**SAS(SHARED ACCESS SIGNATURE)**

**LINKED SERVICES FOR A PARTICULAR AZURE RESOURCE:**

**Why Do We Have Linked Services in Azure Data Factory (ADF)?**

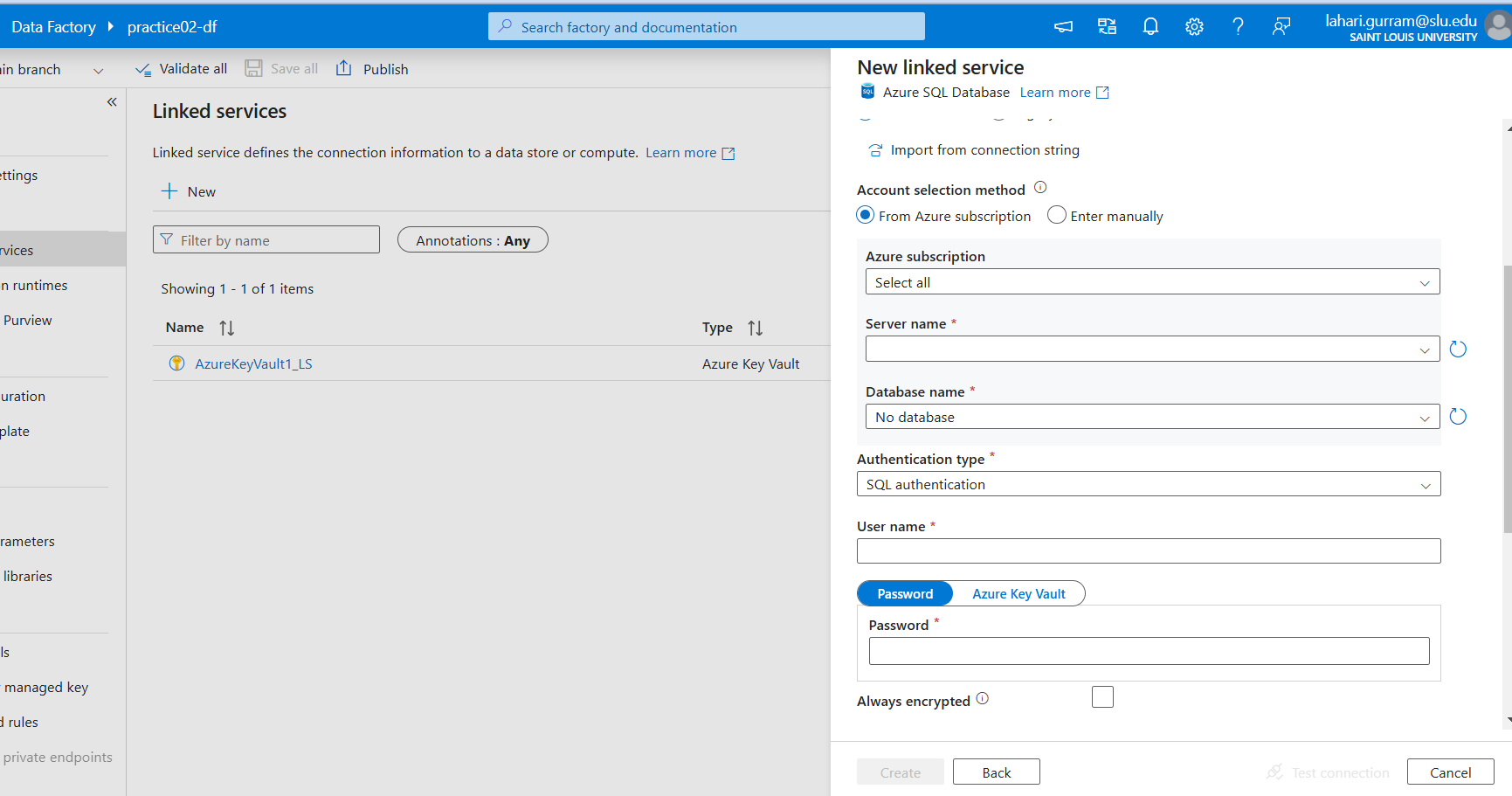
A Linked Service in Azure Data Factory (ADF) acts as a connection string that allows ADF to connect to external data sources like Azure Storage, Azure SQL Database, or on-premises databases. It defines how ADF should authenticate and interact with the data source.

