**AUTOMATED ETL with DATA LAKE STORAGE**

**Project statement**

Utilize Azure Data Factory to orchestrate an ETL pipeline that ingests data from various sources into Azure Data Lake Storage. Use Azure Databricks for complex data transformations.

**Project Overview**

The goal of the project, based on the outlined steps, is to create a robust ETL (Extract, Transform, Load) pipeline using Azure services.

**Project Requirements**

o Azure Data Factory

o Azure Databricks

o Azure Data Lake Storage

o Optimized Data Storage and Retrieval

o Documentation

o Testing and Validation

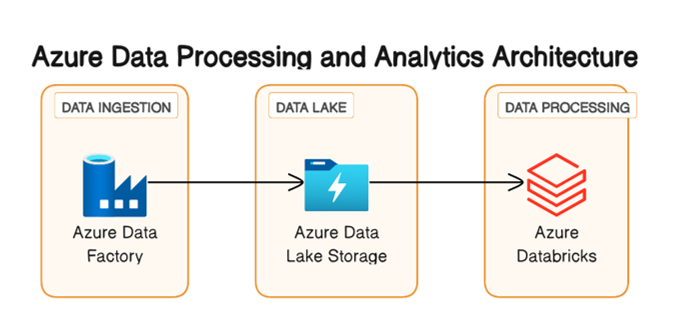
**Prerequisites**

o Azure Subscription: An active Azure subscription to deploy and manage resources.

o Access Rights: Appropriate rights for Azure Data Factory, Data Lake Storage and Databricks.

o Data Sources: Access to the source data that needs to be moved to Azure Data Lake Storage.

**Architecture Diagram**



Azure data factory for data movement and organization

Azure data lake storage for storing data

Azure databricks for data processing and analytics

**Tools and Technologies**

• Azure Data Factory: For data movement and organization

• Azure Databricks: For data processing and analytics

• Azure Data Lake Storage (ADLS): For storing structured and unstructured data

• Azure Monitor and Logging Services: For monitoring and logging the data pipeline.

**Execution Overview**

**Set Up Azure Data Lake Storage:**

o Create necessary containers and directories to organize data.

**Configure Azure Data Factory:**

o Set up linked services to connect to various data sources and Azure Data Lake Storage.

o Define data pipelines for data movement and organization.

**Data Movement:**

o Use Azure Data Factory to move data from sources to Azure Data Lake Storage.

**Azure Databricks Integration:**

o Configure Azure Databricks workspace and cluster.

**Data Processing and Analytics:**

o Develop notebooks or jobs in Azure Databricks for processing and analytics.

**Monitor and Log:**

o Set up Azure Monitor and Logging services for tracking the data pipeline's performance and health.

**Implementation Steps**

**Set Up Azure Data Lake Storage:**

o Create necessary containers and directories to organize data.

**Configure Azure Data Factory:**

o Set up linked services to connect to various data sources and Azure Data Lake Storage.

o Define data pipelines for data movement and organization.

**Data Movement:**

o Use Azure Data Factory to move data from sources to Azure Data Lake Storage.

**Azure Databricks Integration:**

o Configure Azure Databricks workspace and cluster.

**Data Processing and Analytics:**

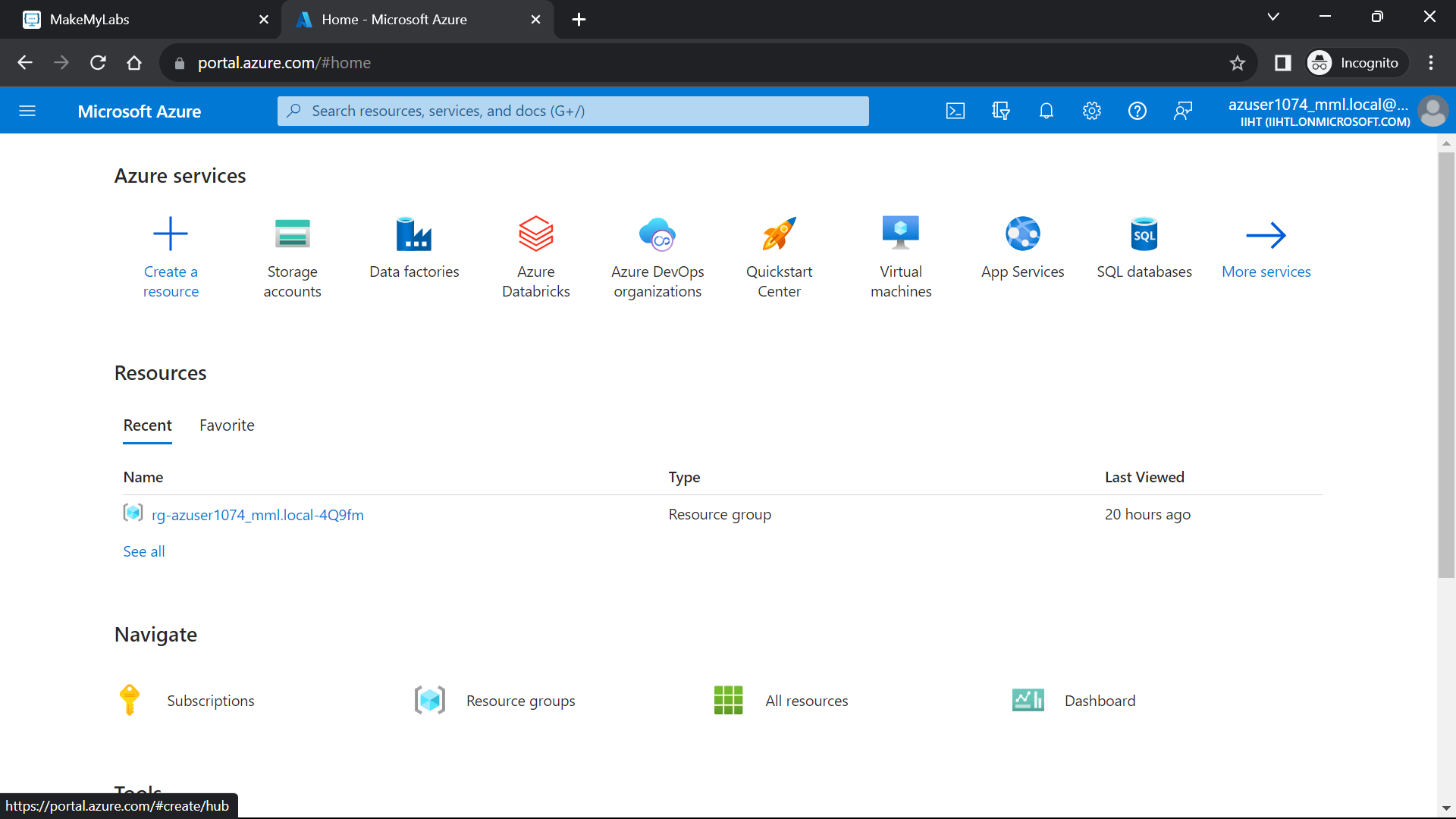
o Develop notebooks or jobs in Azure Databricks for processing and analytics.

Monitor and Log:

