Madiha Aimon Tappal

Data Engineering

batch-1

Data-Engineering-Project-Using-Azure Databricks

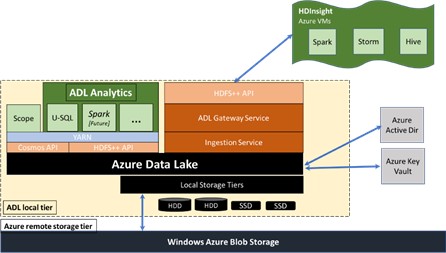
Integrate Azure Data Factory with Azure Data Lake Storage

# Project Overview:

**Objective:**

This project aims to establish a data pipeline using Azure Data Factory (ADF) to transfer data between various folders within an Azure Data Lake Storage (ADLS) account.

# Architecture diagram:



## Azure Data Factory (ADF)

* + **Data Factory Workspace:** A cloud-based environment within Azure to develop and manage data pipelines.
  + **Linked Services:** Connections established within ADF to access external data sources and sinks, including the source and destination ADLS folders in this case.
  + **Data Flows :** A visual interface within ADF to design data transformation logic, including filtering, cleansing, and enriching data (if required).
  + **Pipelines:** Workflows defined in ADF that orchestrate data movement between various sources and destinations. They consist of activities that specify the data operations to be performed.
  + **Monitoring & Management Tools:** Features within ADF to track pipeline execution status, identify errors, and manage pipeline schedules.

## Azure Data Lake Storage (ADLS)

* + **Data Lake Account:** A managed storage service in Azure for storing large datasets in a highly scalable and secure manner.
  + **Folders:** Organizational units within the ADLS account to categorize and manage data. The project will define a source folder containing the data to be moved and a target folder for the transferred data.

The data flow in this project involves transferring data between folders within an Azure Data Lake Storage (ADLS) account using Azure Data Factory (ADF). Here's a breakdown of the steps:

## Source:

* + ADF initiates the process by accessing the **source folder** within your ADLS account.
  + It identifies the data files based on specific criteria like file format, naming convention, or last modified date (configurable within the pipeline).

## Data Processing :

* + This step is **optional** and depends on your specific needs.
  + If data transformation is required, ADF utilizes its **data flow**

capabilities to perform the necessary operations.

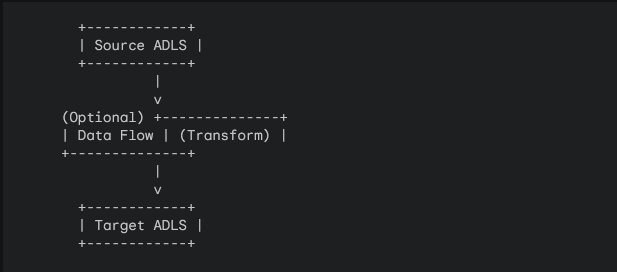
* + This data flow can involve:
    - **Filtering:** Selecting specific data based on defined criteria.
    - **Cleansing:** Removing inconsistencies or errors from the data.
    - **Enriching:** Adding additional data points from other sources.
  + The transformed data becomes the input for the next step.

## Destination:

* + The processed data (or original data if no transformation is applied) is then transferred to the designated **target folder** within ADLS.
  + ADF ensures the data is written to the target location in the desired format.

## Visualization:

Here's a simplified visualization of the data flow:



## Additional Points:

* + The data flow can be configured to handle various file formats commonly used in data lakes, such as CSV, Parquet, and Avro.
  + Error handling mechanisms can be implemented within the pipeline to address any issues during data transfer, such as file corruption or network errors.
  + The entire process can be scheduled to run at specific intervals using ADF's scheduling capabilities, ensuring data is transferred and potentially transformed regularly.

# How it works:

This project outlines three pipelines using Azure Data Factory (ADF) to move data between various locations:

## ADLS to ADLS (adls-adls):

This pipeline transfers data from one folder to another within the same Azure Data Lake Storage (ADLS) account.

## Components:

* + **Source Linked Service:** Connection to the source ADLS folder.
  + **Sink Linked Service:** Connection to the destination ADLS folder.
  + **Copy Activity:** The activity responsible for copying data between the folders.

## Steps:

* + **Define Linked Services:** Configure linked services for both the source and destination ADLS folders, specifying the account name, access key, and other relevant details.
  + **Create Pipeline:** Design a new pipeline in ADF.
  + **Add Copy Activity:** Drag and drop a "Copy Data" activity onto the pipeline canvas.

## Configure Copy Activity:

* + - **Source:** Select the source linked service and specify the source folder path.
    - **Destination:** Select the sink linked service and specify the destination folder path.
    - **File format:** Choose the appropriate file format based on your data (e.g., CSV, Parquet).
  + **Schedule and Run:** Set up a schedule (optional) or run the pipeline manually.

## ADLS to Azure Blob Storage (adls-blob):

This pipeline transfers data from an ADLS folder to an Azure Blob Storage container.

## Components:

* + **Source Linked Service:** Connection to the source ADLS folder.
  + **Sink Linked Service:** Connection to the destination Azure Blob Storage container.
  + **Copy Activity:** The activity responsible for copying data between ADLS and Blob storage.

## Steps:

* + **Define Linked Services:** Configure linked services for both the source ADLS folder and the destination Blob storage container.
  + **Create Pipeline:** Design a new pipeline in ADF.
  + **Add Copy Activity:** Drag and drop a "Copy Data" activity onto the pipeline canvas.

## Configure Copy Activity:

* + - **Source:** Select the source linked service and specify the source folder path.
    - **Destination:** Select the sink linked service and specify the destination Blob storage container and the desired blob name (optional).
    - **File format:** Choose the appropriate format based on your data.
  + **Schedule and Run:** Set up a schedule (optional) or run the pipeline manually.

## Azure Blob Storage to ADLS (blob-adls):

This pipeline transfers data from an Azure Blob Storage container to an ADLS folder.

## Components:

* + **Source Linked Service:** Connection to the source Azure Blob Storage container.
  + **Sink Linked Service:** Connection to the destination ADLS folder.
  + **Copy Activity:** The activity responsible for copying data between Blob storage and ADLS.

## Steps:

* + **Define Linked Services:** Configure linked services for both the source Blob storage container and the destination ADLS folder.
  + **Create Pipeline:** Design a new pipeline in ADF.
  + **Add Copy Activity:** Drag and drop a "Copy Data" activity onto the pipeline canvas.

## Configure Copy Activity:

* + - **Source:** Select the source linked service and specify the Blob storage container name and the blob name (optional).
    - **Destination:** Select the sink linked service and specify the destination ADLS folder path.
    - **File format:** Choose the appropriate format based on your data.
  + **Schedule and Run:** Set up a schedule (optional) or run the pipeline manually.

## Overall Working:

These pipelines utilize the "Copy Data" activity in ADF to transfer data between specified locations. You can schedule the pipelines to run periodically or trigger them based on specific events. By combining these individual pipelines, you can achieve complex data movement workflows within your data lake environment.

# Tools Used:

1. **Azure Data Factory (ADF):** Cloud-based service for orchestrating data movement and transformation.
2. **Azure Data Lake Storage (ADLS):** Managed storage service for storing large datasets in a scalable and secure manner.

# Execution Overview:

This project aims to establish three data pipelines using Azure Data Factory (ADF) to automate data movement between various locations in your Azure cloud environment:

1. **ADLS to ADLS (adls-adls):** Transfers data between folders within the same Azure Data Lake Storage (ADLS) account.
2. **ADLS to Blob Storage (adls-blob):** Transfers data from an ADLS folder to an Azure Blob Storage container.
3. **Blob Storage to ADLS (blob-adls):** Transfers data from an Azure Blob Storage container to an ADLS folder.

# Execution Flow:

* 1. **Define Linked Services:** Establish connections between ADF and your data sources and sinks. This involves creating separate linked services for each source and destination in each pipeline (ADLS accounts, Blob storage containers).
  2. **Develop Pipelines:** Design individual pipelines in ADF for each data transfer scenario (adls-adls, adls-blob, blob-adls).
  3. **Configure Copy Activities:** Within each pipeline, utilize the "Copy Data" activity to specify the source and destination locations, file format, and other relevant settings.
  4. **Schedule and Run Pipelines:** Choose the execution model for each pipeline (e.g., manual, scheduled). Schedule the pipelines to run periodically or trigger them manually based on your needs.
  5. **Monitor Execution:** Track the execution status and history of each pipeline using the ADF monitoring tools. This allows you to identify any errors or issues and ensure successful data movement.

# Benefits:

* **Automation:** Eliminates manual data transfer tasks, ensuring consistency and reliability.
* **Scalability:** Handles large datasets efficiently, adapting to your data processing needs.
* **Scheduling:** Runs pipelines at specific intervals or based on events, keeping data up-to-date.
* **Monitoring:** Provides centralized oversight of data movement processes for troubleshooting and management.

# Azure Resources Used for this Project:

This project utilizes several Azure resources to facilitate data transfer between different storage locations:

## Azure Data Factory (ADF):

* + **ADF Workspace:** A cloud-based environment within Azure to develop, manage, and orchestrate data pipelines.

This serves as the central hub for designing and executing the data movement workflows.

## Azure Data Lake Storage (ADLS):

* + **ADLS Account:** A managed storage service in Azure for storing large datasets in a highly scalable and secure manner.
  + This project utilizes ADLS accounts in two ways:
    - **Source:** Stores the data to be transferred in one or more folders within the ADLS account.
    - **Destination:** Holds the transferred data in a designated folder within the ADLS account.

## Azure Blob Storage (Optional, used in adls-blob and blob-adls pipelines):

* + **Blob Storage Container:** A container within an Azure Blob Storage account that serves as a repository for object data.
  + This project utilizes Blob storage containers in two ways:
    - **Source:** Stores the data to be transferred in the adls-blob pipeline.
    - **Destination:** Holds the transferred data in the blob-adls pipeline.

## Additional Resources (Optional):

* + **Azure Key Vault:** A managed service that securely stores and manages cryptographic keys and secrets.

This can be used to store access keys for ADLS and Blob storage accounts, improving security by avoiding hardcoding them in the pipelines

# Project Requirements:

This project outlines the requirements for developing three data pipelines using Azure Data Factory (ADF) to automate data movement between various storage locations:

## Functional Requirements:

* + **Data Transfer:** The pipelines must successfully transfer data between the specified source and destination locations for each scenario (ADLS-to-ADLS, ADLS-to-Blob, Blob-to-ADLS).
  + **File Format Handling:** The pipelines must be able to handle the chosen file formats for the data being transferred (e.g., CSV, Parquet).
  + **Scheduling:** The pipelines should be configurable to run at specific intervals or be triggered manually based on your needs.
  + **Monitoring:** The project should enable monitoring of pipeline execution status and history to track progress and identify any errors.

## Non-Functional Requirements:

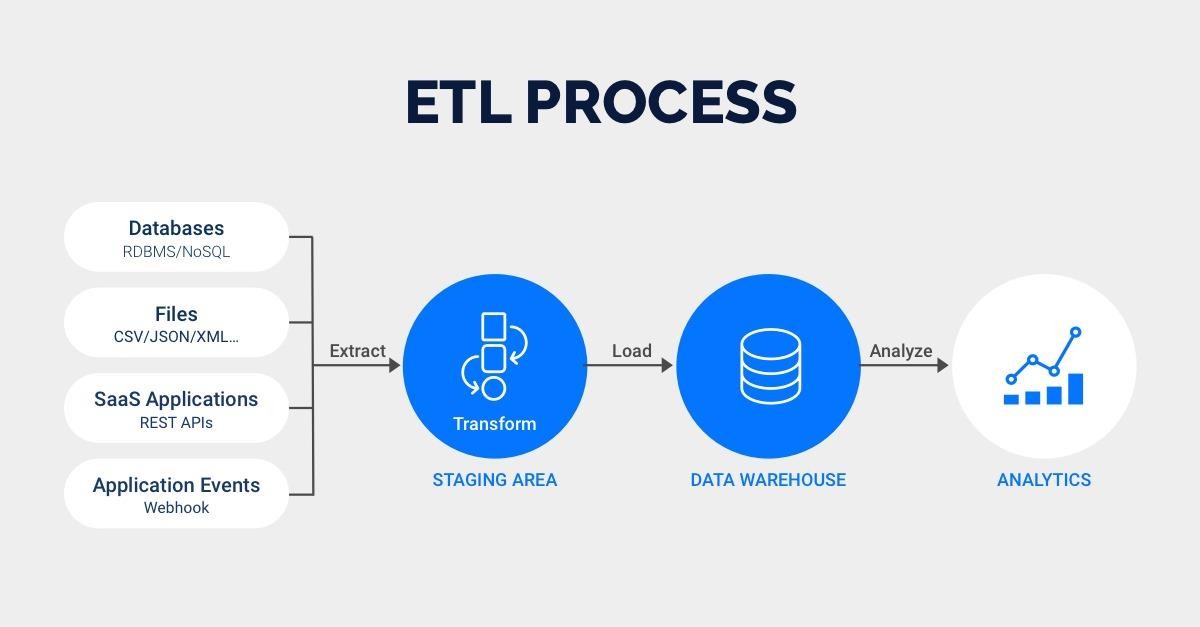
* + **Scalability:** The pipelines should be able to handle large datasets efficiently and scale to meet future data processing needs.
  + **Security:** The project should implement security best practices, such as using Azure Key Vault to store access keys securely and avoiding hardcoding them in the pipelines.
  + **Performance:** The pipelines should be designed and optimized for efficient data transfer, minimizing processing time and resource utilization.
  + **Logging:** The pipelines should log relevant information during execution, such as success/failure status, data volume transferred, and any encountered errors.
  + **Documentation:** The project should be well-documented, including details on pipeline configurations, data formats, and troubleshooting steps.

## Additional Requirements:

* + Familiarity with Azure Data Factory and its capabilities.
  + Access to the necessary Azure resources, including an ADF workspace, ADLS accounts (potentially), and Blob storage accounts (optional).
  + Understanding of the data formats involved in the transfer process.

# ETL pipeline

An ETL pipeline, which stands for Extract, Transform, Load, is a series of processes used to move data from one or more sources to a target system, typically a data warehouse or data lake. It's essentially a mechanism for preparing raw data for analysis and further use.



## Here's a breakdown of the three stages involved:

1. **Extract:** In this stage, data is retrieved from its various sources. These sources can be diverse, ranging from databases and applications to flat files and social media feeds.
2. **Transform:** Once extracted, the data is cleaned, filtered, and manipulated to fit the specific needs of the target system. This might involve correcting errors, formatting inconsistencies, combining data sets, or deriving new information.
3. **Load:** Finally, the transformed data is loaded into the designated target system, where it's readily available for analysis and

reporting. This could be a data warehouse, data lake, or any other system designed to store and manage large amounts of data.

## Benefits of using ETL pipelines:

1. **Improved data quality:** By cleaning and transforming data, ETL pipelines ensure consistency, accuracy, and completeness, leading to more reliable insights.
2. **Centralized data management:** ETL pipelines consolidate data from various sources into a single location, simplifying data access and analysis.
3. **Increased efficiency:** Automating data extraction and transformation saves time and reduces manual effort compared to handling data manipulation tasks individually.
4. **Enhanced decision-making**: By providing clean and readily available data, ETL pipelines empower data-driven decision making within organizations.

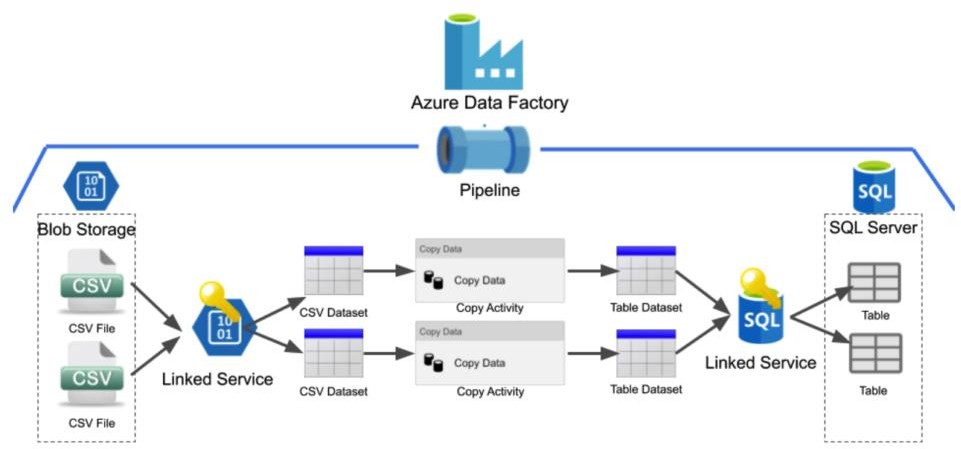
## Use cases for ETL pipelines:

1. **Data migration:** Moving data from legacy systems to new platforms.
2. **Data warehousing**: Preparing data for analysis and reporting.
3. **Data cleansing:** Addressing inconsistencies and errors in data.
4. **Data integration:** Combining data from multiple sources for holistic views.

The project described utilizes an EL (Extract, Load) process rather than a full-fledged ETL pipeline. While data transformation isn't explicitly mentioned, it might be implicitly handled during the copy operation or could be integrated using additional functionalities within the Azure Data Factory (ADF) pipeline.

# Azure Data factory pipeline

Data Factory Pipeline to implement data copy between folders in Azure Data Lake Storage (ADLS). While it doesn't encompass the full ETL (Extract, Transform, Load) process, it leverages a key component of Data Factory Pipelines:



## The Copy Activity:

This activity acts as the engine for moving data between various data stores.

## In this specific scenario, the Copy Activity is configured to:

1. Extract data from a source folder within an ADLS account (using a linked service).
2. Load the data into a designated destination folder within the same or potentially another ADLS account (using another linked service).
3. While data transformation isn't explicitly included in this instance, the Copy Activity might handle basic transformations like file format conversion during the copy process.

## Key characteristics of Data Factory Pipelines:

1. **Visual interface:** ADF provides a user-friendly interface for building and orchestrating data pipelines.
2. **Drag-and-drop functionality**: You can easily create pipelines by dragging and dropping various activities, including the Copy Activity.
3. **Scheduling:** Pipelines can be scheduled to run at specific intervals or triggered by events.
4. **Monitoring and logging:** ADF offers comprehensive monitoring and logging capabilities to track pipeline execution and troubleshoot any issues.

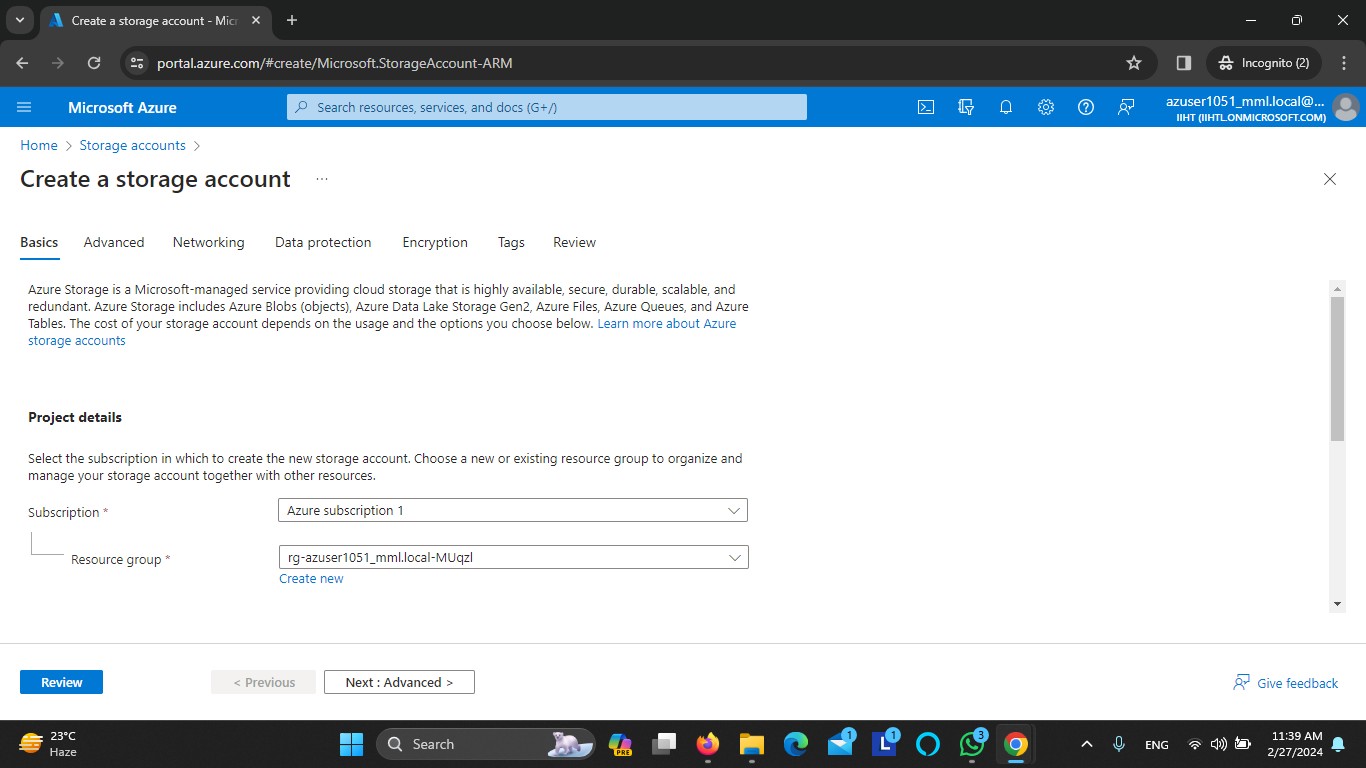
Therefore, while the overall process represents an EL (Extract, Load) approach, it's crucial to understand that a Data Factory Pipeline with the Copy Activity is the fundamental mechanism enabling the data movement between folders within ADLS.