**Why Is a Linked Service Required?**

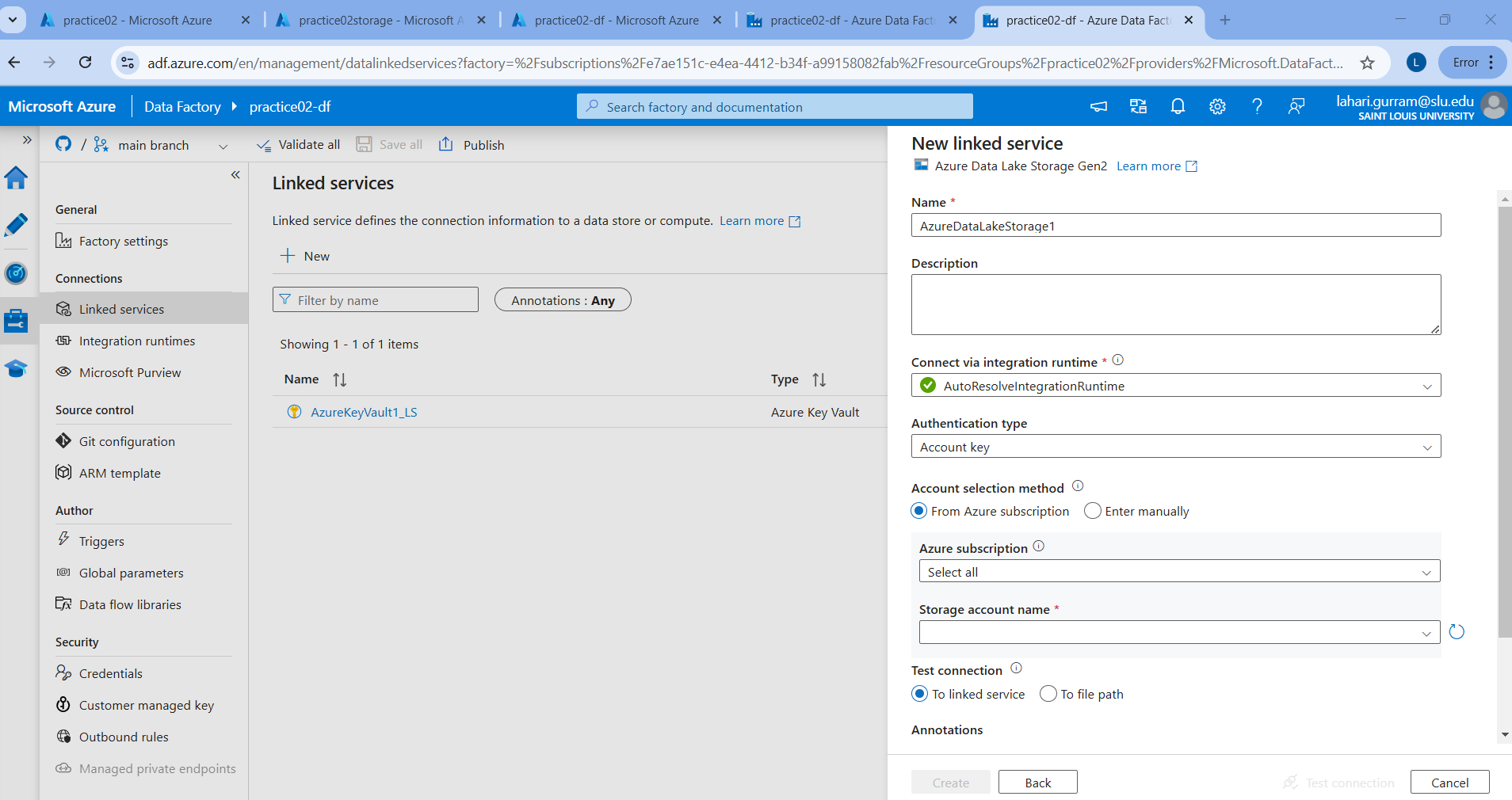
1. **Establishes a Connection** – Allows ADF to securely connect to **Storage Accounts, Databases, APIs, etc.**
2. **Centralized Configuration** – Instead of manually providing connection details in every pipeline, you define it **once** in a Linked Service.
3. **Supports Multiple Authentication Methods** – A Linked Service can use:
   * **Account Keys** (less secure)
   * **Shared Access Signature (SAS) Tokens** (temporary and safer)
   * **Managed Identity** (most secure, avoids storing credentials)
   * **Service Principal** (for role-based access control)
4. **Enhances Security** – Instead of hardcoding credentials in pipelines, Linked Services securely **store and encrypt** authentication details.
5. **Simplifies Reusability** – Once created, multiple **datasets and activities** in ADF can reference the same Linked Service, avoiding redundancy.

A **Linked Service** in Azure Data Factory (ADF) is created **for a specific data source**, such as a **Storage Account, SQL Database, REST API, or other services**. When you create a Linked Service for a **Storage Account**, you are defining a connection to that **particular storage account** so ADF can access and interact with its data.

So we create a linked service for one particular RESOURCE (SECURELY) ; and in the data factory these are CONNECTED TO preform the operations we need( HERE ADF acts as a centralized thing for connecting various azure resources using pipelines etc..)



The above one is linked service for AZURE SQL DATA BASE AND IT USES SERVER NAME, DATABASE NAME, AUTHENTICATION TYPE (Service Principal, SQL authentication, User managed identity , System managed identity) TO CONNECT TO THAT STORAGE ACCOUNT SECURELY



The above one is linked service for STORAGE Account AND IT USES various authentication types like Account key , SAS, Service Principal , User managed Identity, System Managed Identity to connect SECURELY to that particular storage account