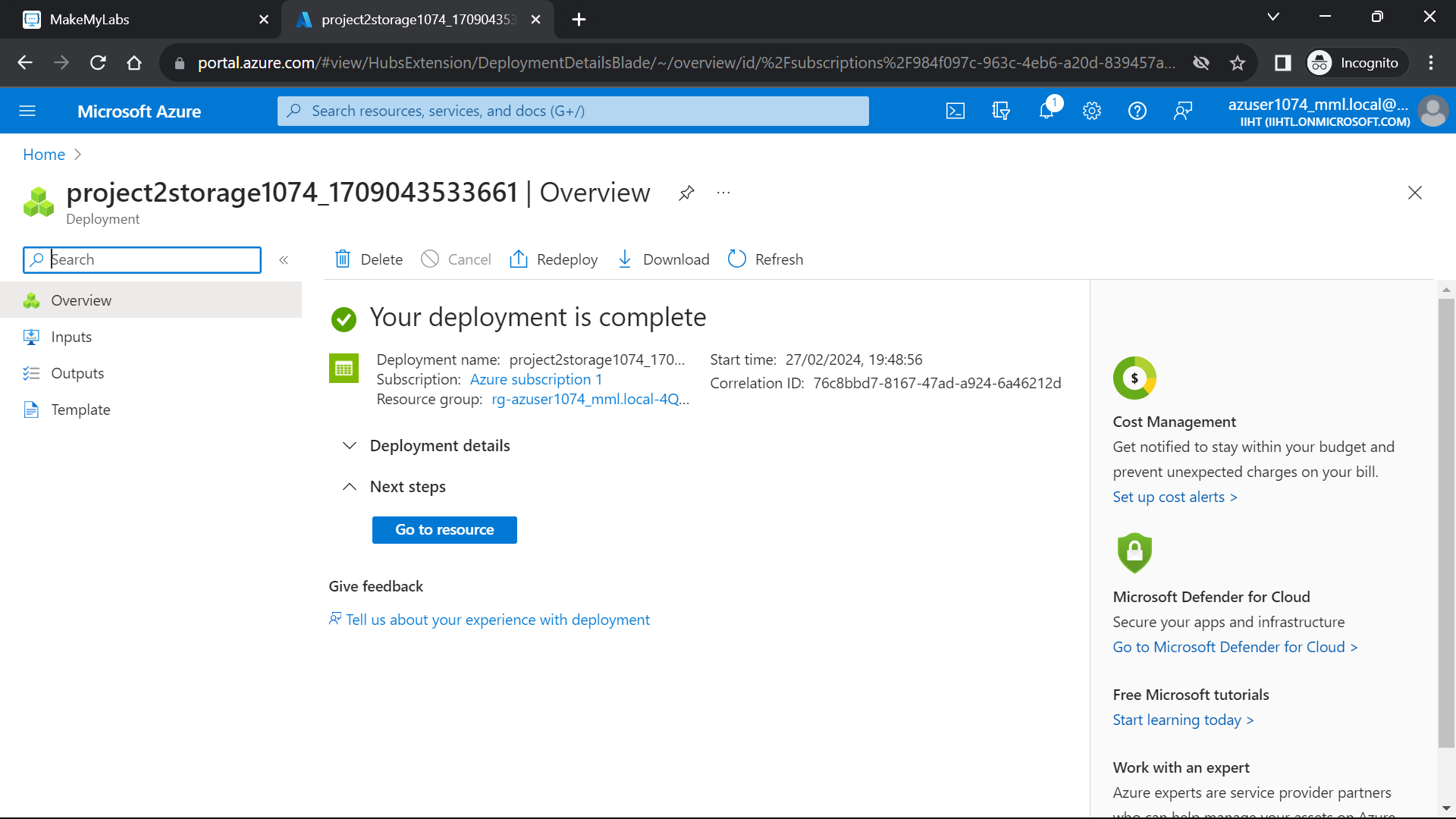
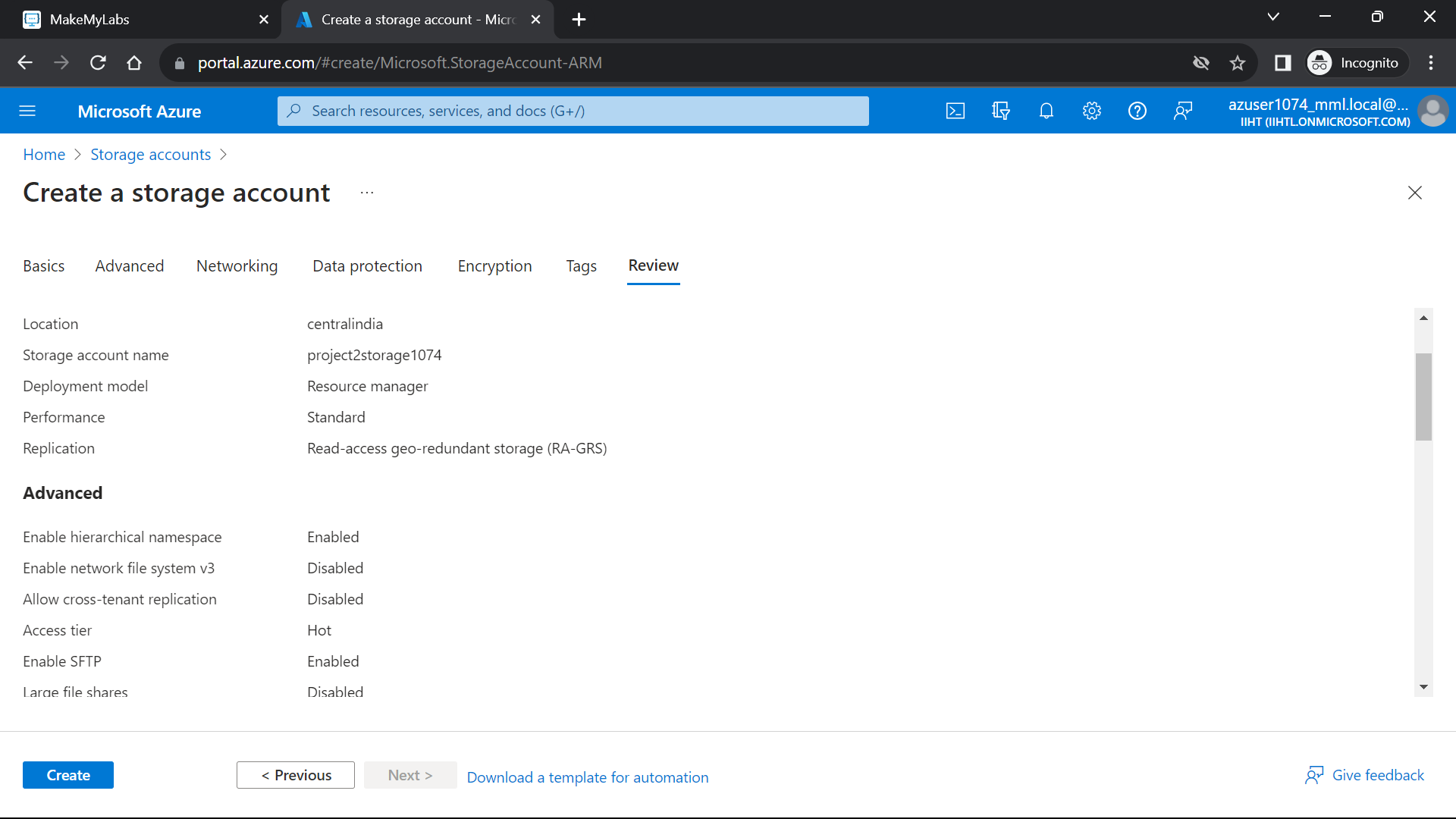
o Set up Azure Monitor and Logging services for tracking the data pipeline performance and health.

