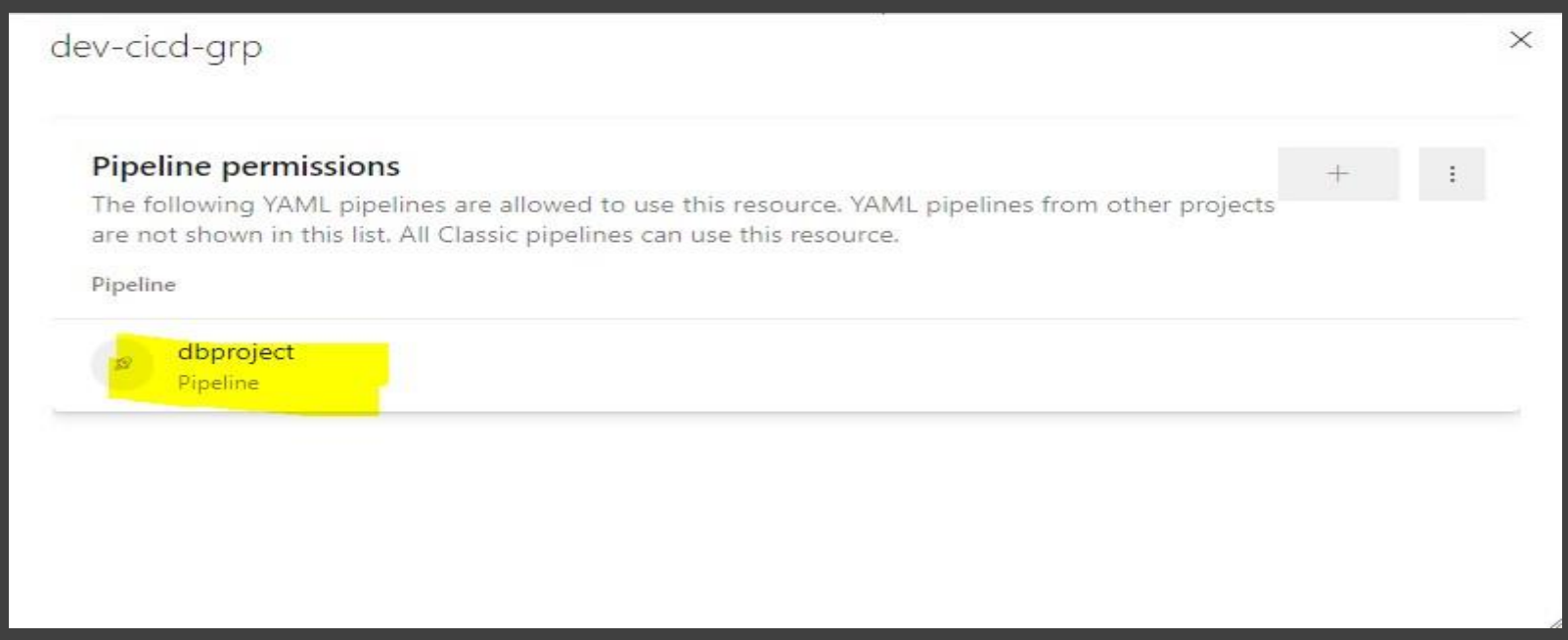
Review your pipeline code



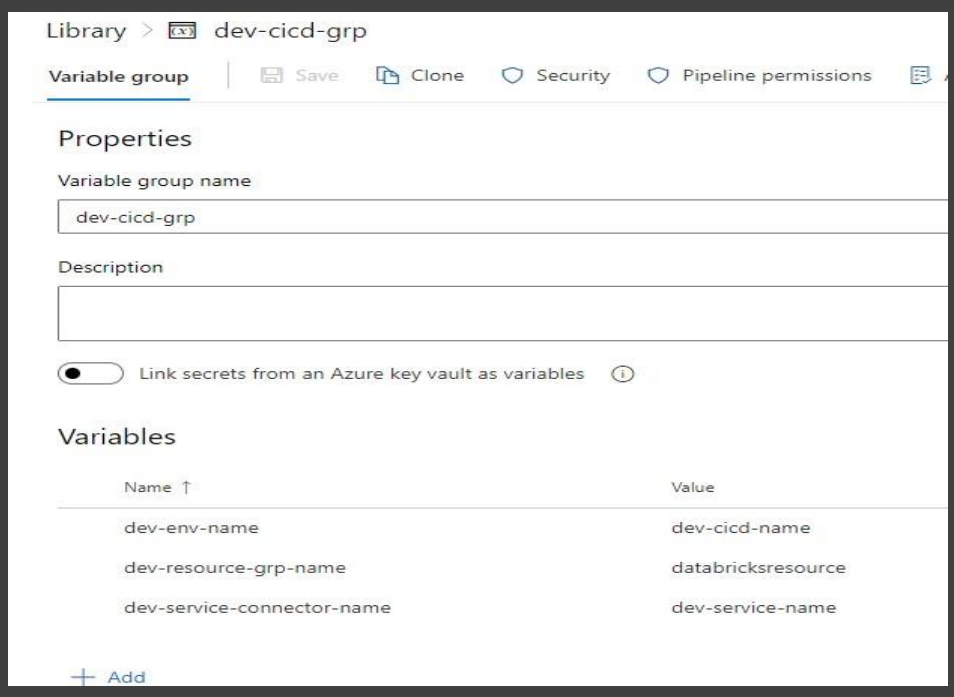
Three important steps:
Enviornment name through library



Pipeline permissions are required

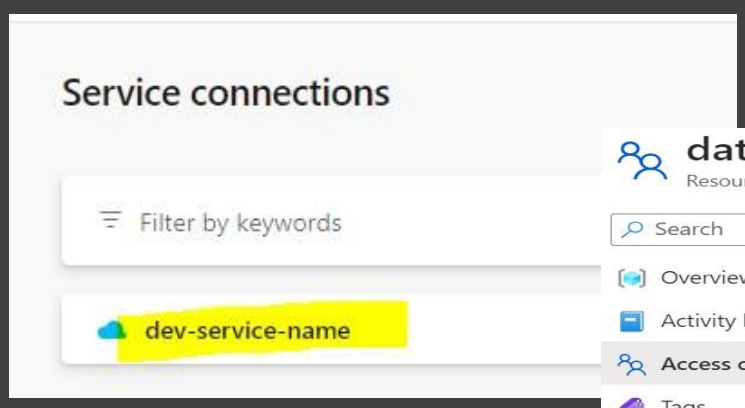


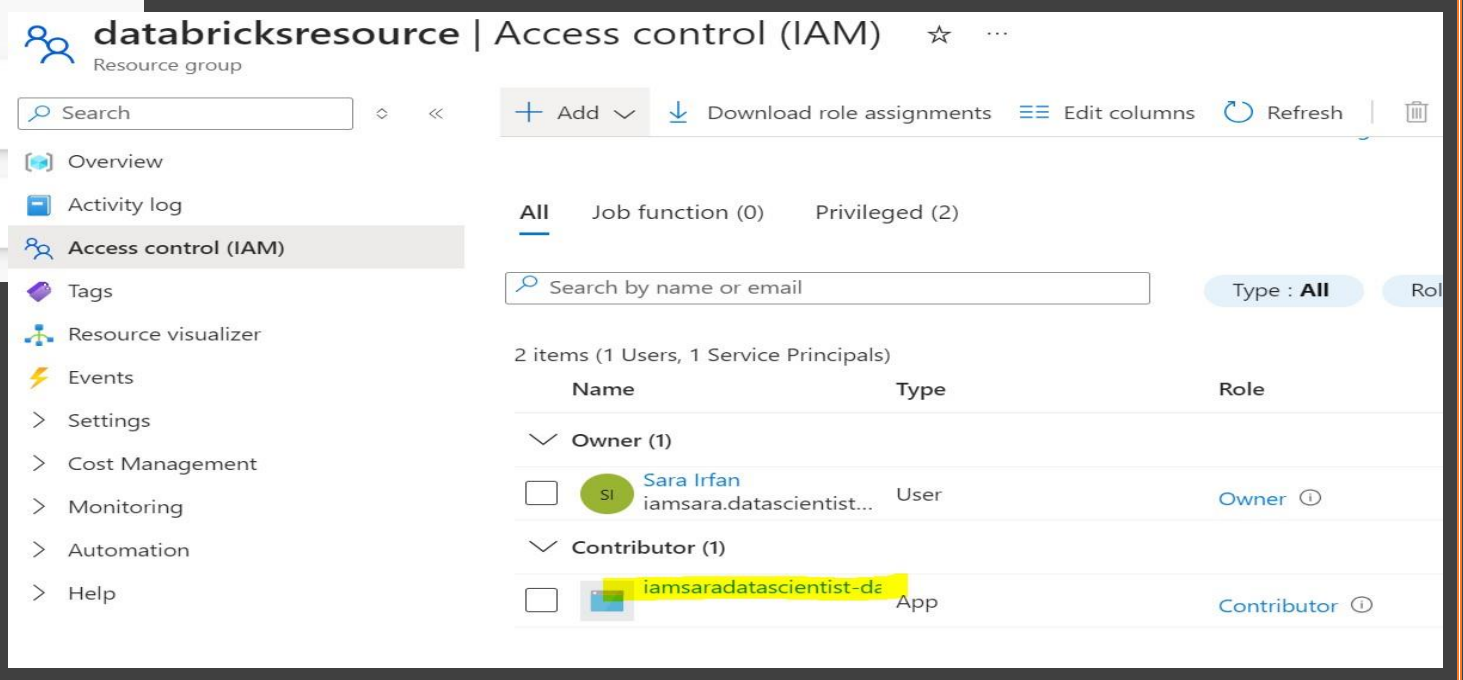
Resource group is same as used in azure for azure devops

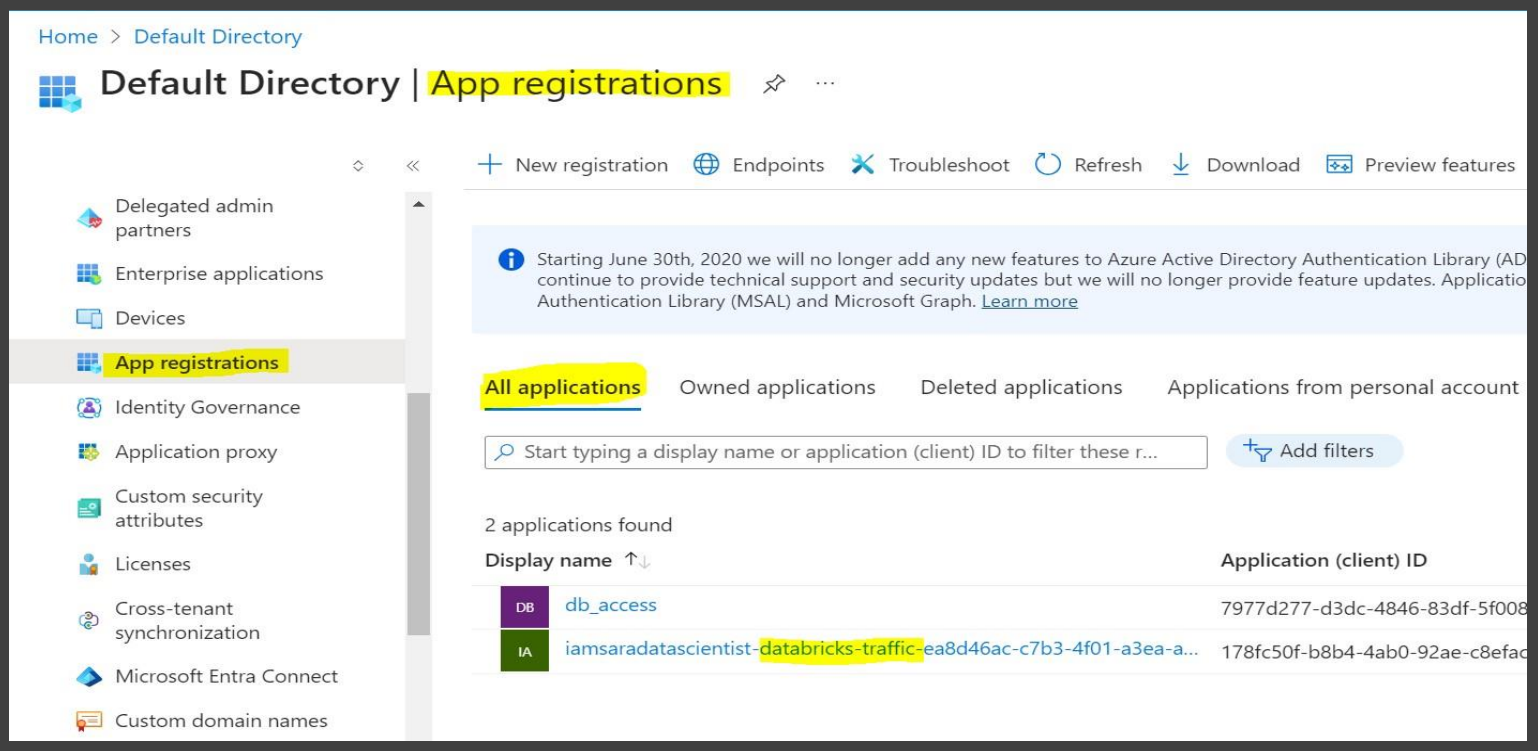


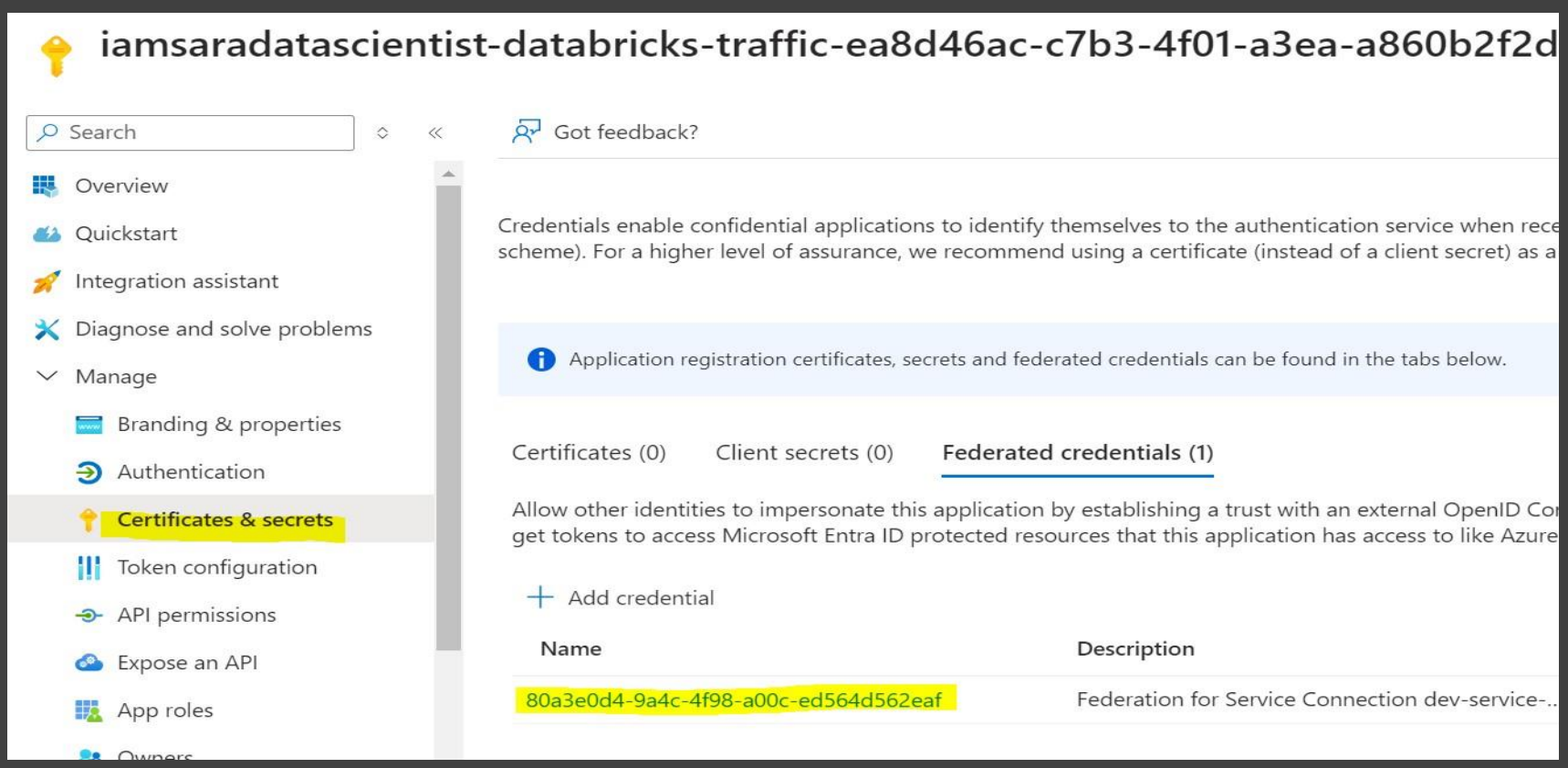
Q. How to give service connection?

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.