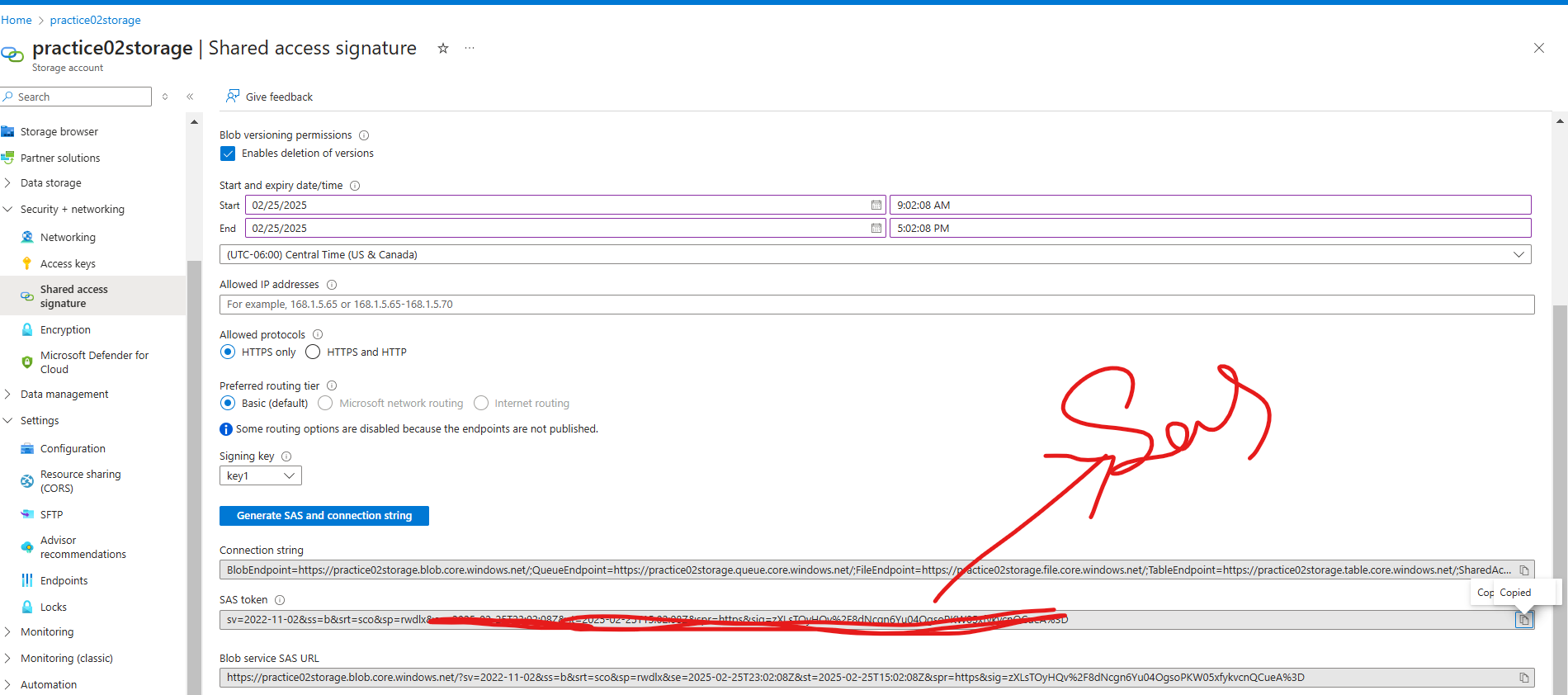
**SAS (SHARED ACCESS SIGNATURE)**

**What is SAS (Shared Access Signature) in Azure?**

A **Shared Access Signature (SAS)** in Azure is a **secure URL** that provides **restricted access** to Azure resources without sharing account keys. It allows users to **grant limited permissions** (such as read, write, delete, or list) to specific Azure services like **Storage Accounts, Blob Storage, Queues, Tables, and Files** for a defined period.

So here :

SAS token has various aspects that we can select as shown in the figure( Note: to use SAS token ; we need to storage access key enabled in the storage account) ; and here we can generate the SAS token