**Practical implement on Azure Portal:**

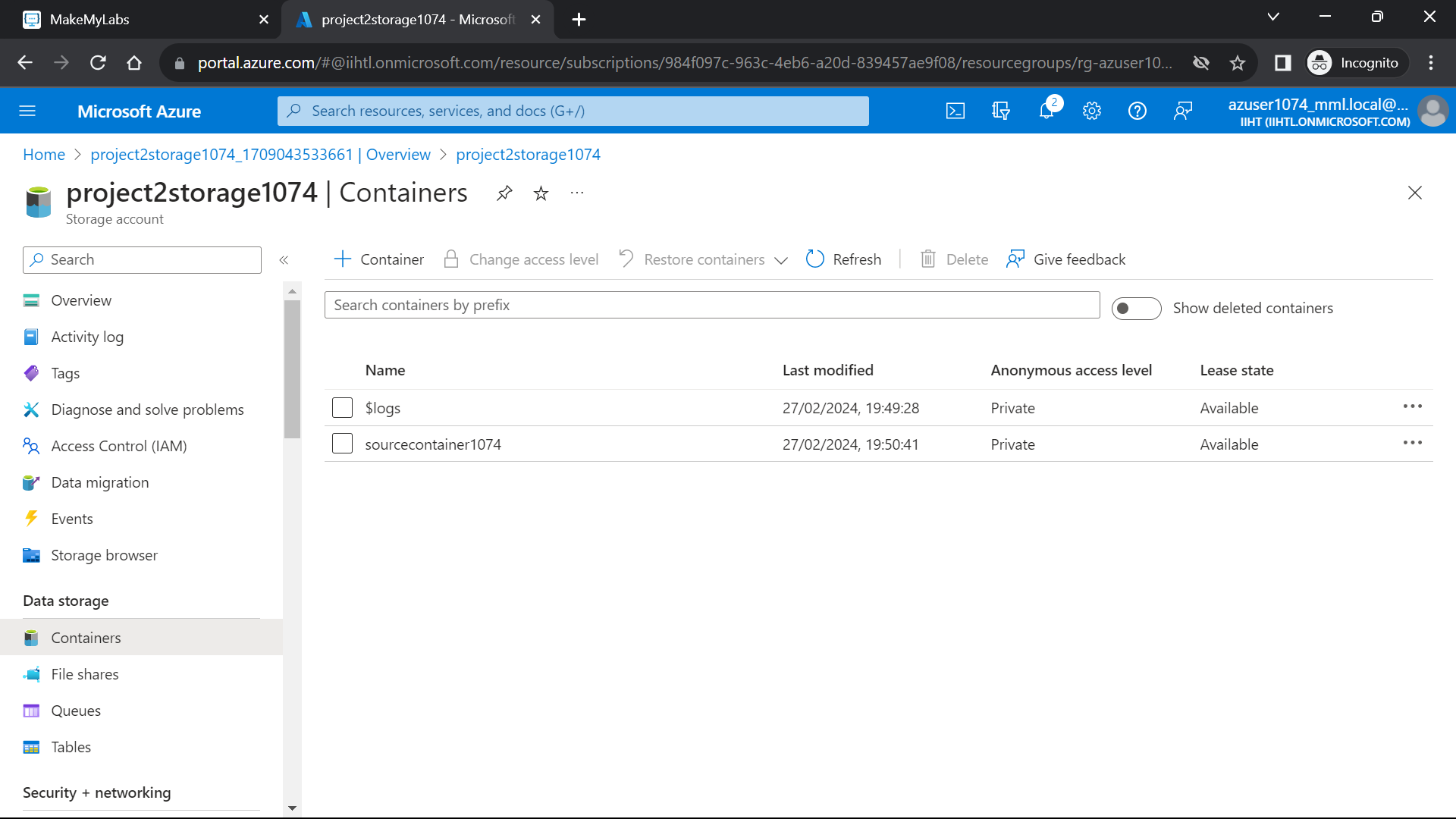
Login to the **Azure portal.**



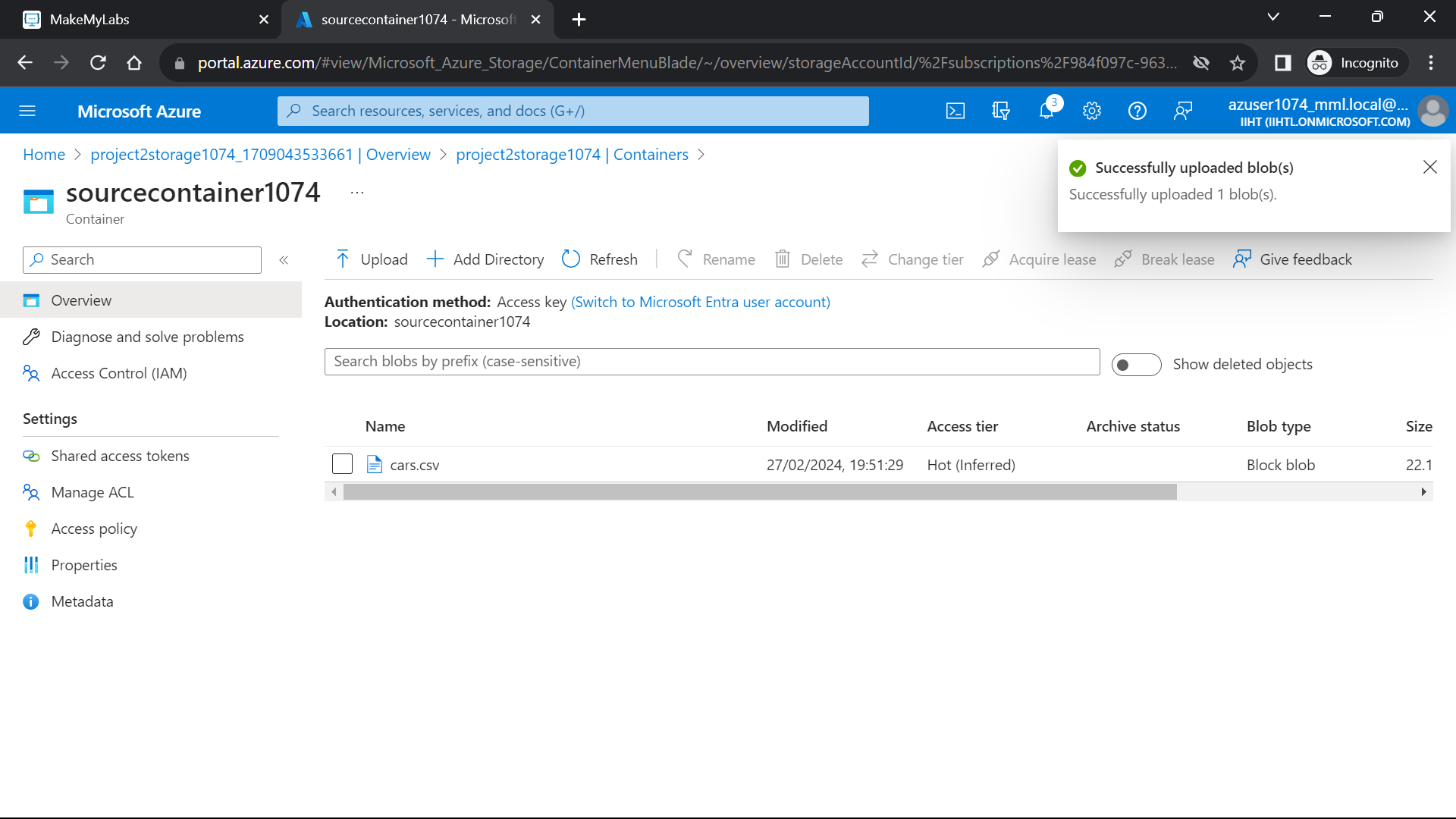
Search for the **Storage account** and create a storage acoount.



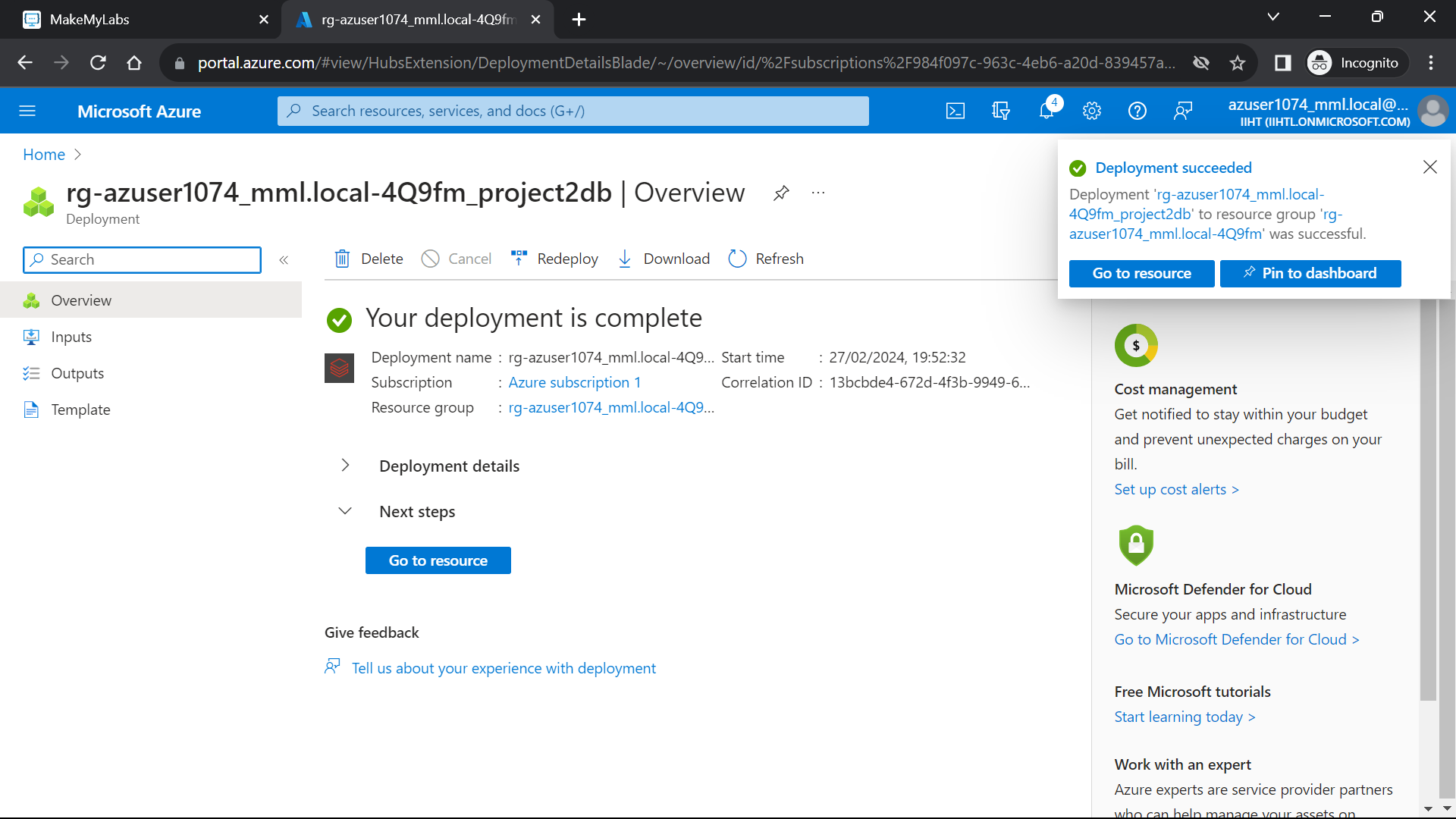
Create a **container**



Go to the container and click on **“upload”** to upload the file in the container.

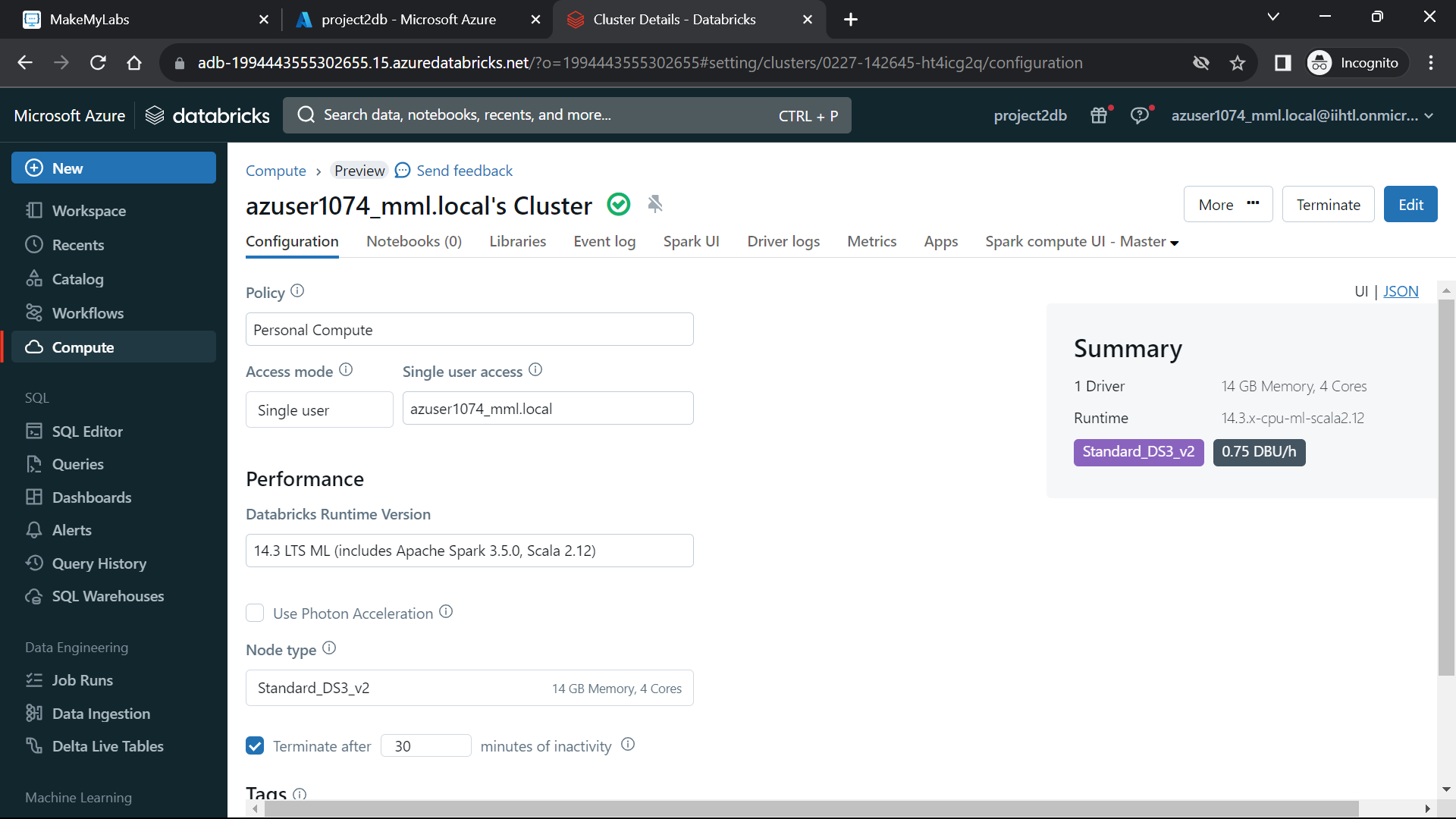


Go to azure home page and select the **“Azure databricks**” and create the new azure databricks workspace.