# Tasks performed:

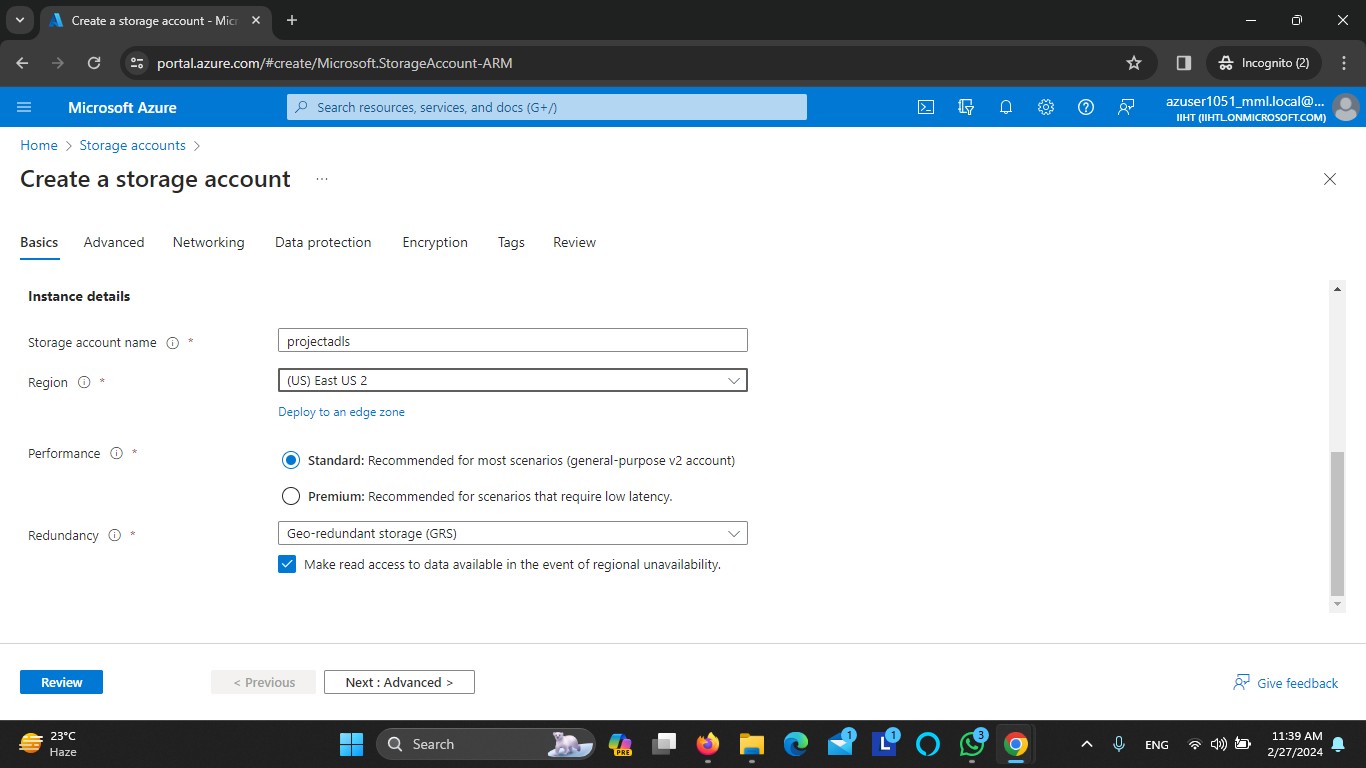
## . Create a Pipeline to copy the data task from

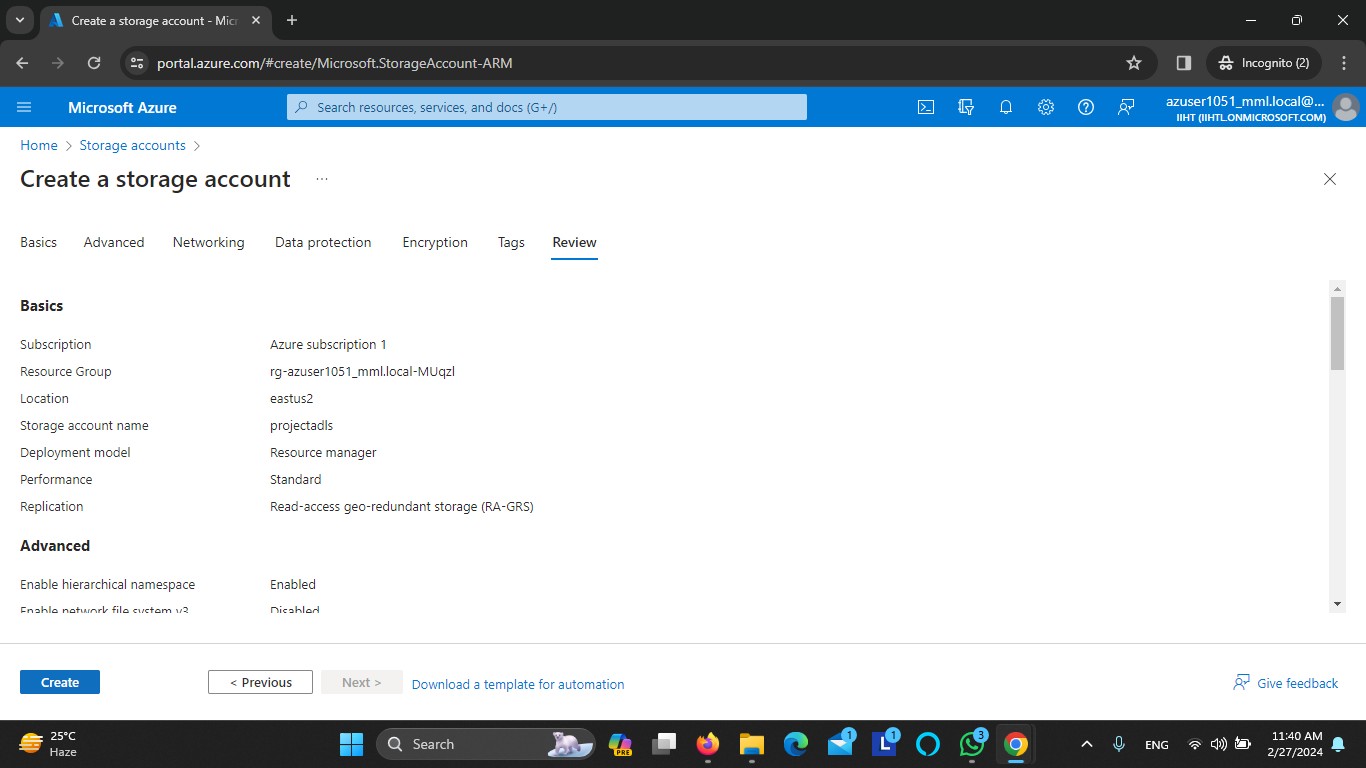
**Azure Data Lake Storage Gen2 to Azure Data Lake Storage Gen2 Steps:**

Login in to Azure Portal and create a Storage Account Storage Account Name - projectadls