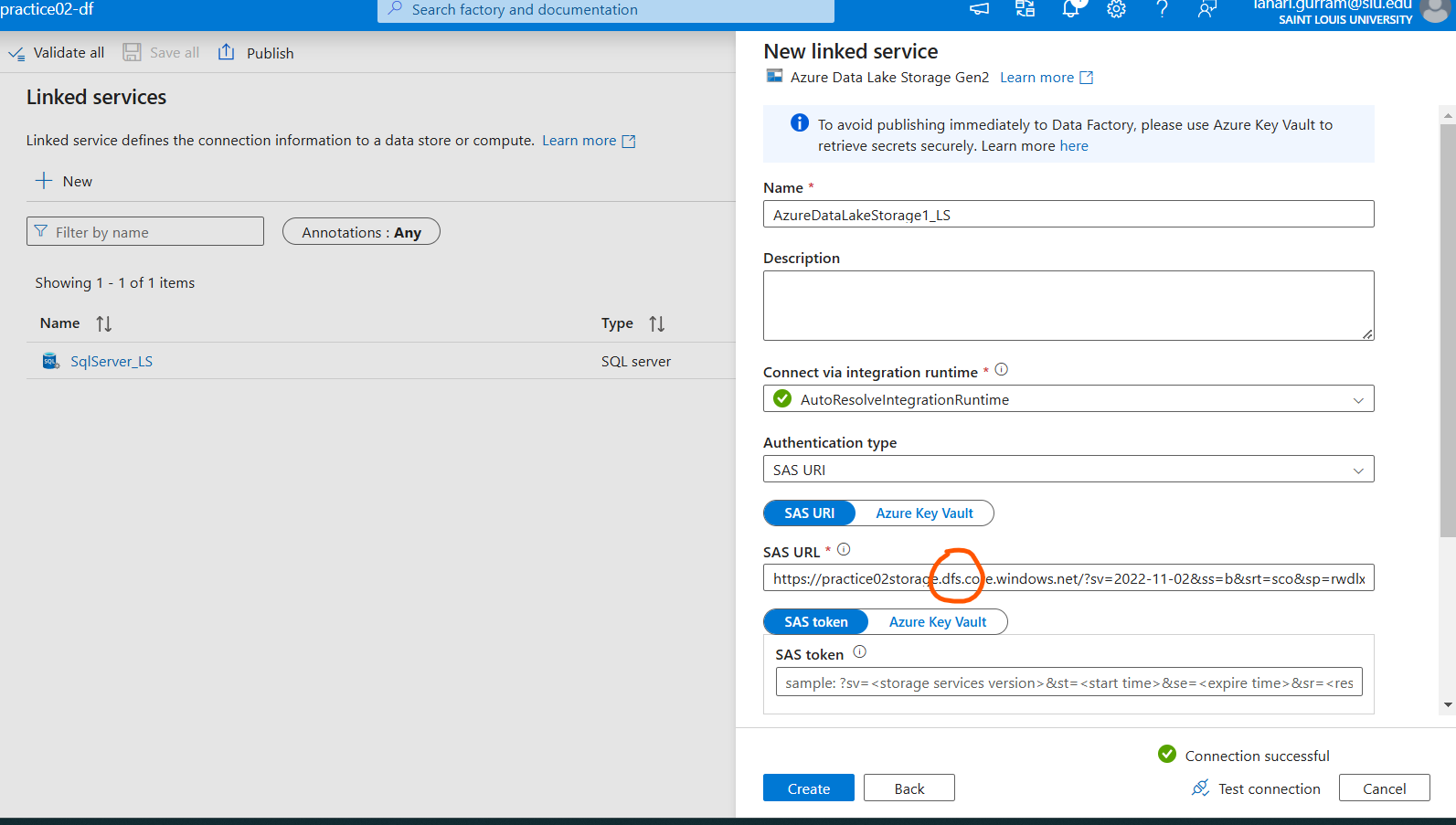


So here to verify permissions within the ADF I am trying to exhibit this scenario:

"I have an existing SQL server with some databases, and I want to test writing data into a storage account using a SAS token in Azure Data Factory. I will enable the write option for the storage account, write the data, then delete it and disable the write option to demonstrate that the SAS token is functioning correctly, ensuring it controls write access to the storage account."

So for this I created a linked service for AZURE SQL DATA BASE WHICH IS USED AS SOURCE FOR MOVING THE DATASET

And for storing data I am using AZURE DATA LAKE STORGAE AND ENABLING THE SAS TOKEN :



Here we have many endpoints but why does we have to use only **dfs**

When selecting an endpoint for your **Azure Data Lake Storage Gen2 (ADLS Gen2) linked service**, your instructor chose:

arduino

CopyEdit

https://practice02storage.dfs.core.windows.net/

Here's why this is the correct choice:

**1. Understanding the Different Endpoints**

Your storage account provides multiple services, each with different endpoints:

* **Blob Service** → https://practice02storage.blob.core.windows.net/
* **File Service** → https://practice02storage.file.core.windows.net/
* **Queue Service** → https://practice02storage.queue.core.windows.net/
* **Table Service** → https://practice02storage.table.core.windows.net/
* **Data Lake Storage (DFS)** → https://practice02storage.dfs.core.windows.net/

**2. What is Data Lake Storage (DFS) Endpoint?**

* **ADLS Gen2** is built on top of the Azure **Blob storage**, but it provides **Hierarchical Namespace (HNS)** support, meaning it works like a file system (like folders and directories).
* The **"dfs.core.windows.net"** endpoint is specifically for **Azure Data Lake Storage Gen2** and supports **file system operations**, such as:
  + Creating directories
  + Moving and renaming files
  + Setting ACLs (Access Control Lists)

**3. Why Not Use the Blob Endpoint (blob.core.windows.net)?**

* The **blob endpoint** (blob.core.windows.net) is used when you interact with **flat blob storage**, meaning it doesn’t support the hierarchical namespace.
* If you use the blob endpoint for **ADLS Gen2**, you might lose some **file system** capabilities like directory structure and ACL-based permissions.

**4. Why Did Your Instructor Choose the DFS Endpoint?**

Your instructor chose:

arduino

CopyEdit

https://practice02storage.dfs.core.windows.net/

**because**:

* It's the correct endpoint for **Data Lake Storage Gen2**.
* It ensures your **linked service** can work with **hierarchical namespaces and file system operations**.
* It allows you to properly manage files and directories inside **ADLS Gen2**.