Now click on **“Compute”** and create the cluster.

The cluster is ready and good to use.



Now create a **notebook** and connect the cluster.

Now to **mount the storage account** to the azure databricks use the below code

**Code:**

storage\_account\_name = “<storageaccountname>"

container\_name = "containername"

mount\_point = "/mnt/mount\_name "

# Use the access key obtained in Step 1

storage\_account\_key = "<key>"

# Create the mount point

dbutils.fs.mkdirs(mount\_point)

# Mount Azure Storage

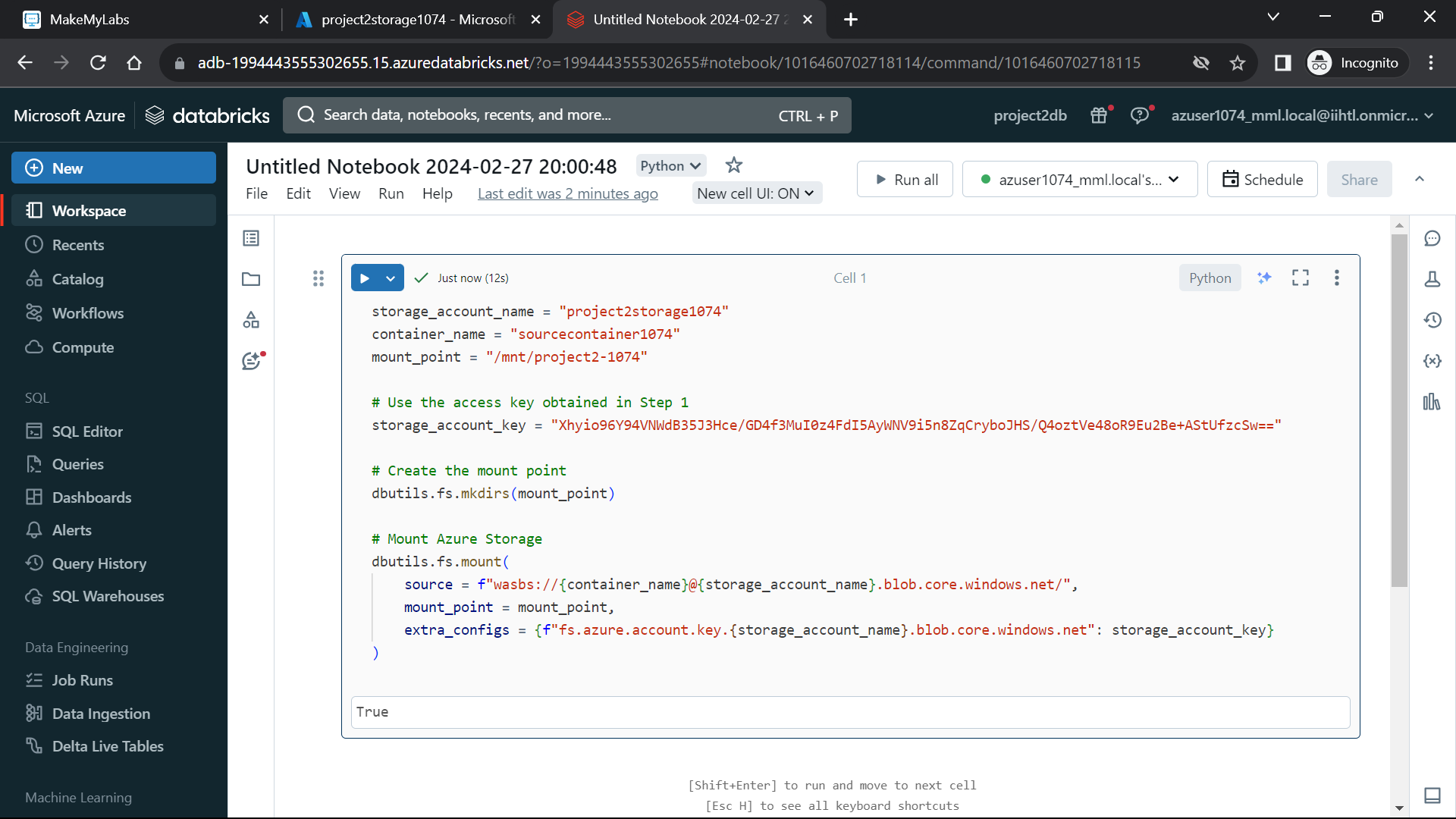
dbutils.fs.mount(

source = f"wasbs://{container\_name}@{storage\_account\_name}.blob.core.windows.net/",

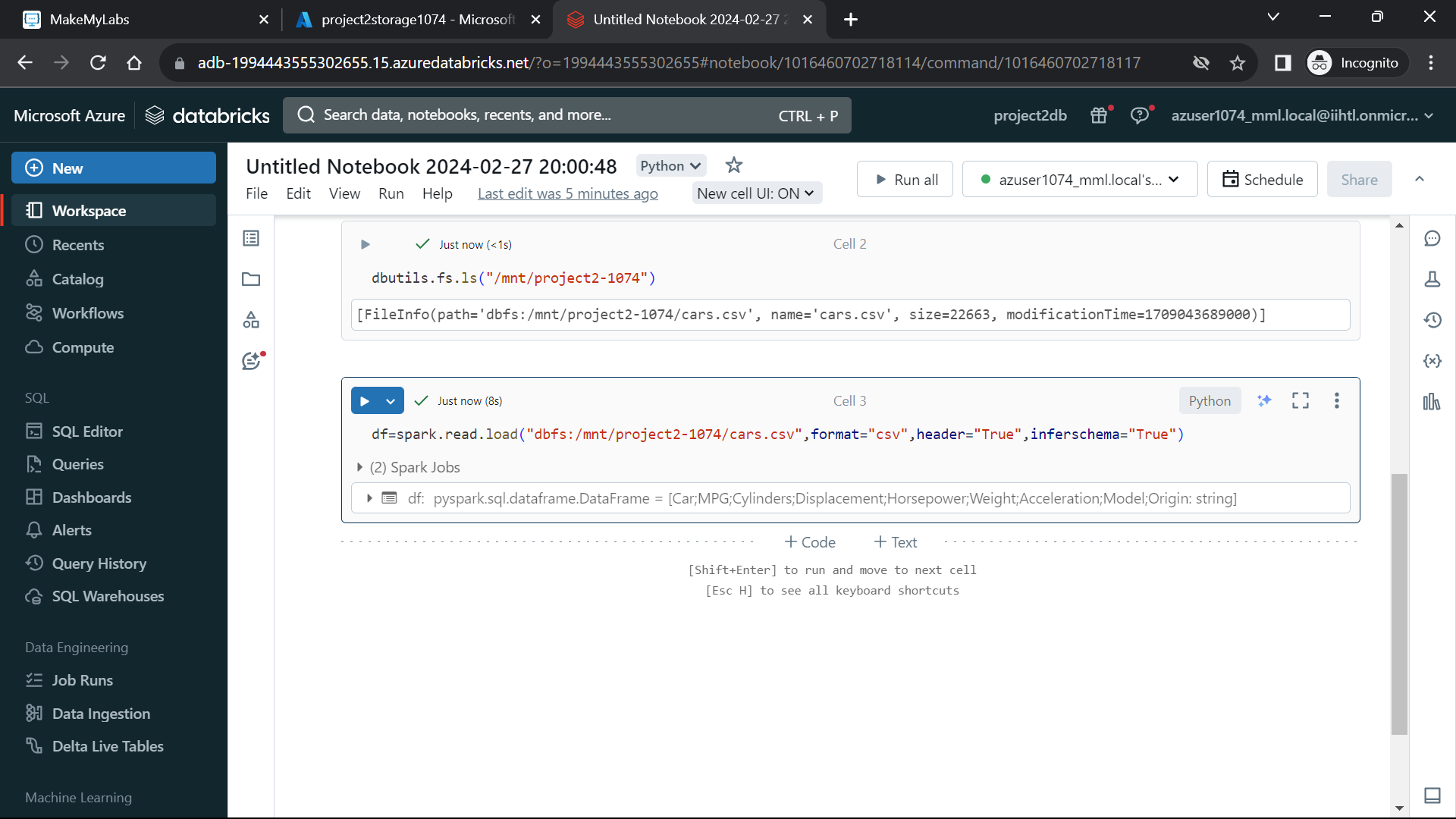
mount\_point = mount\_point,

extra\_configs = {f"fs.azure.account.key.{storage\_account\_name}.blob.core.windows.net": storage\_account\_key}