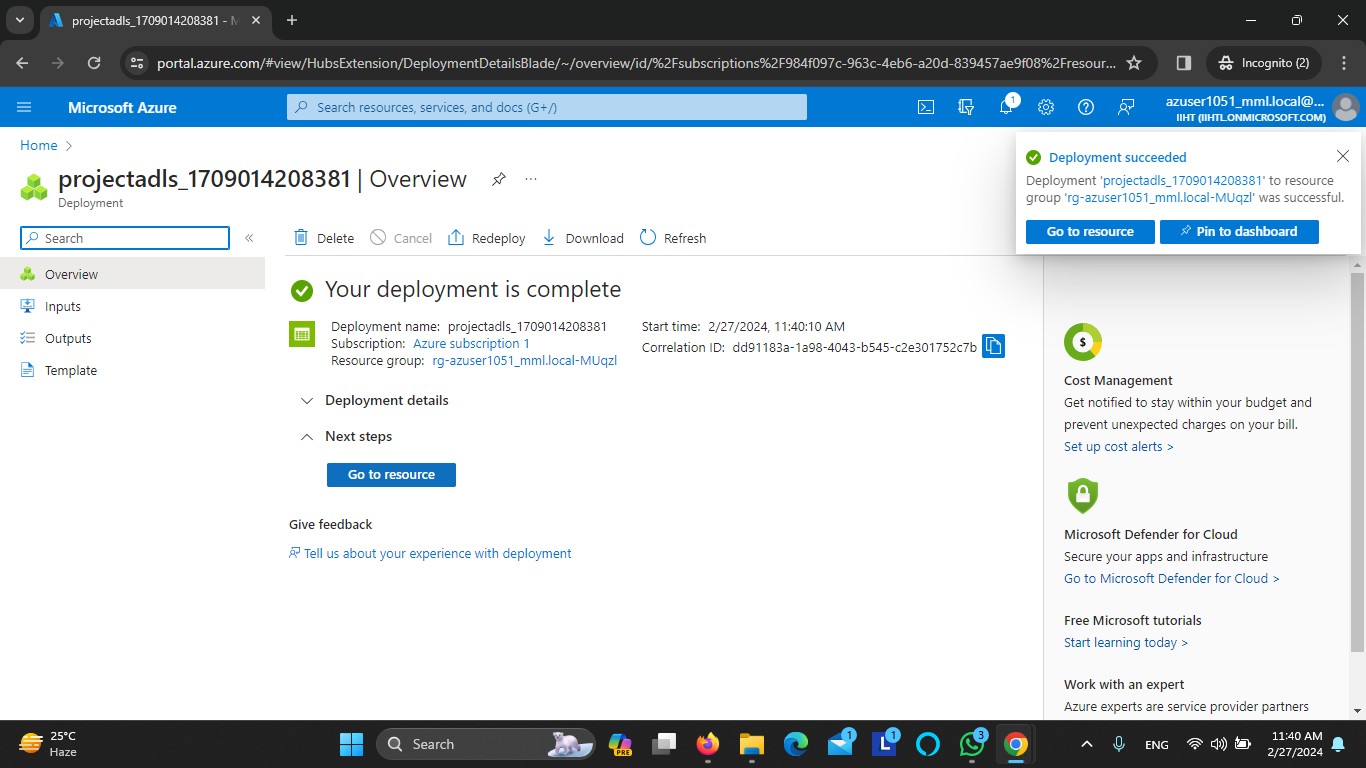


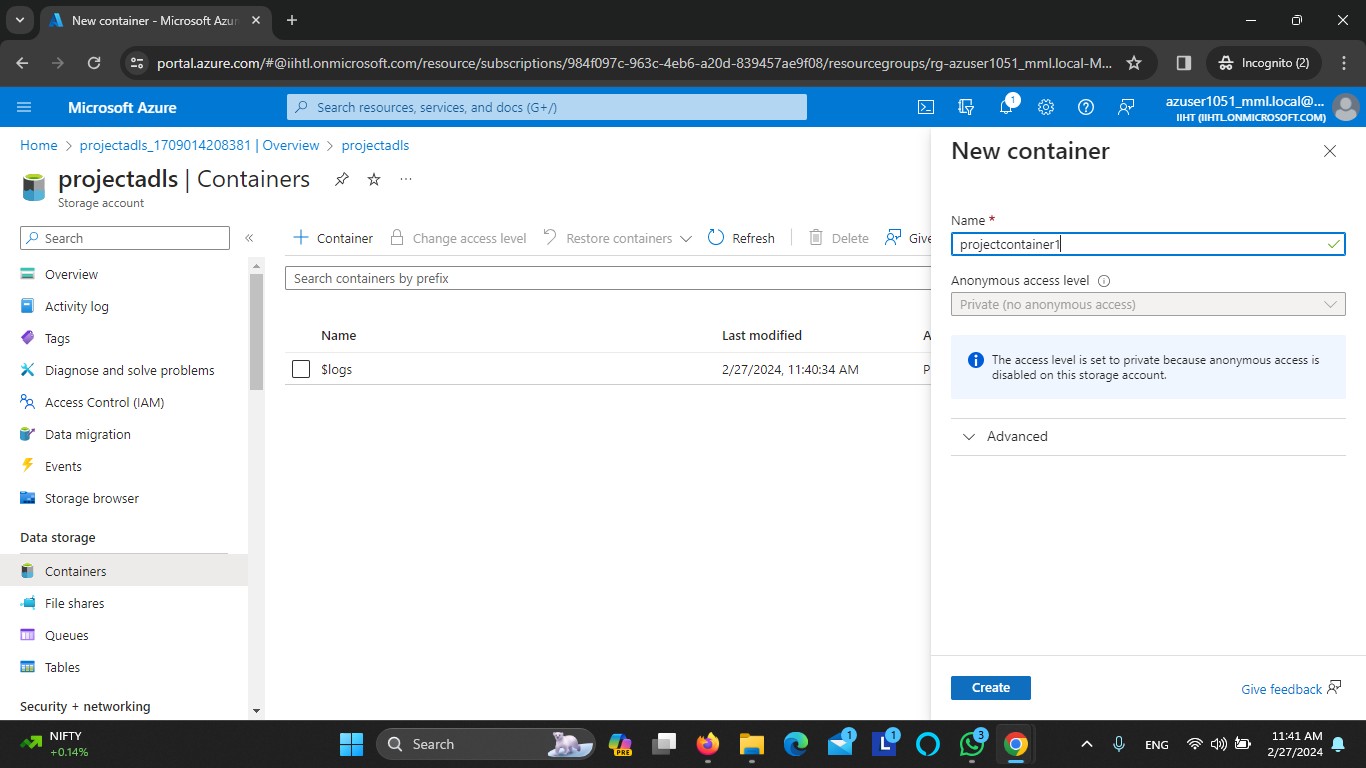
Region - East US



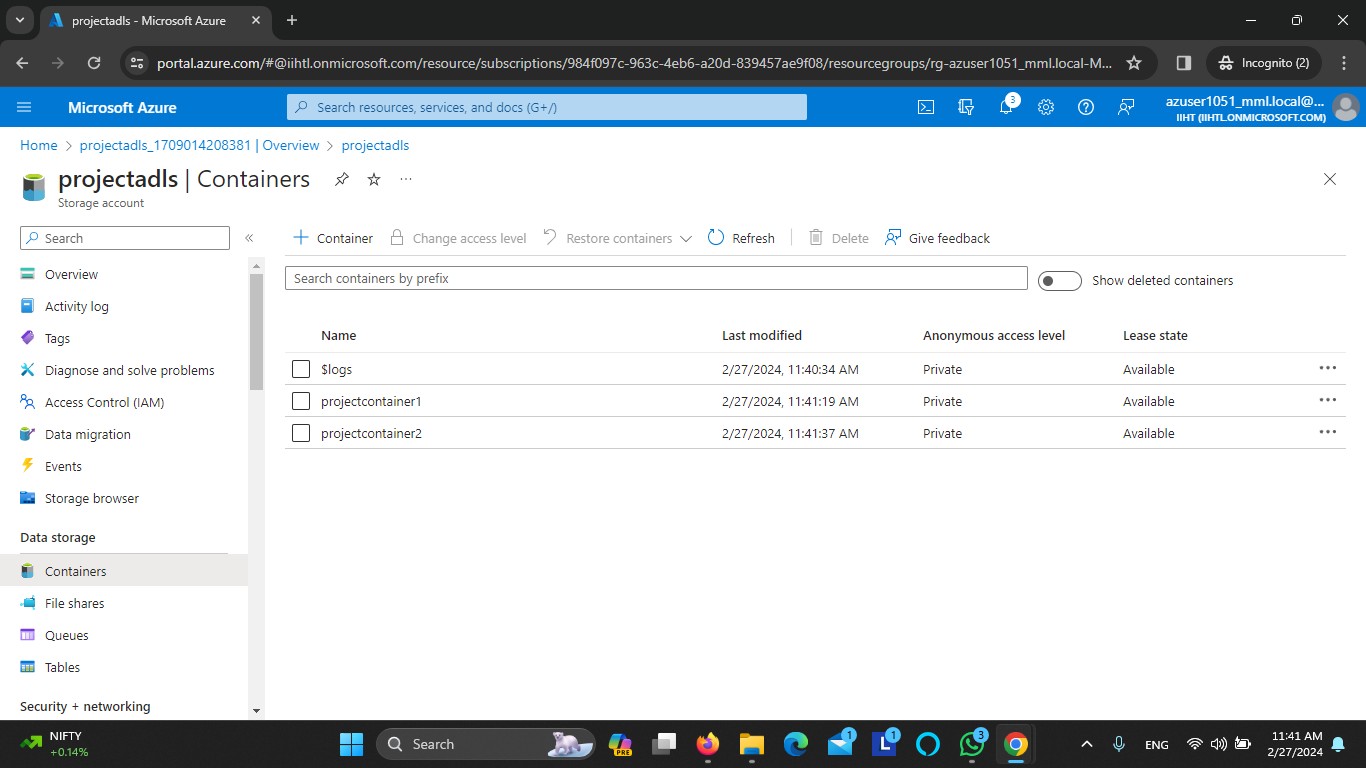


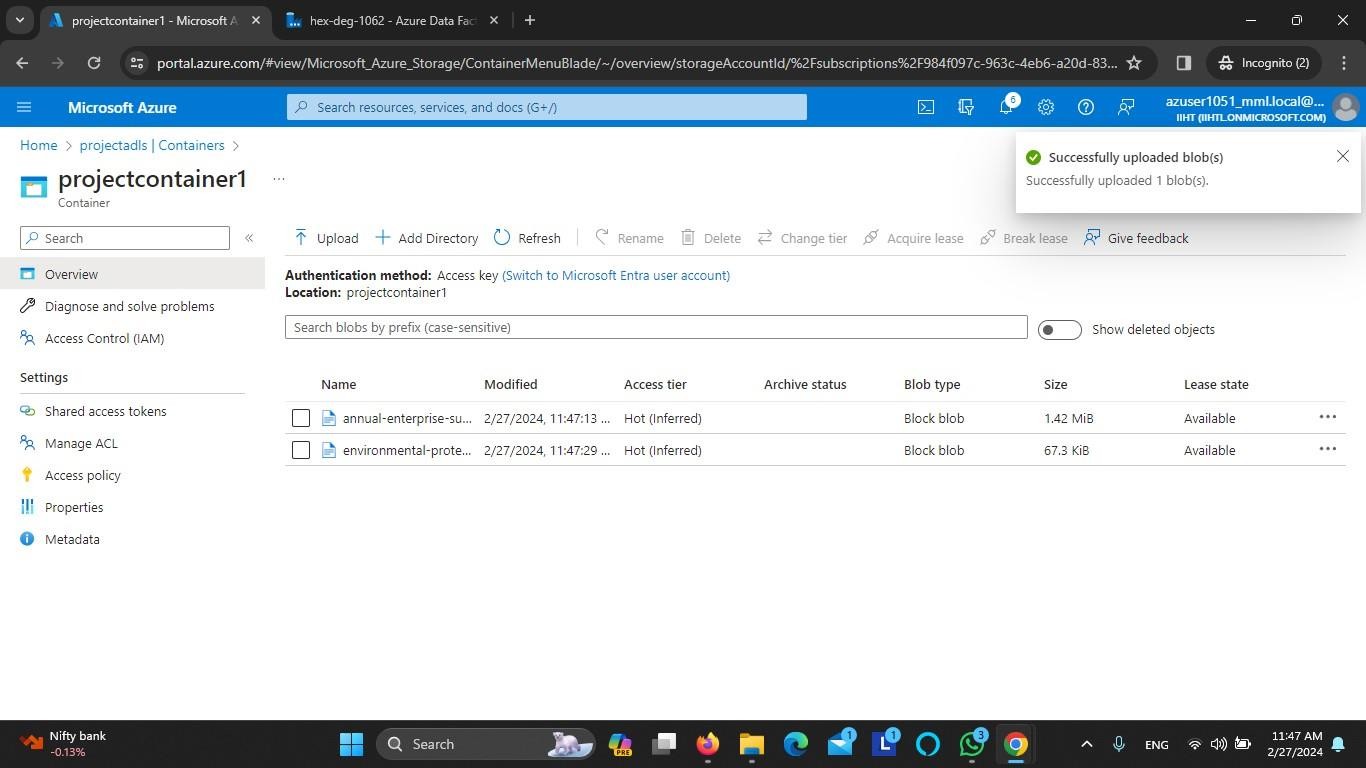
Storage Account Created





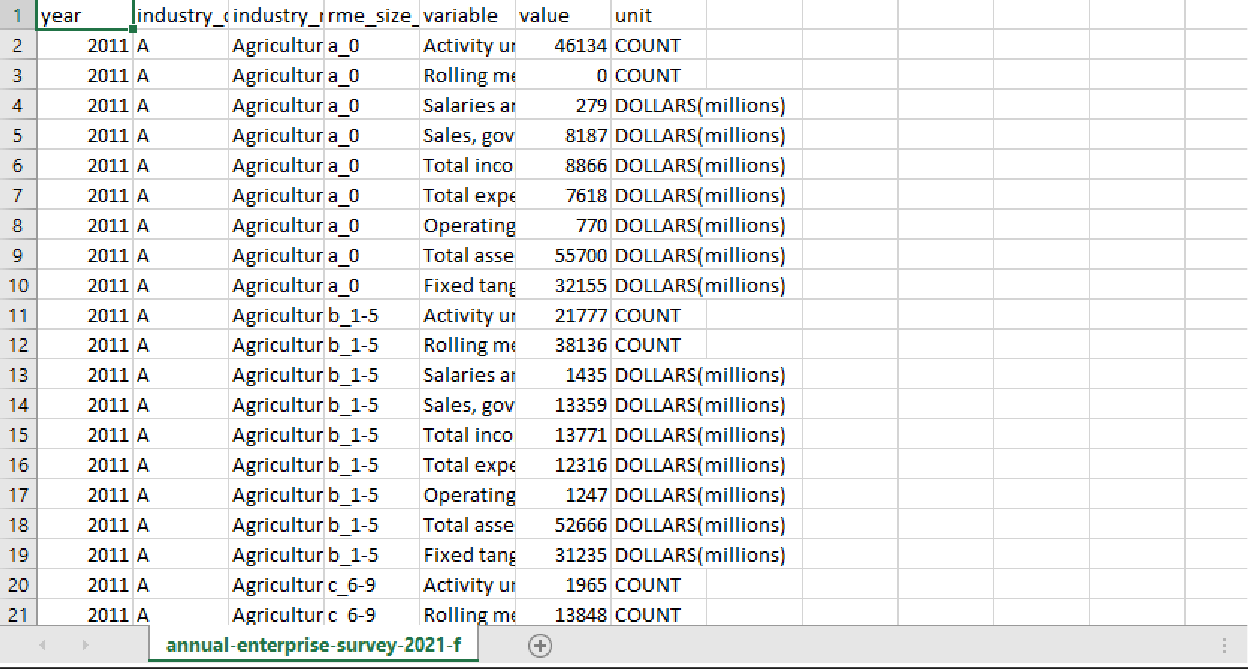
Create container-projectcontainer2

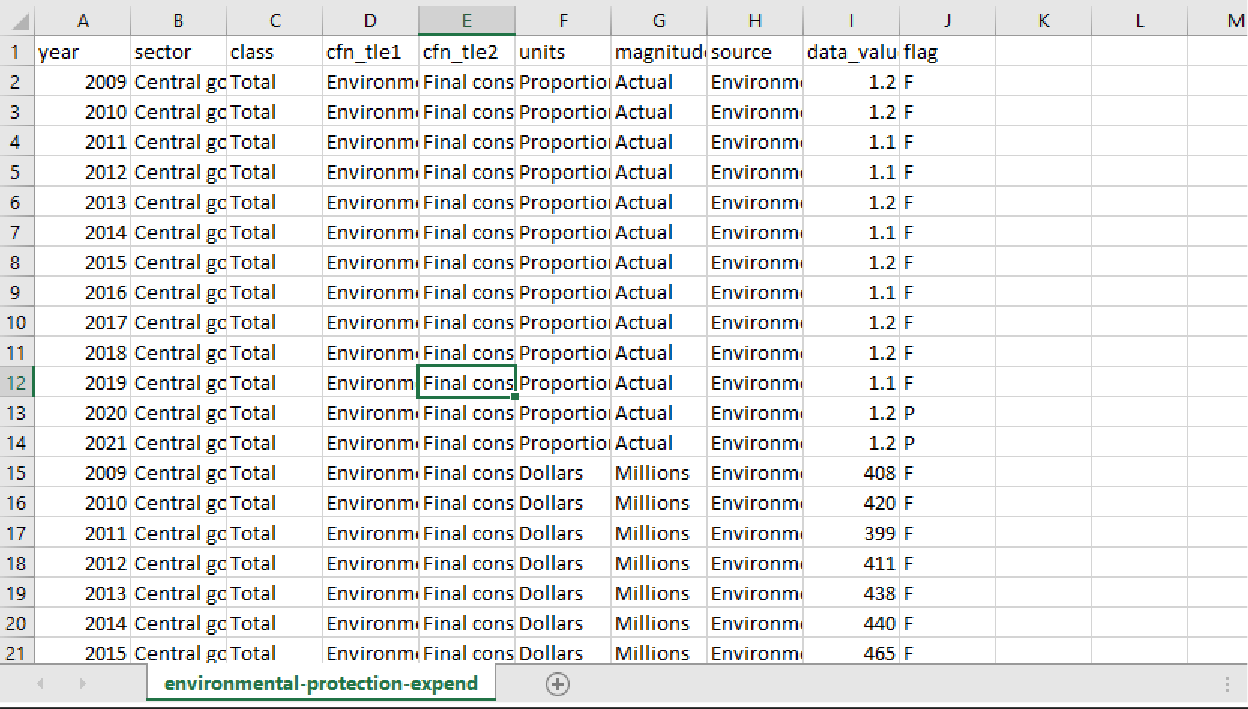




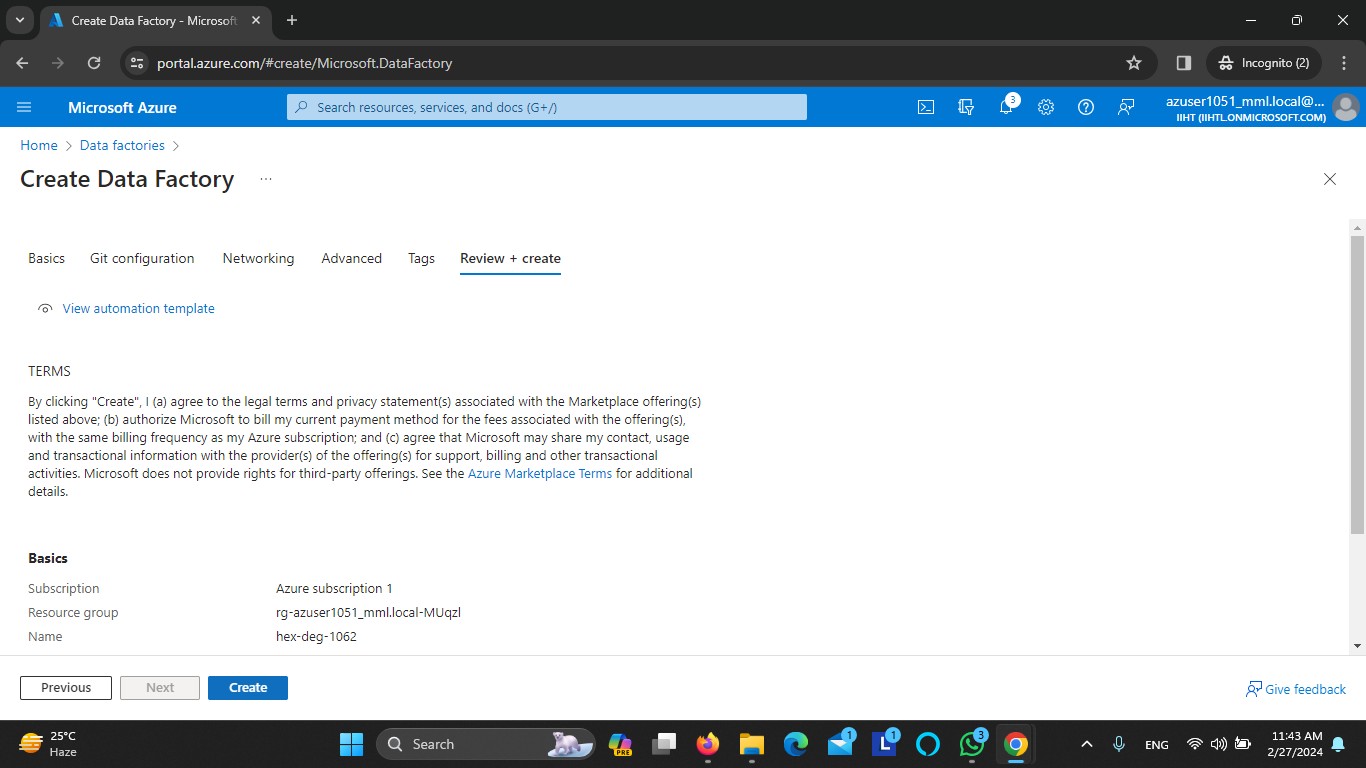
Sample data

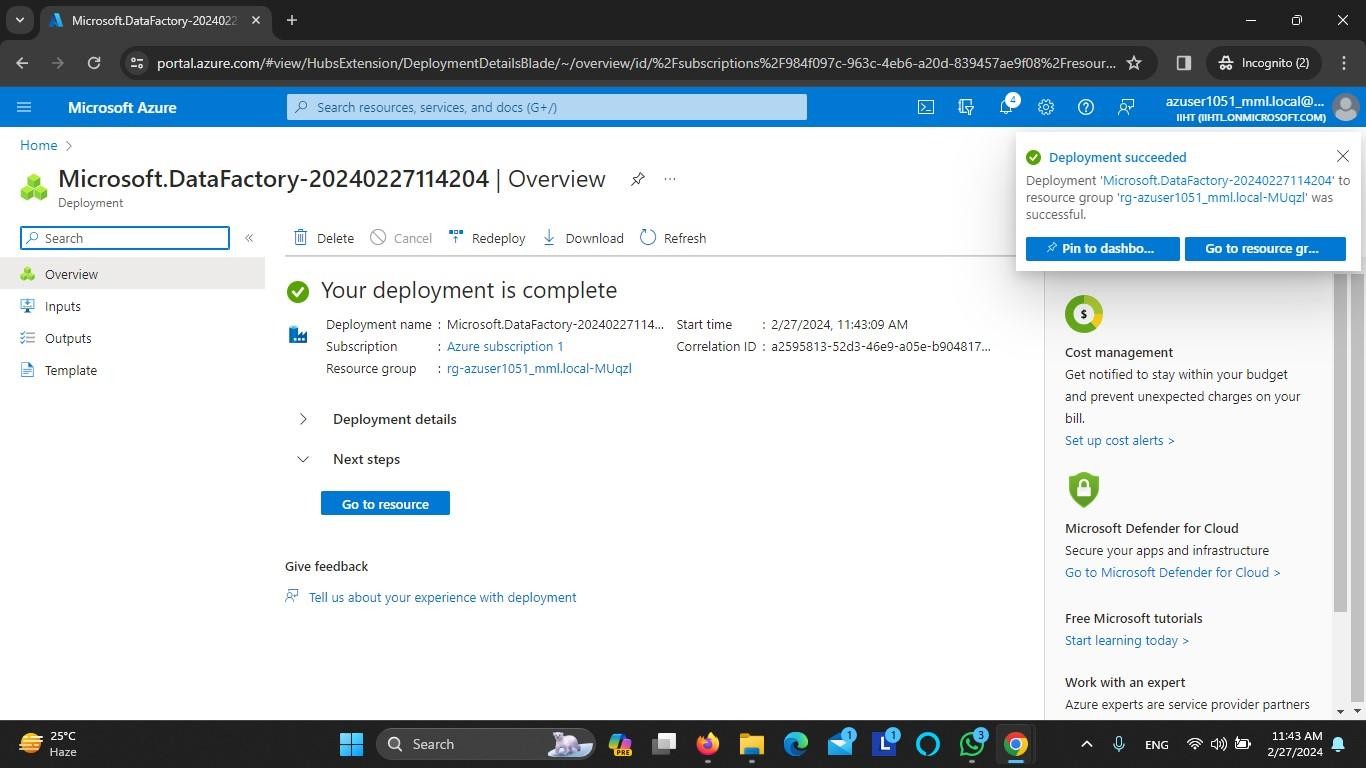
Annual-Enterprise-Survey-2021

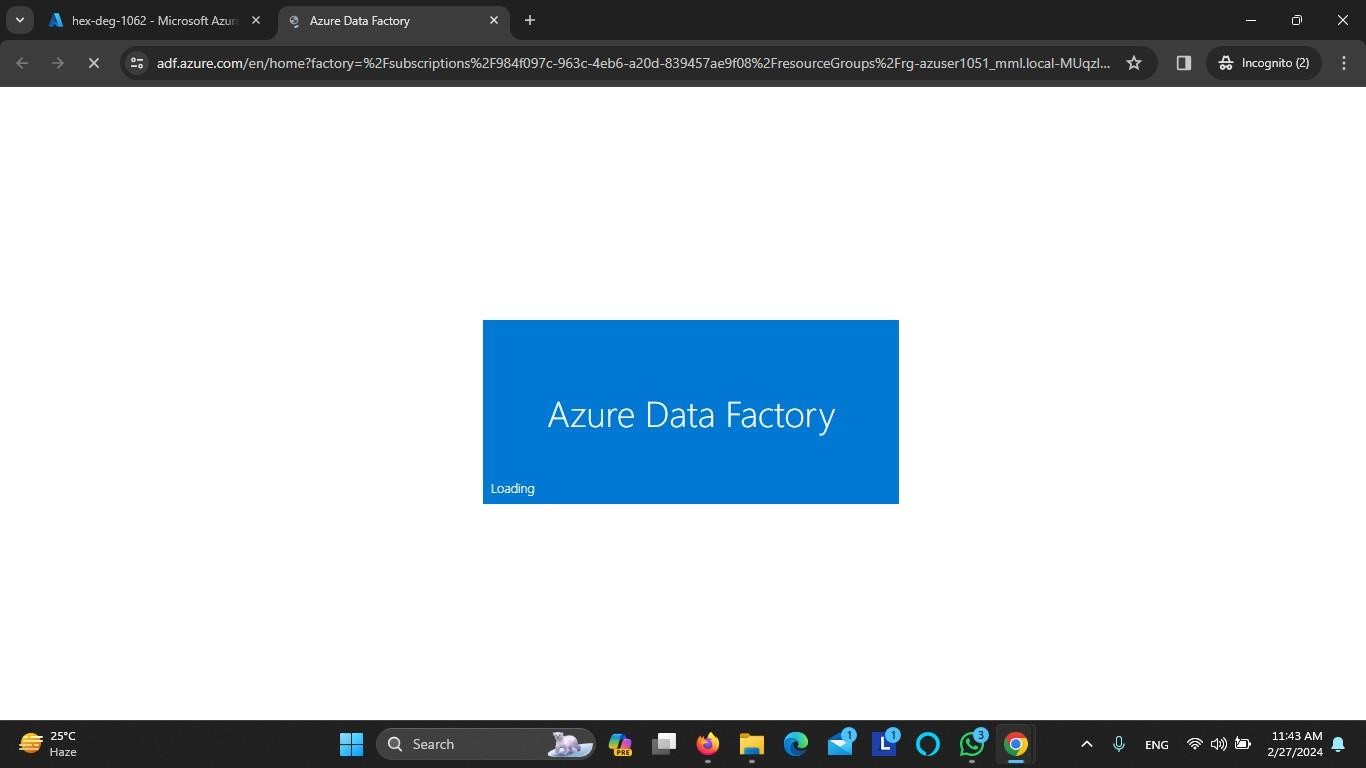




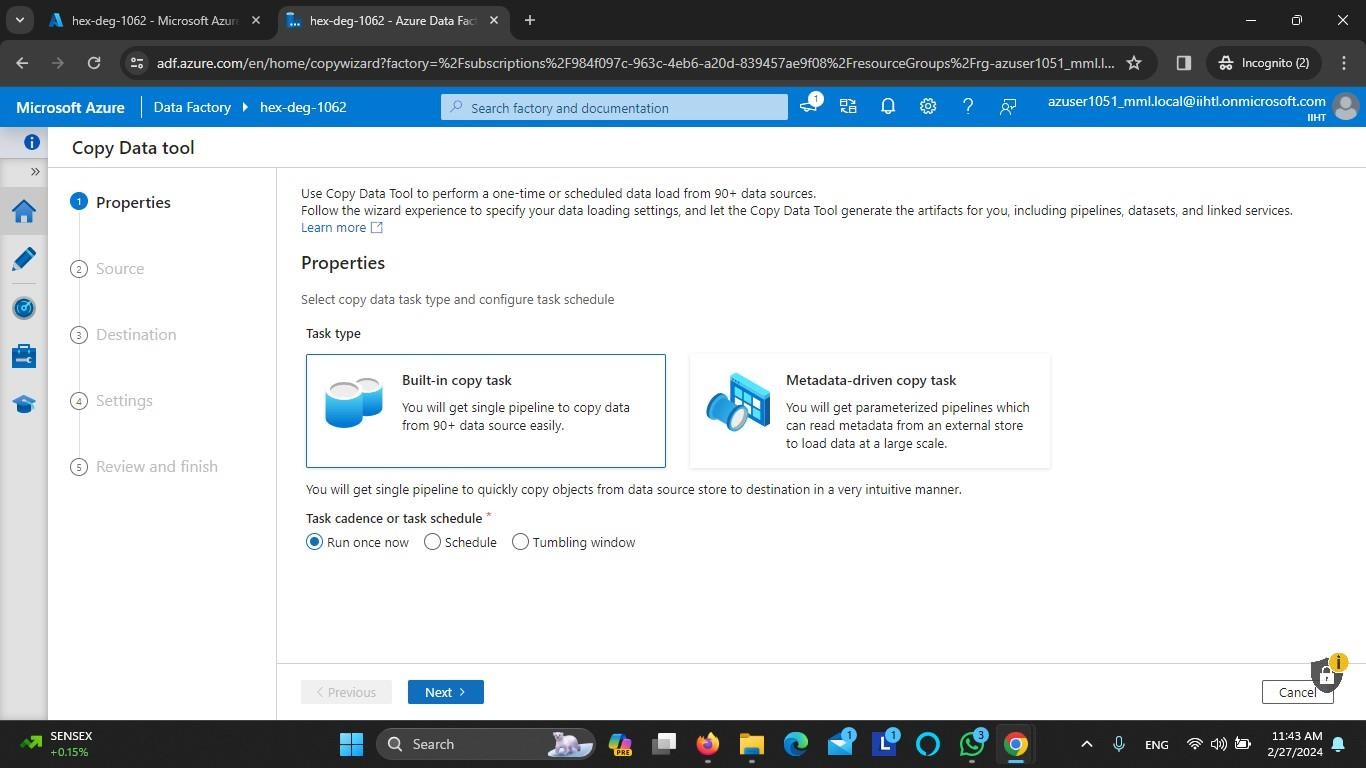
Create Azure data factory



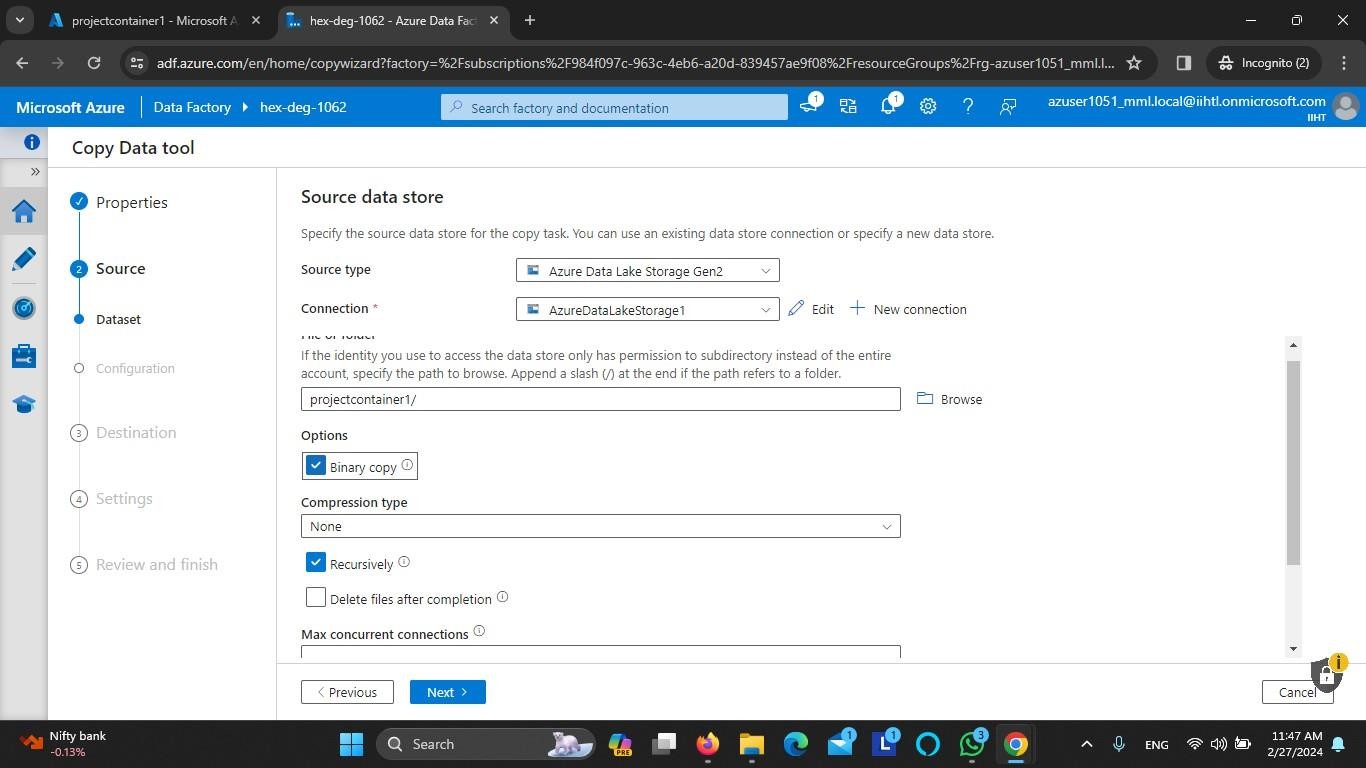


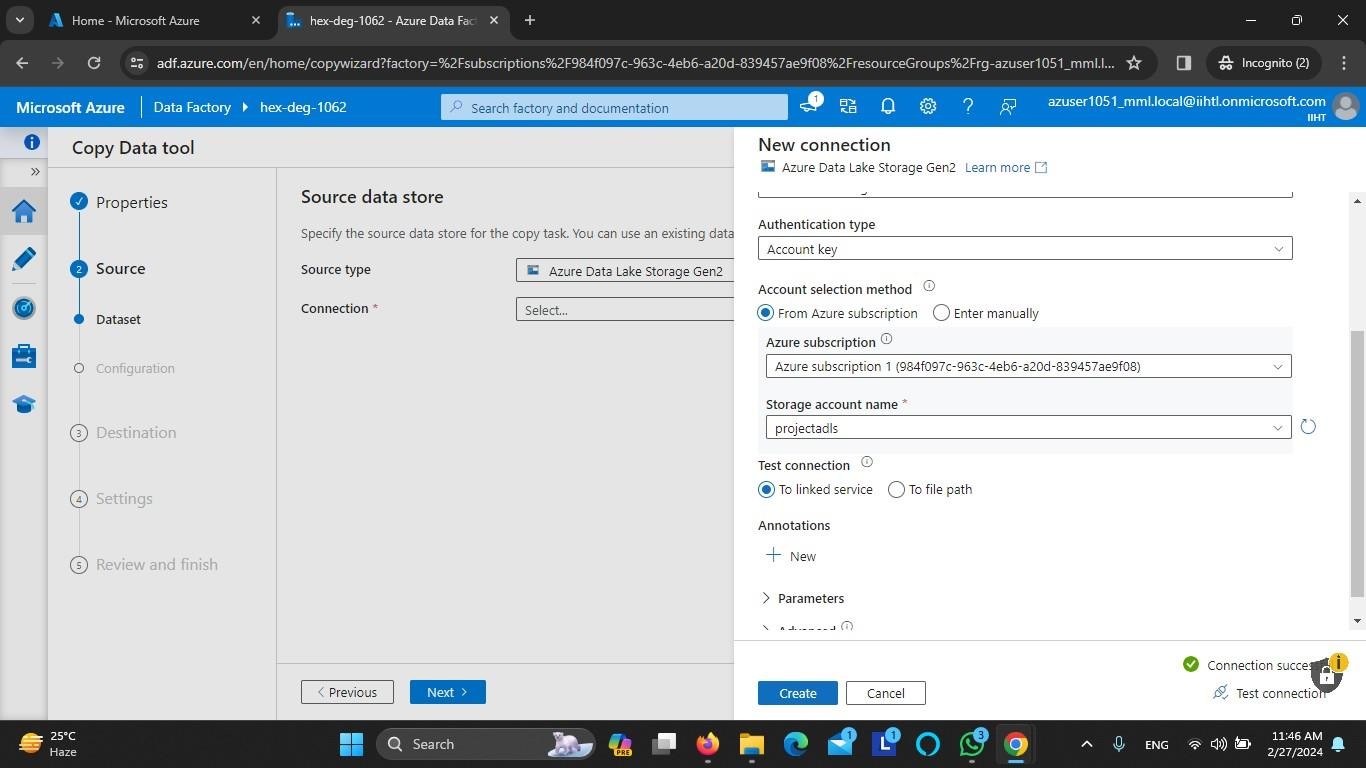


Built in copy task

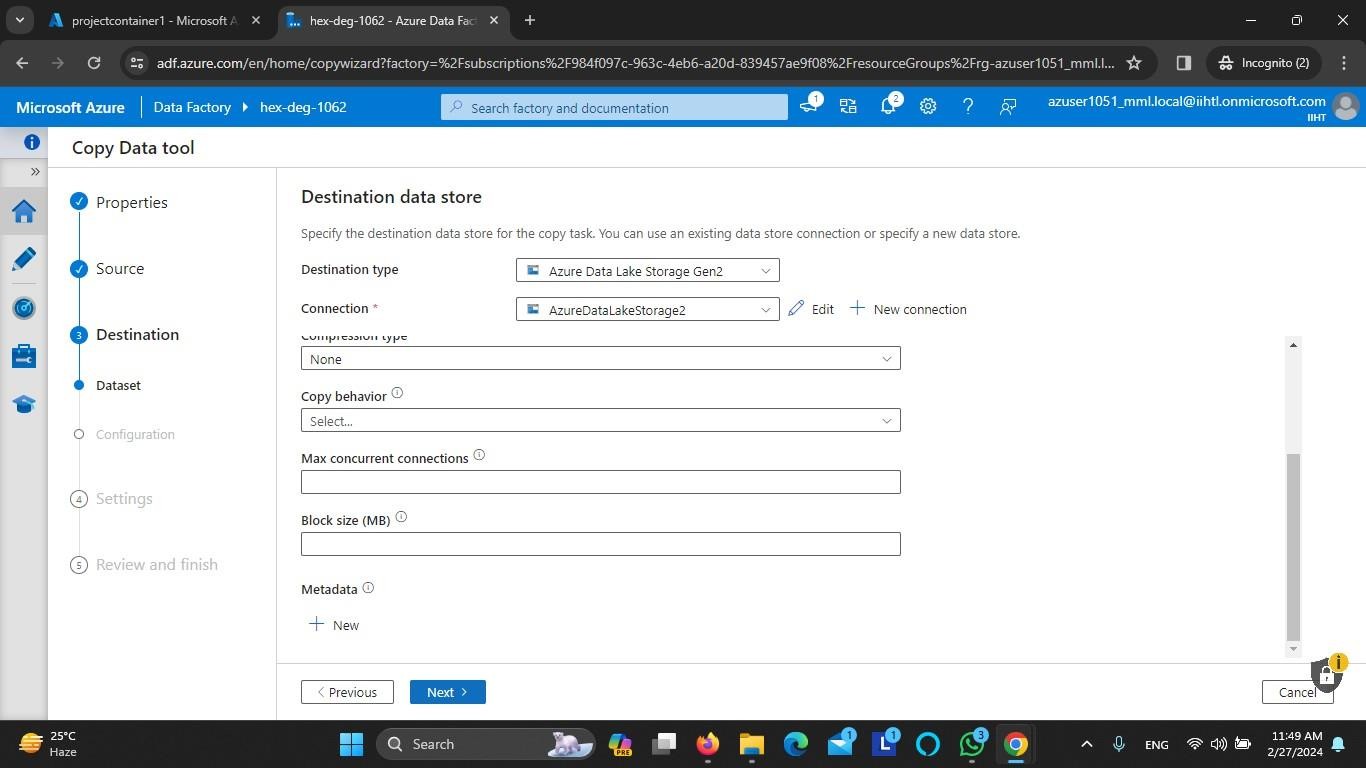