**✅ Final Answer**

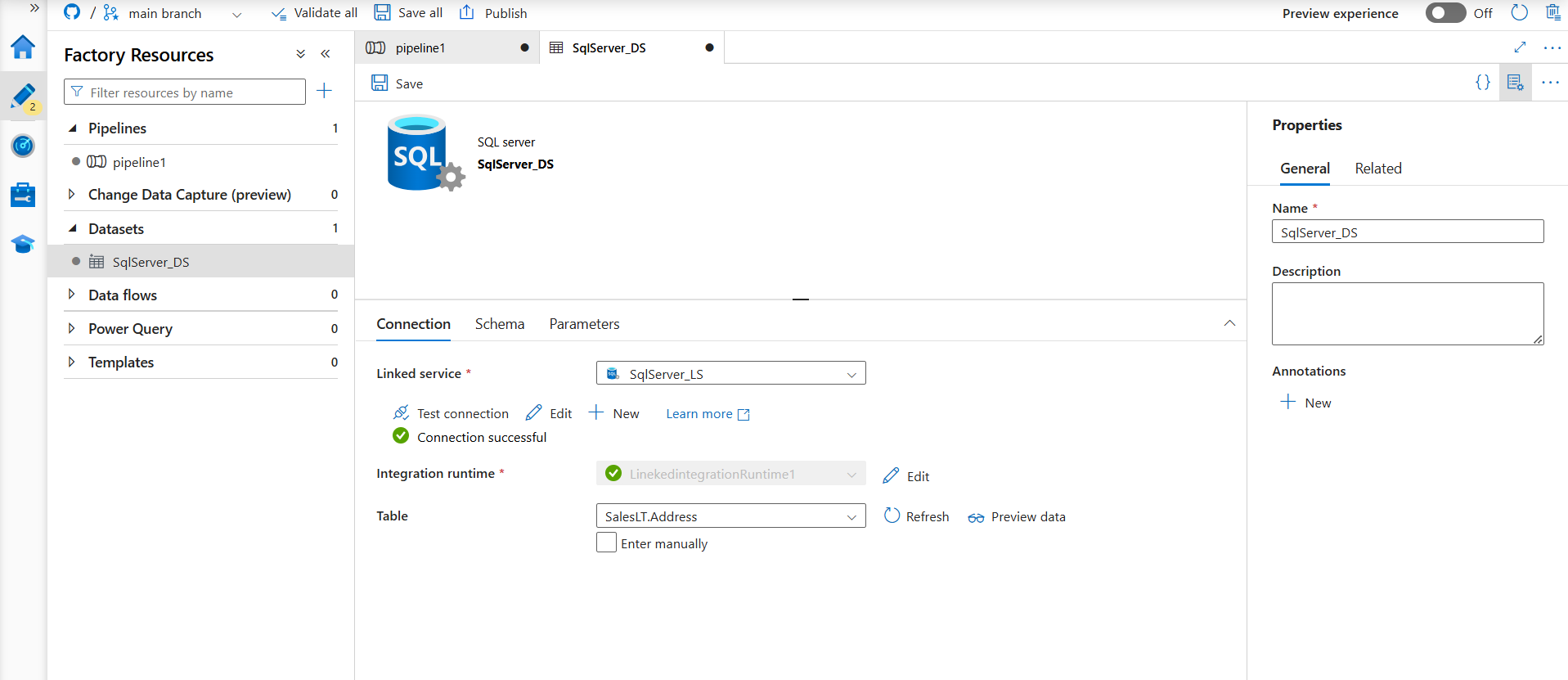
When configuring a **Linked Service for ADLS Gen2**, you should always use:

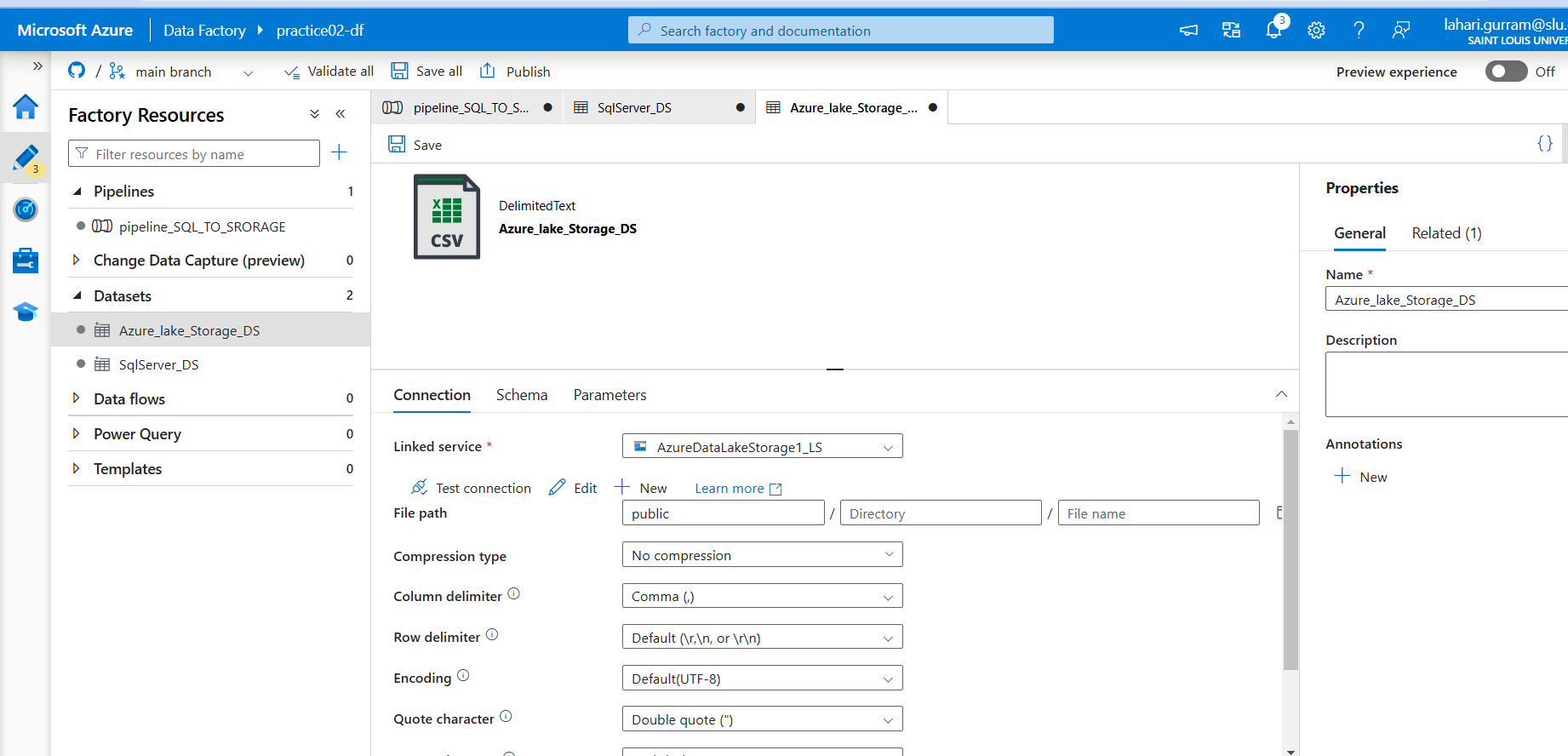
https://<storage-account-name>.dfs.core.windows.net/