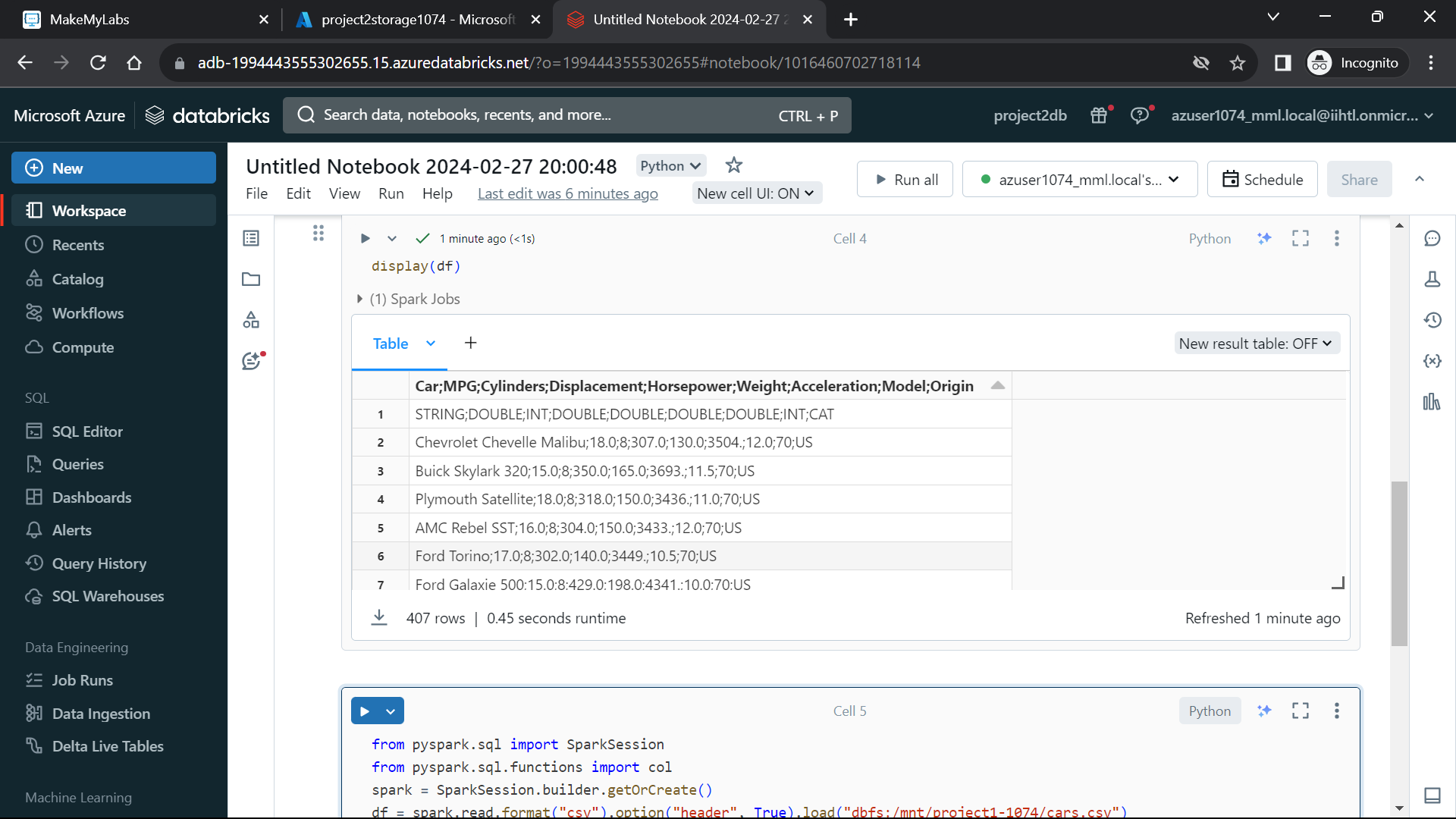
)

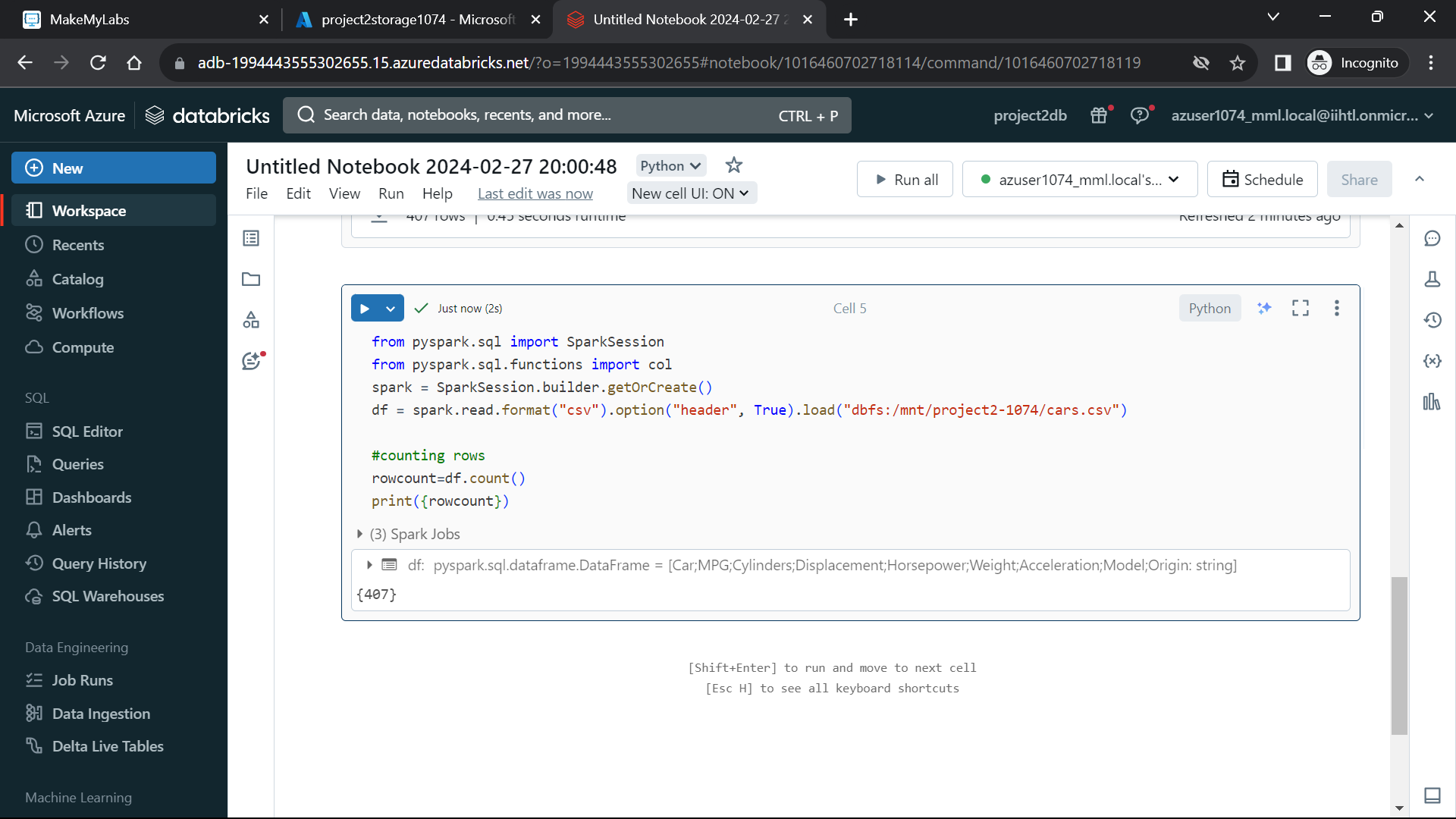


Now load the file into the databricks by using the **“spark.read.load(“filepath”)”.**