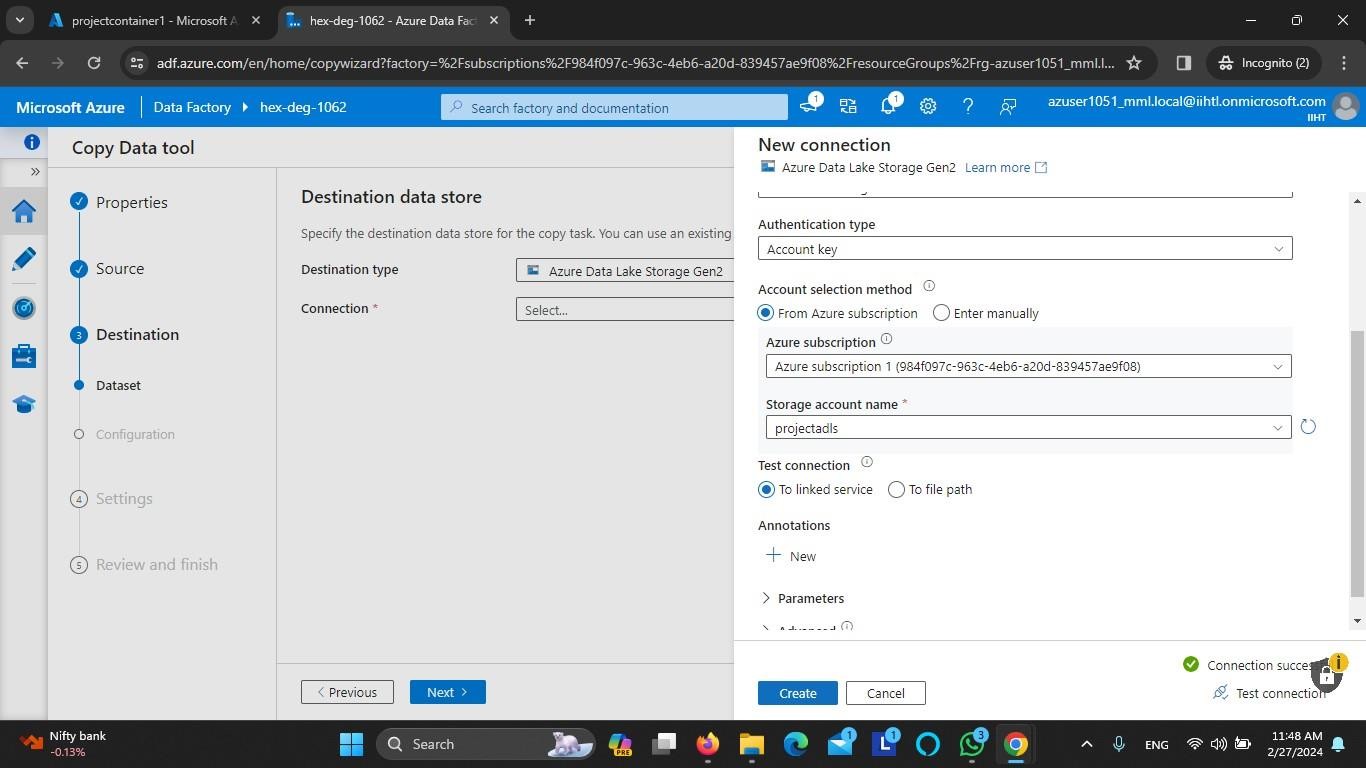
Specify the source data store for copy task Azure Data Lake Storage Gen2



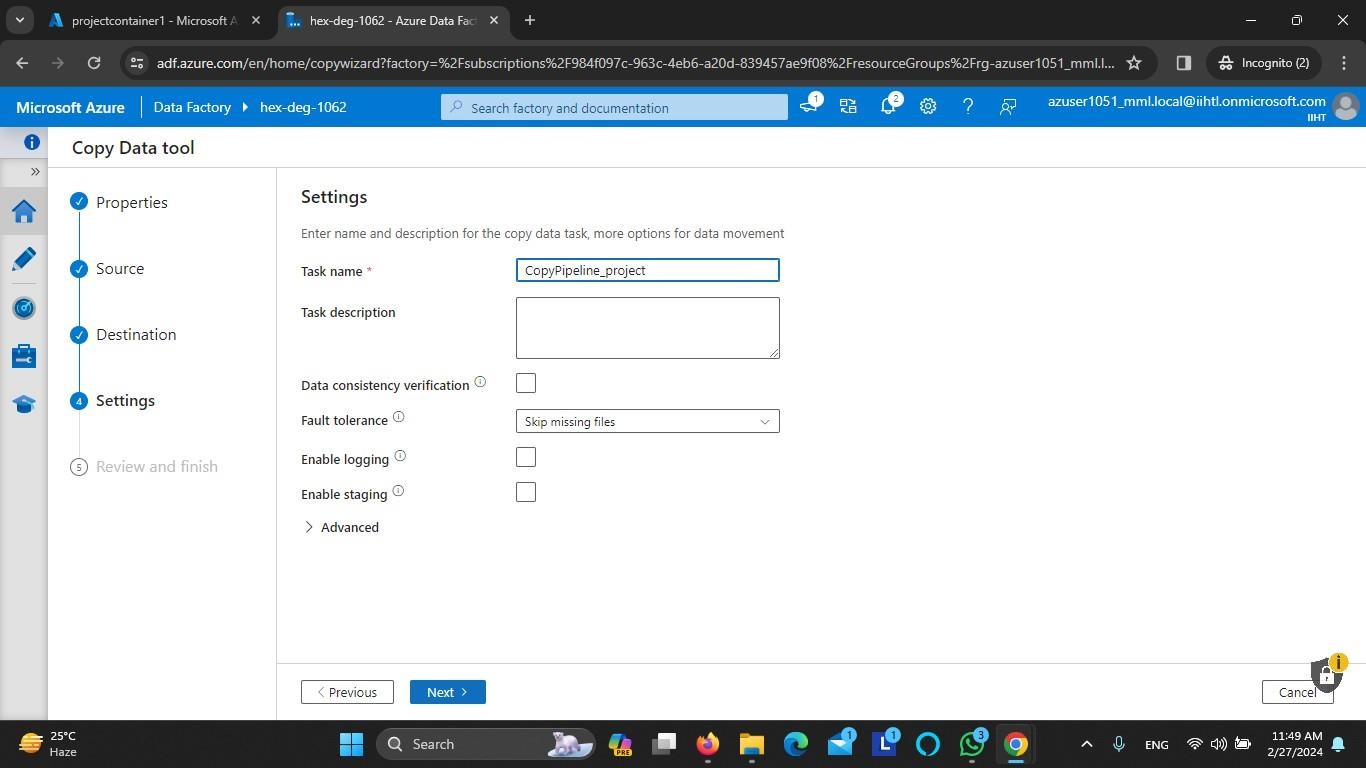


Specify the destination data store for copy task Azure Data Lake Storage Gen2

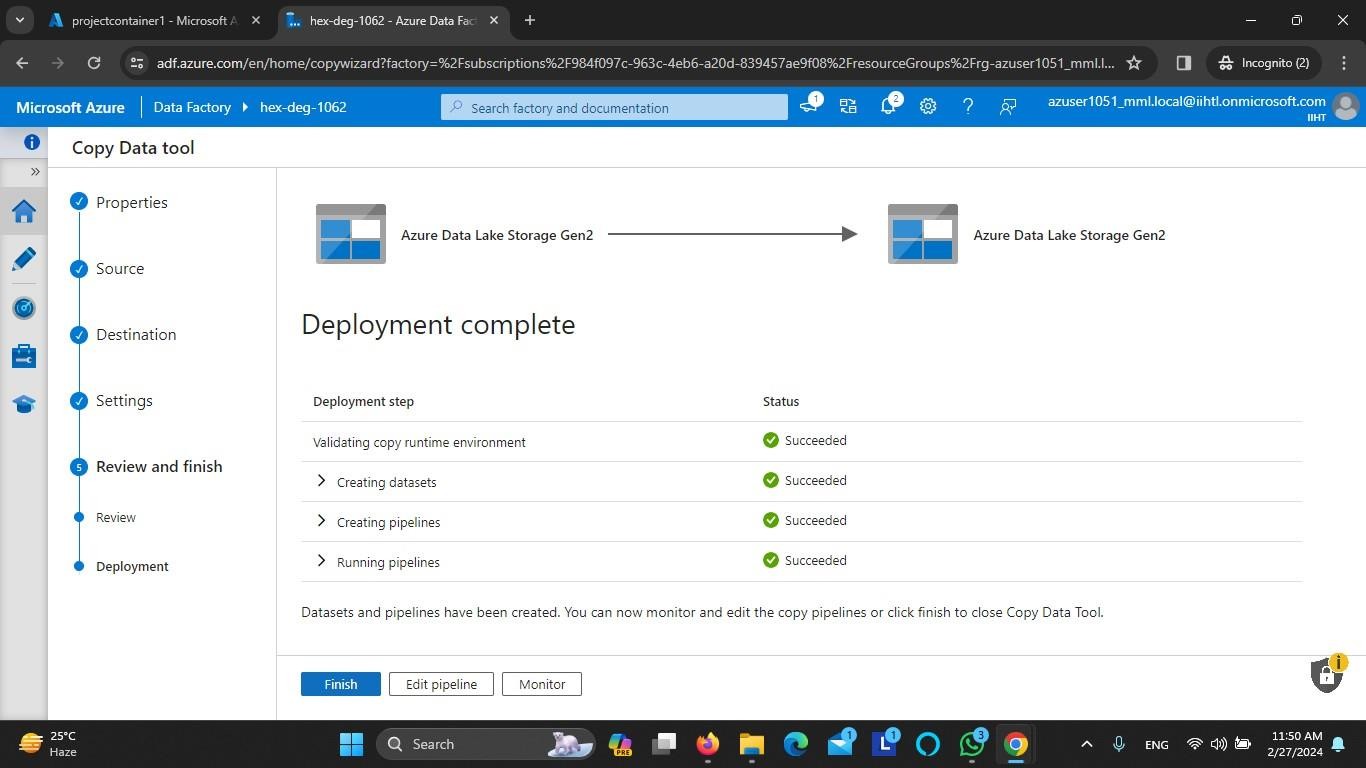




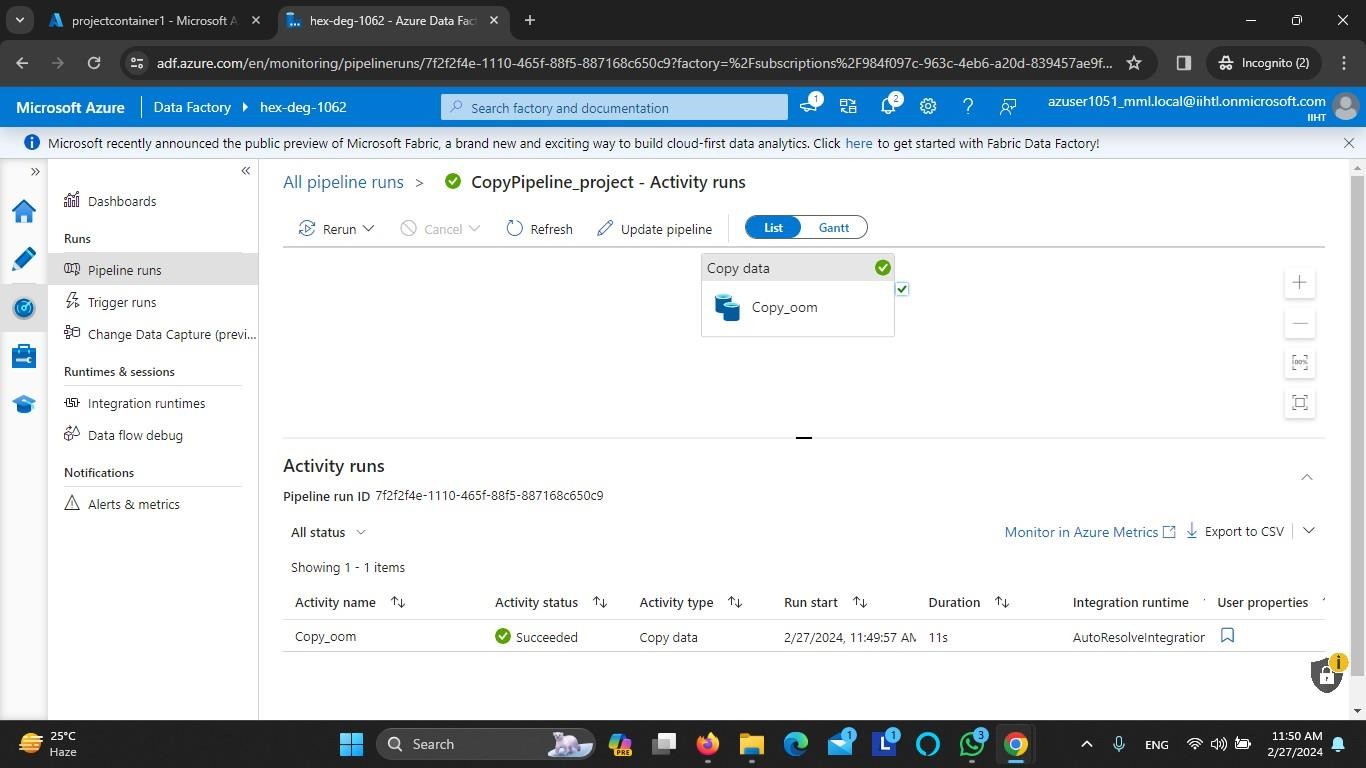
Enter copy data task name CopyPipeline\_project

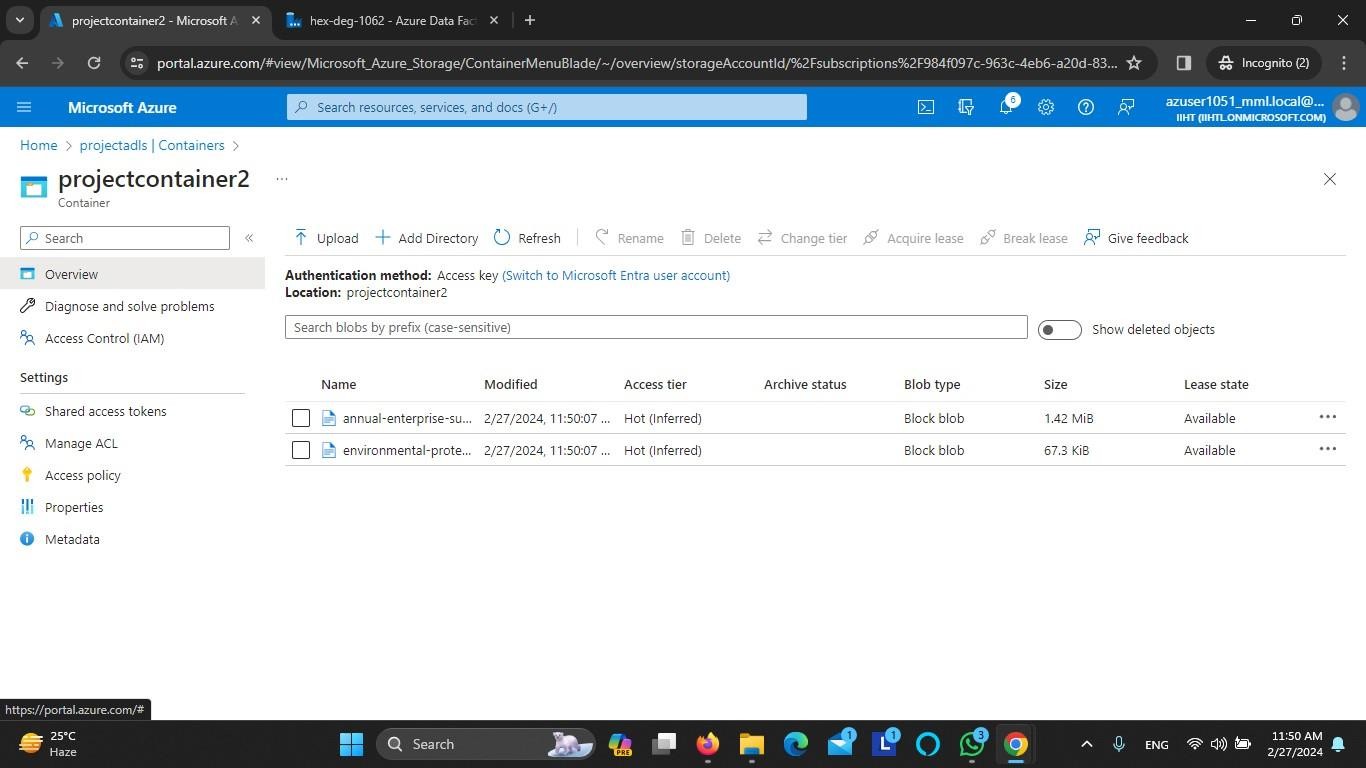


Azure Data Lake Storage Gen2 to Azure Data Lake Storage Gen2



Activity succeeded and pipeline ingested

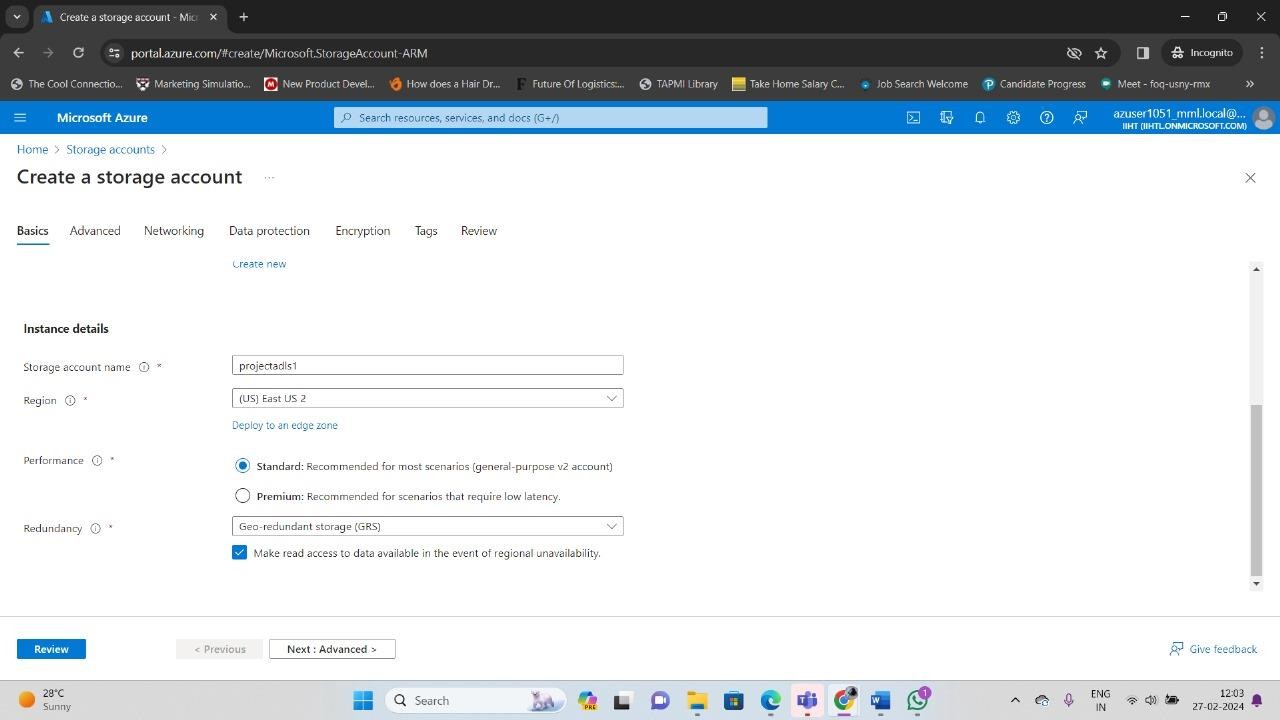




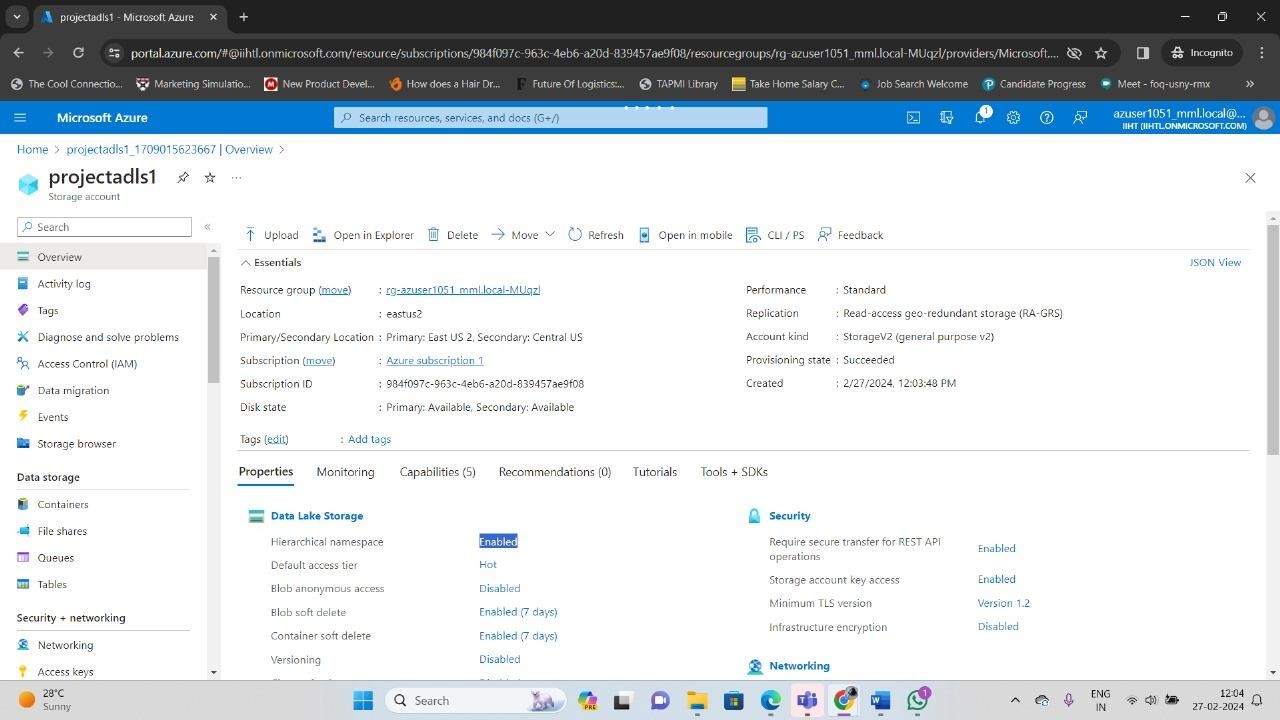
## . Create a Pipeline to copy the data task from Azure Data Lake Storage Gen2 to Azure Blob Storage Steps:

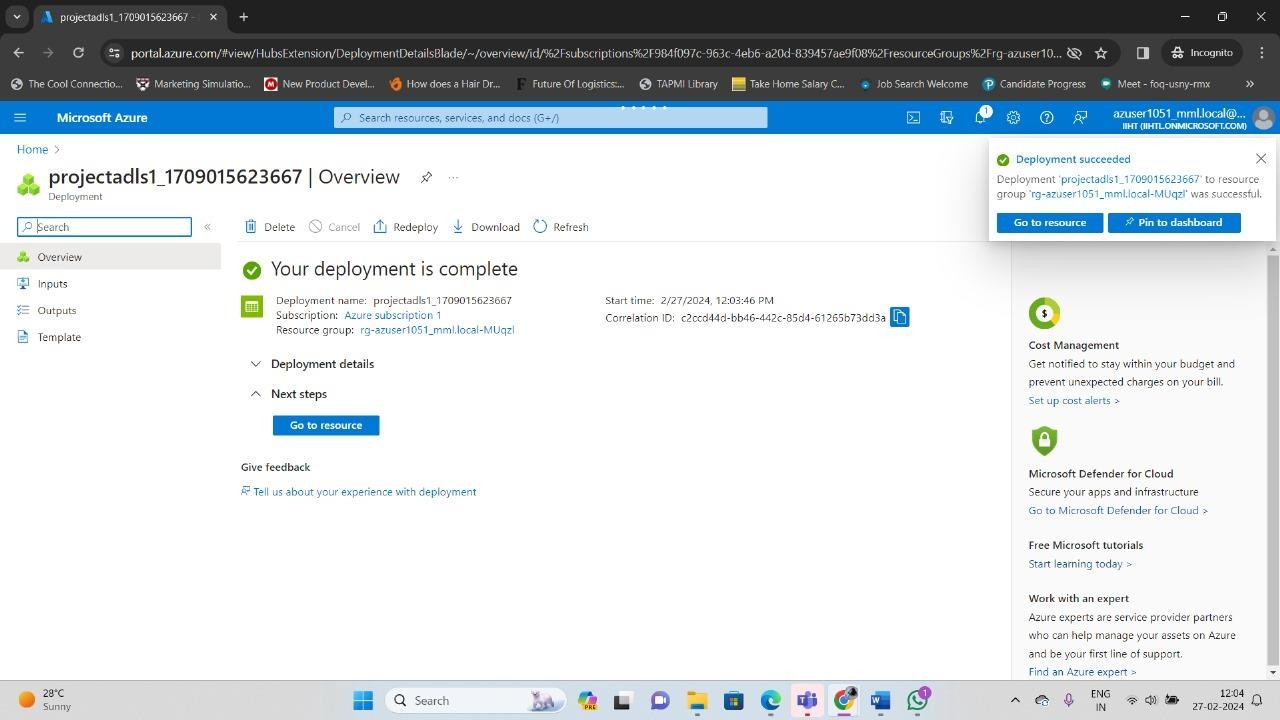
Login in to Azure Portal create a Storage Account

Storage Account Name - projectadls1



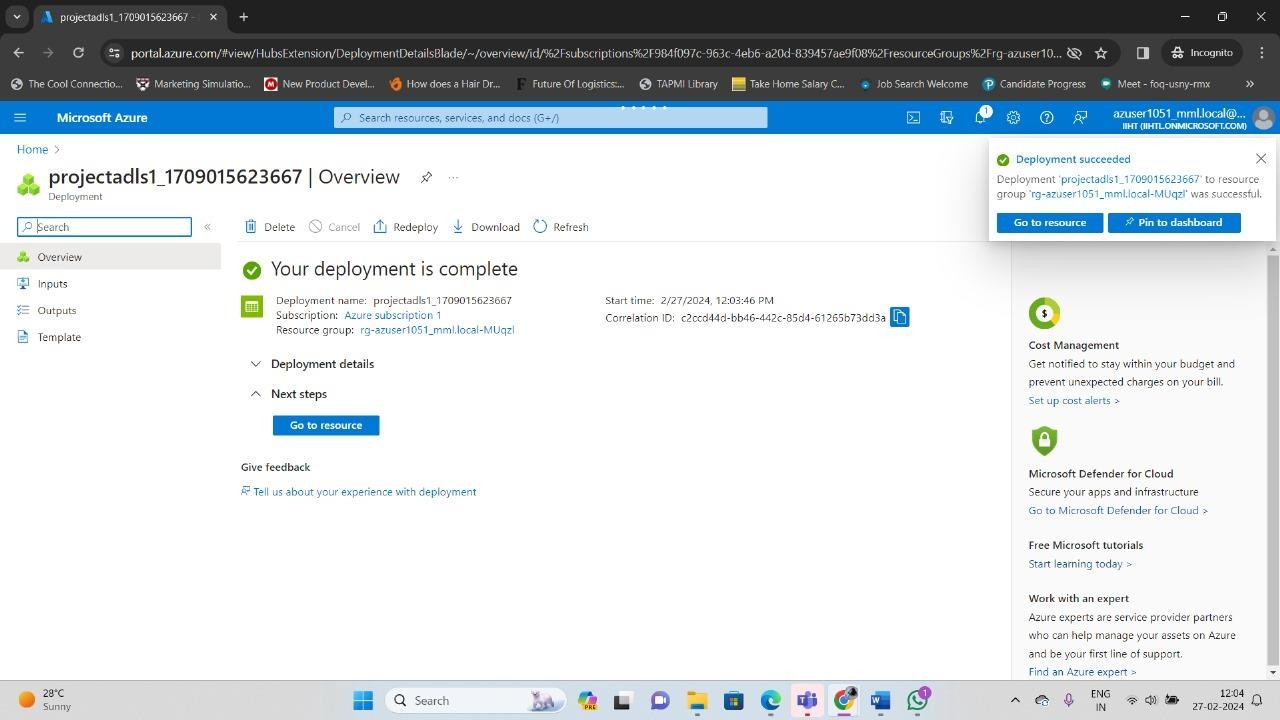
Enable hierarchical namespace

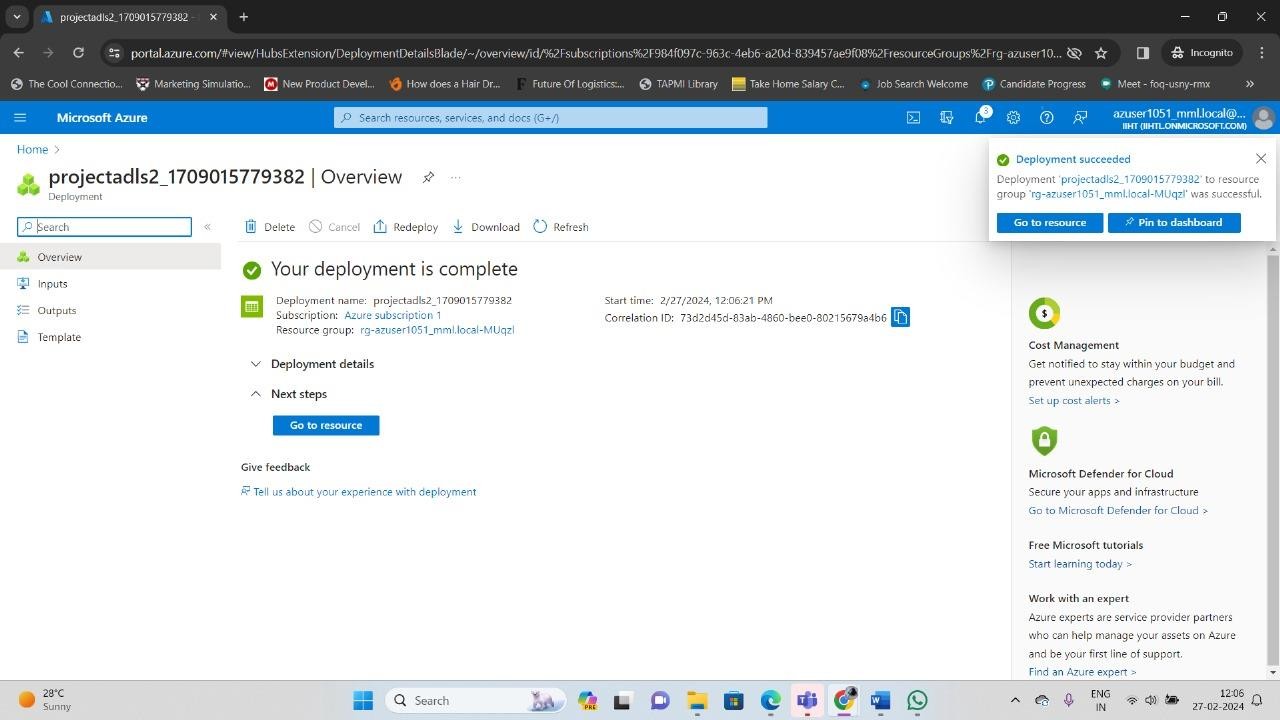




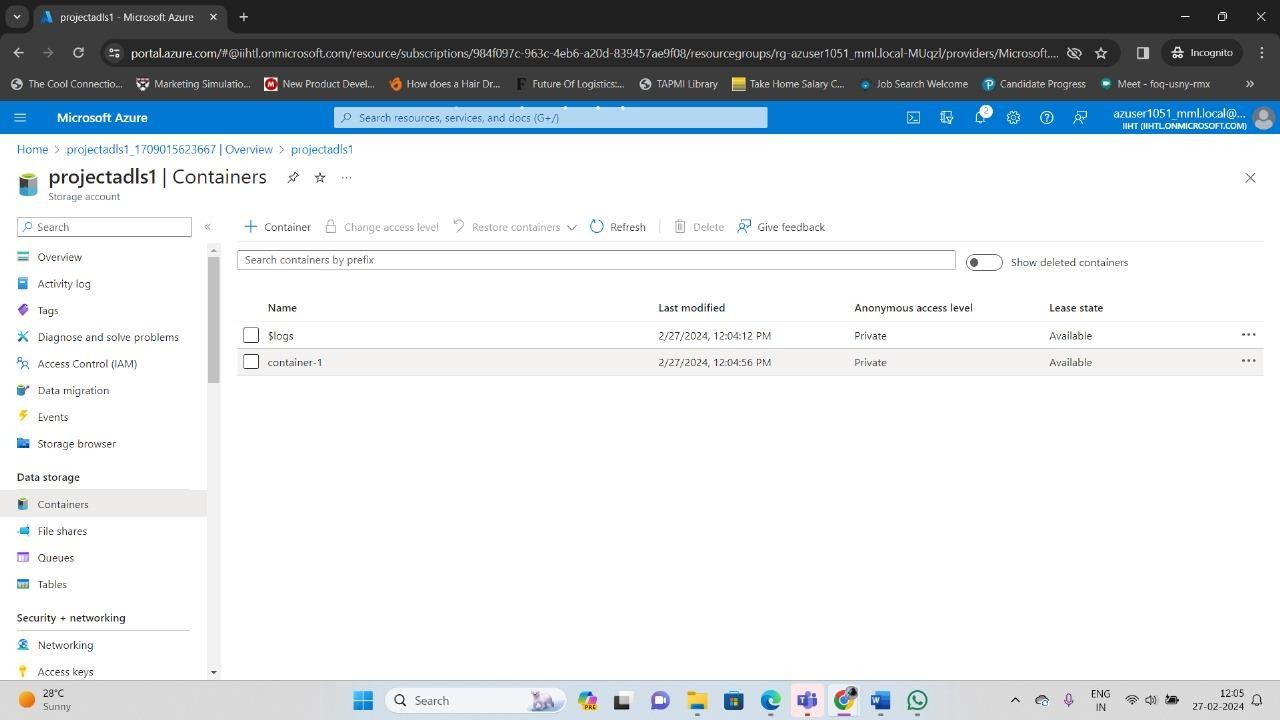
Create Storage account

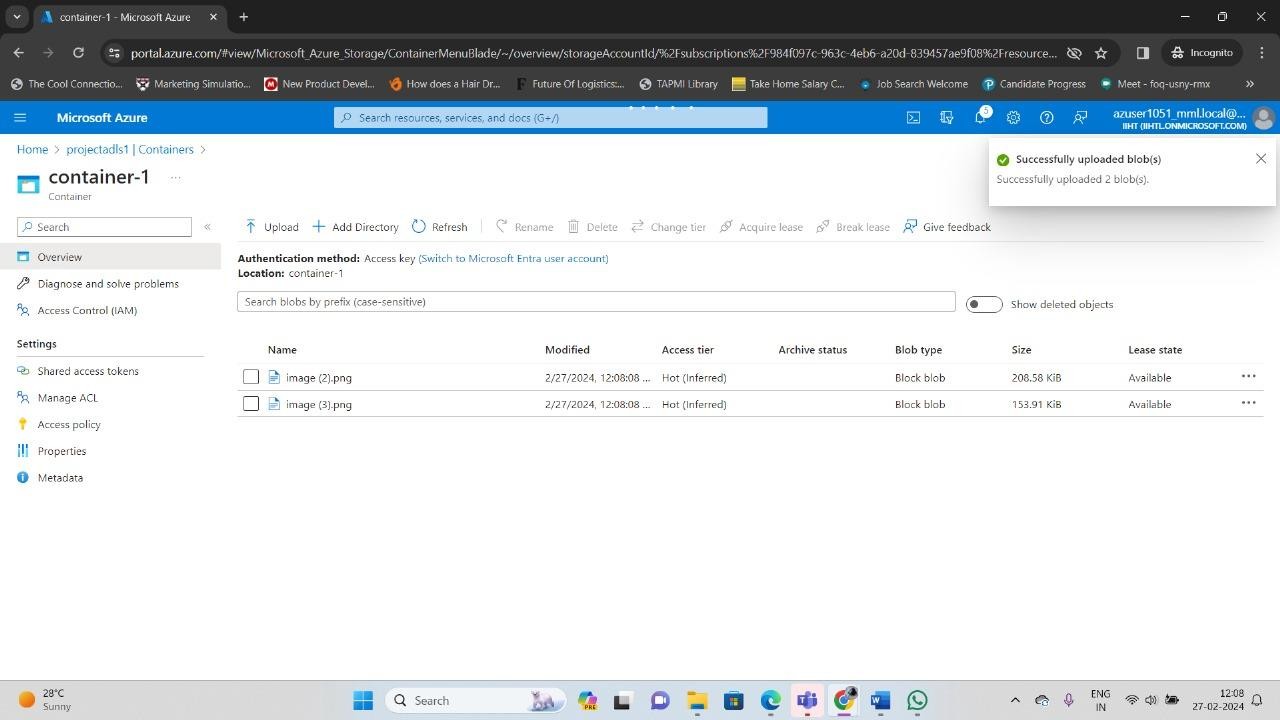
Storage account name - projectadls2



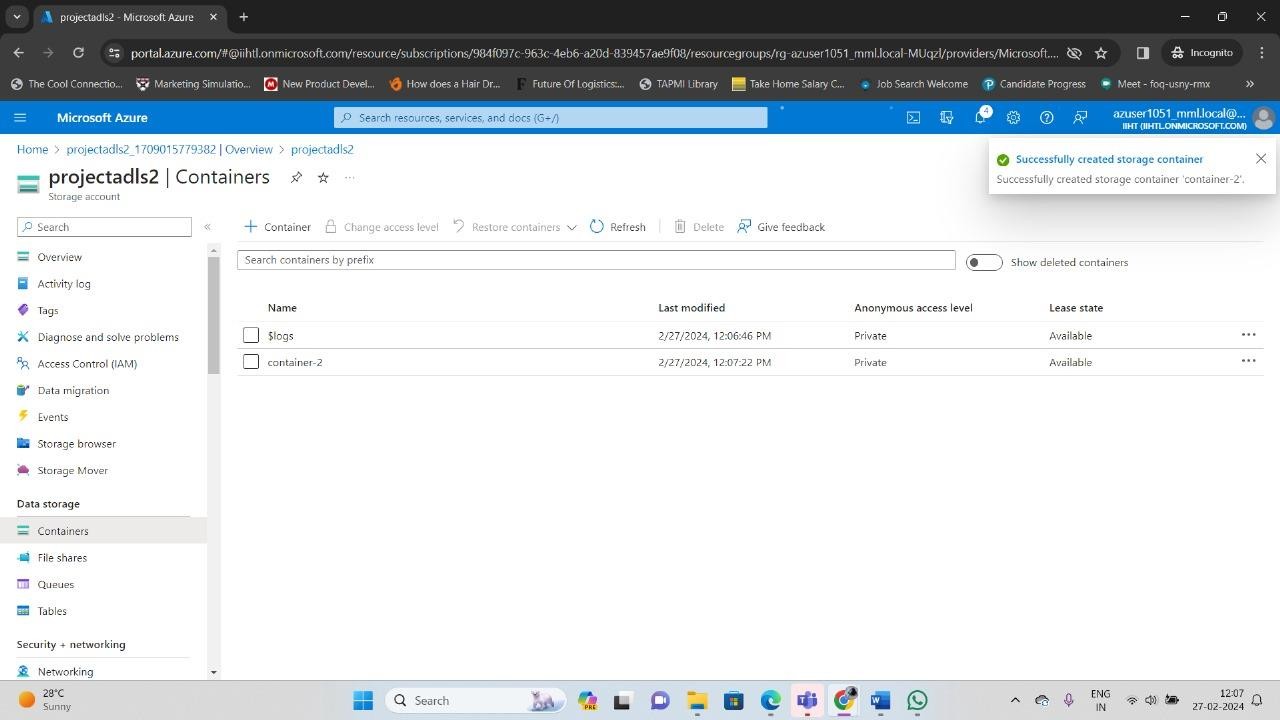


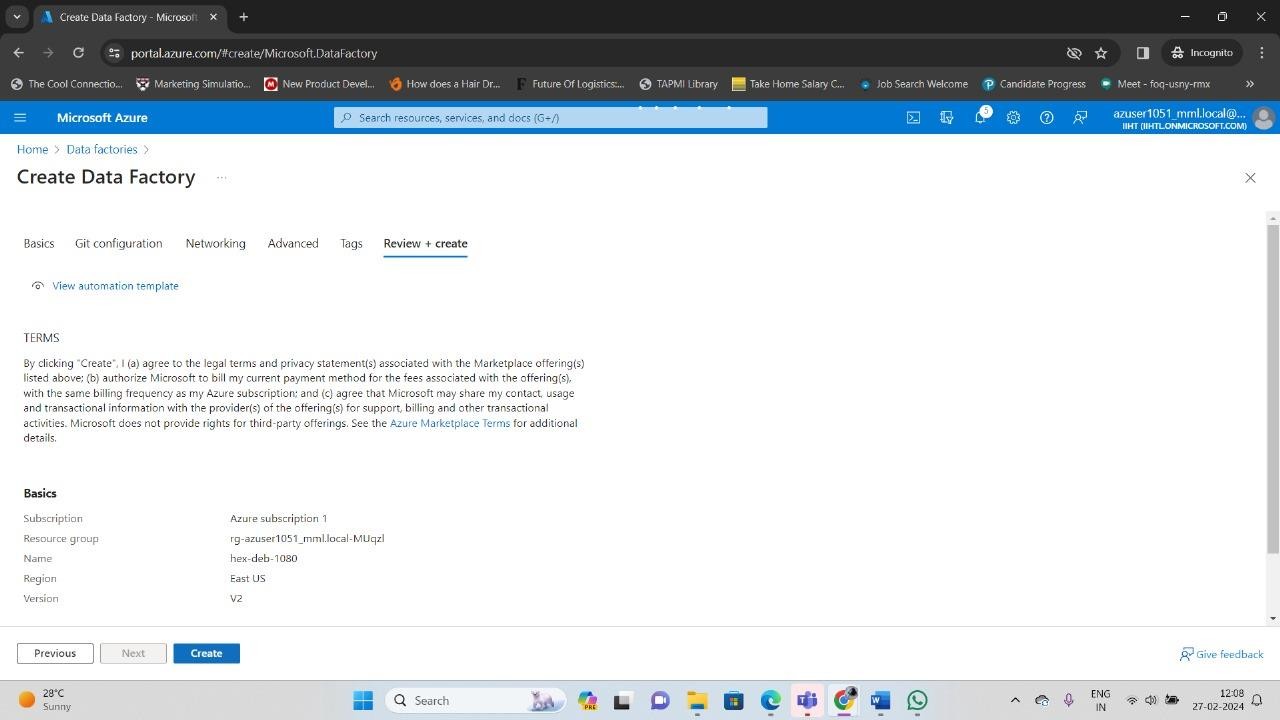
Create container container-1 in projectadls1