This ensures proper **hierarchical namespace support** and allows **full ADLS Gen2 capabilities** in Azure Data Factory.

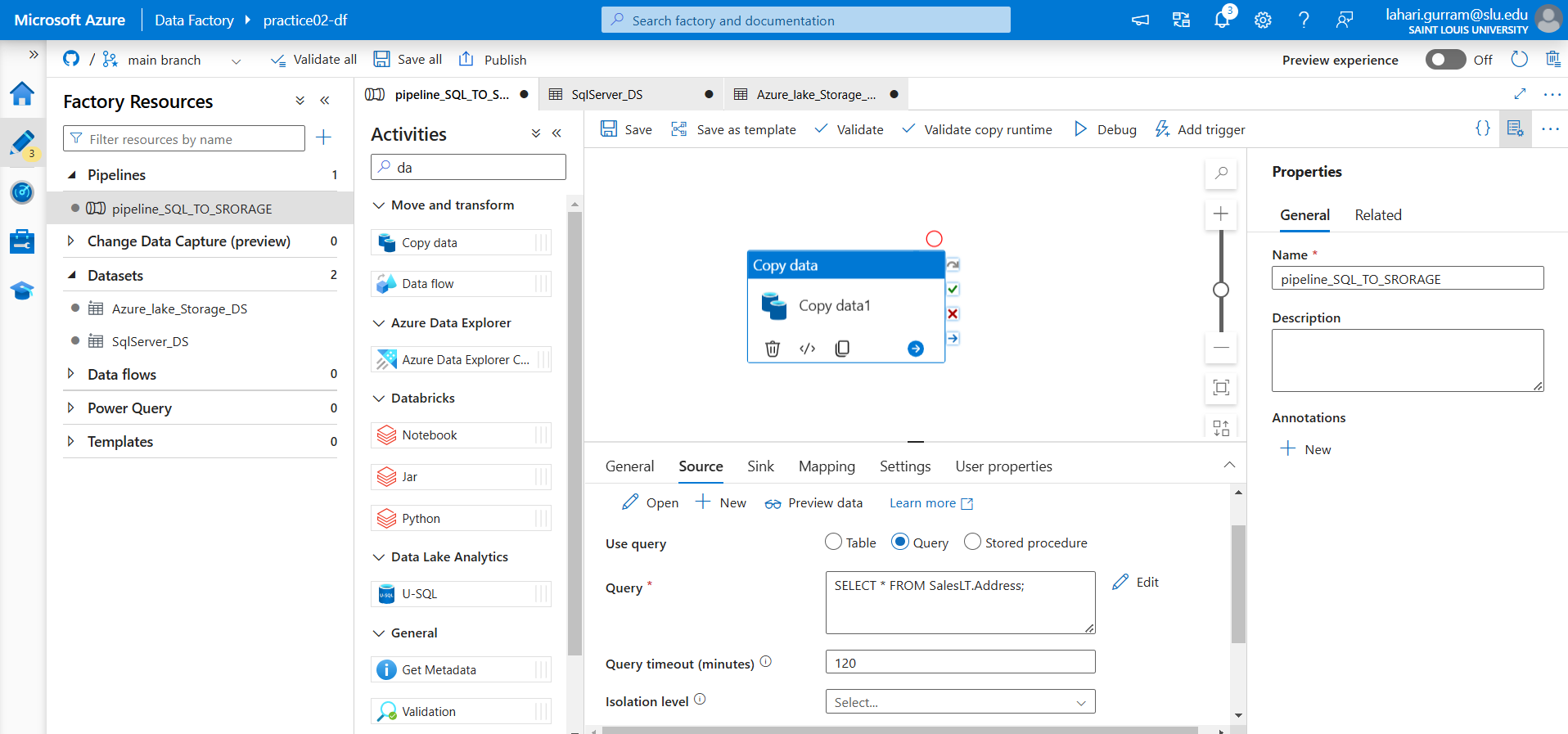
So the connection is successful and I created a linked service for both Database and Datalake