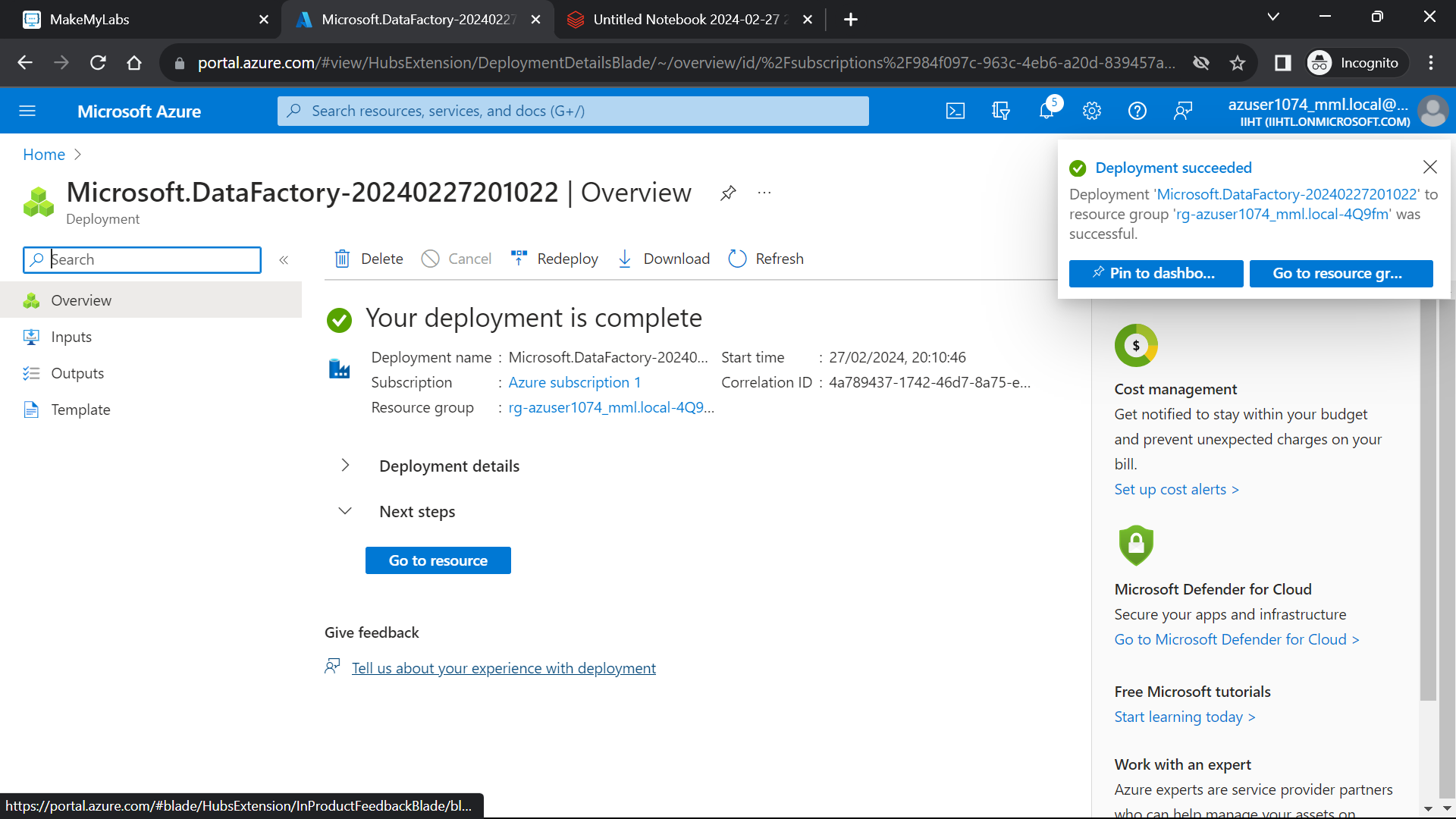


Now perform some transformations on the file

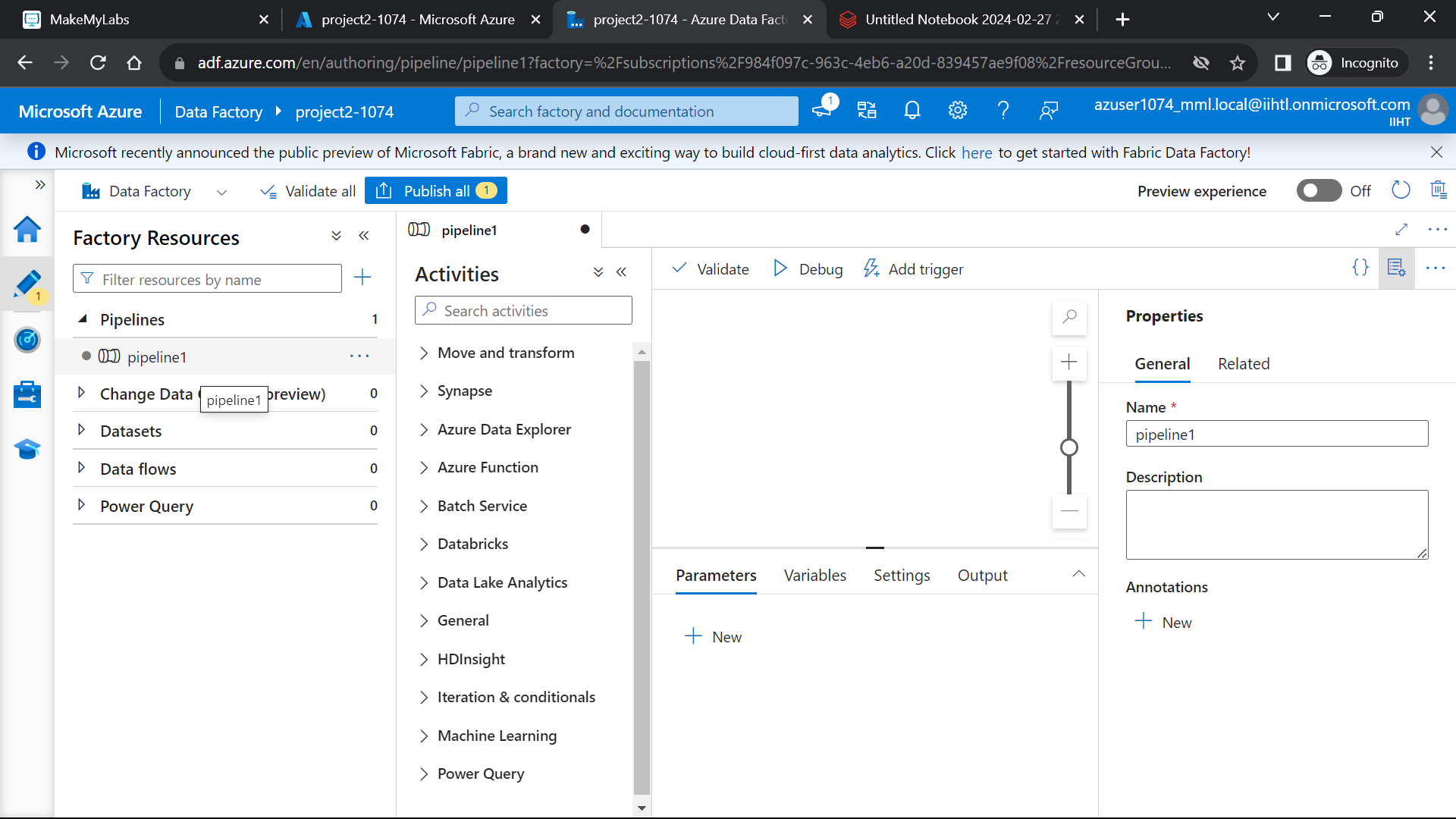




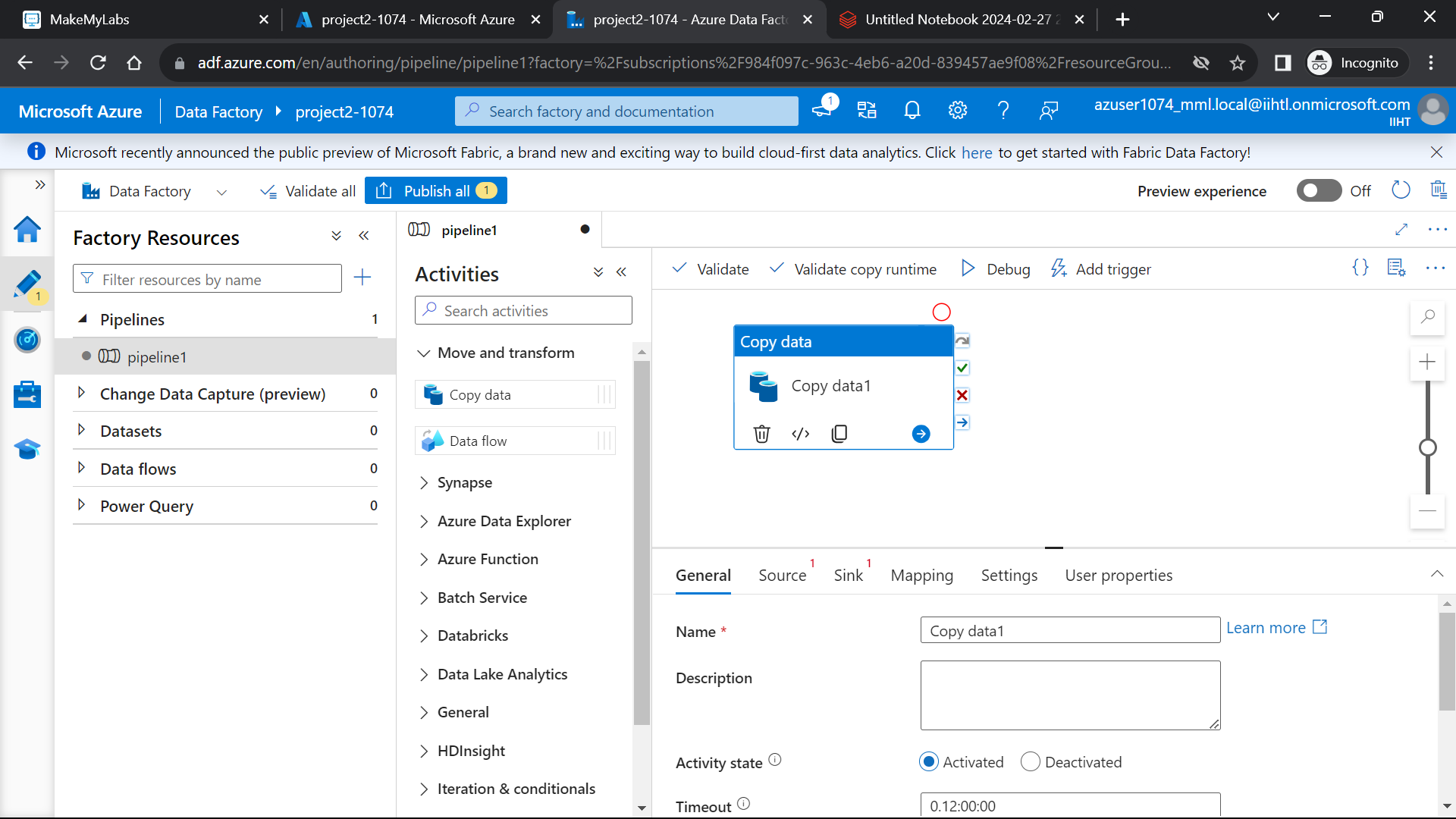
Create the **“data factories”**