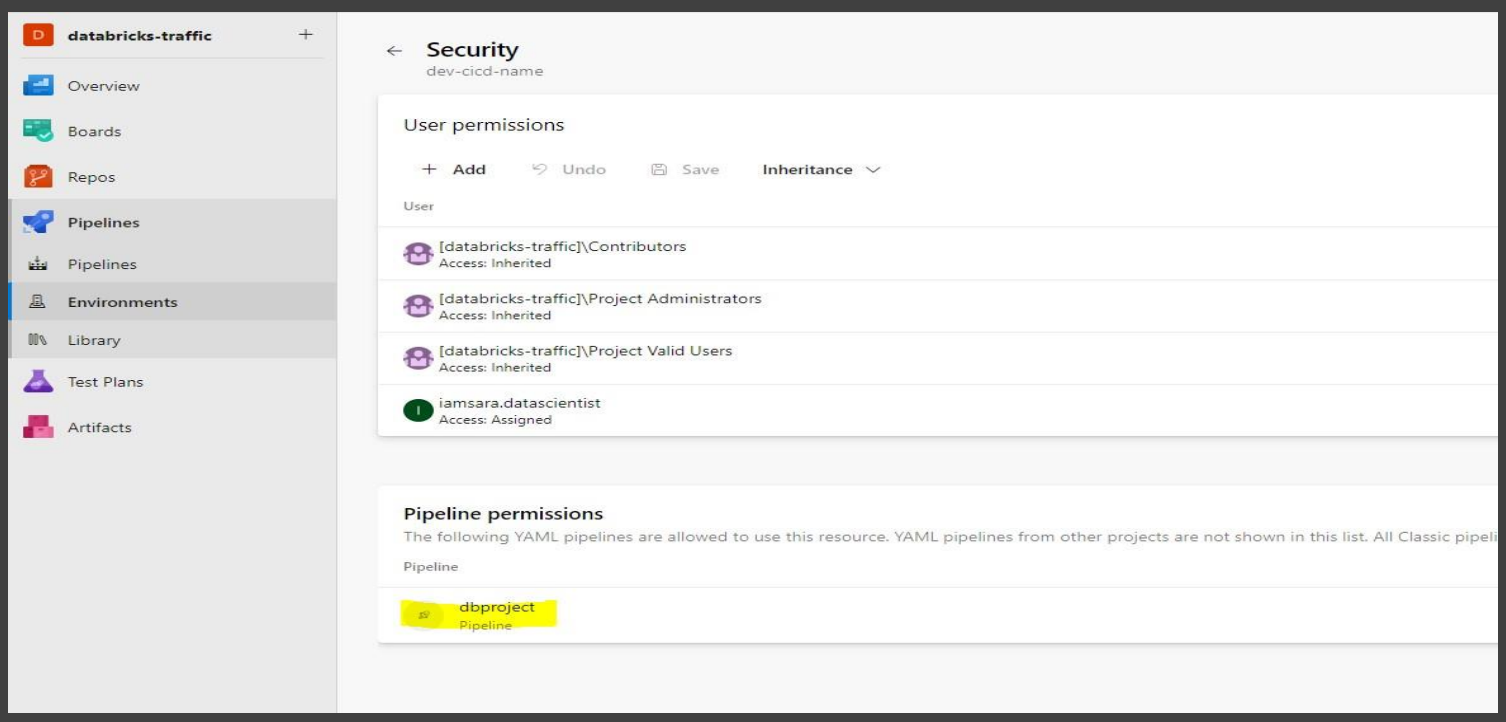


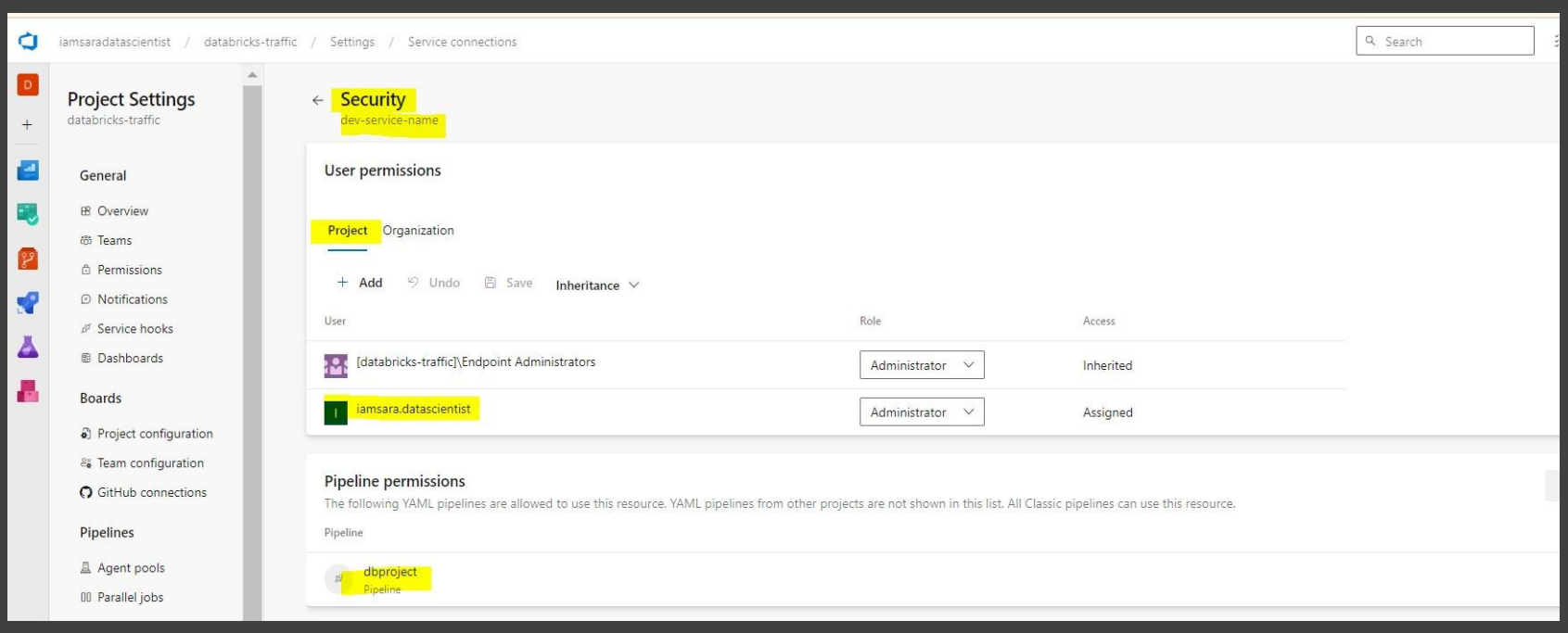


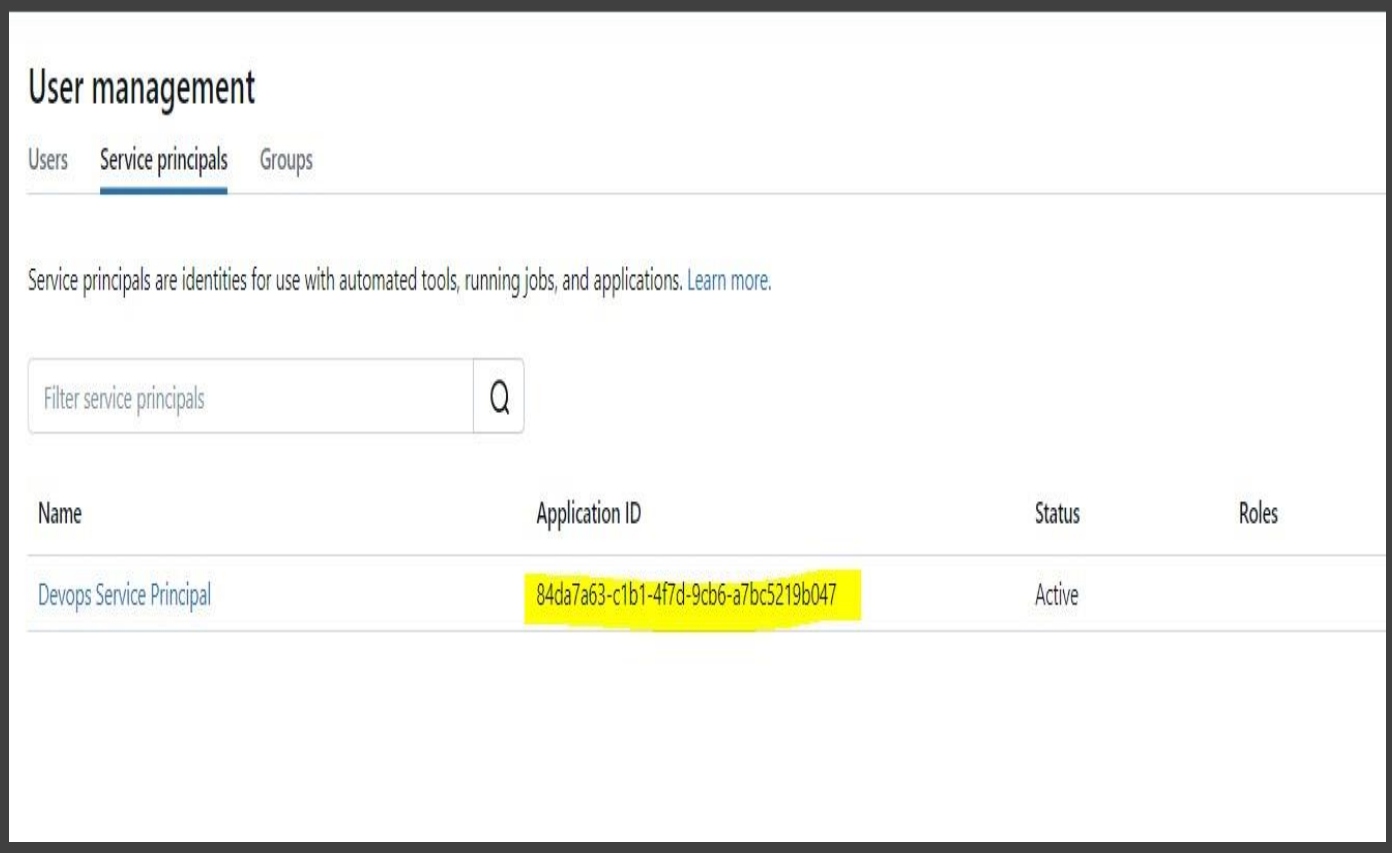

Home > Default Directory
Default Directory | App registrations
New registration
Endpoints
Troubleshoot
Refresh
Download
Preview features
Delegated admin partners
Enterprise applications
Devices
App registrations
Identity Governance
Application proxy
Custom security attributes
Licenses
Cross-tenant synchronization
Microsoft Entra Connect
Custom domain names
Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (AD
continue to provide technical support and security updates but we will no longer provide feature updates. Applicatio
Authentication Library (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 r...
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-a...
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 rece
scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a
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 Co
get tokens to access Microsoft Entra ID protected resources that this application has access to like Azure
Add credential
Name
Description
80a3e0d4-9a4c-4f98-a00c-ed564d562eaf
Federation for Service Connection dev-service-..