I am creating a pipeline where I am creating two datasets and (not dynamic just to make it simple for now) ; now I am seeing if I can actually write the data in storage account using the SAS( NOW THE WRITE PERMISSION IS ENABLED)

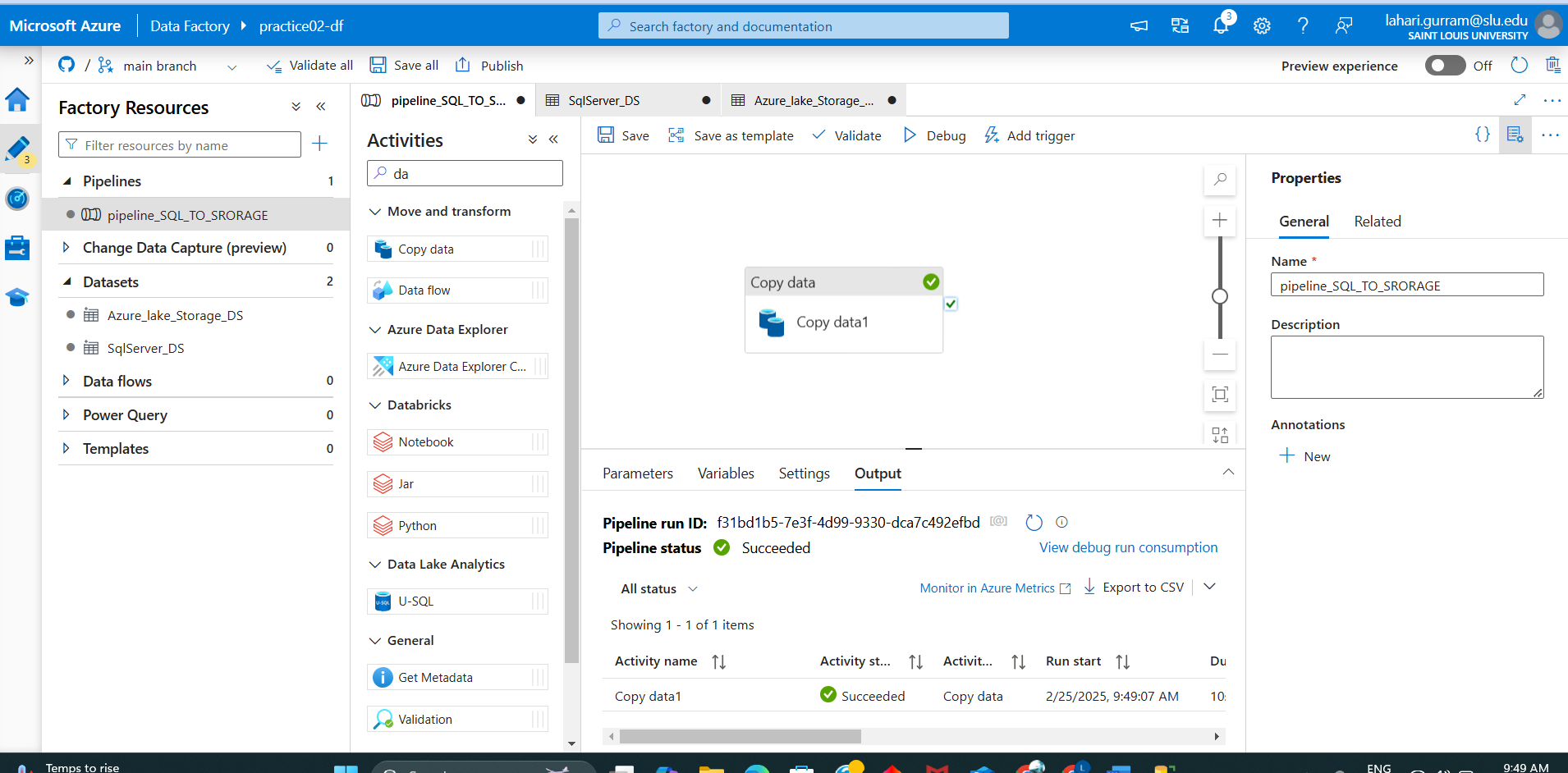