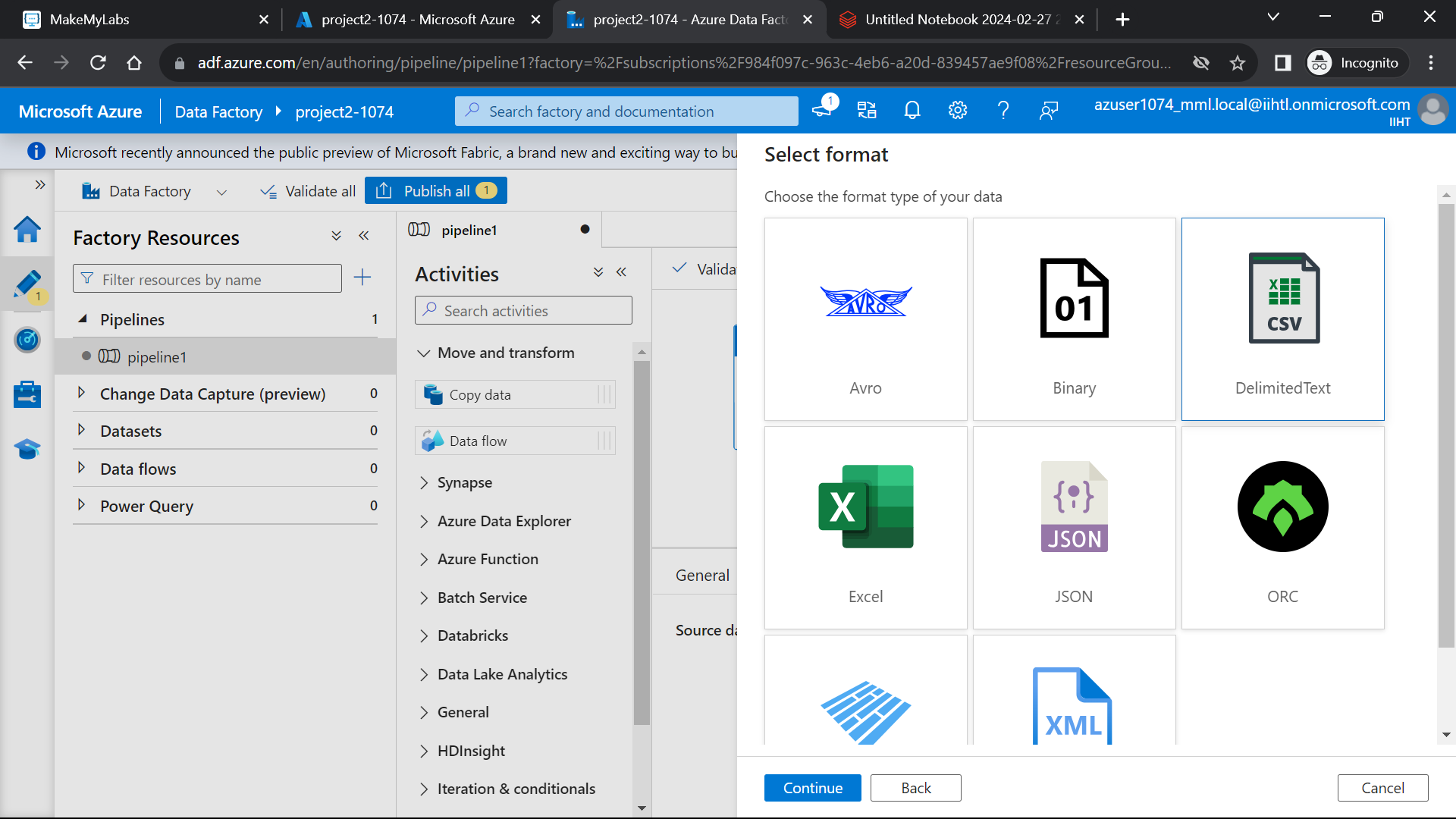


Create a **new pipeline**.

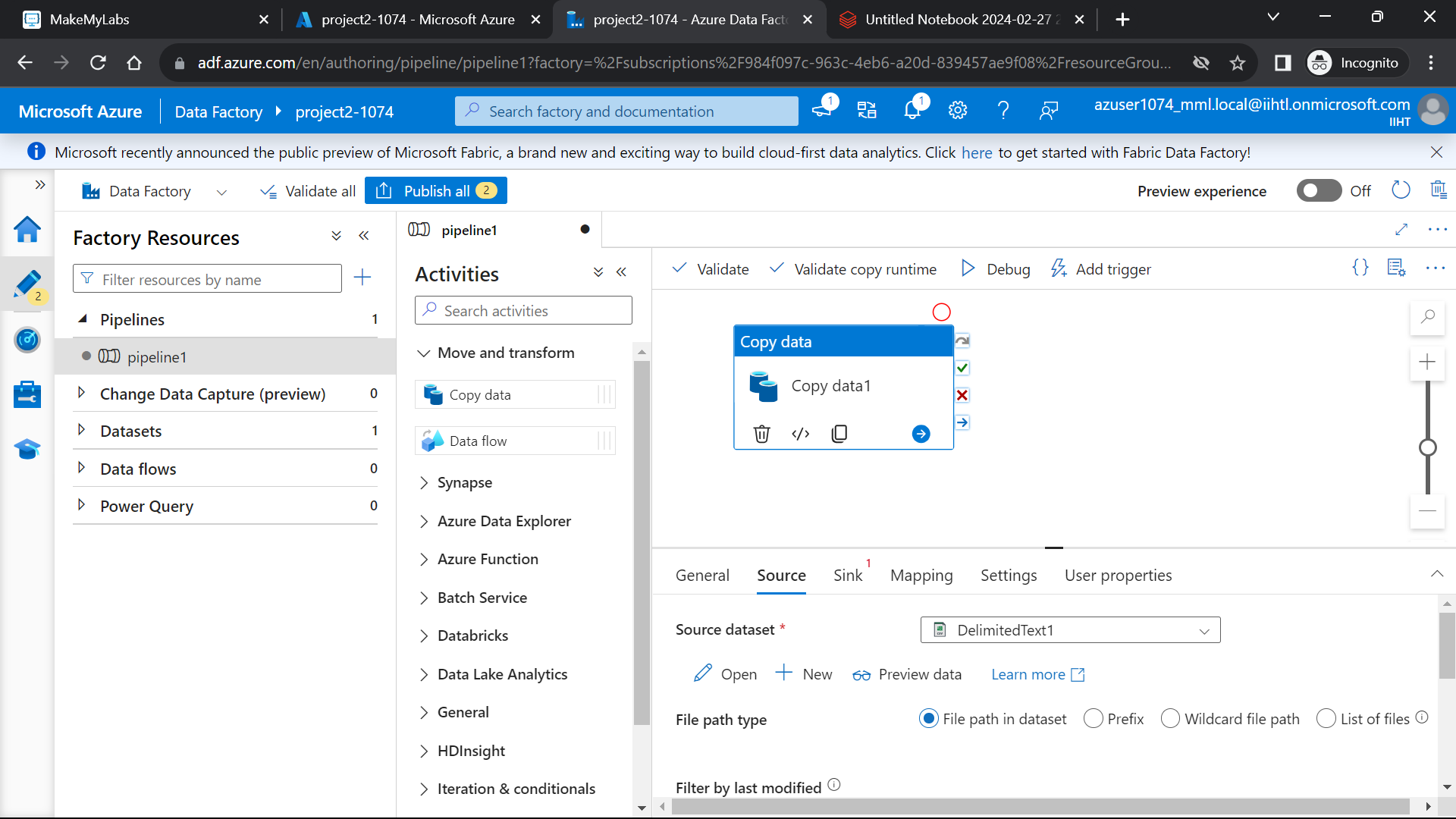


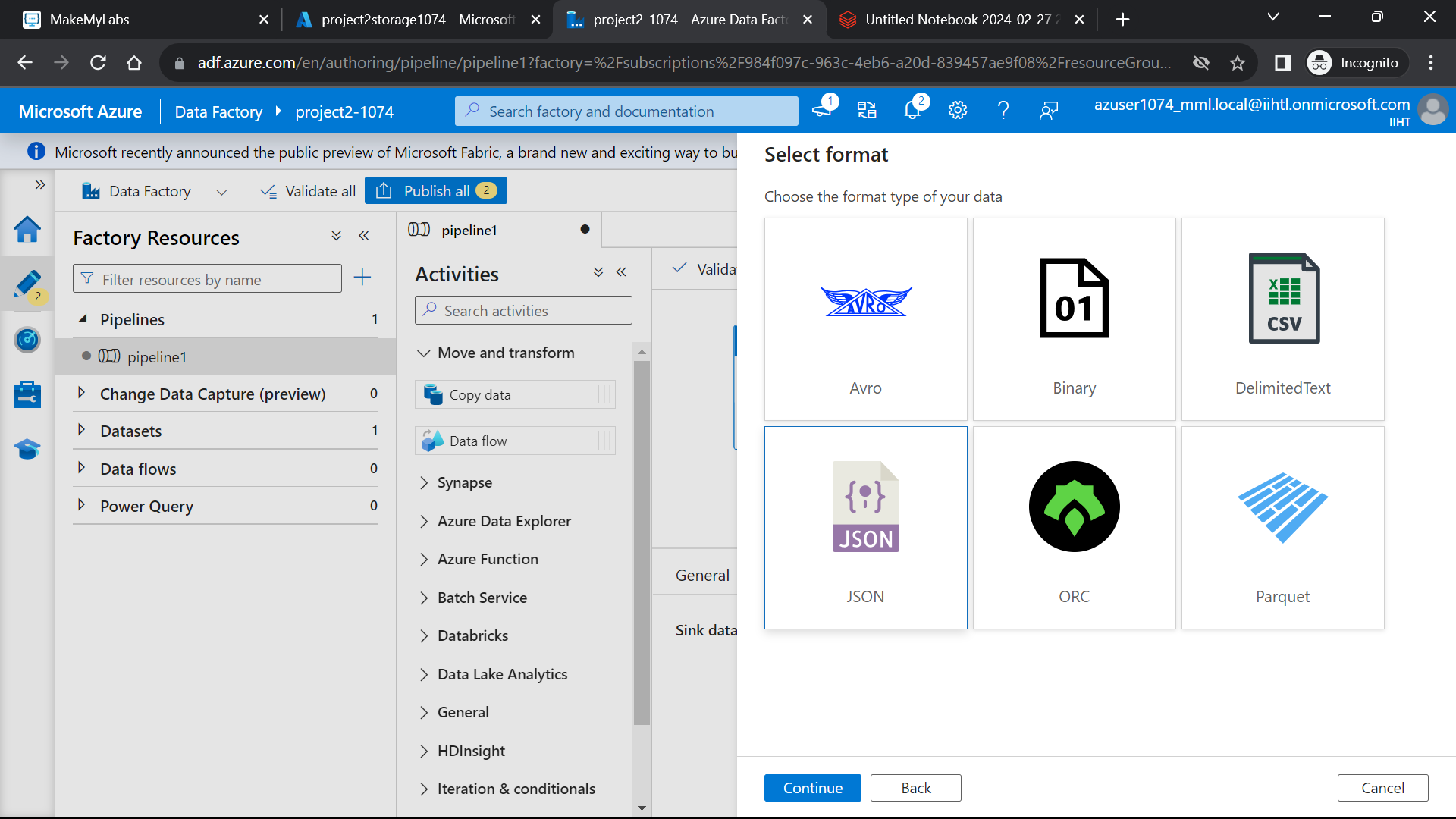
Now include the **copy data activity** to the pipeline.