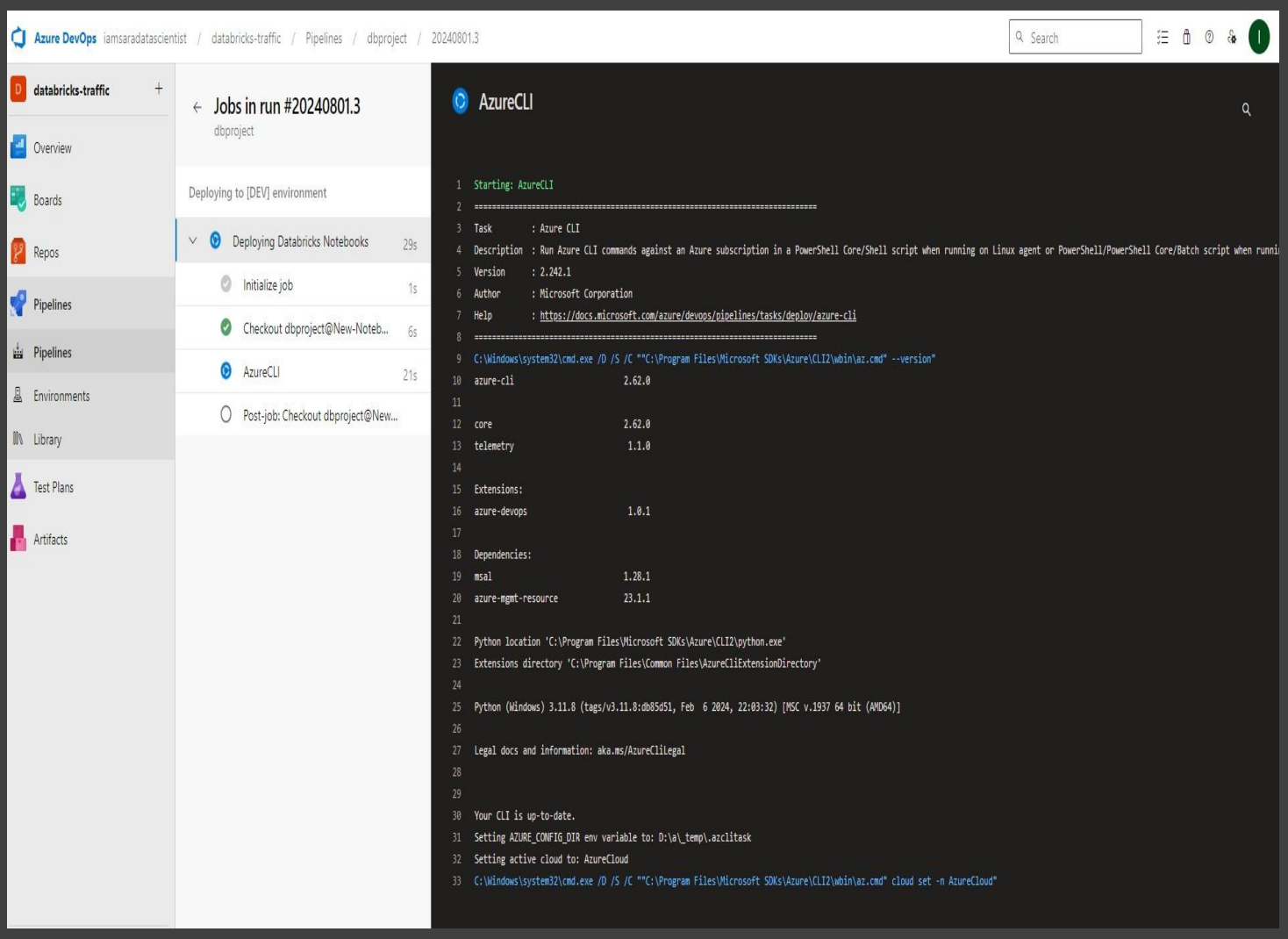
Permissions for user is also required



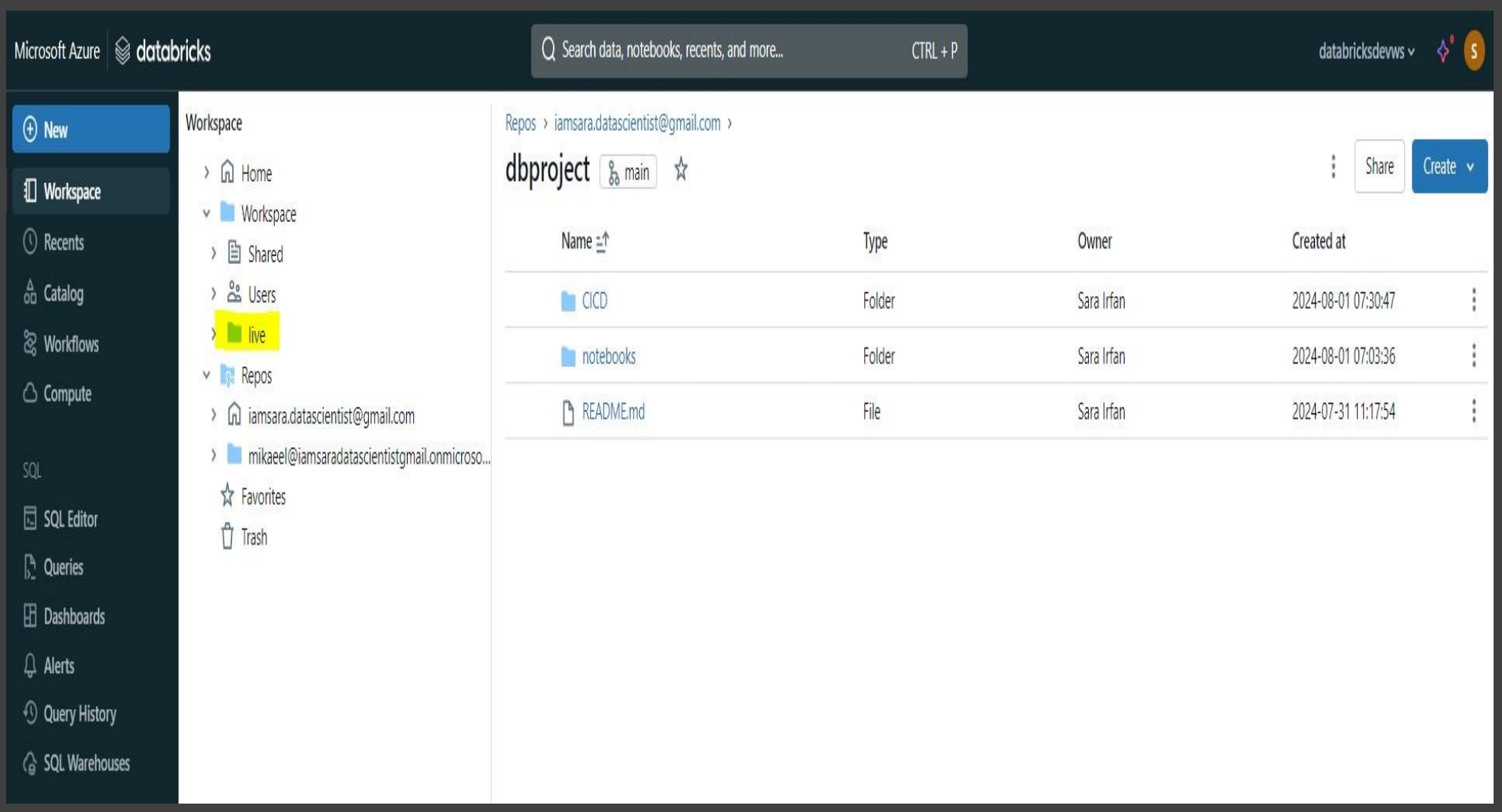




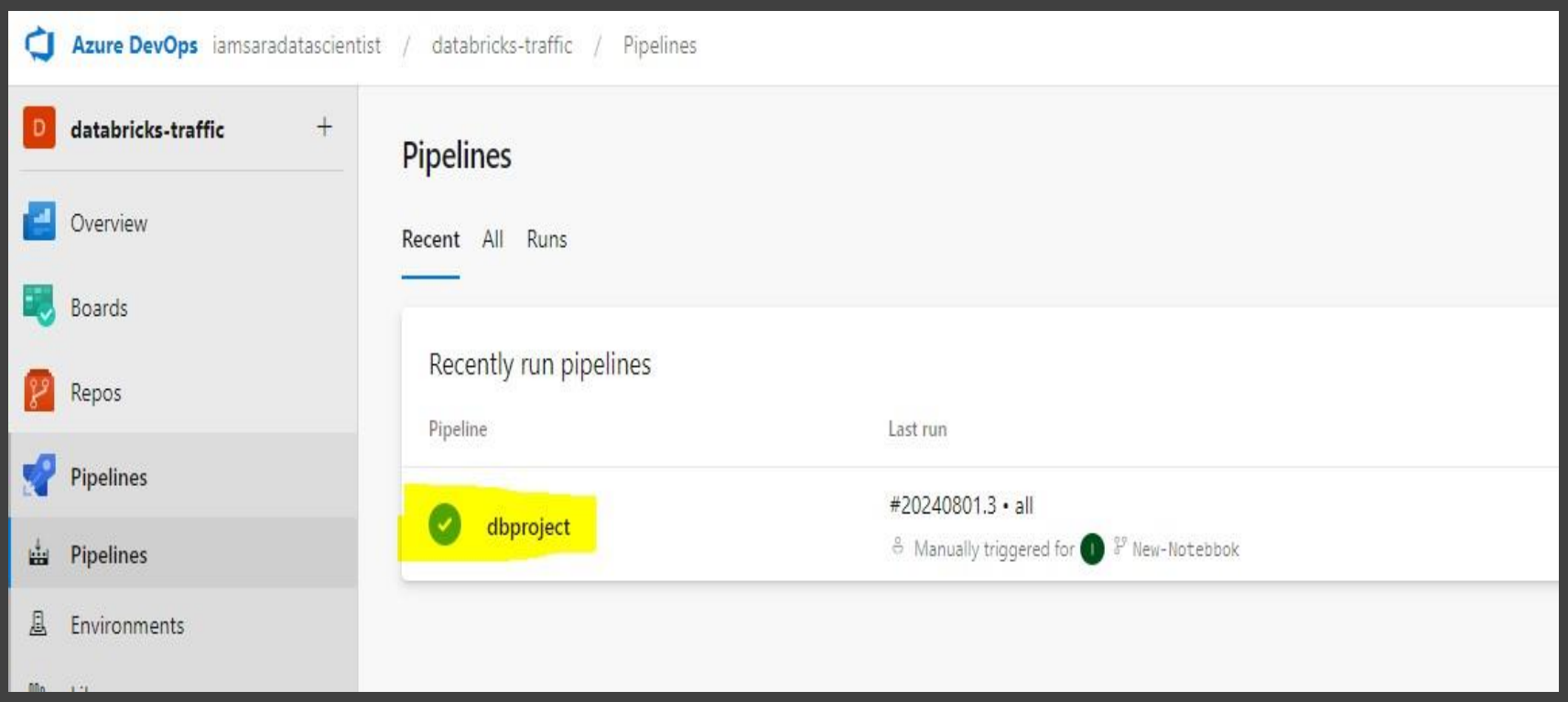
Deploying databricks notebooks

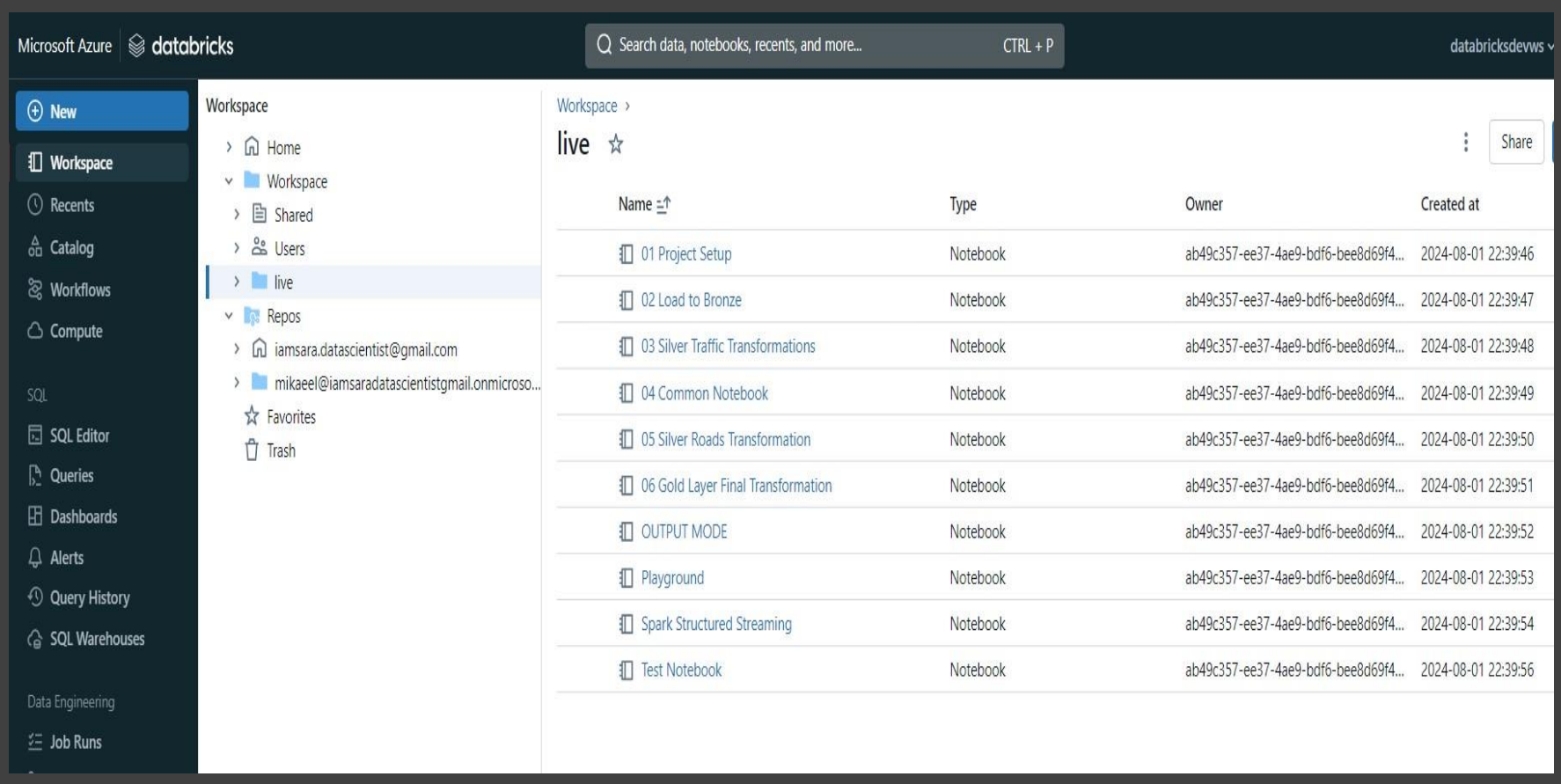


Live folder has been created:



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

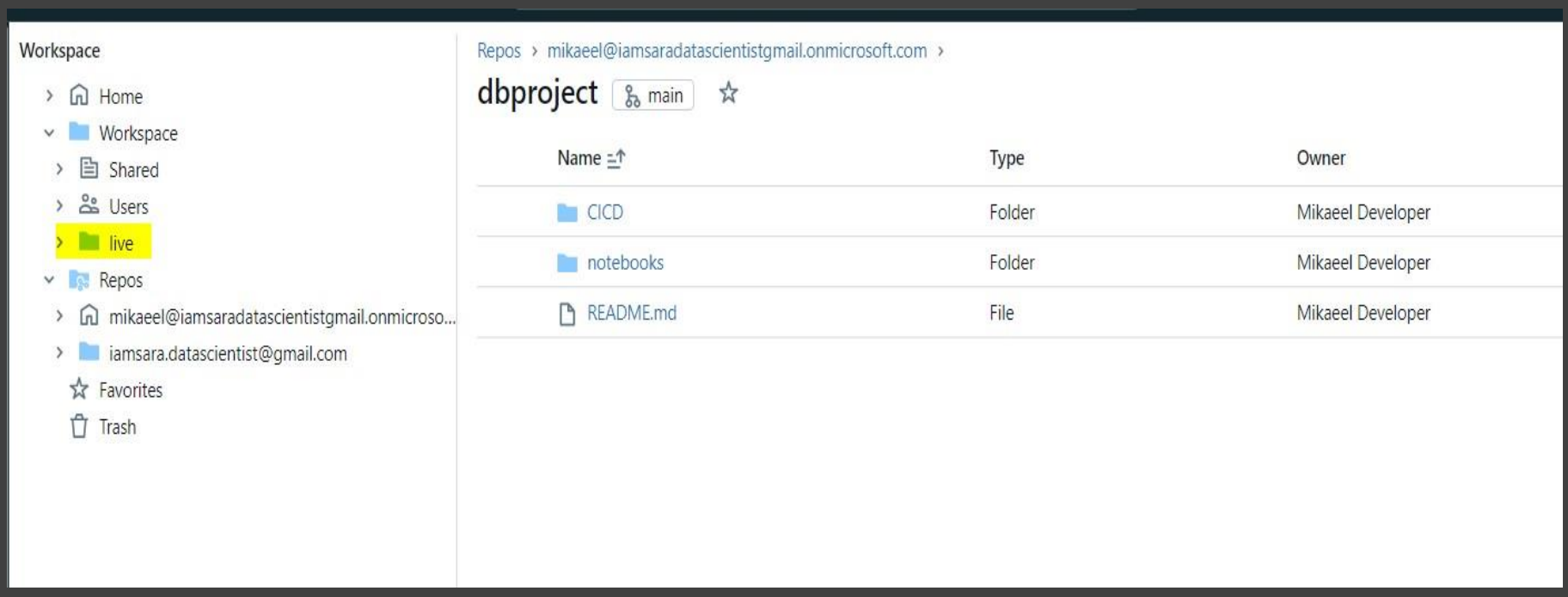




Give admin permissions to Mikaeel:

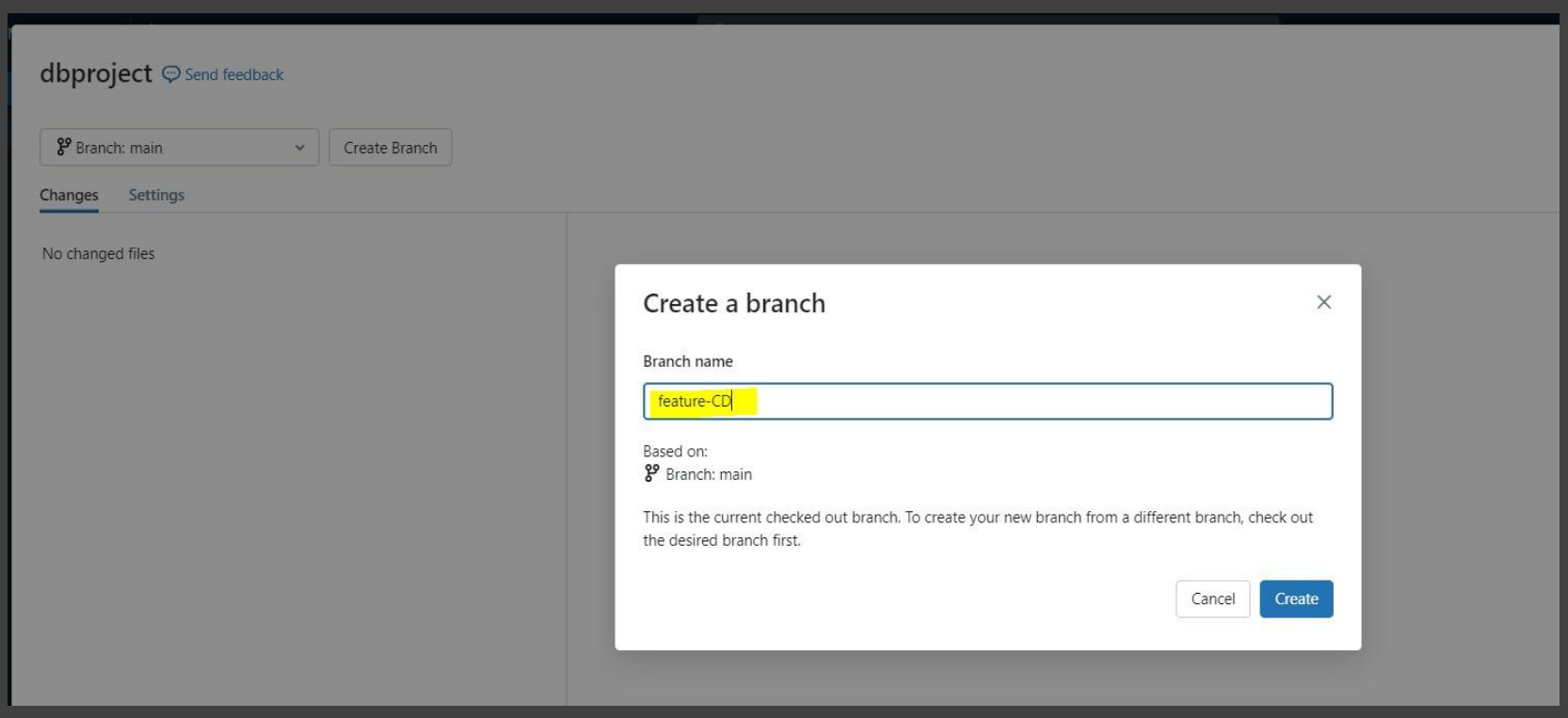
Mikaeel Developer (mikaeel@iamsaradatascientistgmail.onmicrosoft.com)	User	Admin
Sara Irfan (iamsara.datascientist@gmail.com)	User	Admin

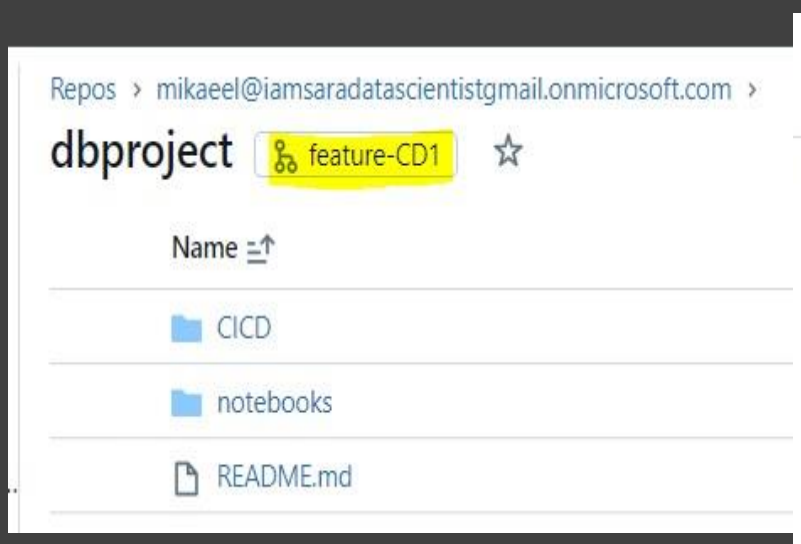
Live folder is also in Mikael's folder:

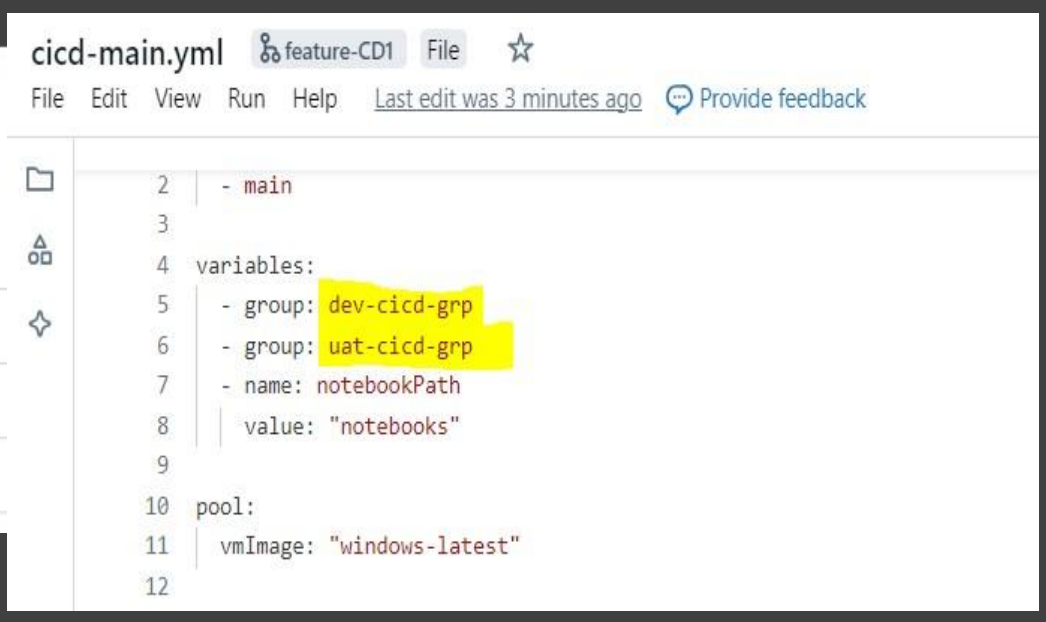


Create live folder for uat resource group:

Create a branch:



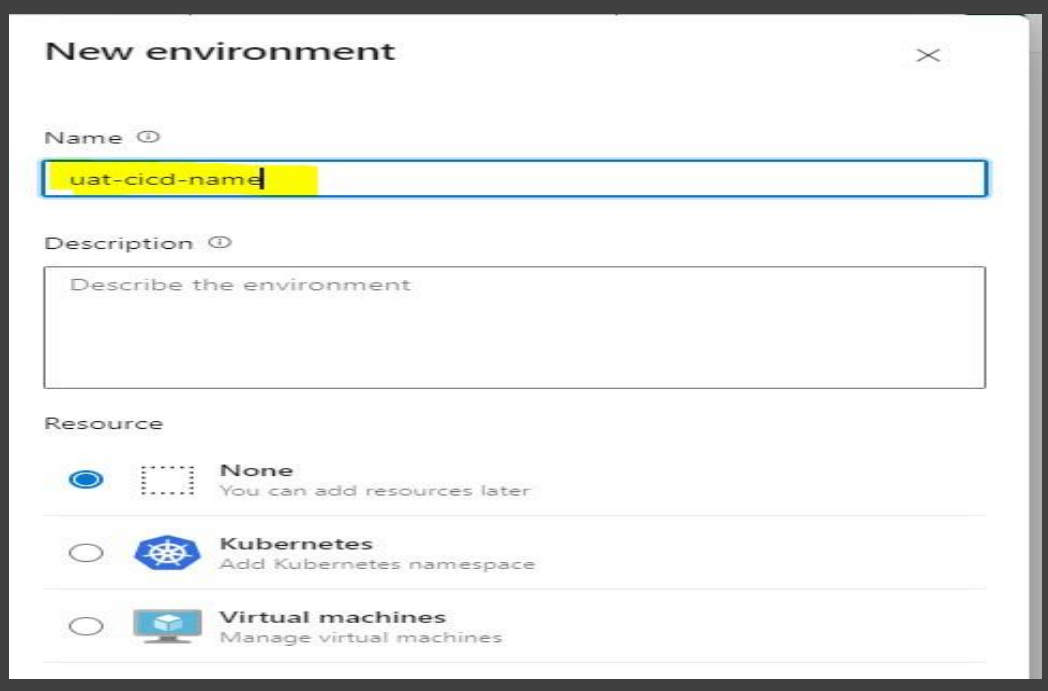




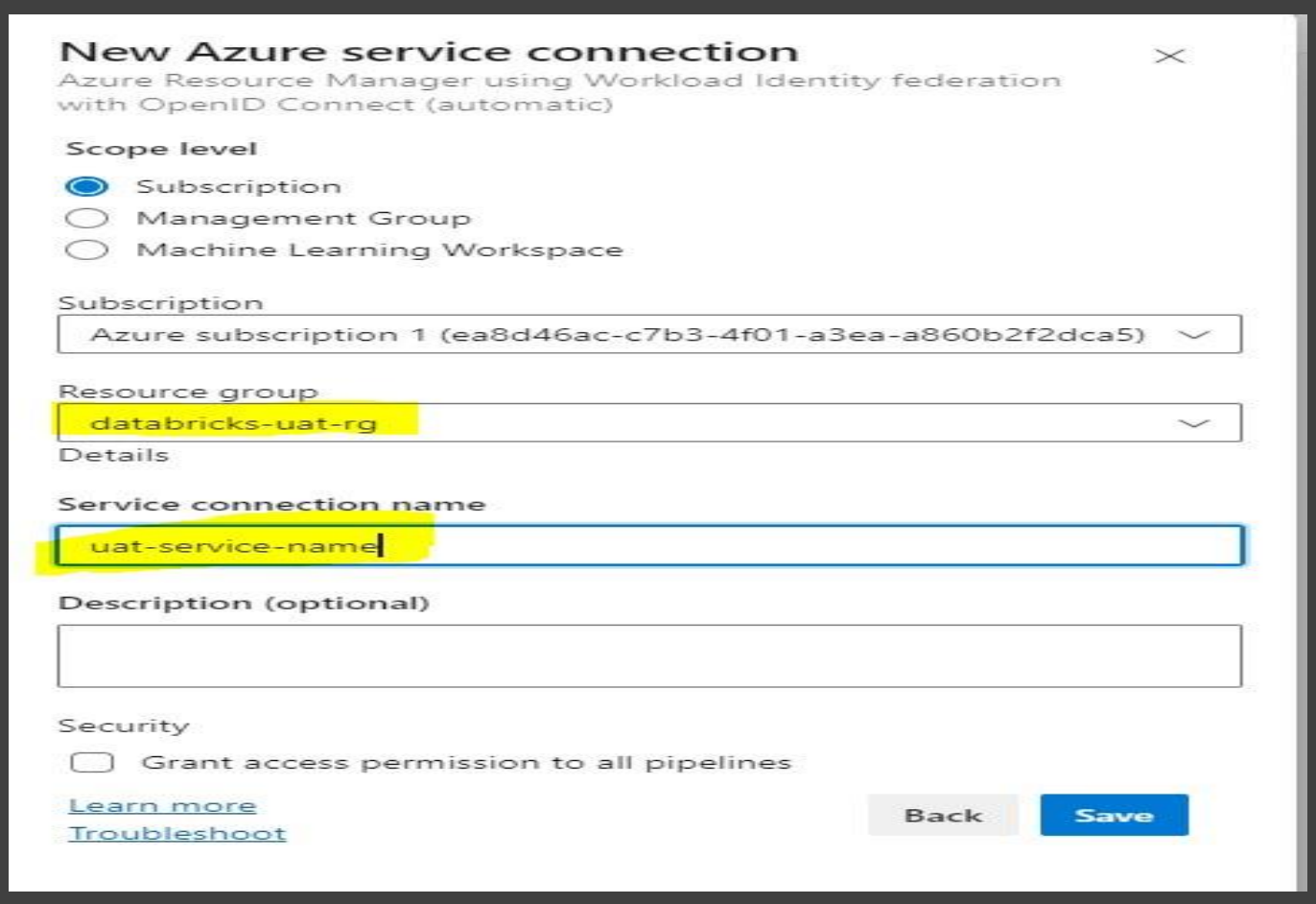
Do these modifications in cicd-yaml file:

```yaml
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)
```