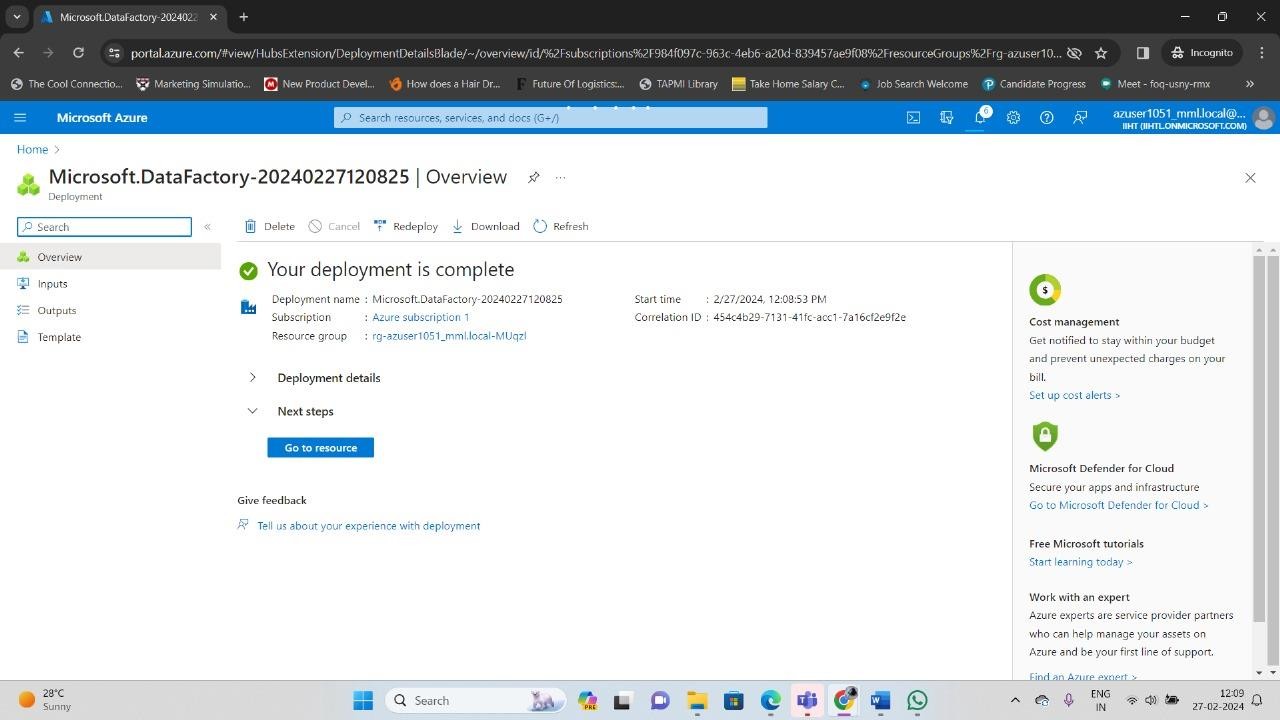


Create container-2 in projectadls2

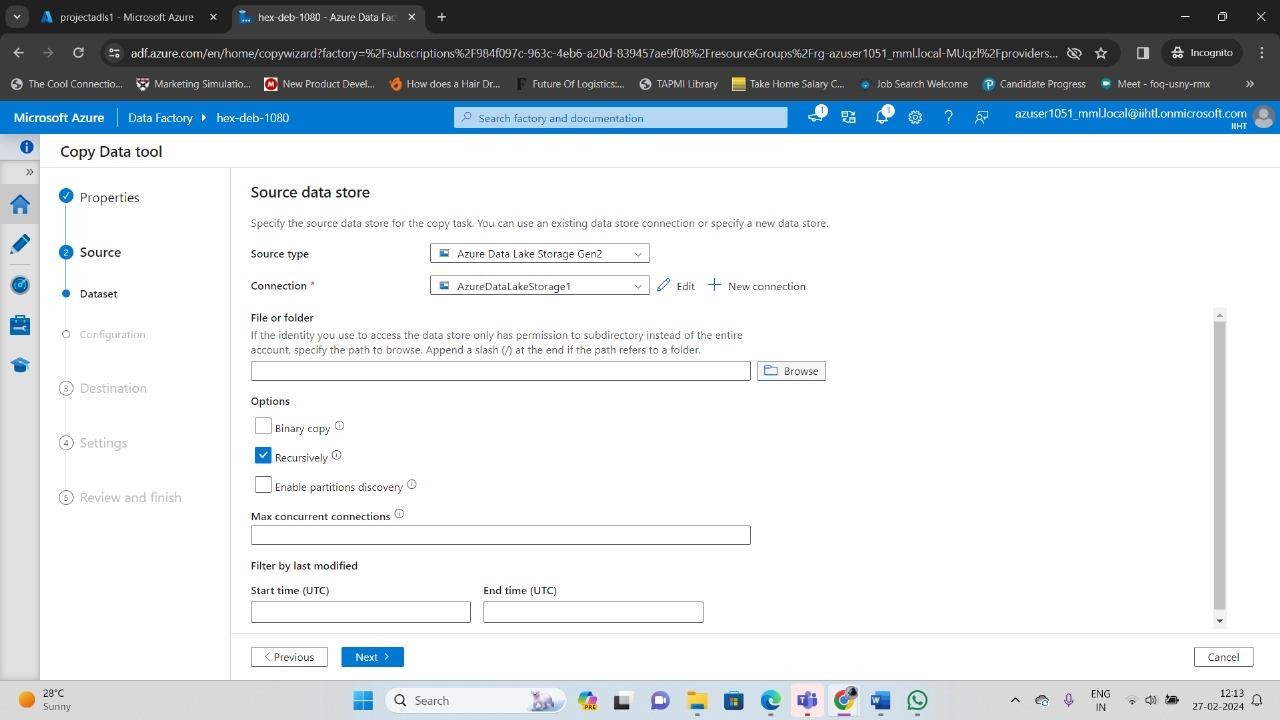




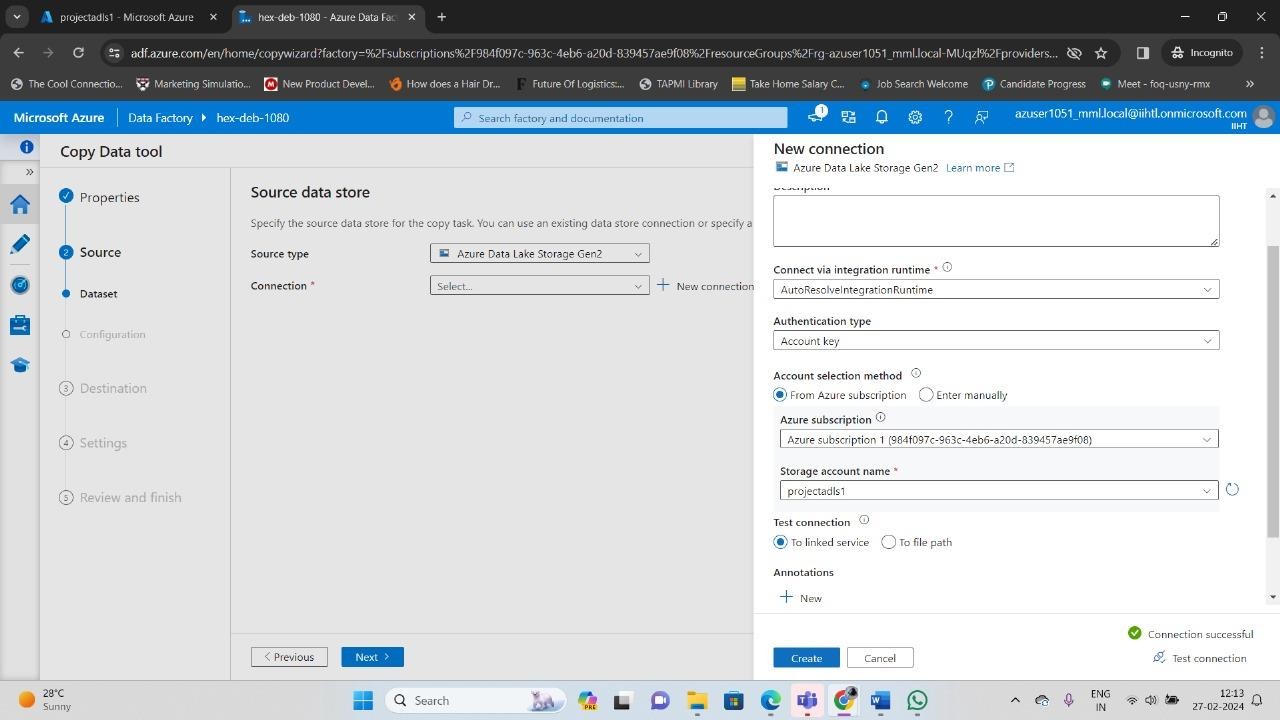
Azure data factory created



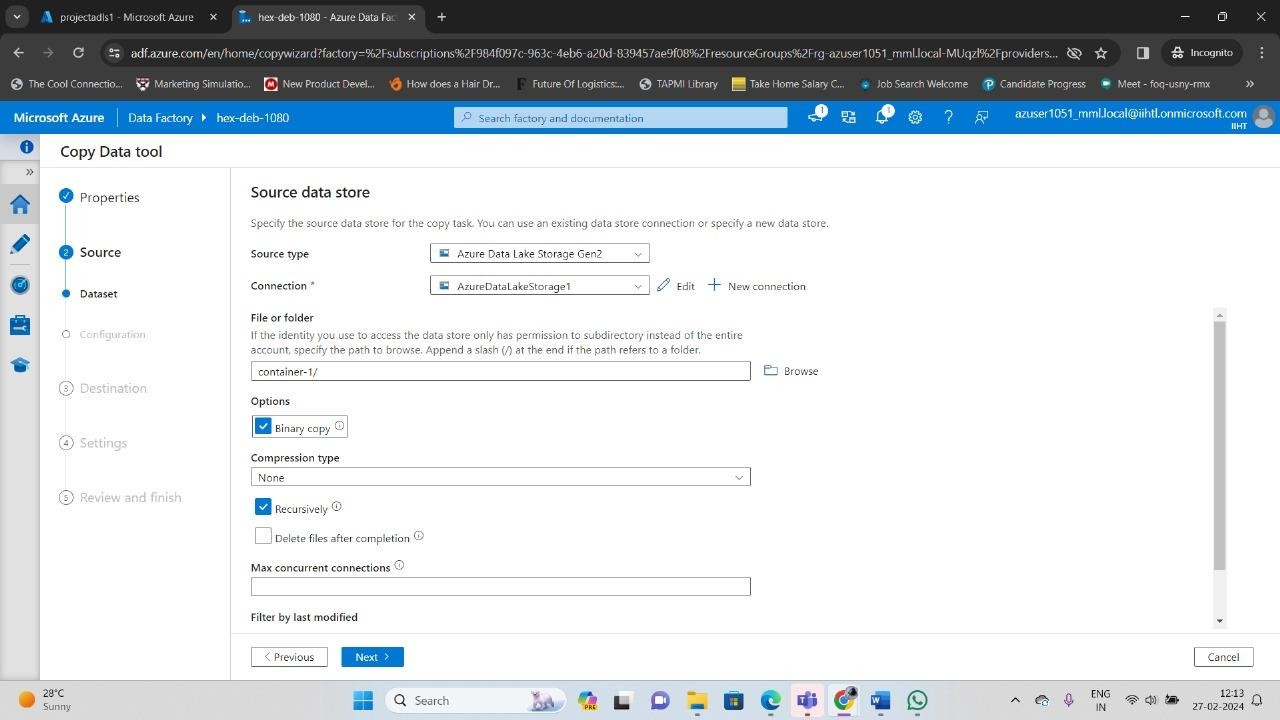
Azure Data Lake Storage Gen2



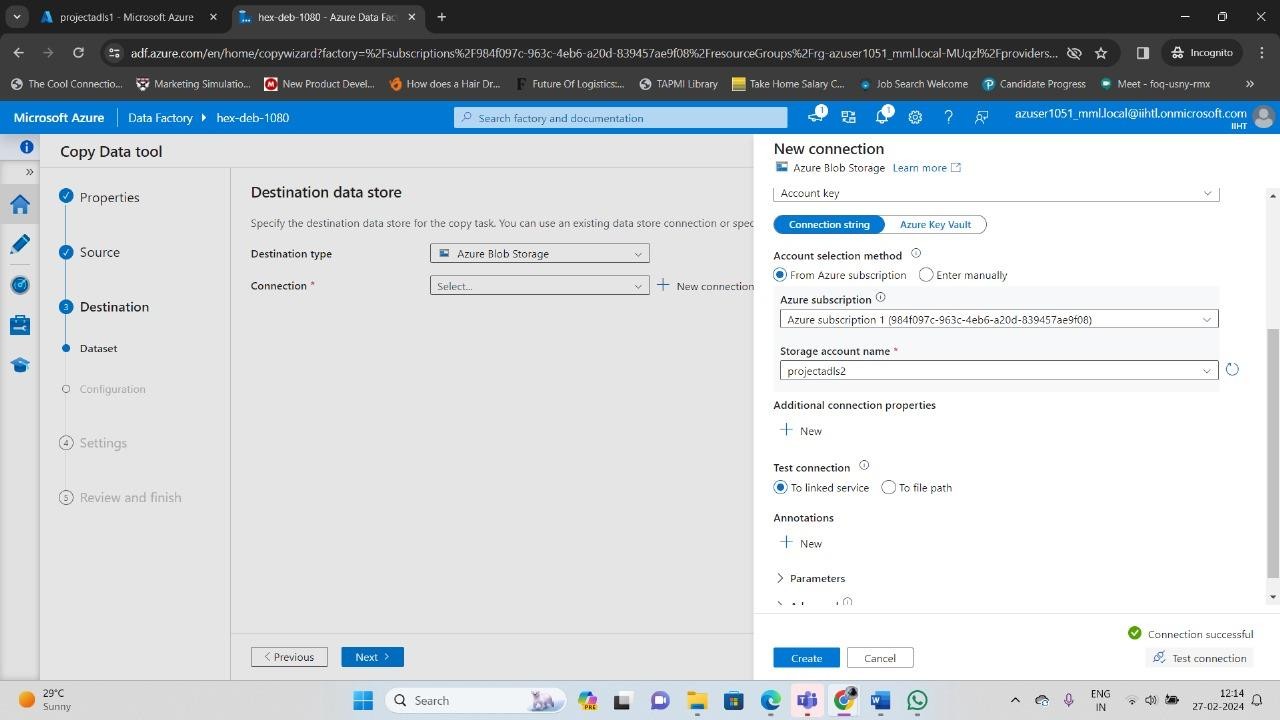
Create connection and specify storage account name