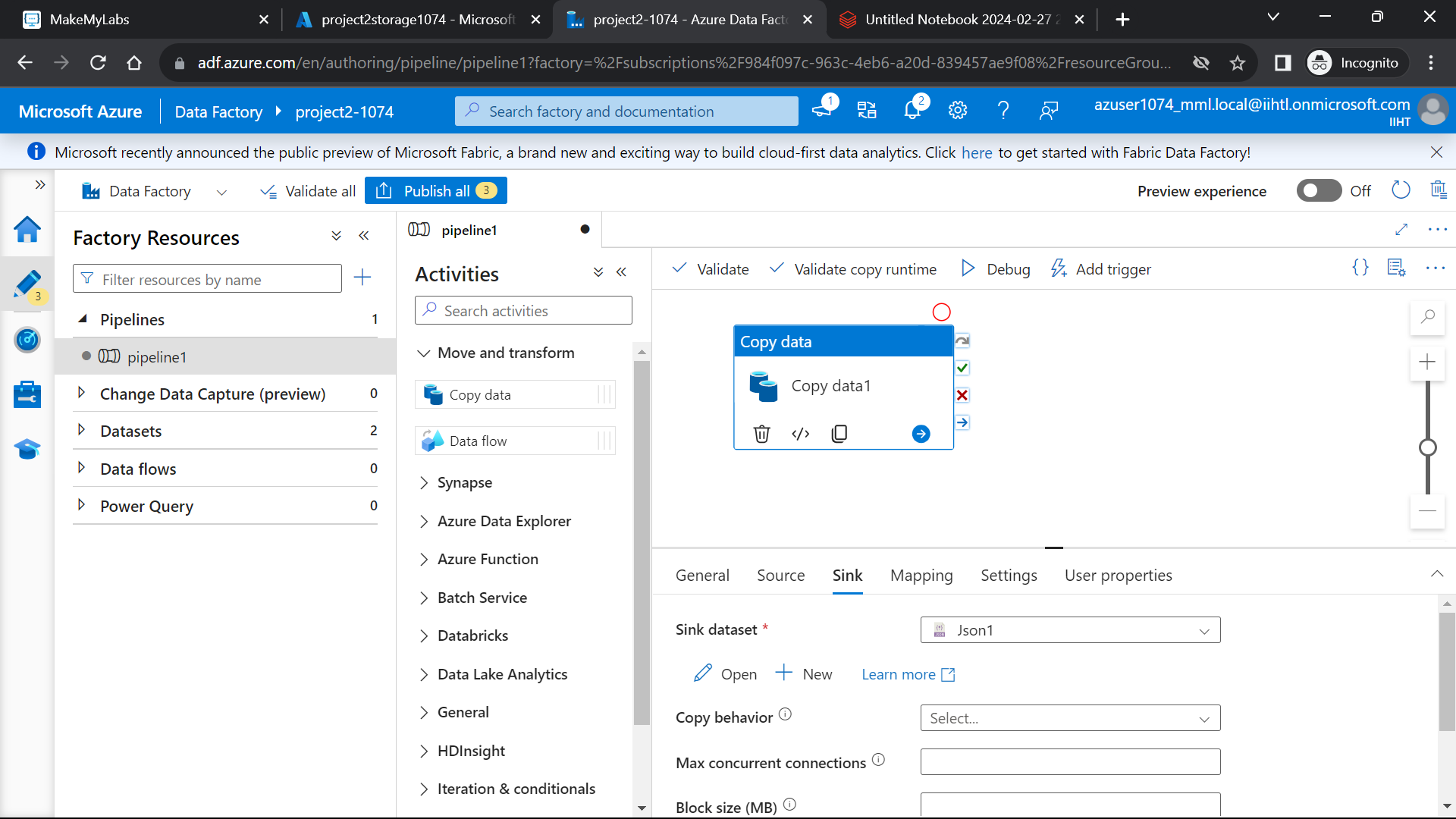




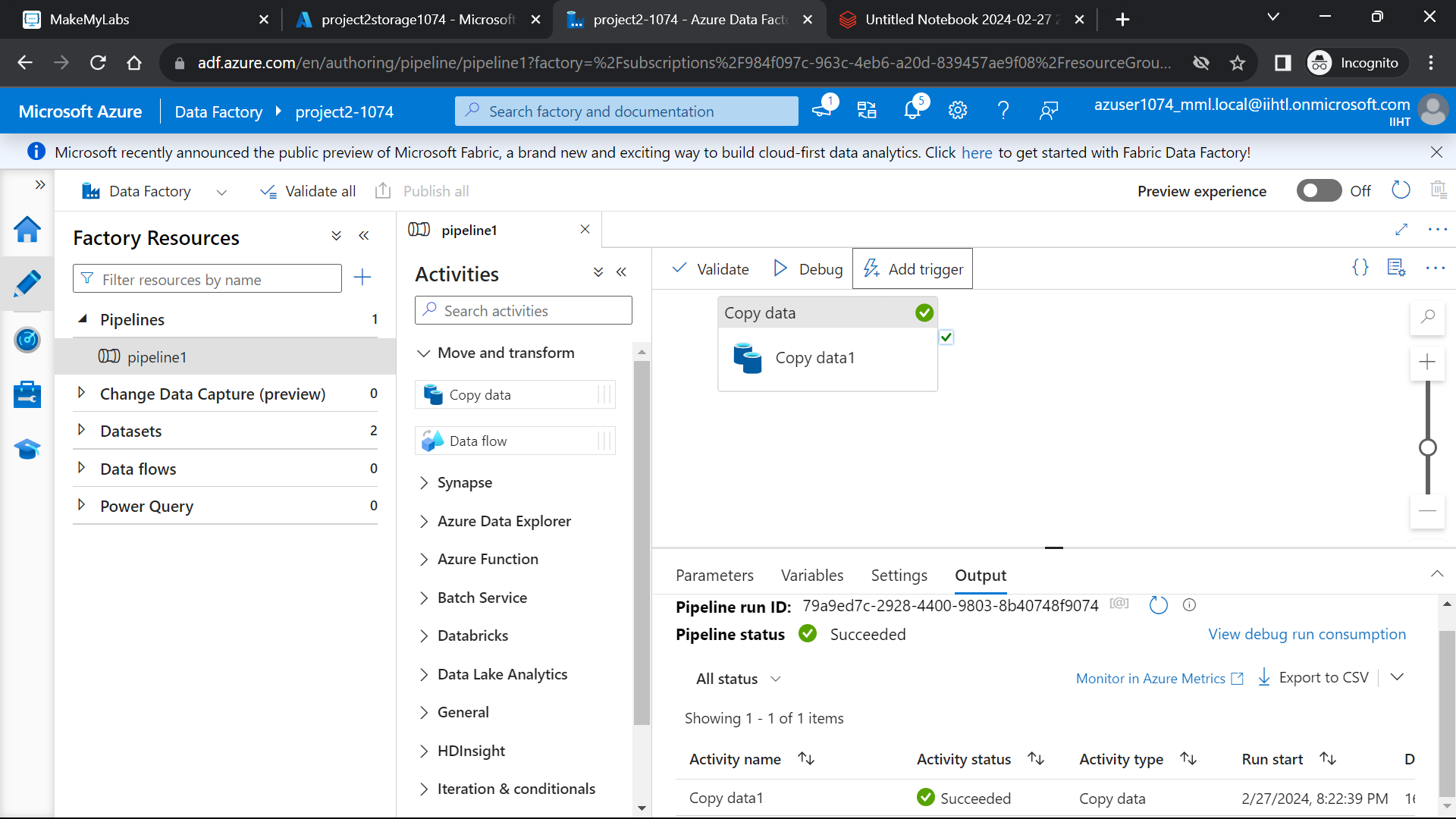
Now create a linked service by providing the azure subscription and the storage account name.







Now create a **new trigger** and **debug** the pipeline. The status of the pipeline can be monitored in the **output section.**



After the successful completion of pipeline, go to the destination storage path and verify the file is copied or not.



The **json file** is copied to the **destination storage container**.

We can view the json file by clicking and going to the edit and can also download the file to the local system by clicking on **“download”** option.

**OUTPUTS:**