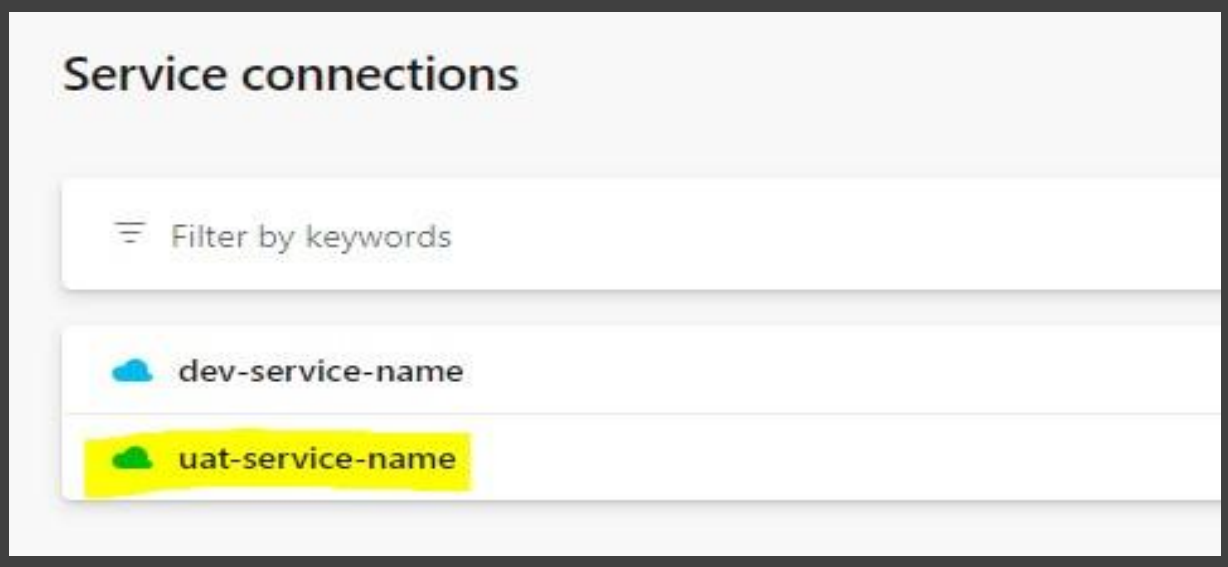
Create an enviornment uat-cicd-name:



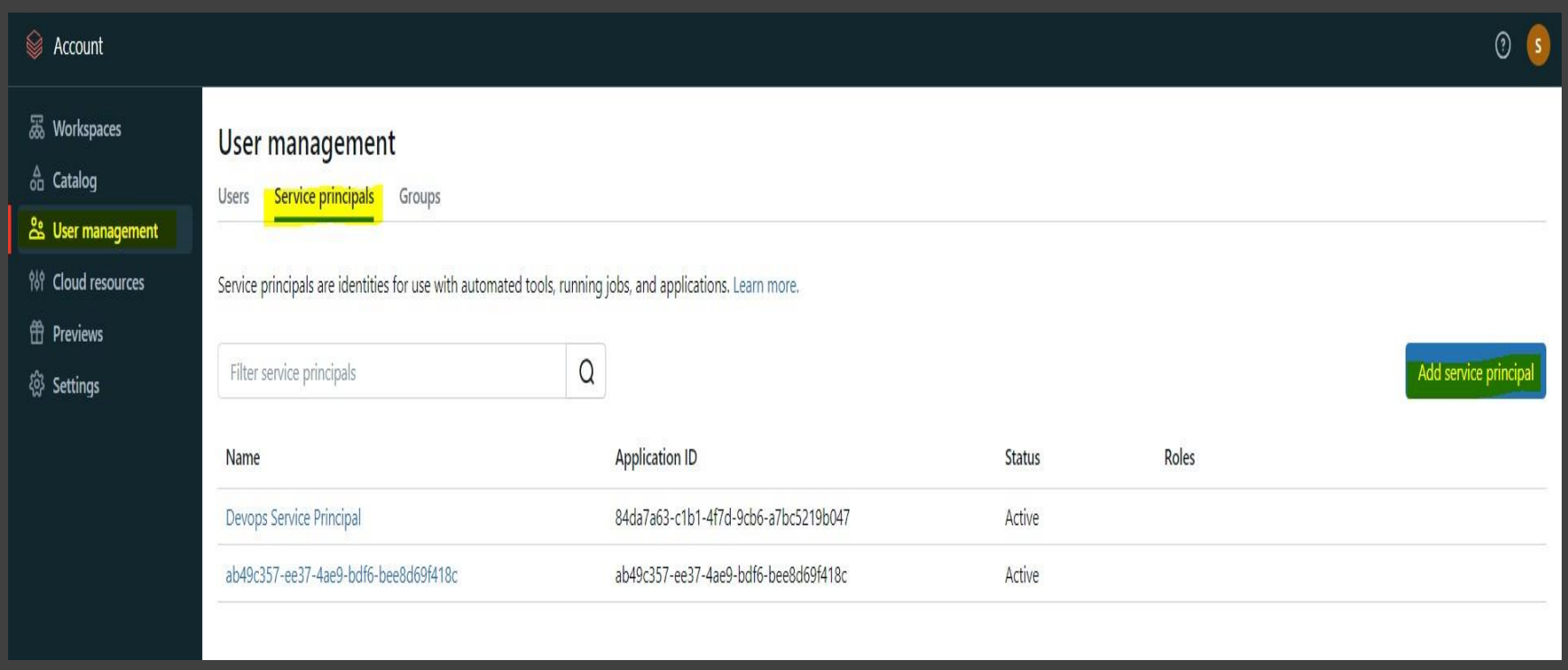
Create azure service connection, with service principals:



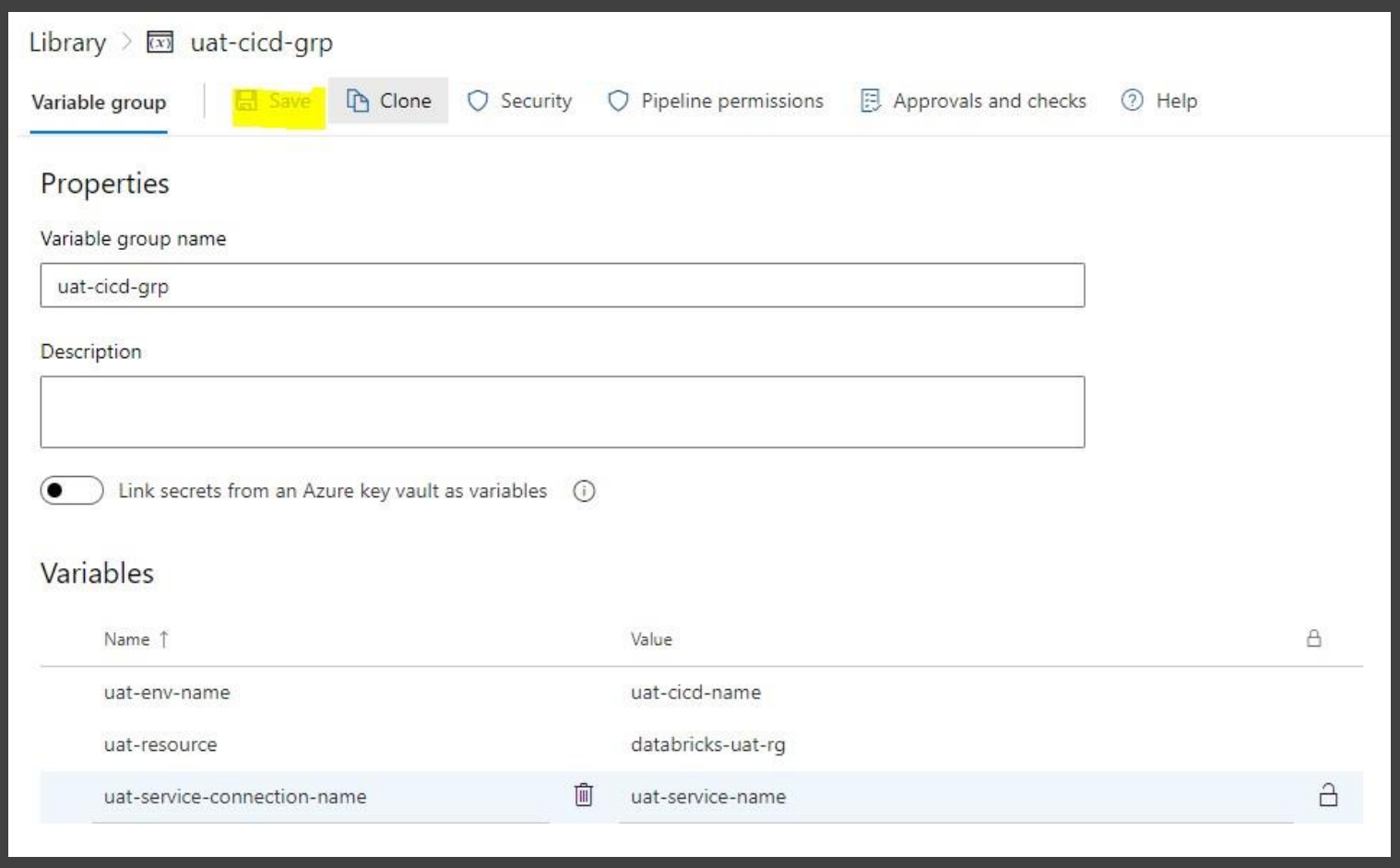
Check out the service connection:

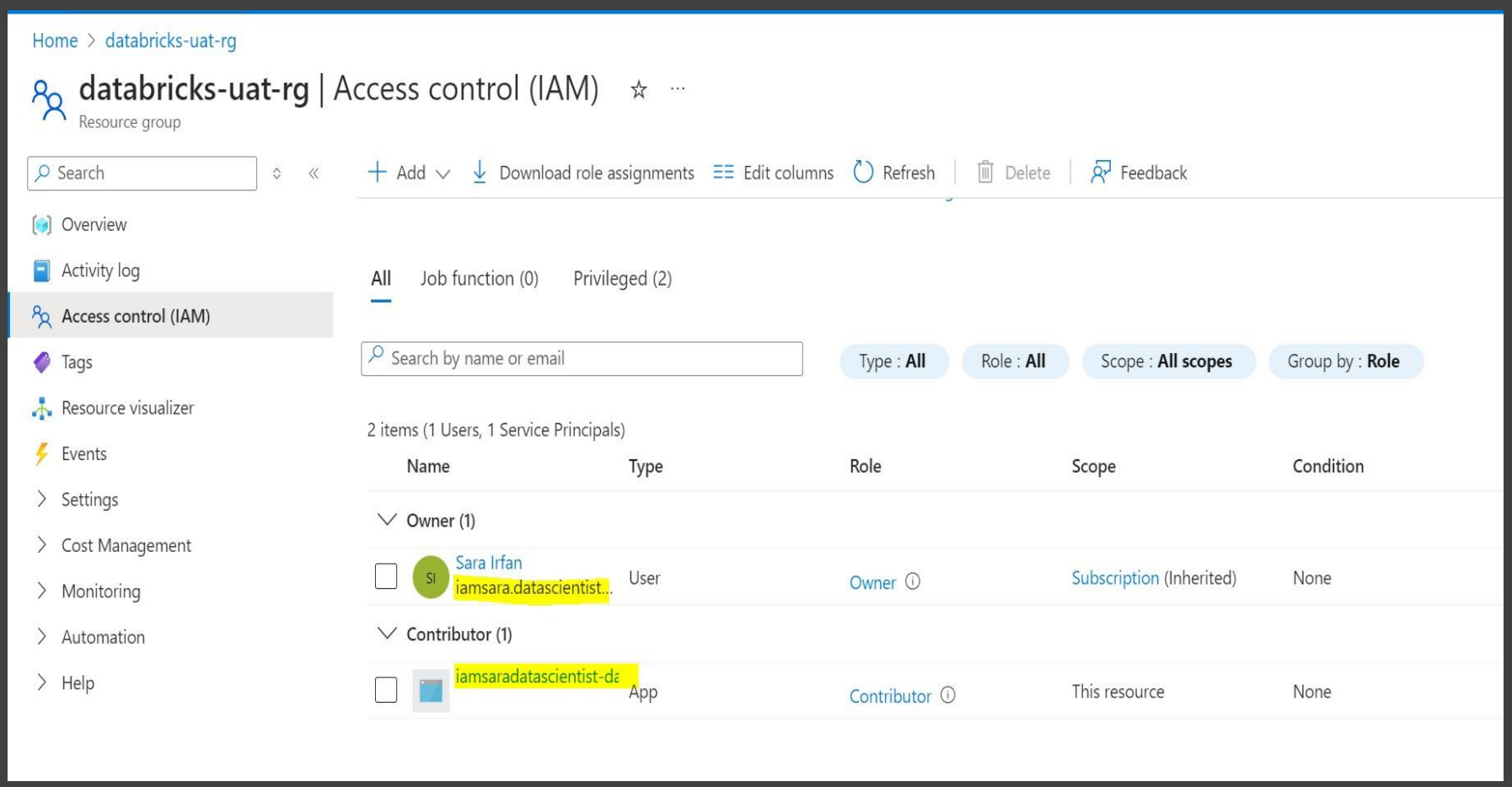


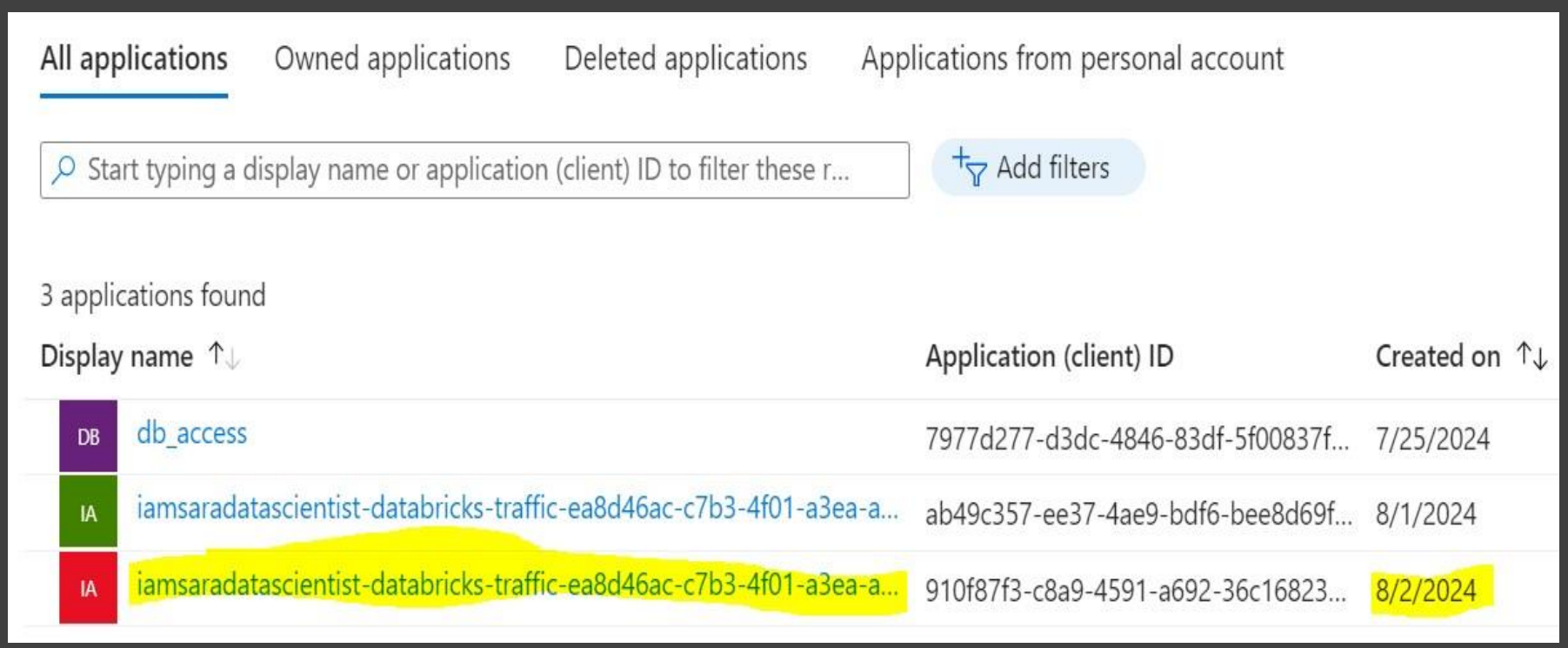
Now go to manage account of specfied workspace, open user management, service principal:

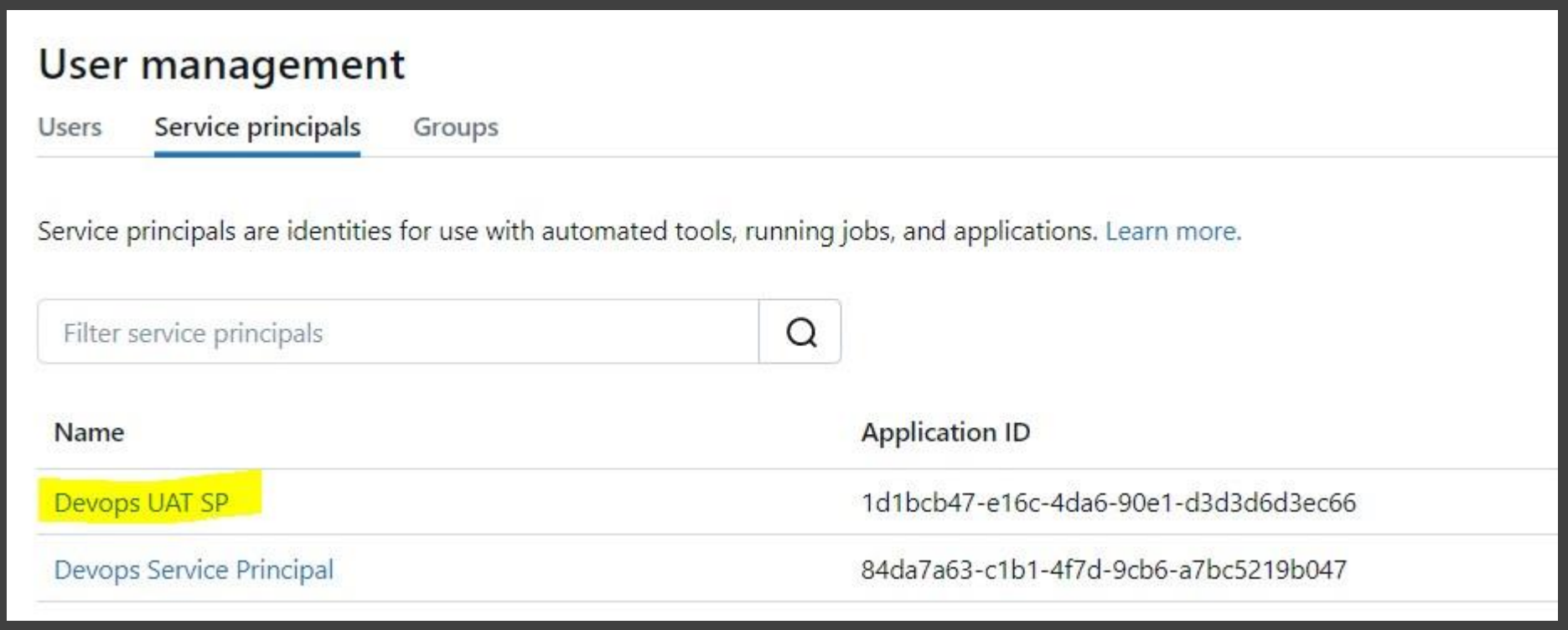


Click on SAVE:



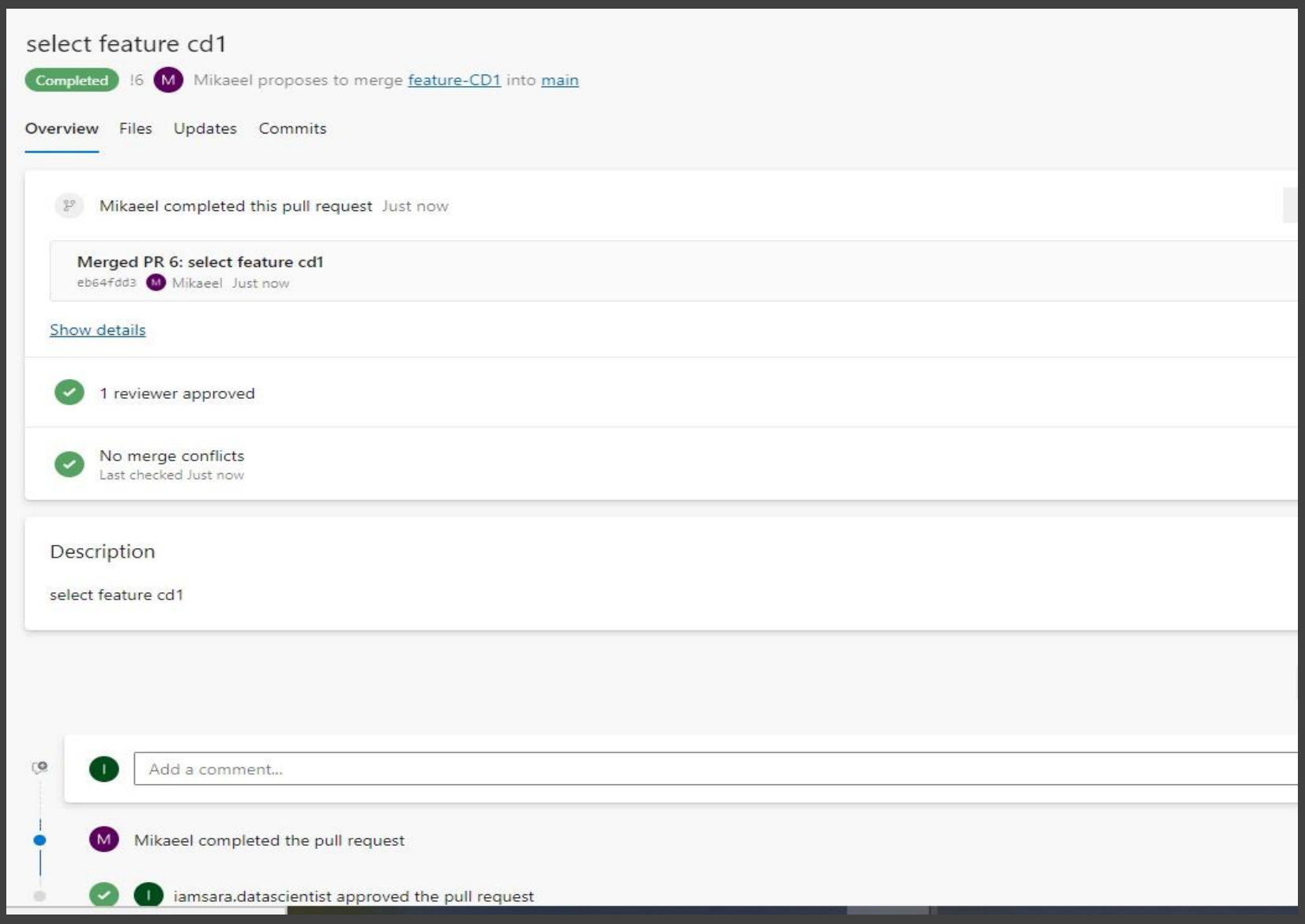


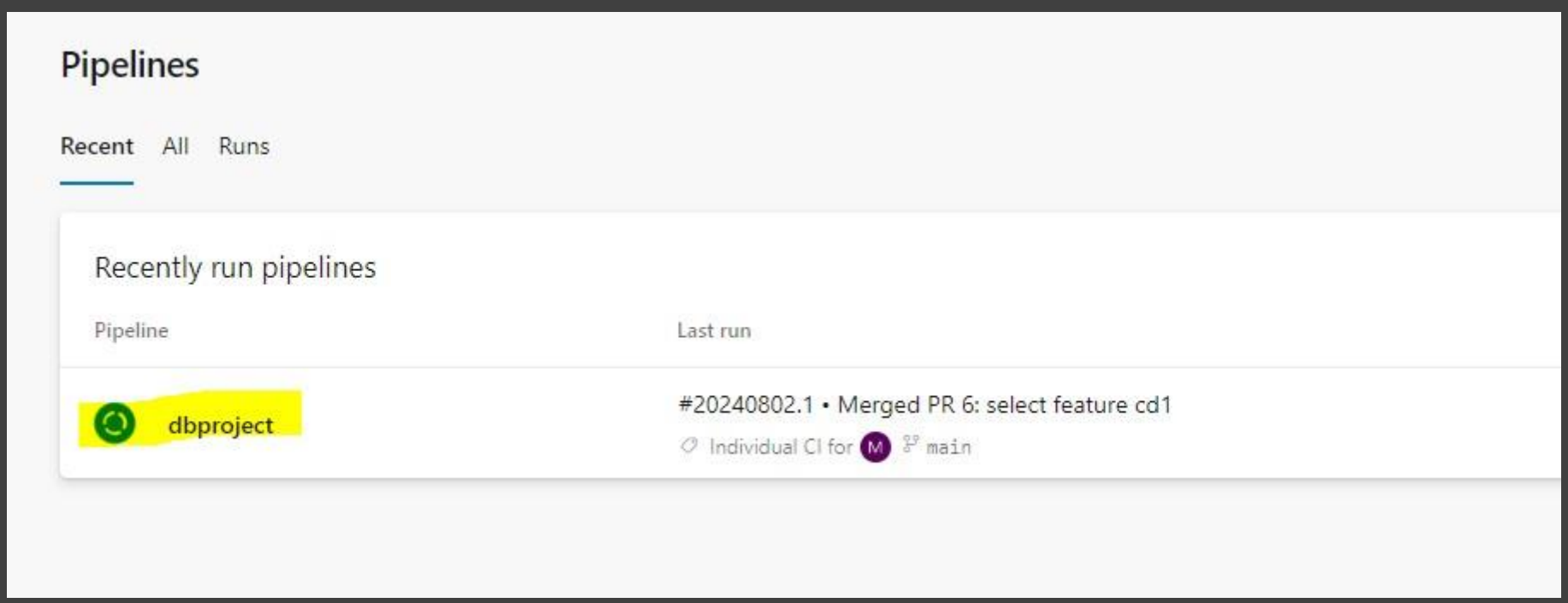




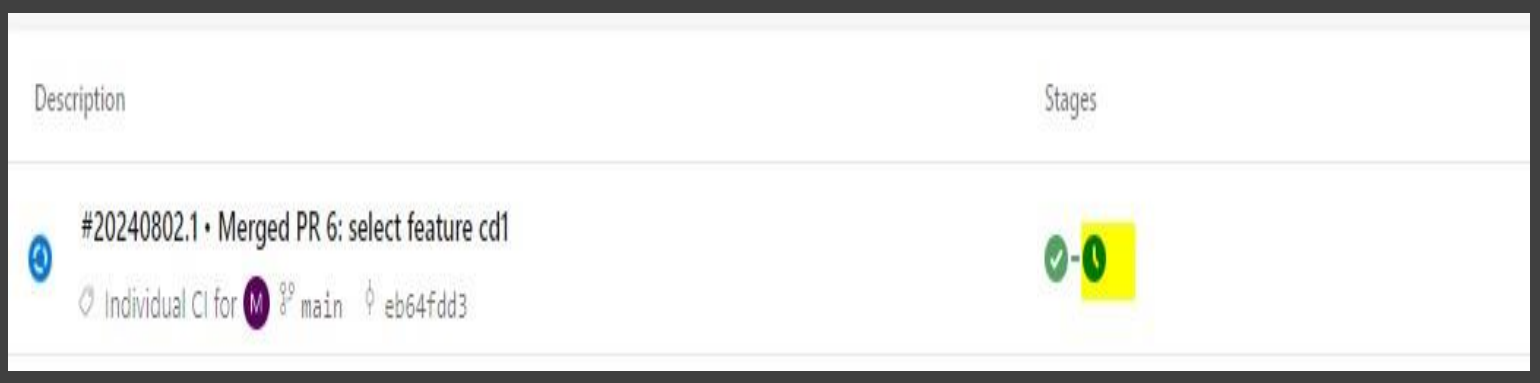
Creating feature branch, Giving approvals, commit and push, create a pull request:

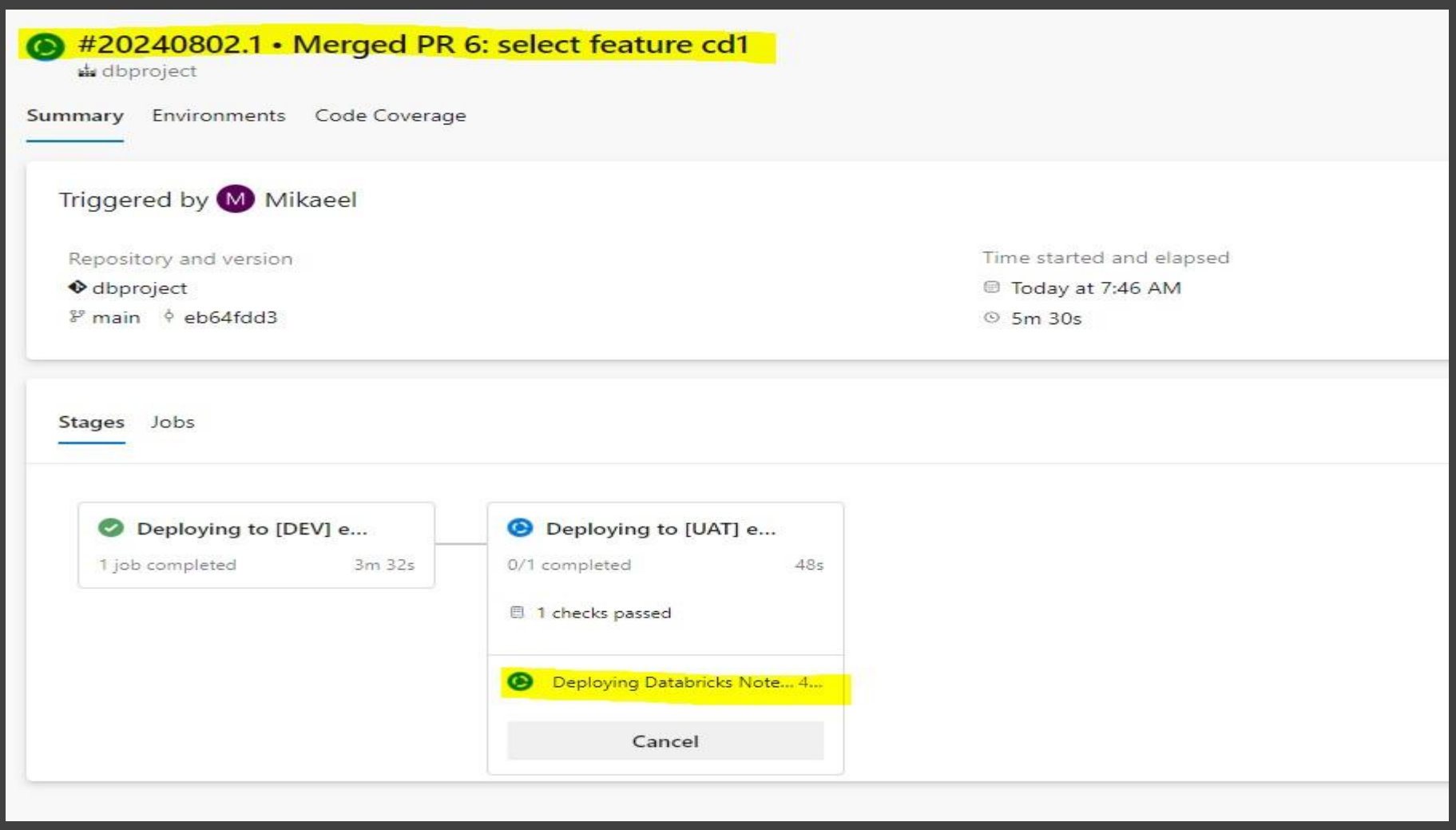
Approvals:



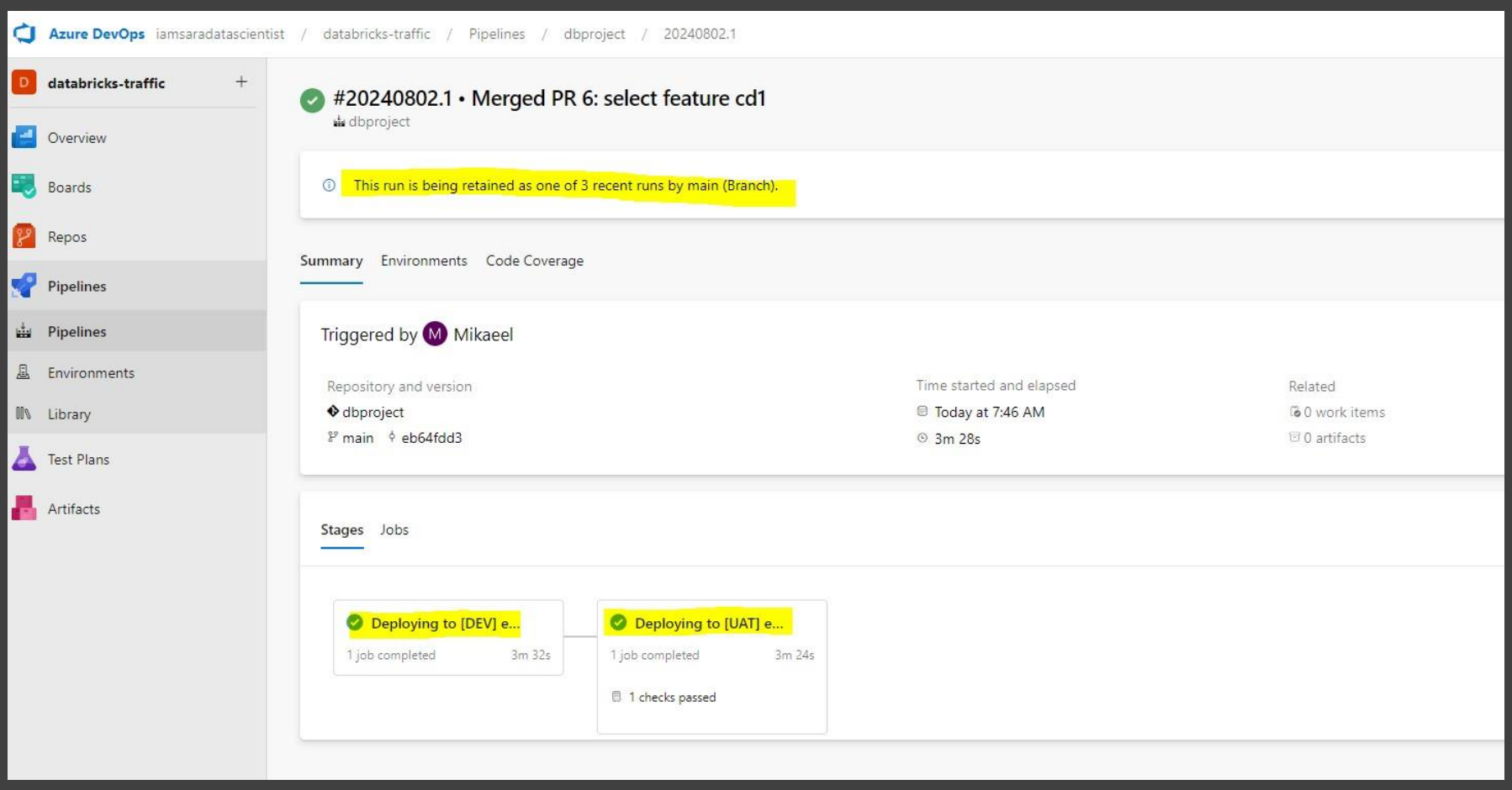


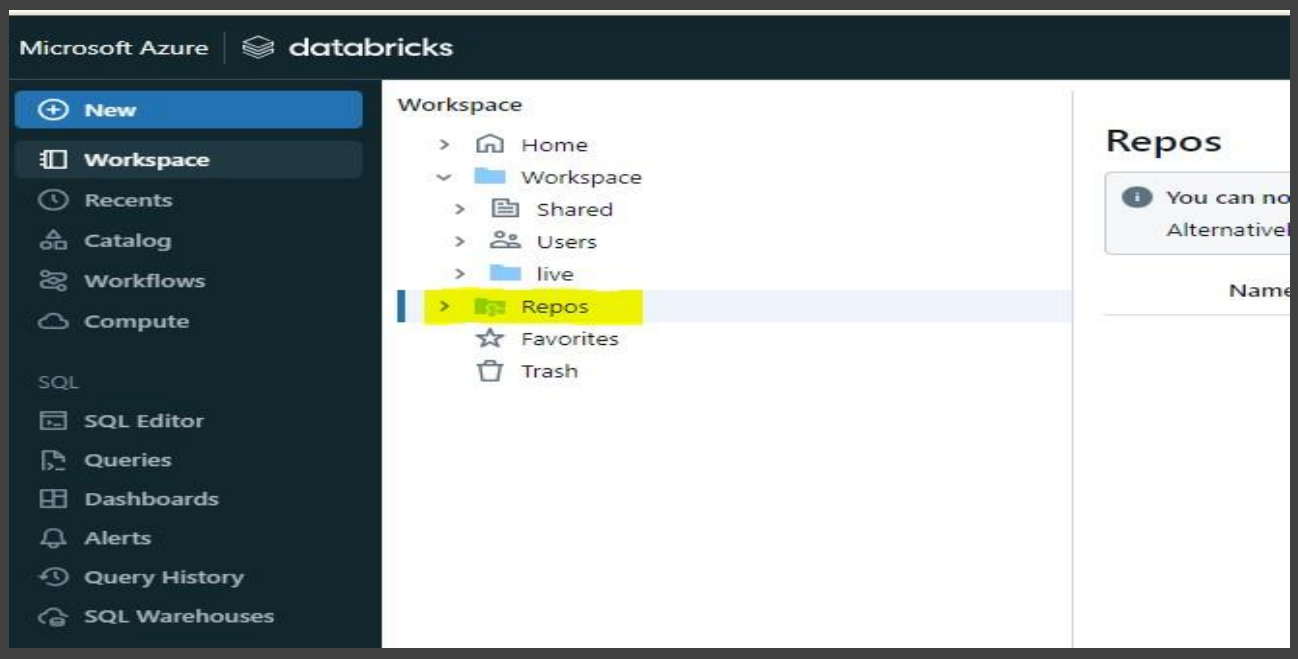
Now you can see, another pipeline is deploying:



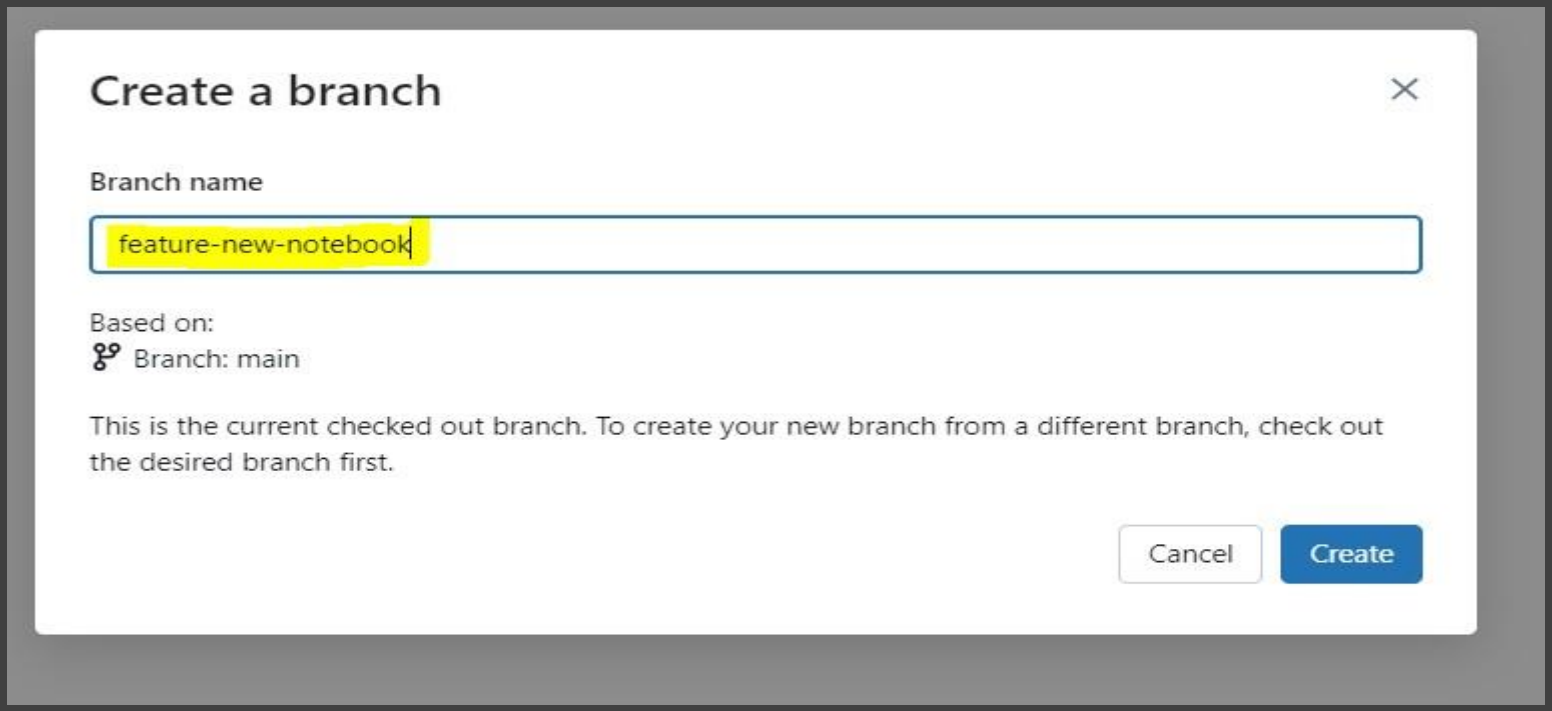


Green tick sign shows, its suceeded:

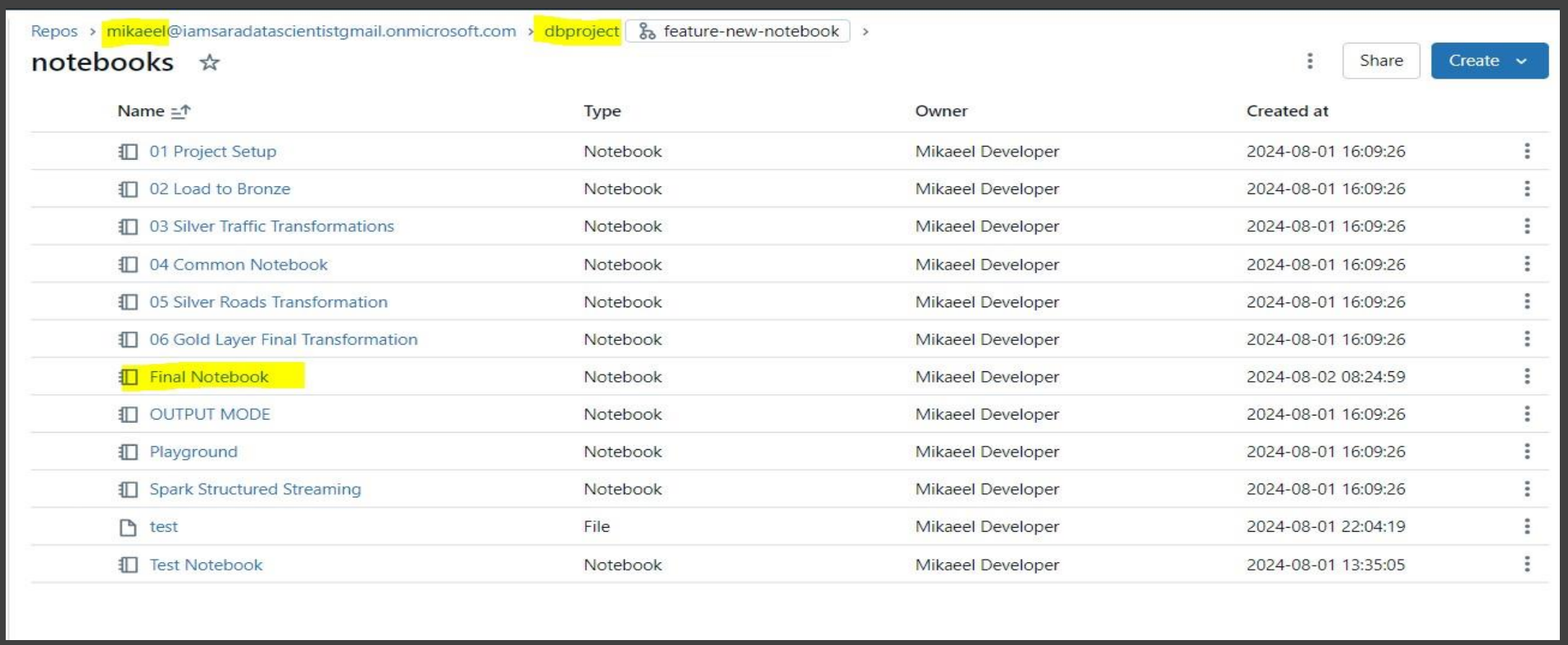


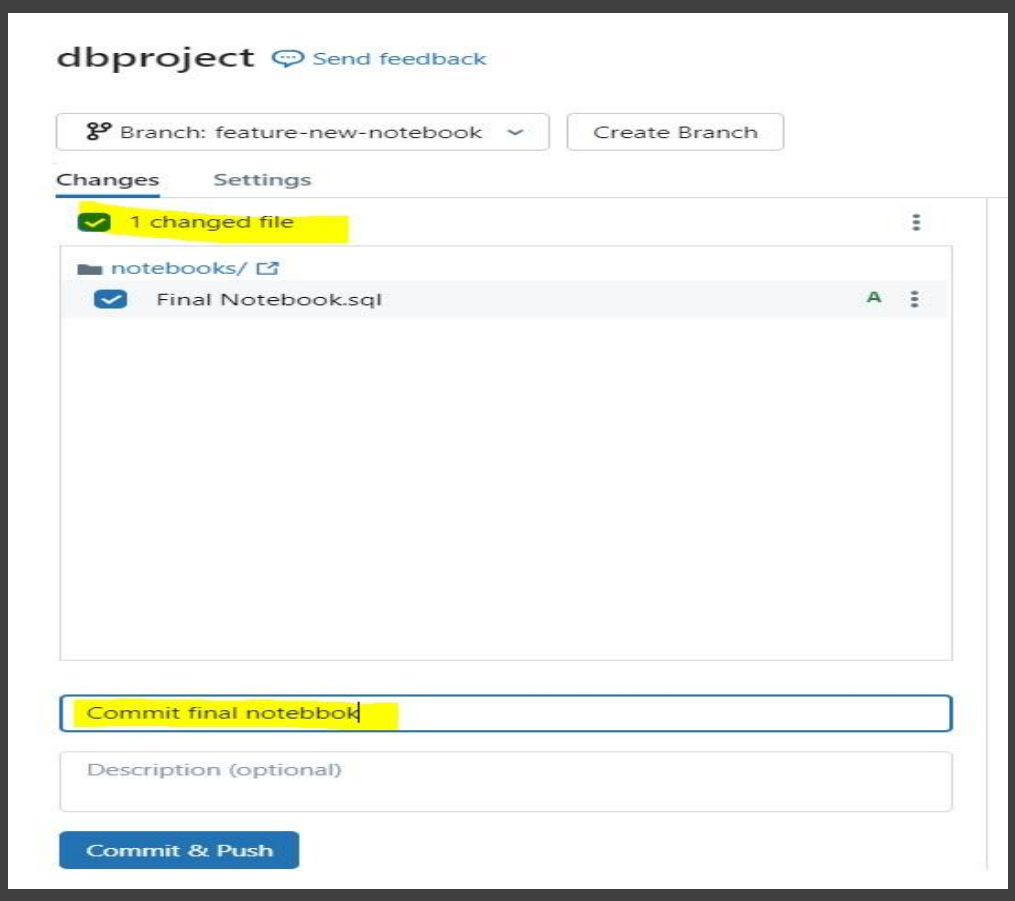


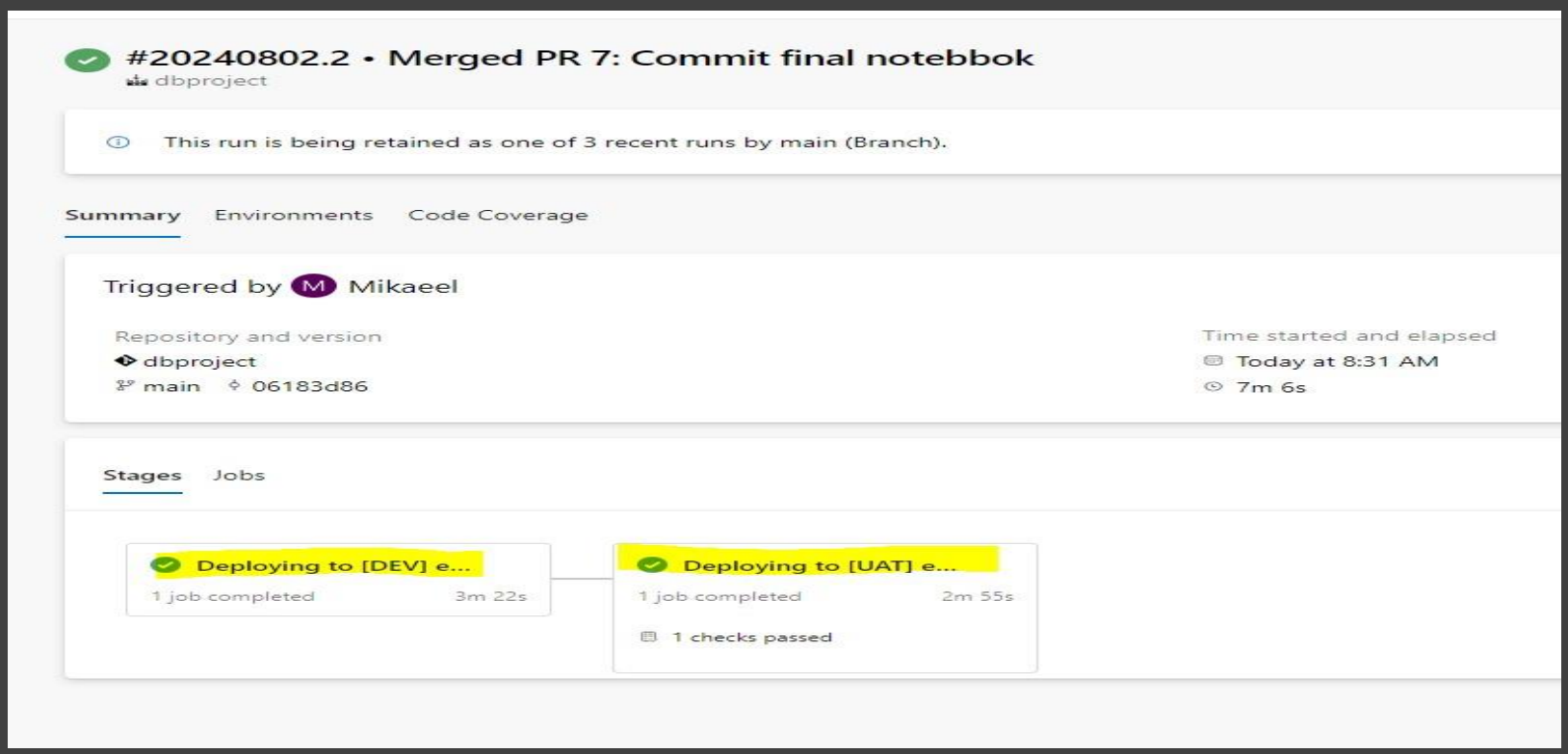
Now create another feature notebook for final results:

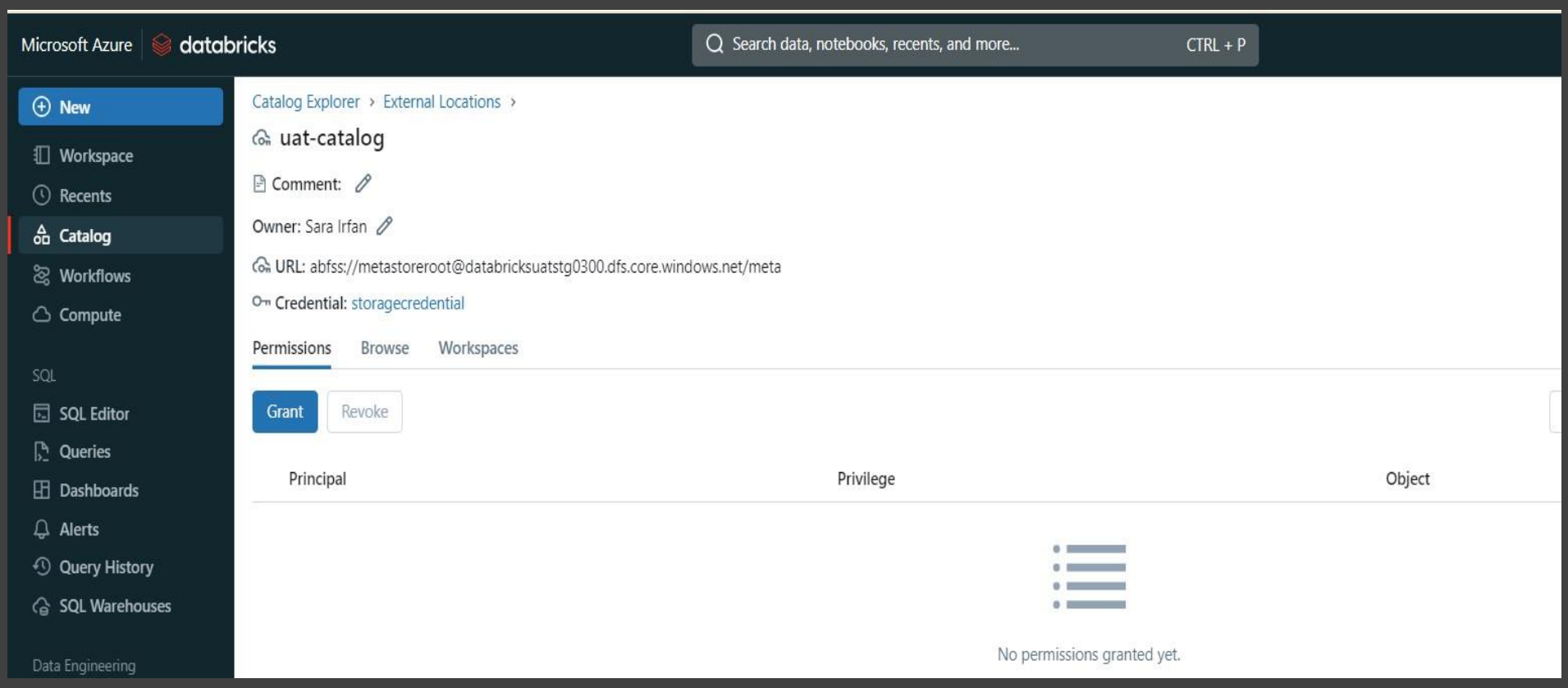


Create a final notebook inside notebooks folder:









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?