AzureDataLakeStorage1



Create connection and specify storage account name

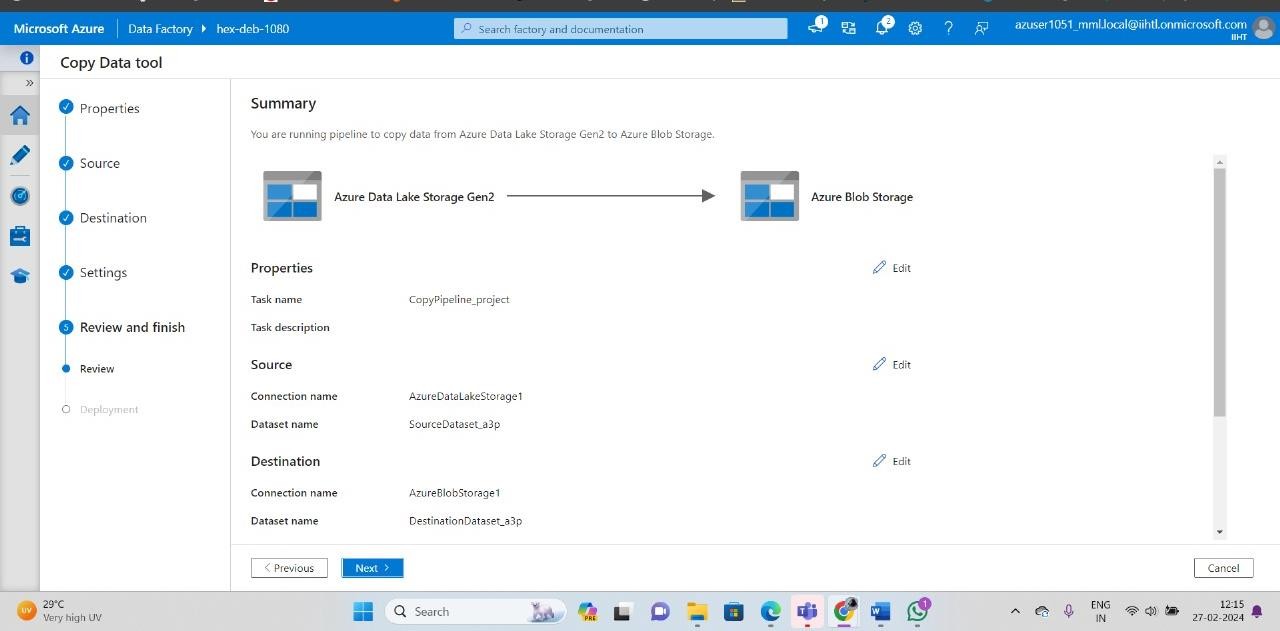


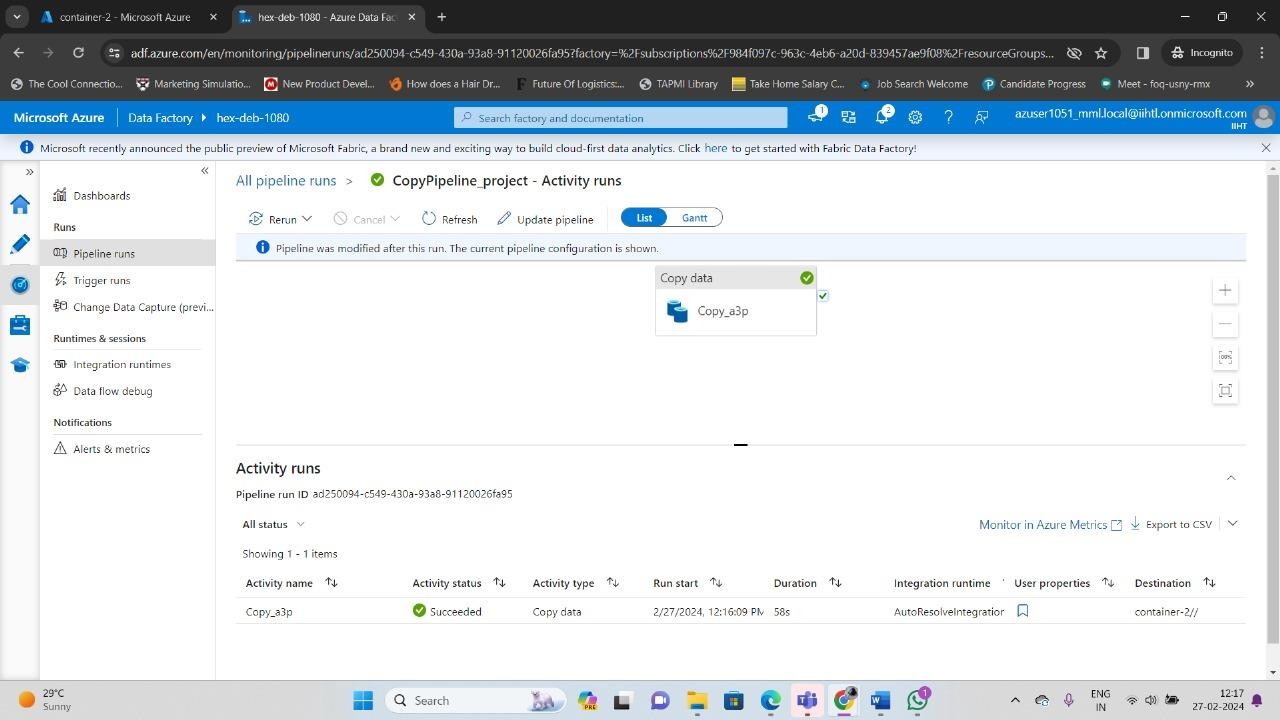
CopyPipeline\_project



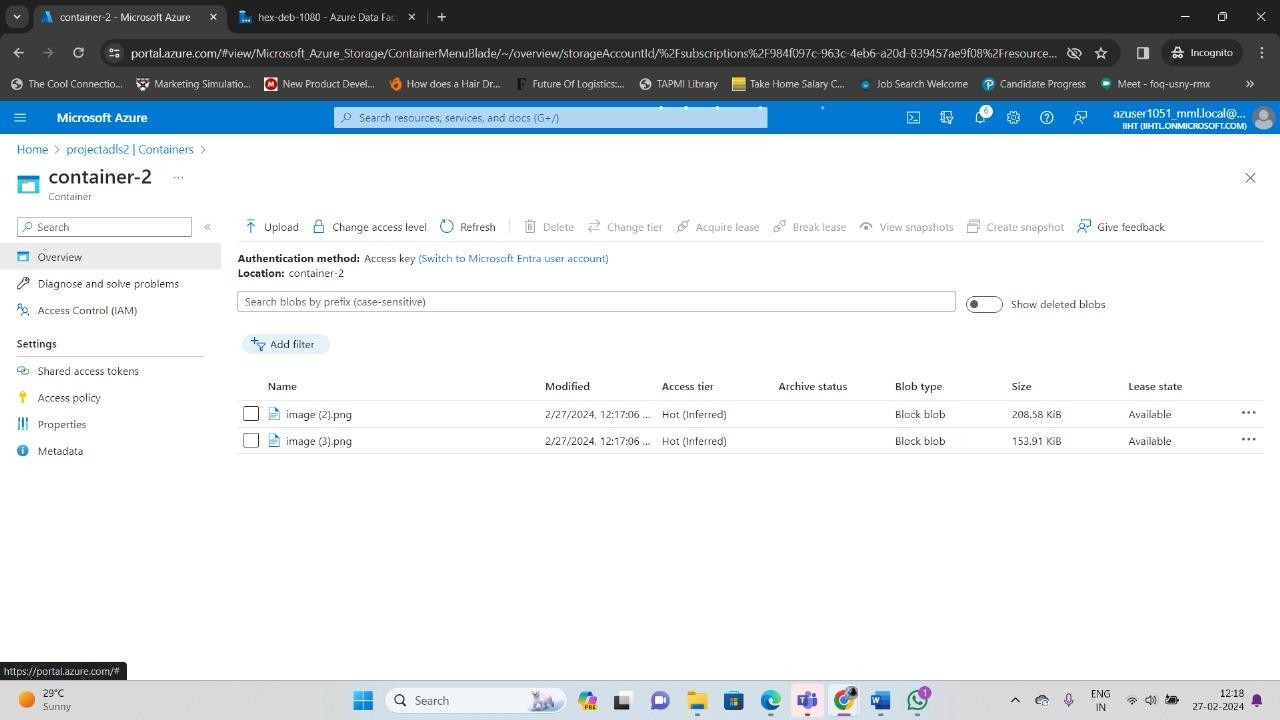
Running pipeline to copy data from

Azure data lake storage gen2 to Azure blob storage



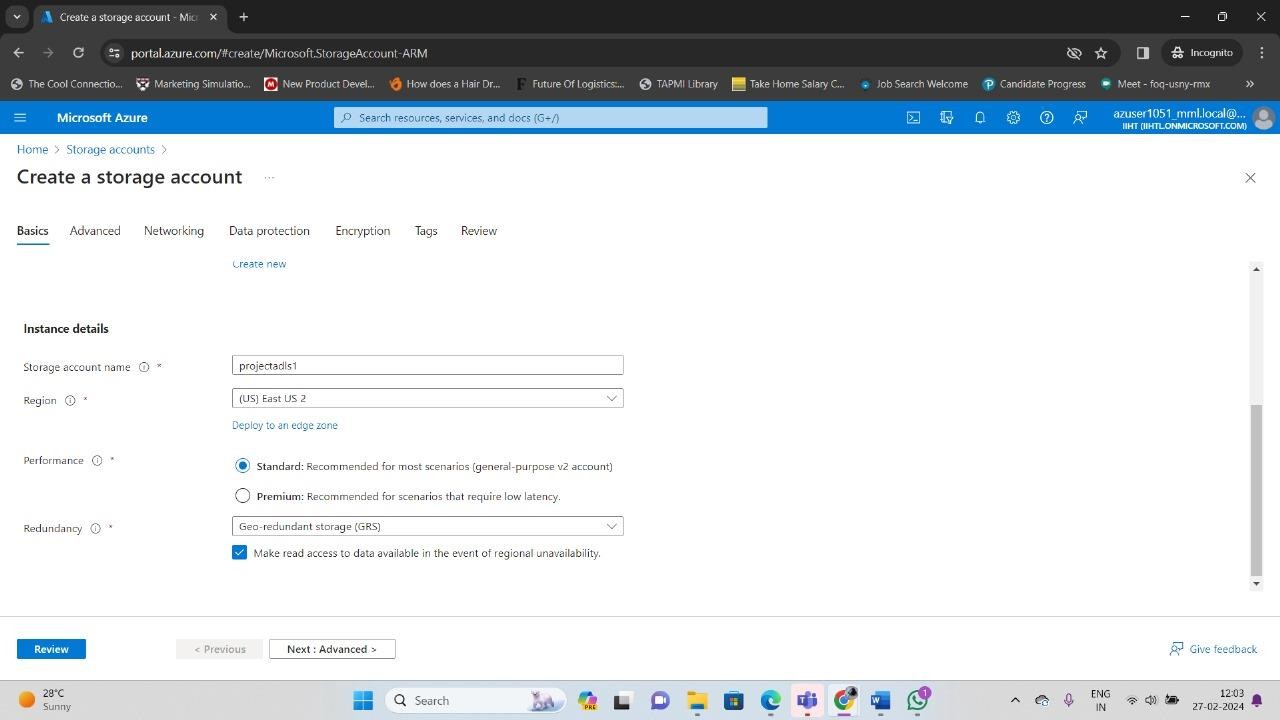


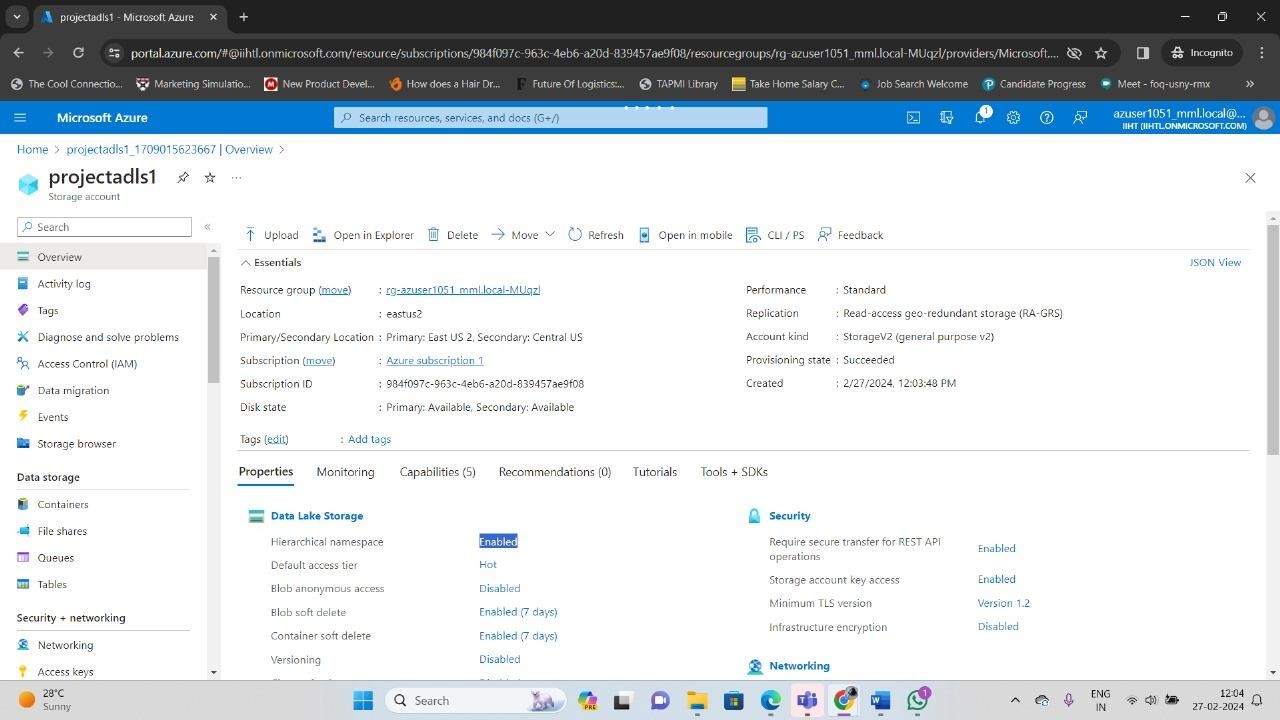
Data copied in container-2



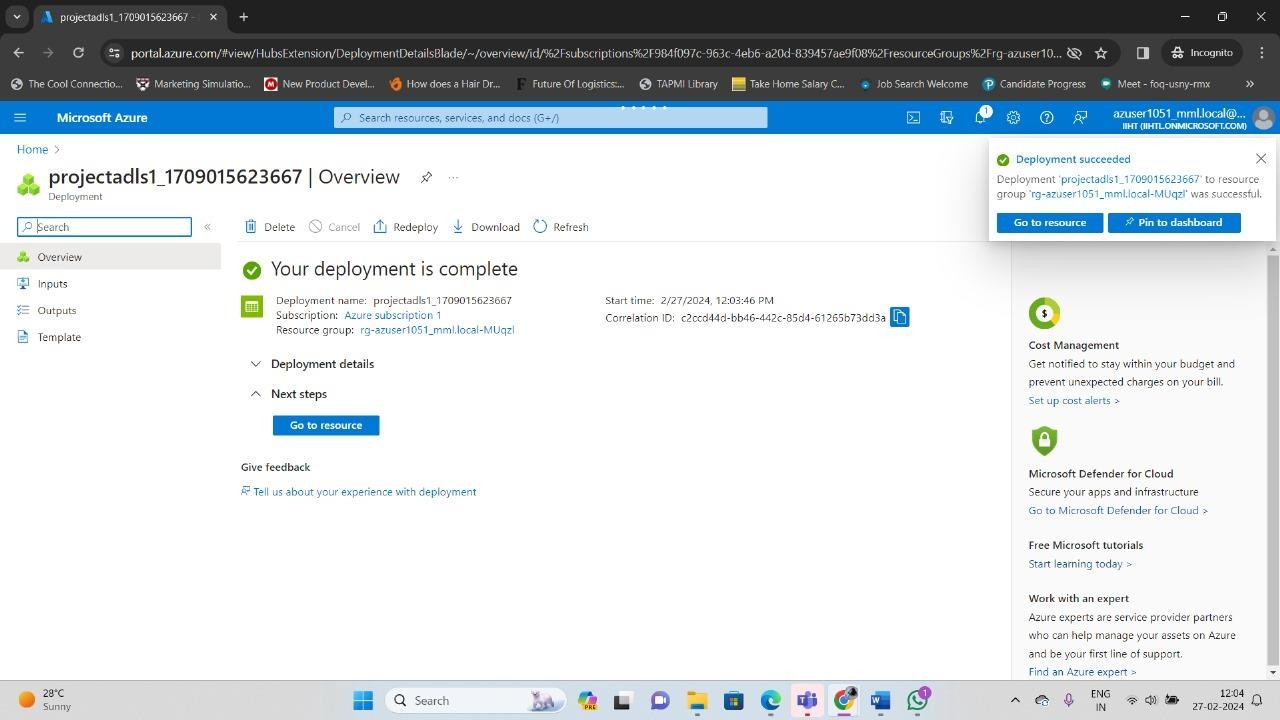
## 2 . Create a Pipeline to copy the data task from Azure Blob Storage to Azure Data Lake Storage Gen2 Steps:

Login in to Azure Portal create a Storage Account Storage Account Name - projectadls1