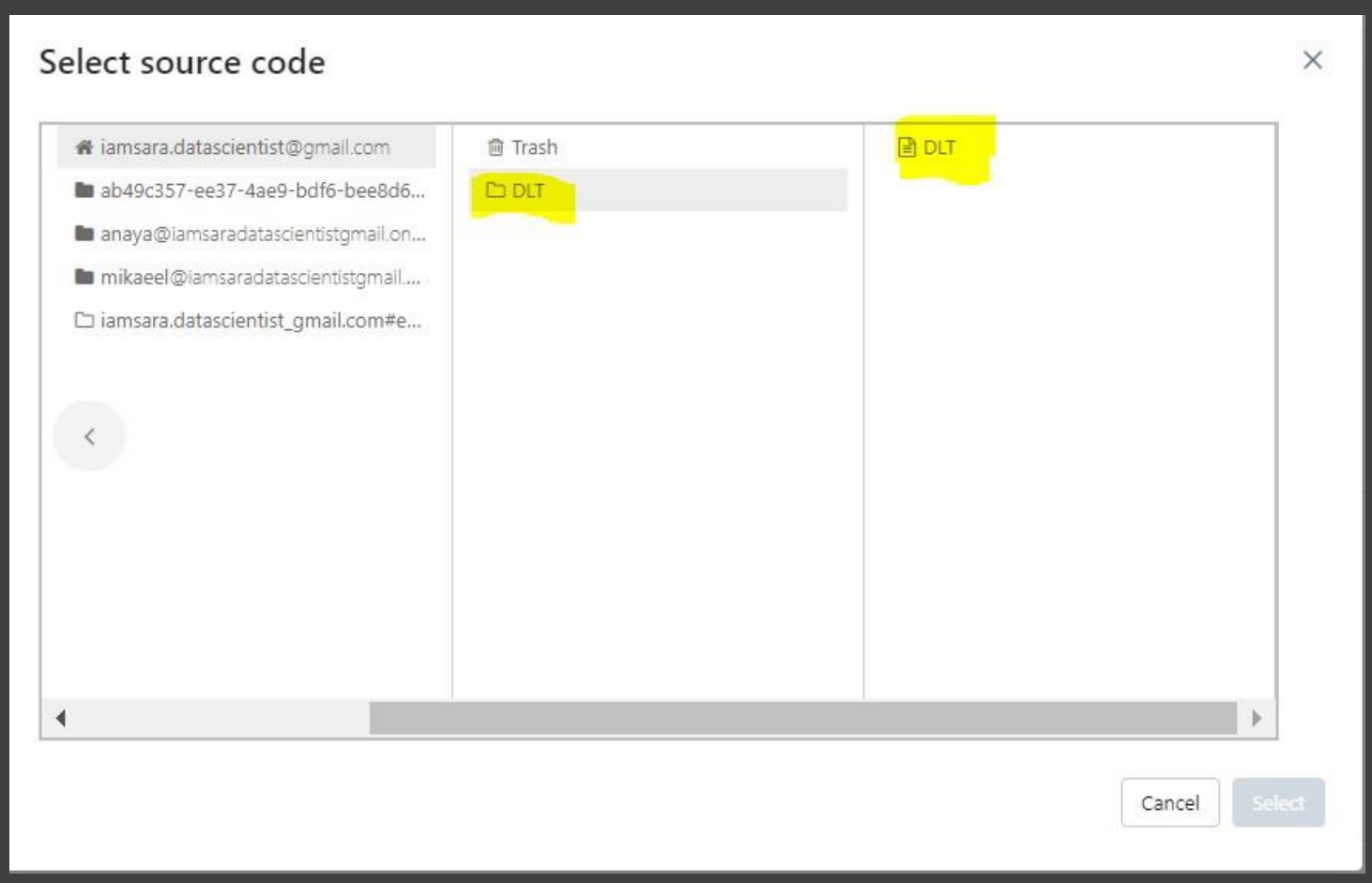
- A **premium workspace** is required to execute Delta Live Tables, as it supports the necessary features and integrations.
- Delta Live Tables supports **Python and SQL**. Interaction through magic commands is not supported, and **only one language** can be used **per notebook**.
- Delta Live Tables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

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

Step-by-Step Answer:

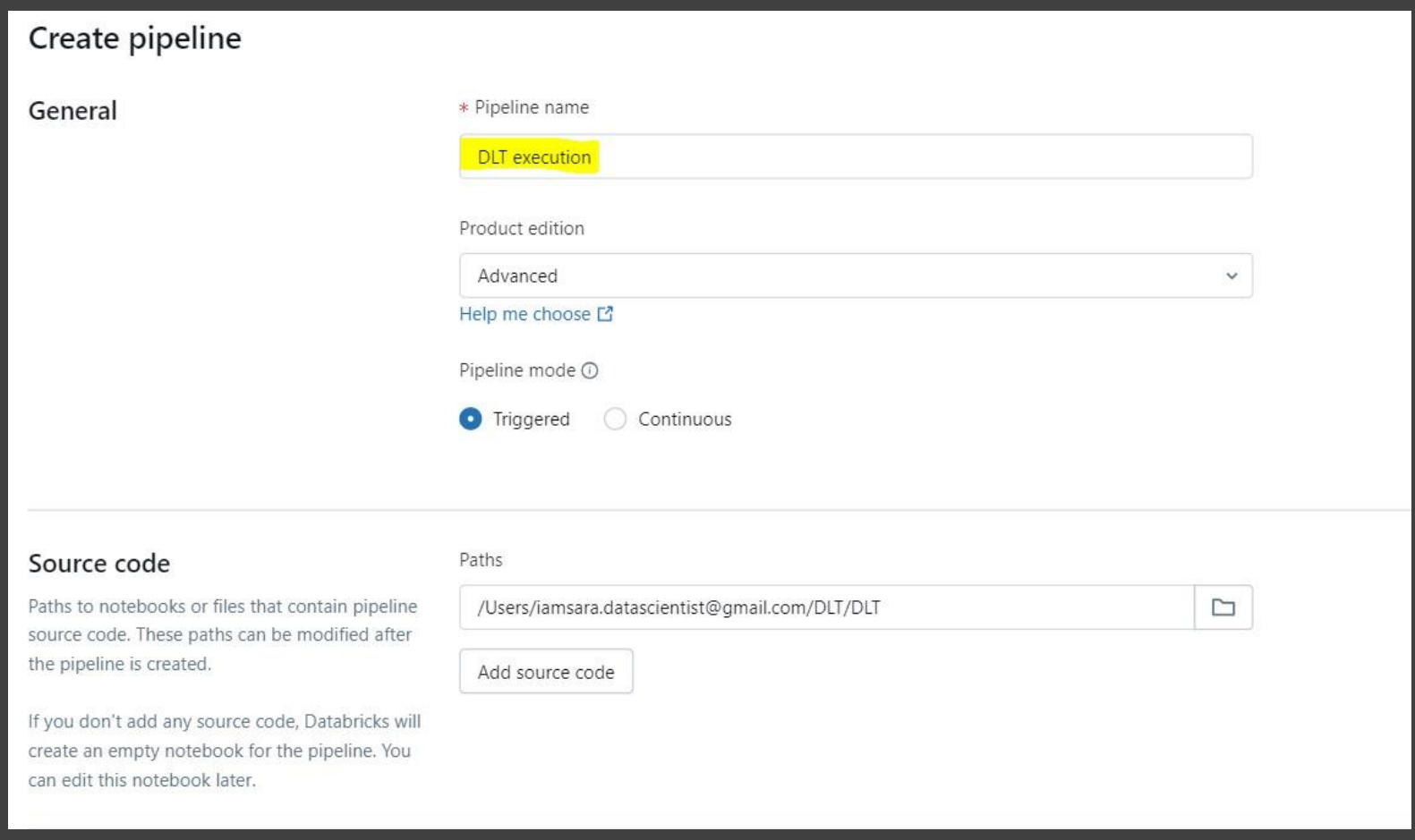
1. **To create a Delta Live Table pipeline:**
   - **Create a notebook in the workspace:**
     - **Navigate to the workspace and create a new folder and notebook.**
   - **Write the code in the notebook:**
     - **Write the code that will create the Delta Live Tables.**
     - **Use the Autoloader 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.**



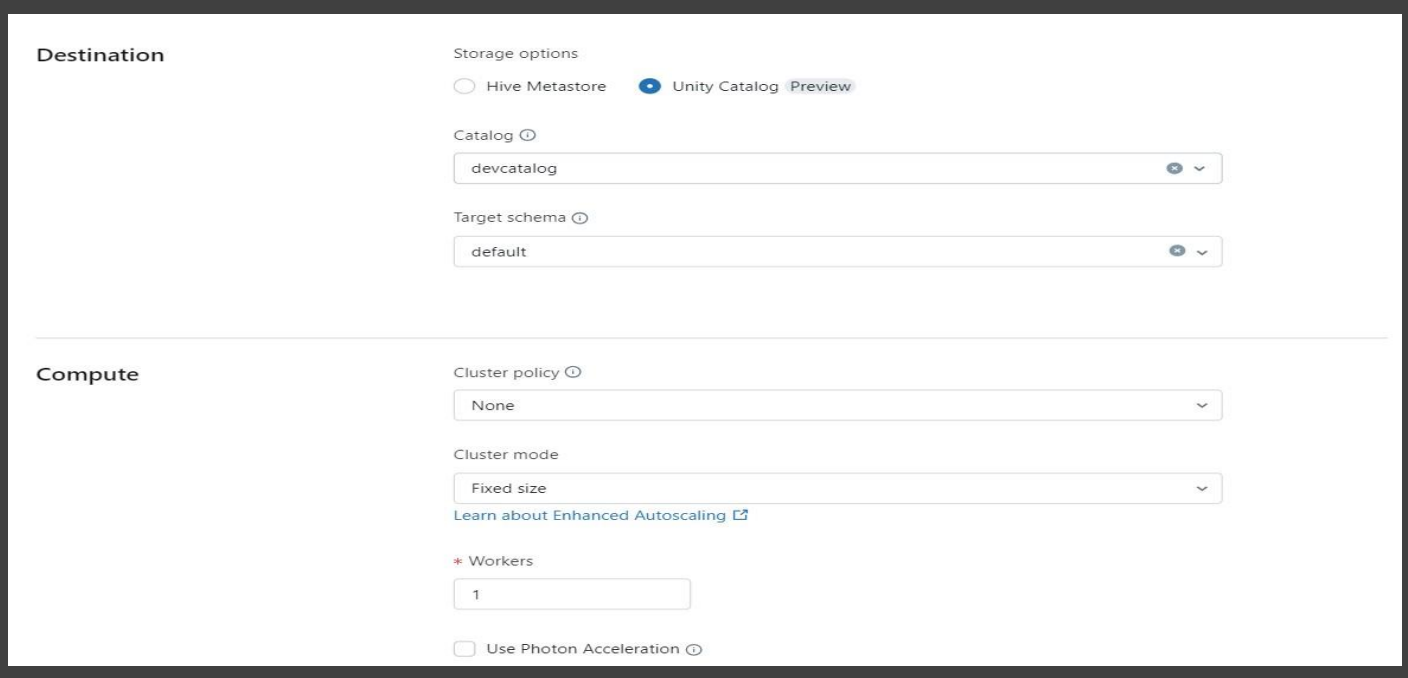
   - **Create the pipeline:**
     - **Choose the option to create a Delta Live Table pipeline.**
     - **Name the pipeline and specify the path to the notebook.**

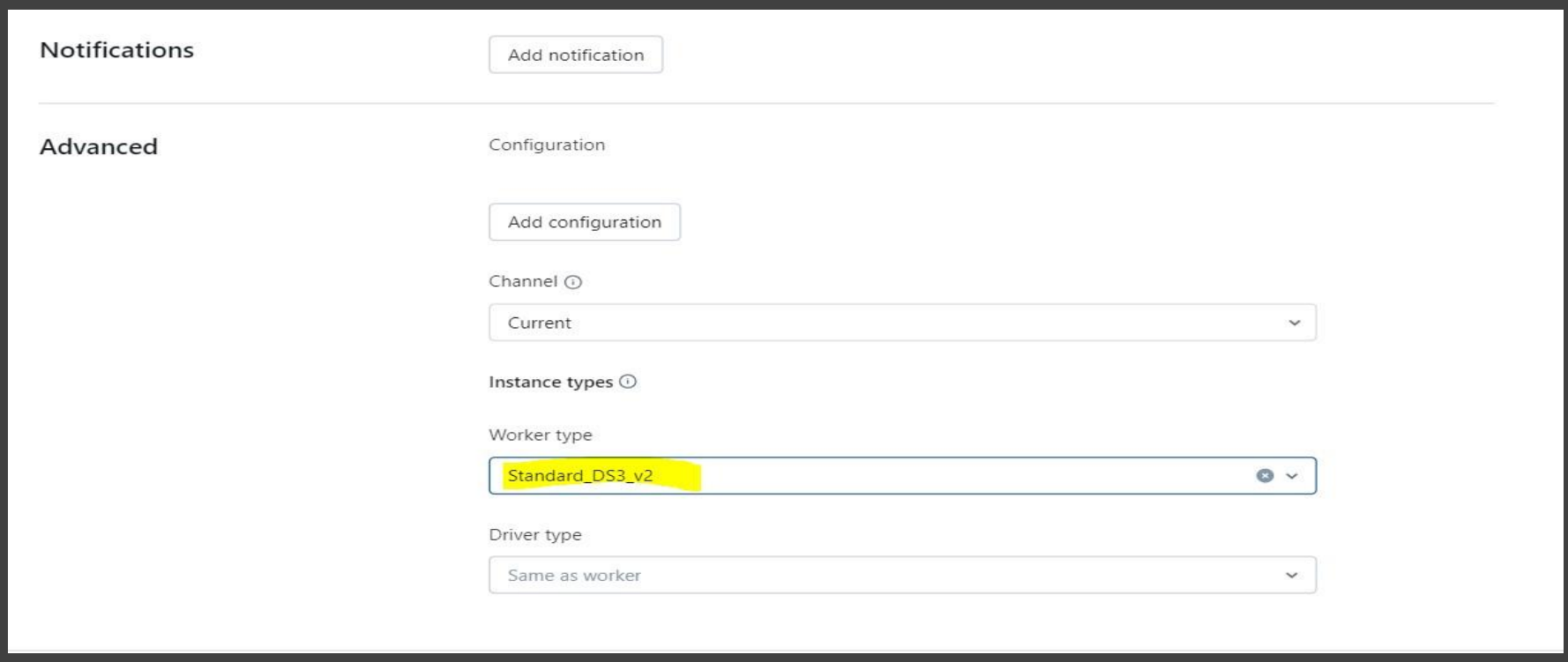
- **Select the Unity Catalog and schema.**

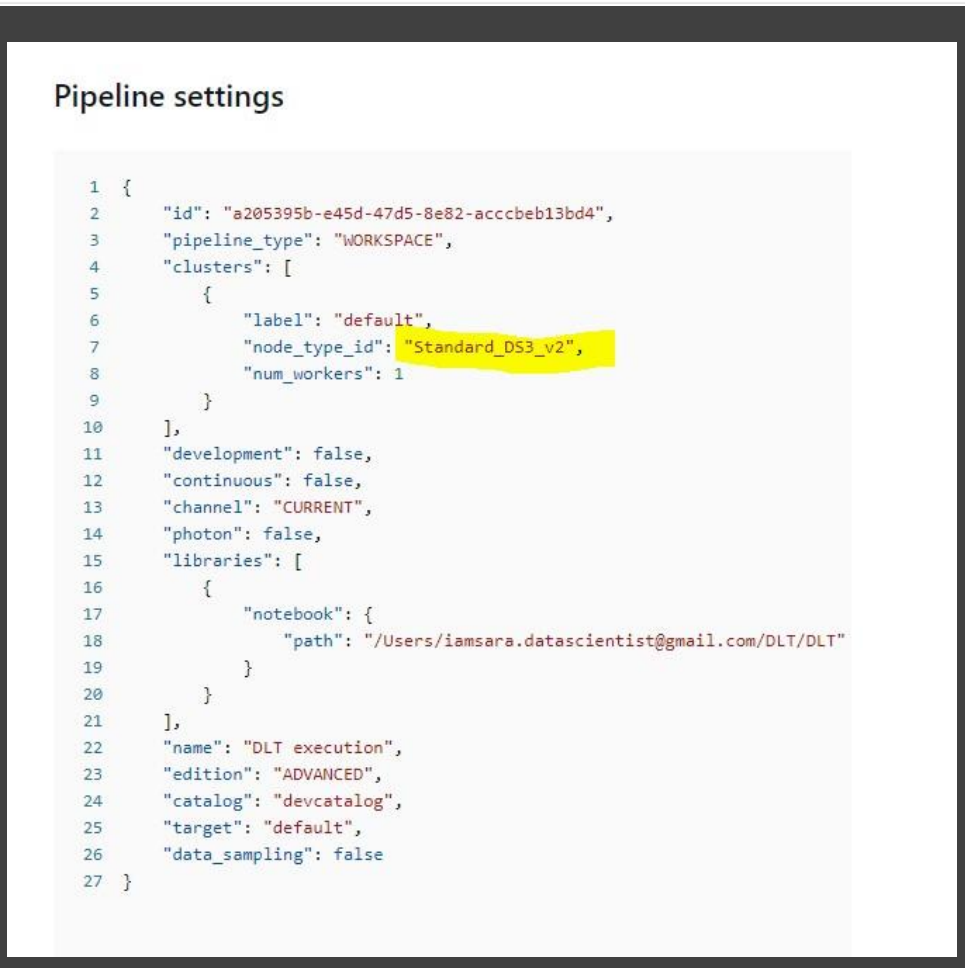


- **Configure the compute settings:**
  - **Specify the cluster node and node type.**
  - **Use the JSON editor to specify the node type ID.**
- **Start the pipeline:**
  - **Click the Start button to run the pipeline.**
  - **Choose between development or production mode and monitor the execution.**

Selection of destination and compute:



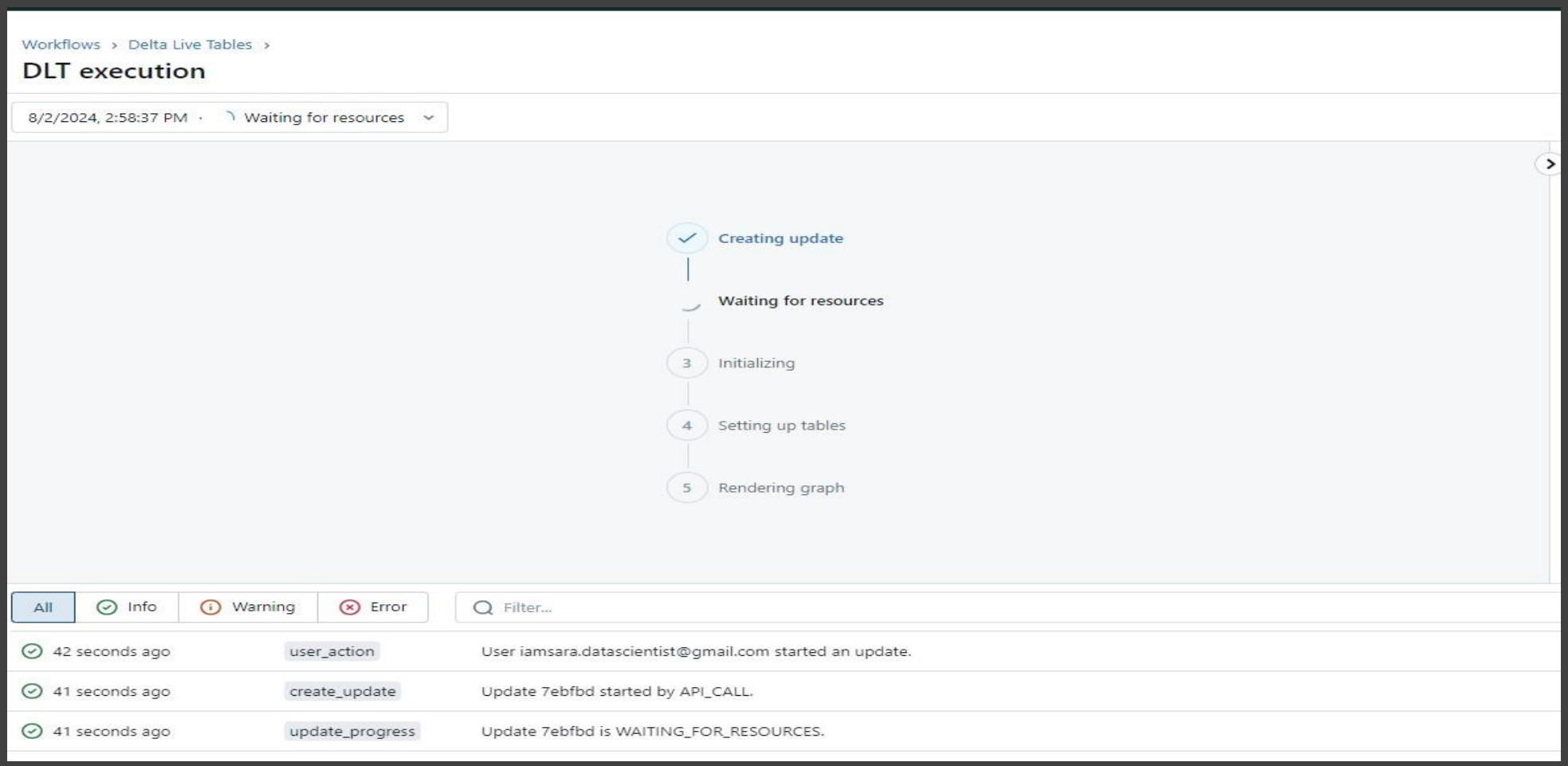




Q. How to implement data quality checks in a Delta Live Table pipeline?

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 on_violation="drop" to drop rows if the data quality rule is violated.
   - Define on_violation="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.

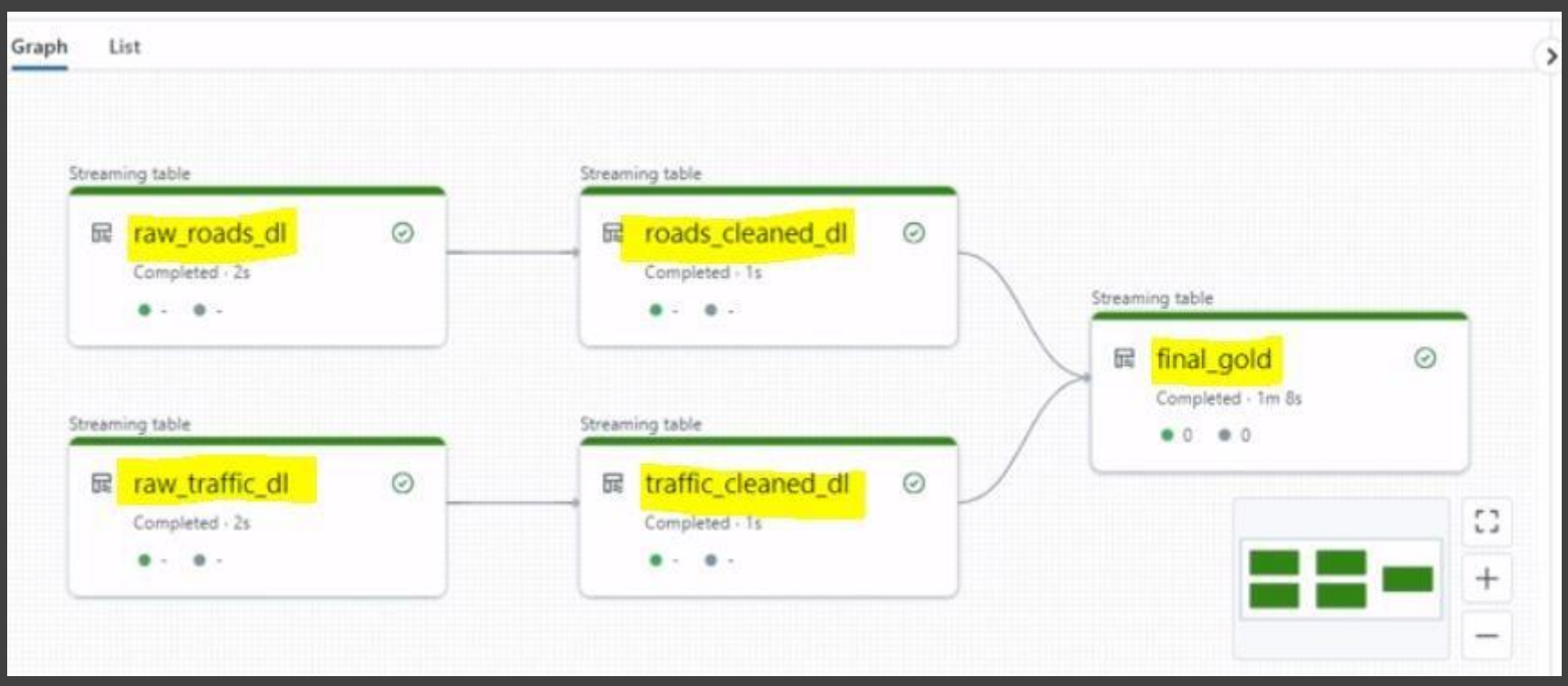


Q. How to create and interpret a dependency graph in a Delta Live Table pipeline?

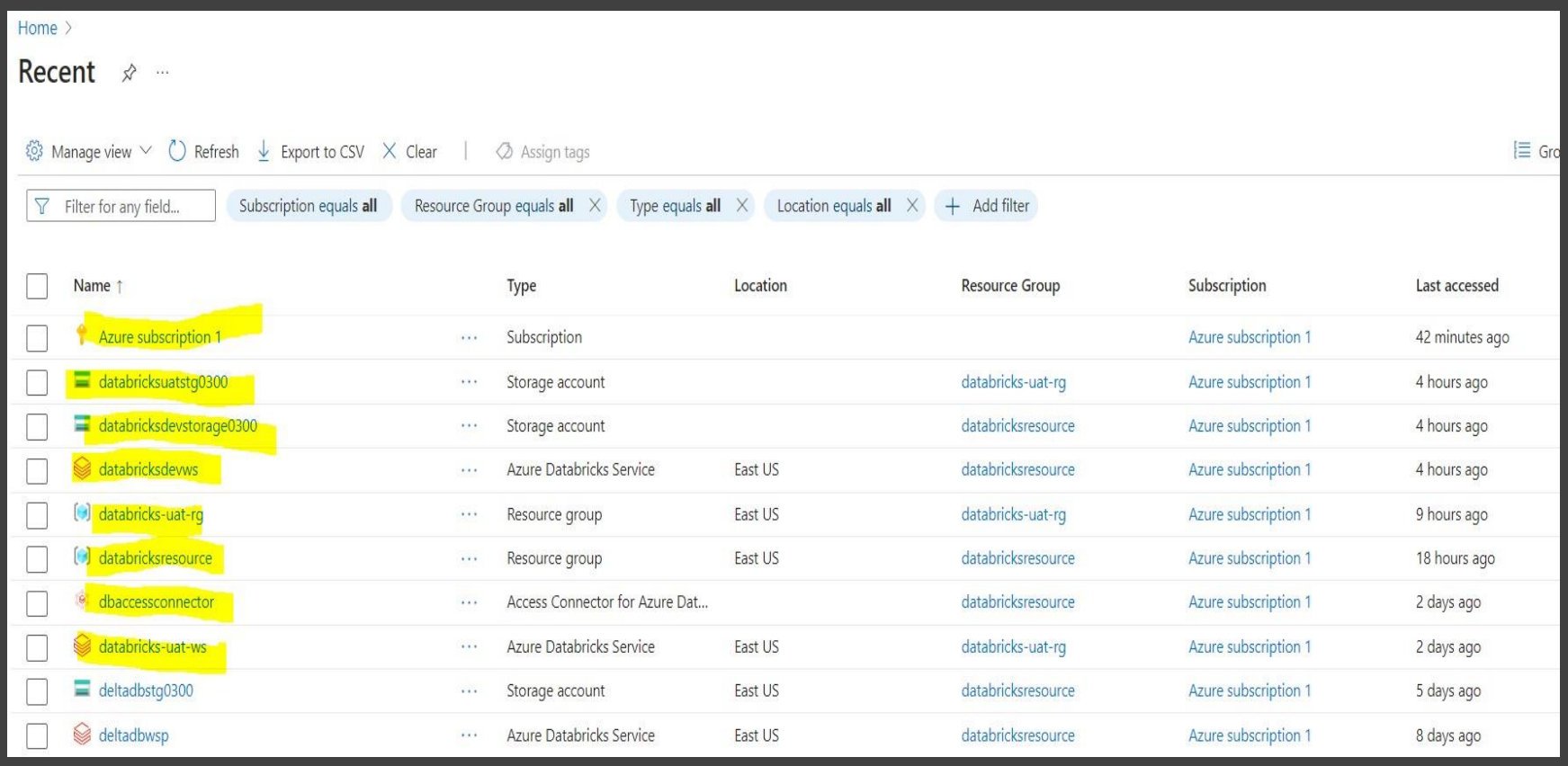
Step-by-Step Answer:

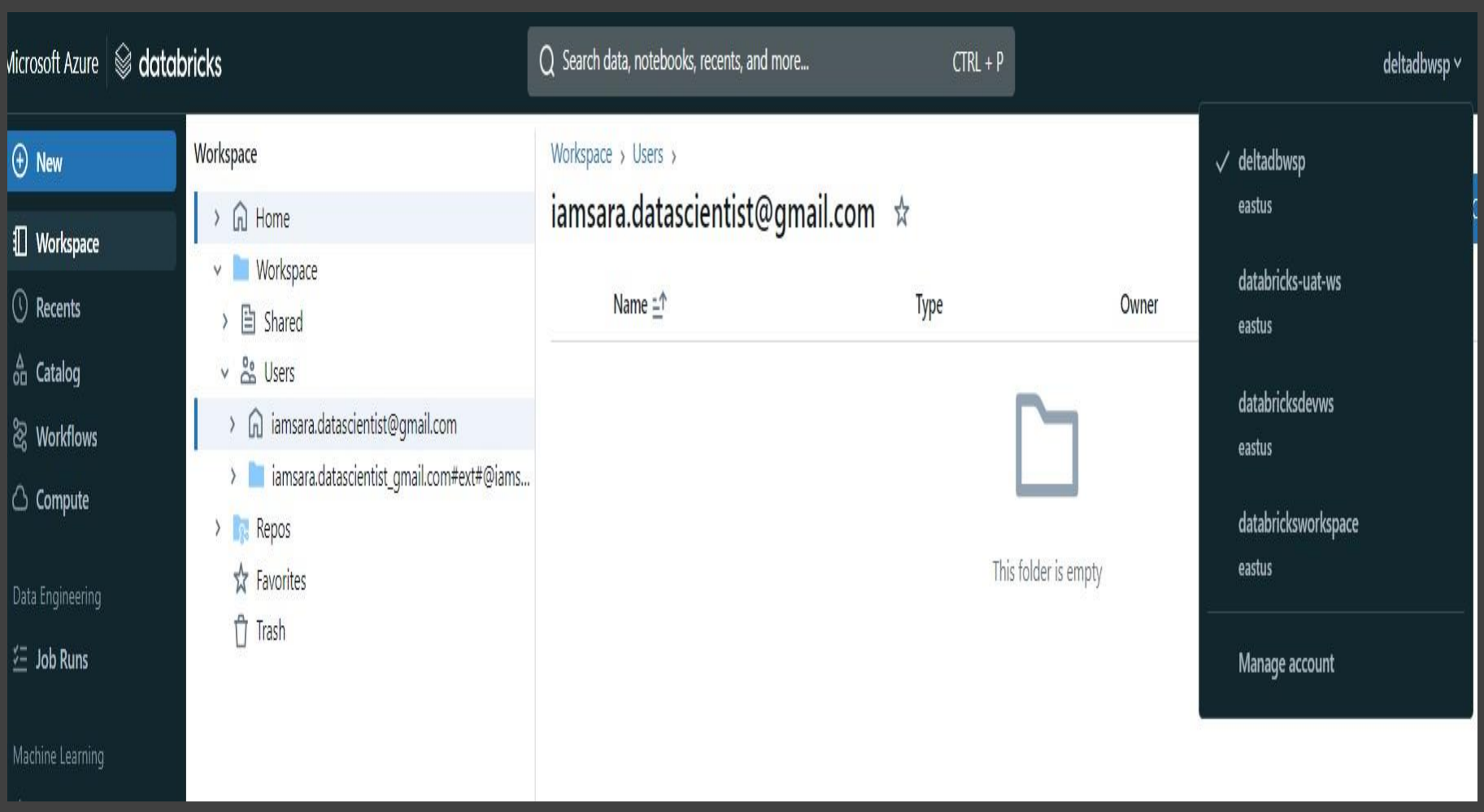
1. To create a dependency graph:
   - Write the code for each table that performs transformations and creates new tables.
   - For example, define multiple tables with the @dlt.table decorator that depend on each other.
   - After writing the notebook code, run the pipeline.
   - Delta Live Tables automatically 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.

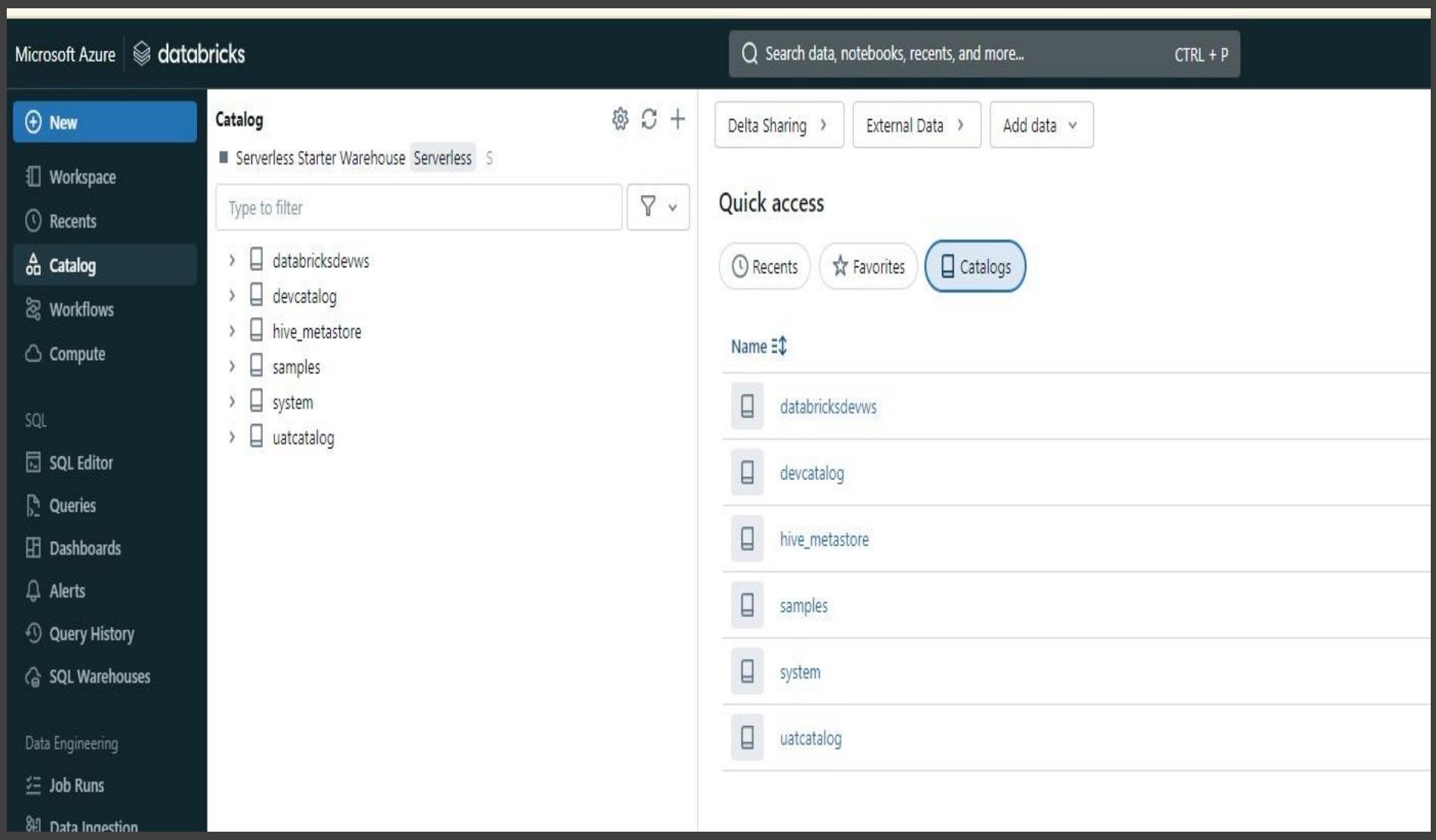
Live Delta Pipeline



Resources used in my project:






Microsoft Azure
databricks
Search data, notebooks, recents, and more...
CTRL + P
New
Workspace
Recents
Catalog
Workflows
Compute
SQL
SQL Editor
Queries
Dashboards
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
Catalog
Serverless Starter Warehouse
Serverless
Type to filter
databricksdevws
devcatalog
hive_metastore
samples
system
uatcatalog
Delta Sharing
External Data
Add data
Quick access
Recents
Favorites
Catalogs
Name
databricksdevws
devcatalog
hive_metastore
samples
system
uatcatalog

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