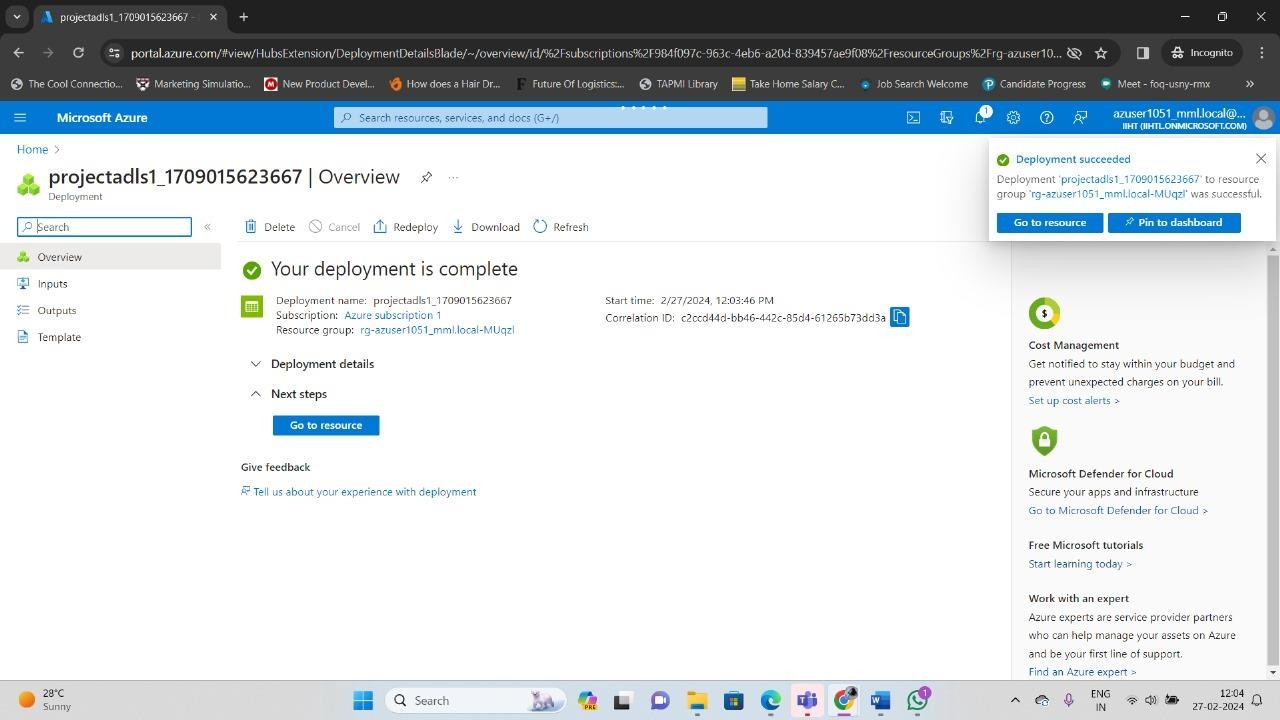




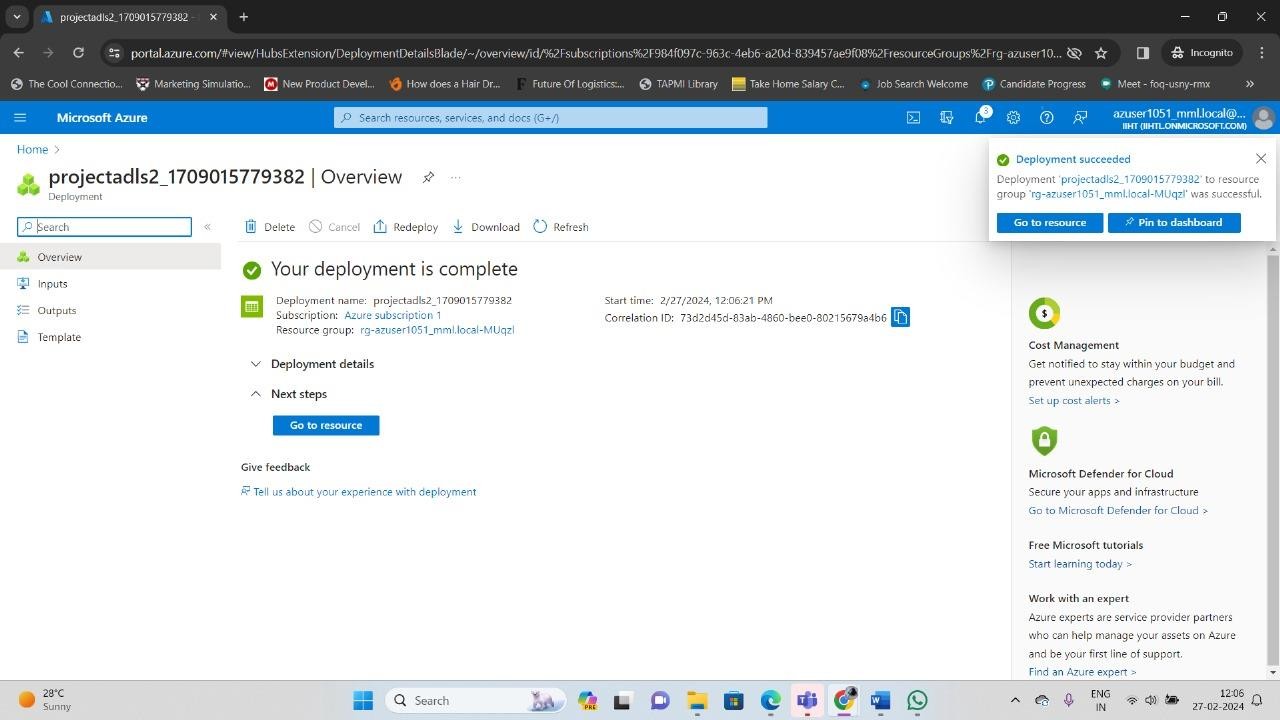
Storage account created

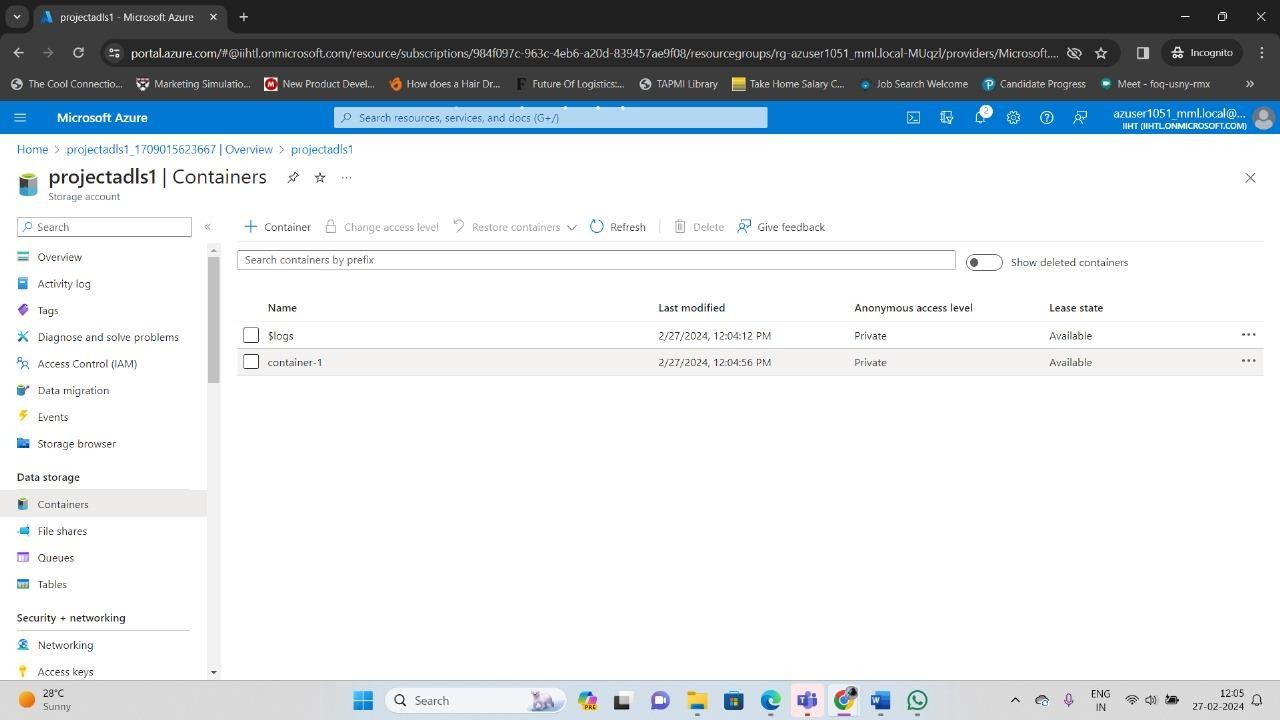


Storage account name - projectadls2

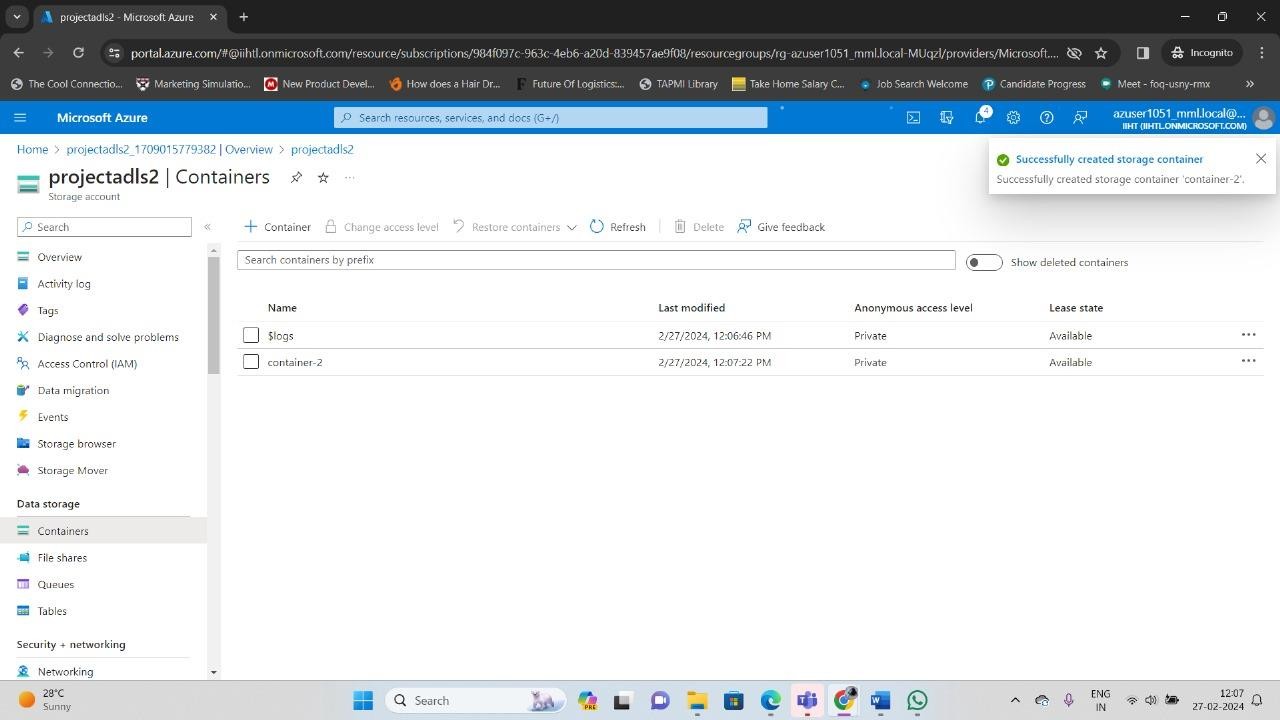


Storage account created



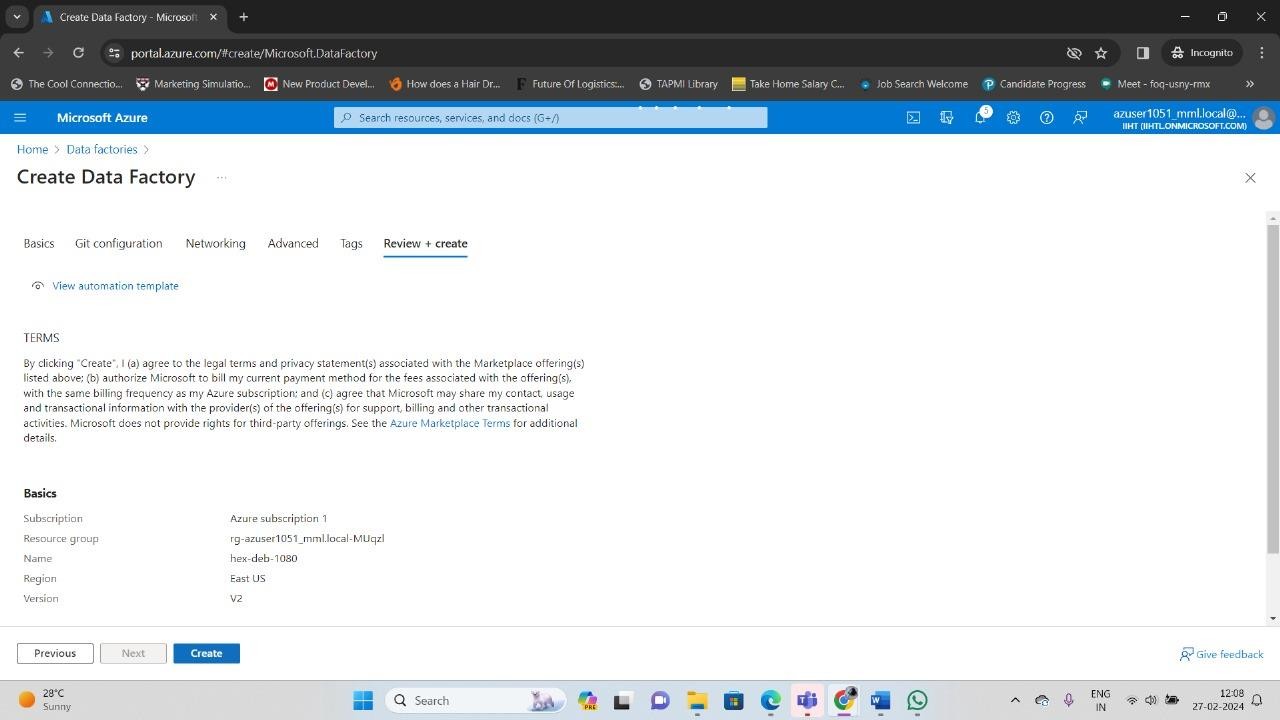


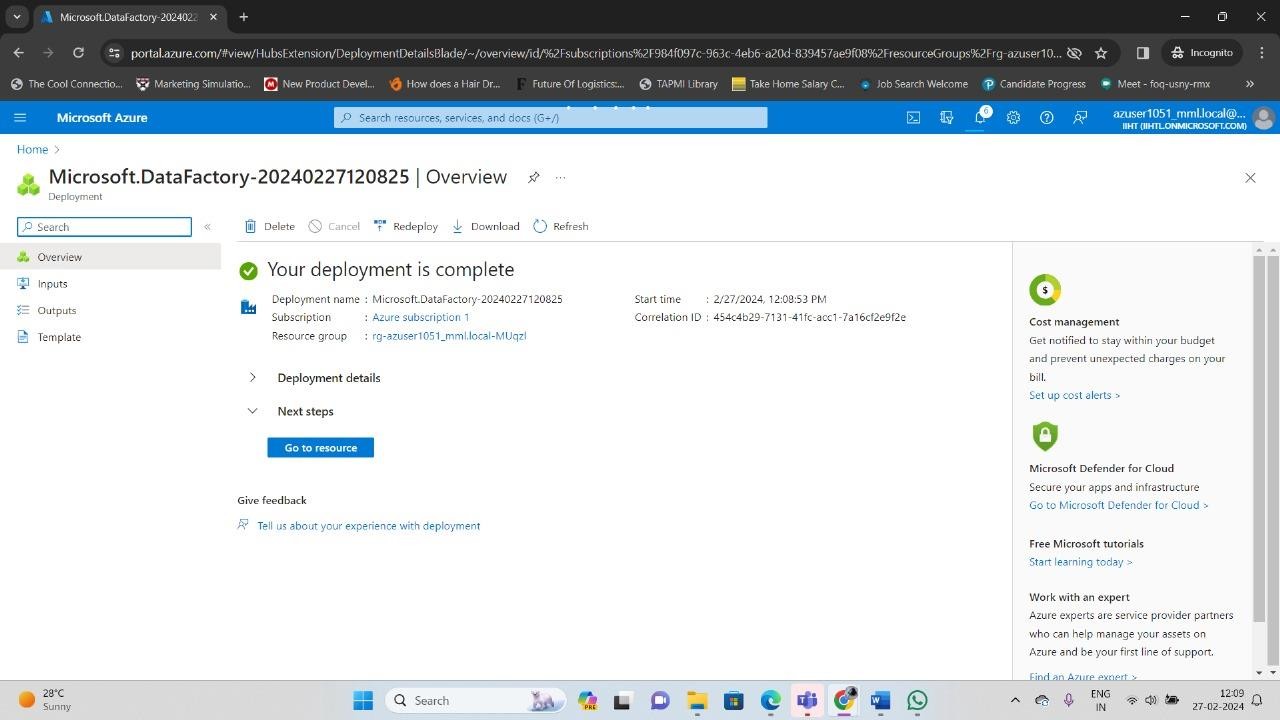
Create container-2 in projectadls2



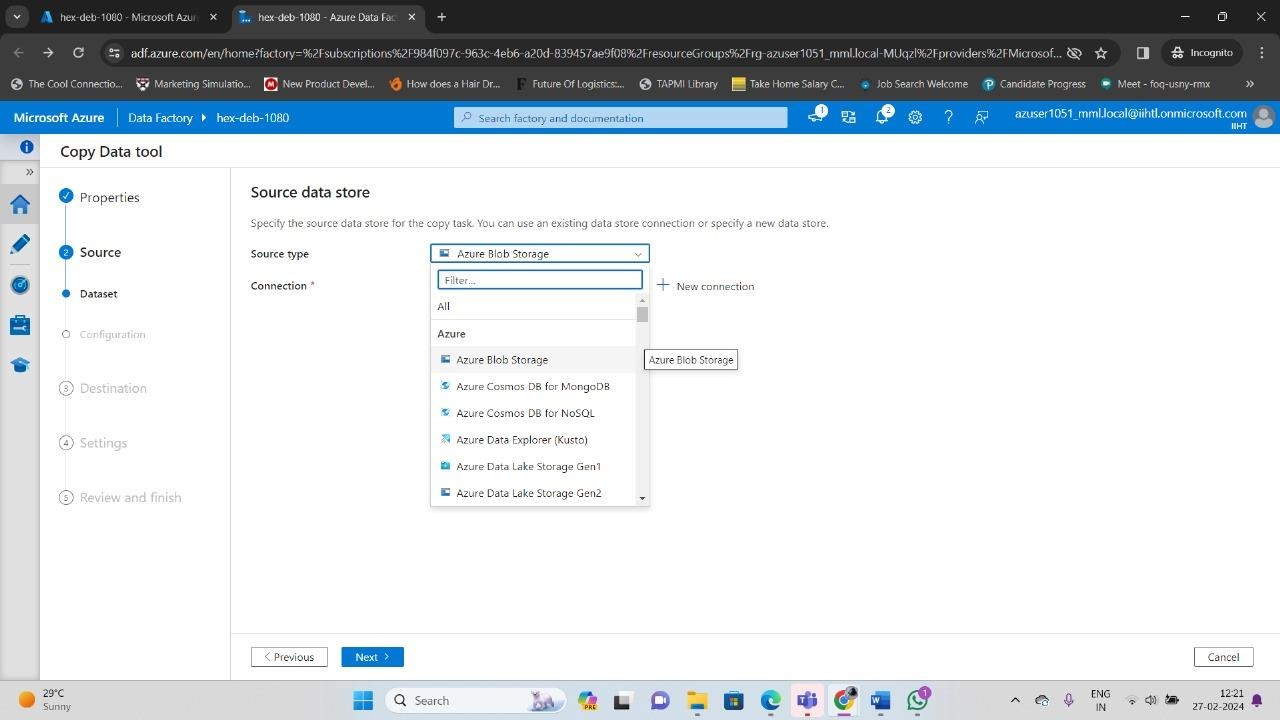


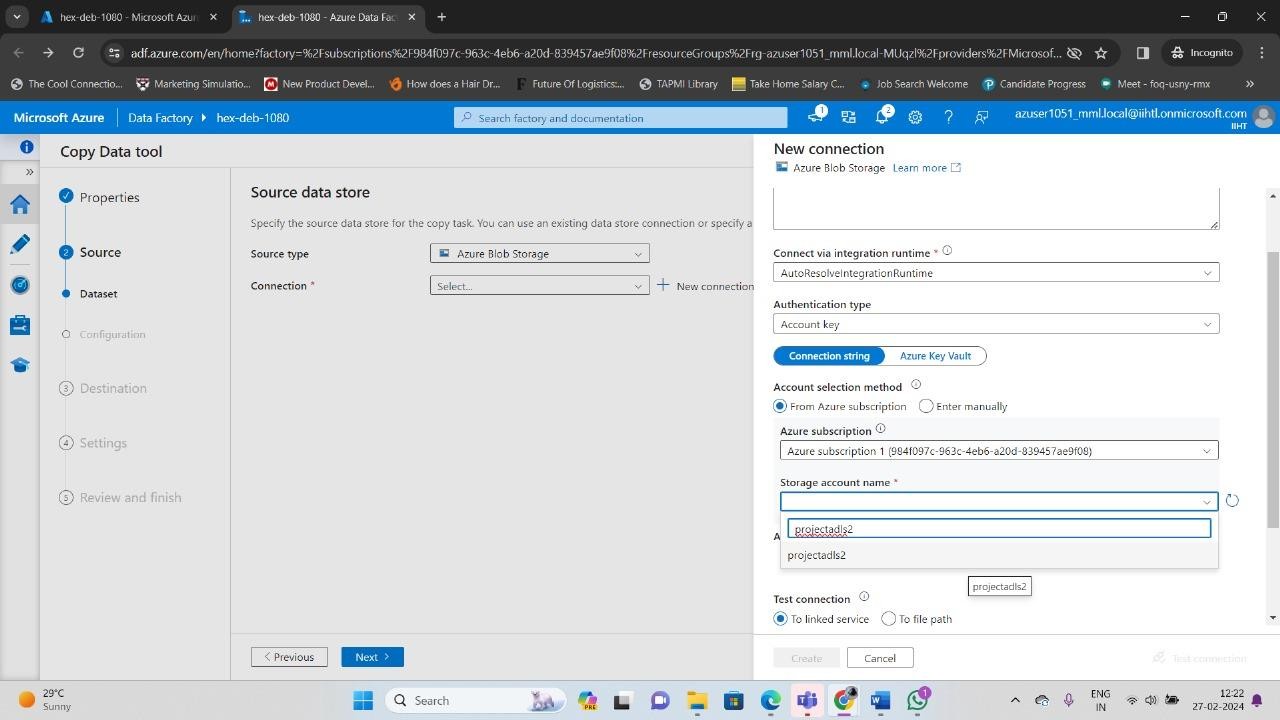
Create azure data factory



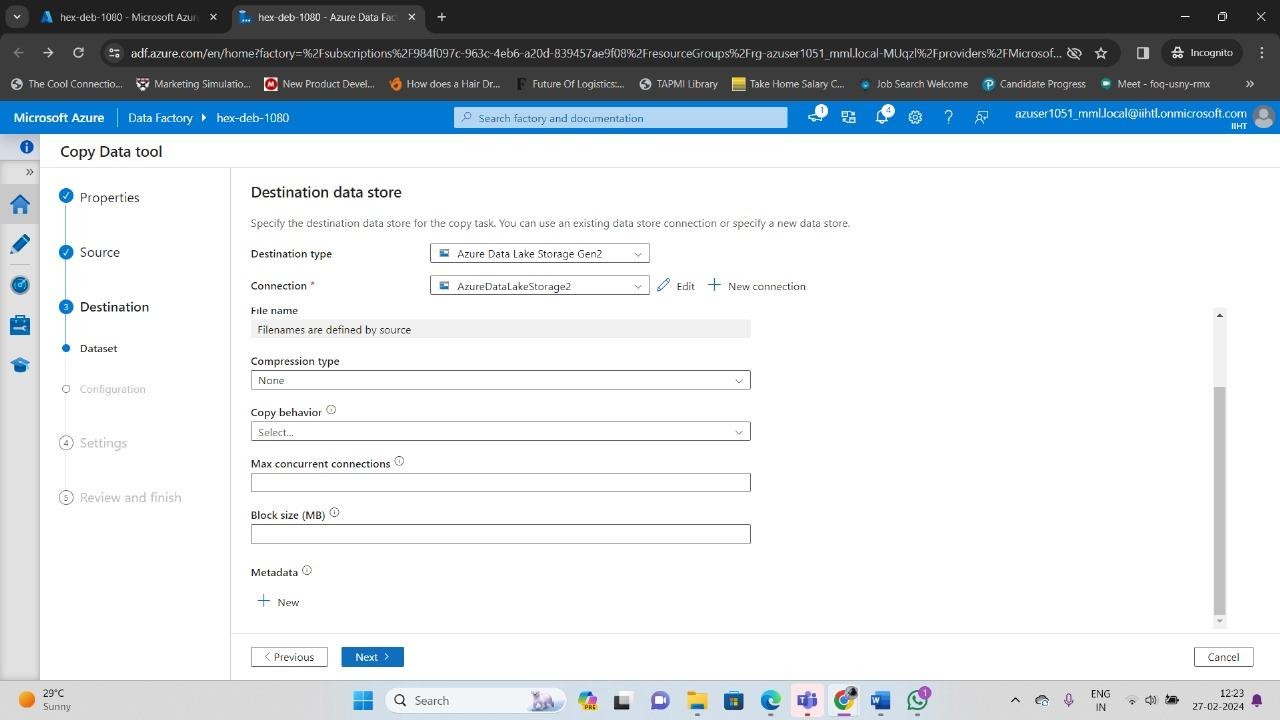


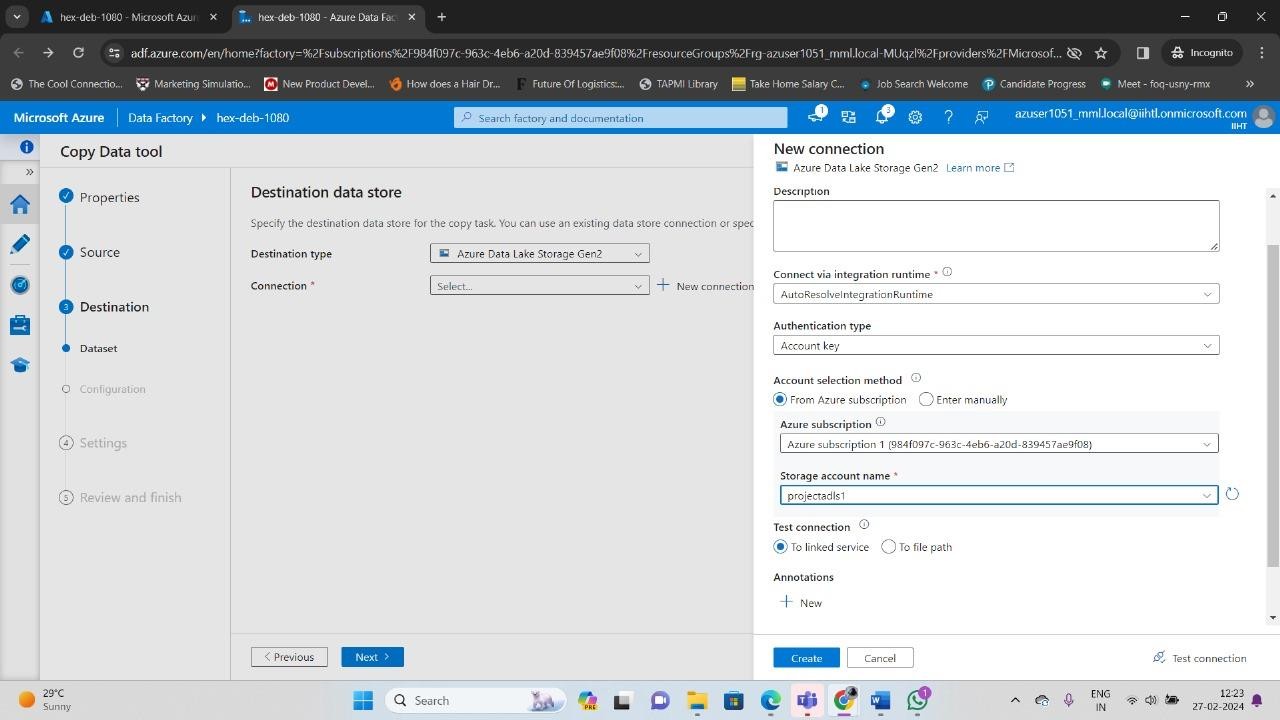
Specify the source data store for the copy task Azure blob storage



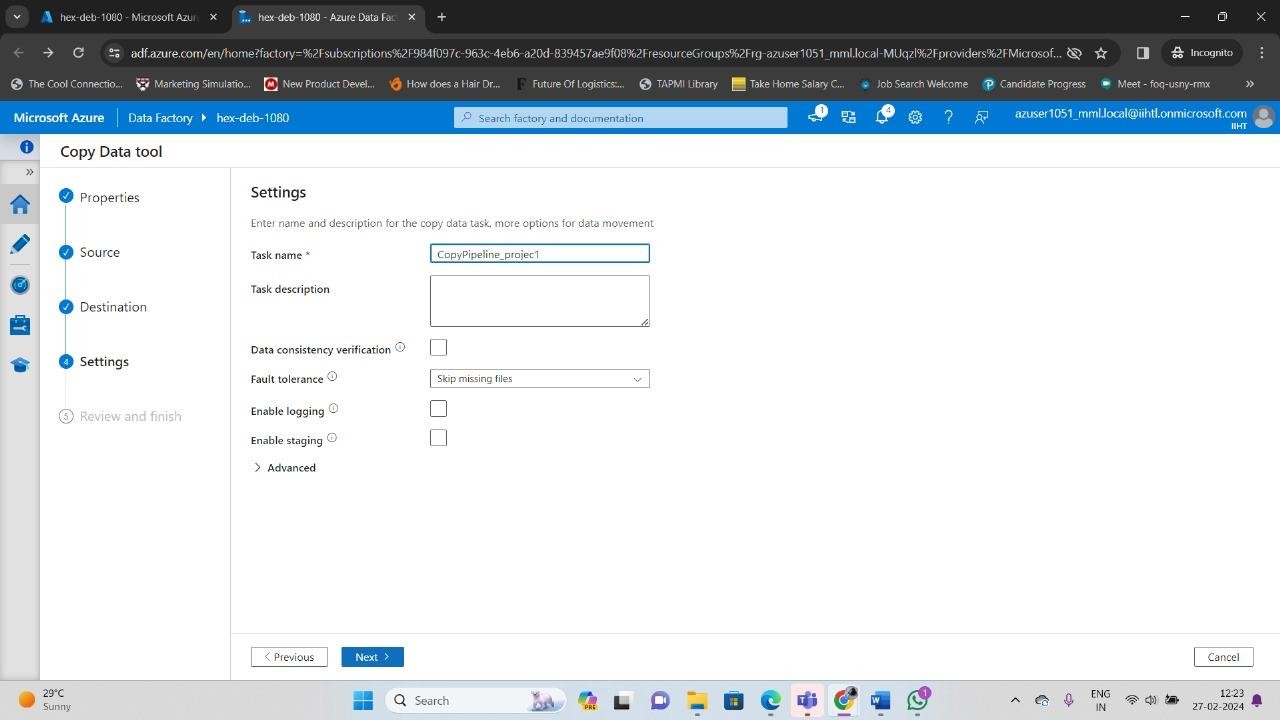


Specify the destination data store for the copy task

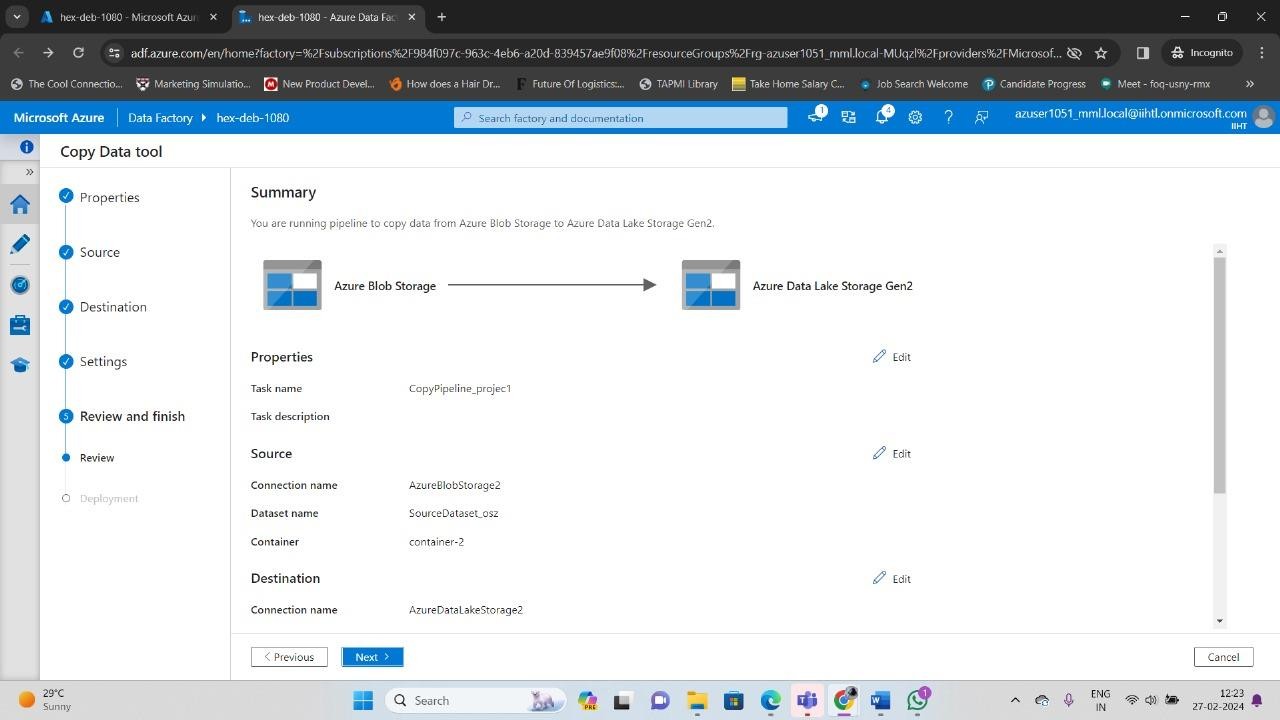




Enter the name of the copy data task CopyPipeline\_project1

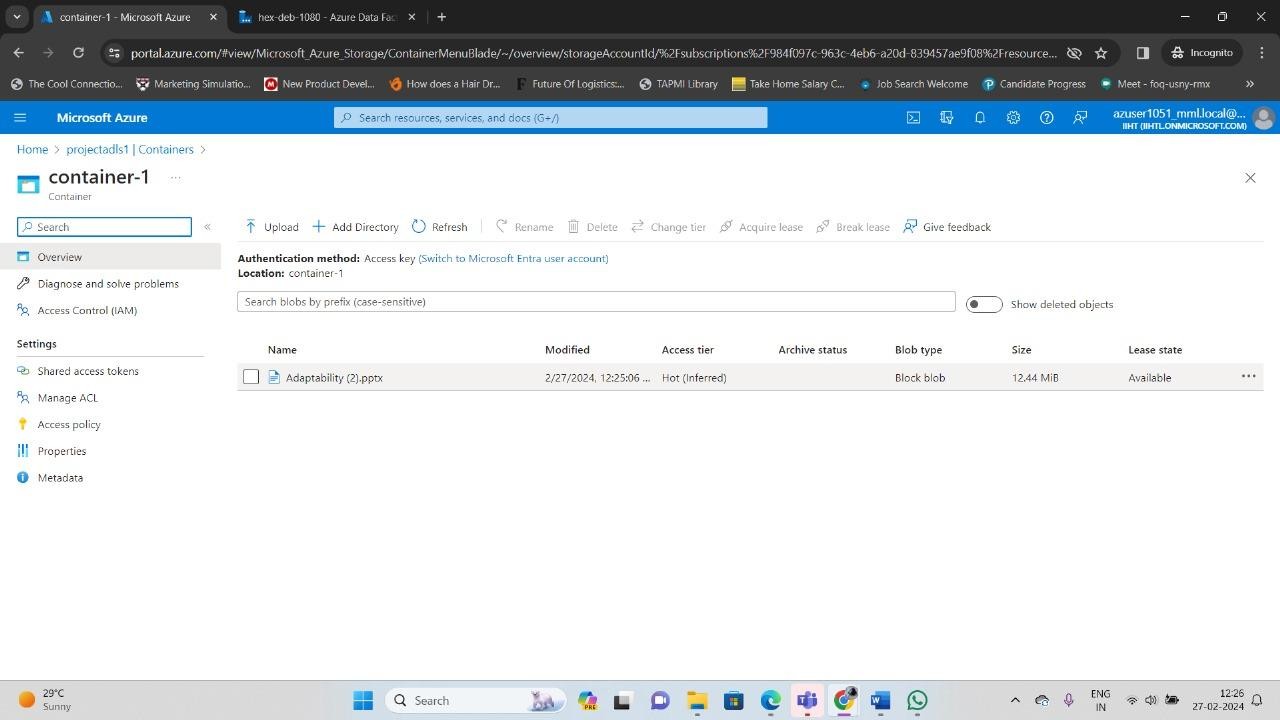


Azure Blob Storage to Azure Data Lake Storage Gen2



Activity succeeded and pipeline ingested





# Reference links:

[https://learn.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-](https://learn.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-introduction) [introduction](https://learn.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-introduction)

[https://learn.microsoft.com/en-us/azure/data-factory/load-azure-data-](https://learn.microsoft.com/en-us/azure/data-factory/load-azure-data-lake-storage-gen2) [lake-storage-gen2](https://learn.microsoft.com/en-us/azure/data-factory/load-azure-data-lake-storage-gen2)