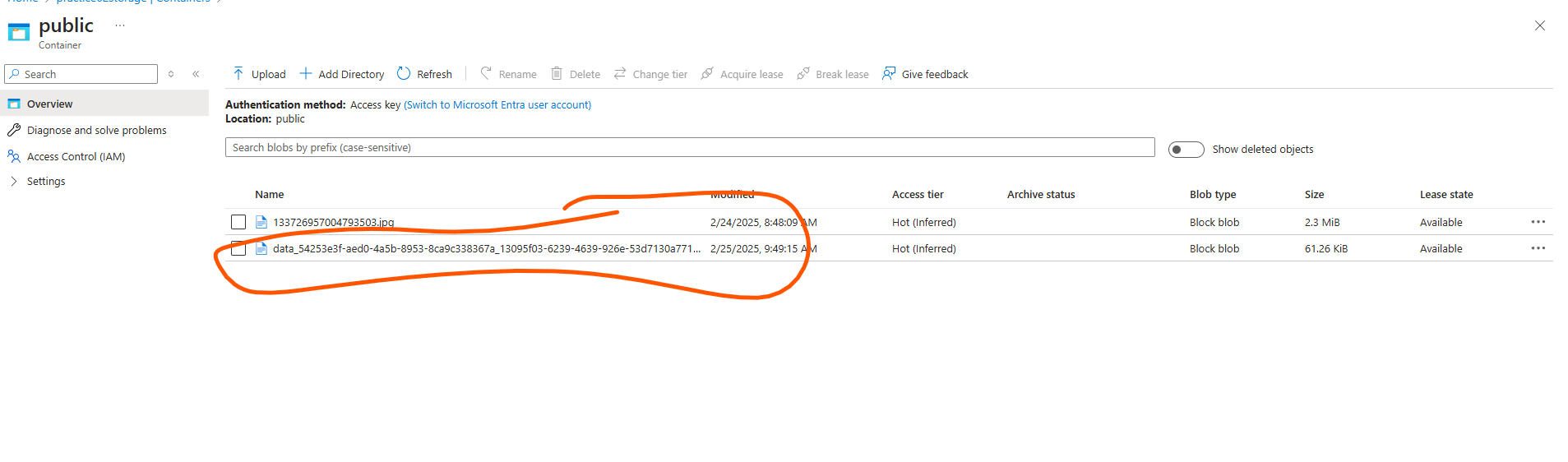
Copy data activity :



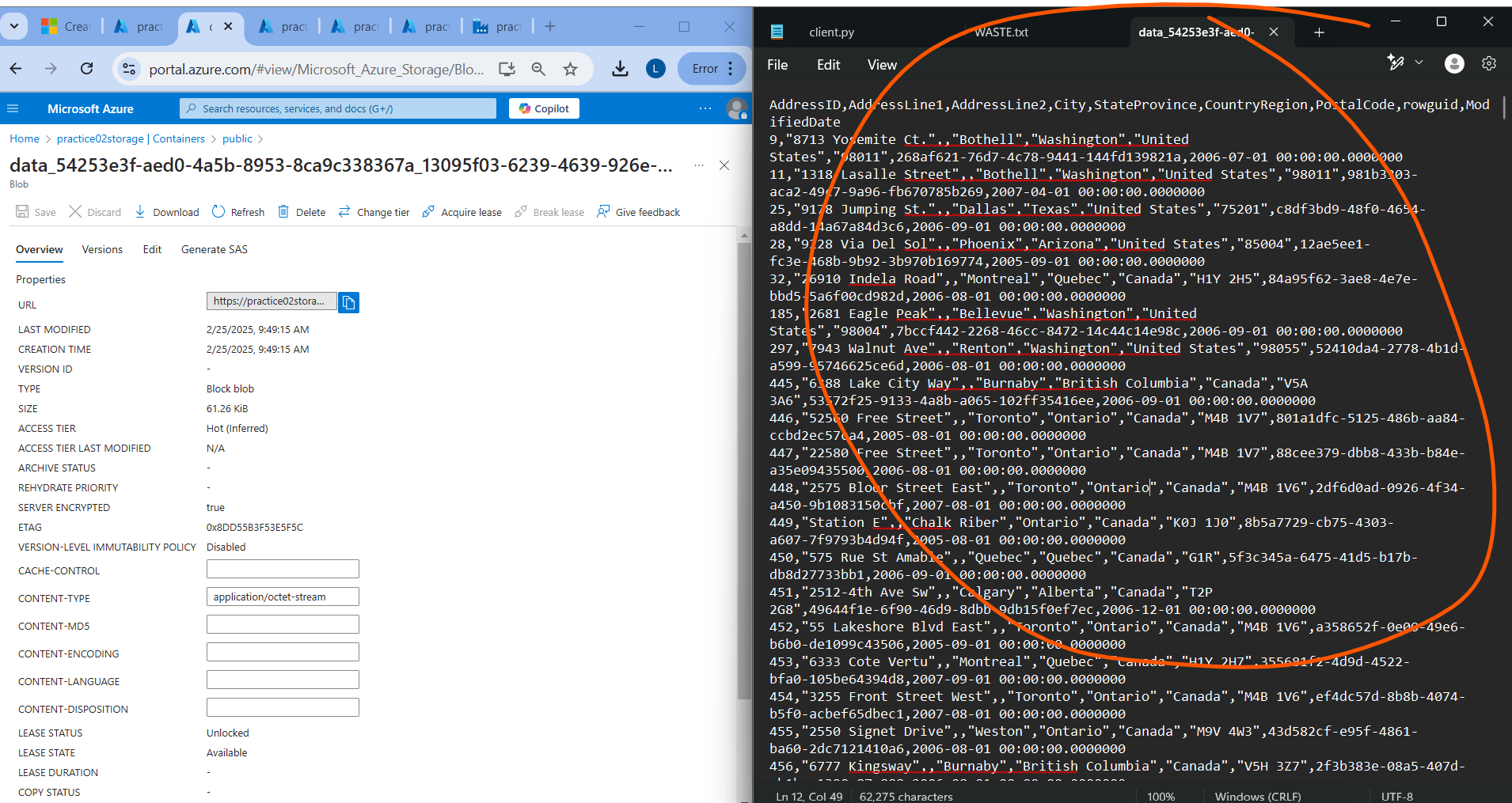
Now the connection is successful and let me check if I can actually copy the data (because the write is enabled)



New file found in ‘Public’ container :



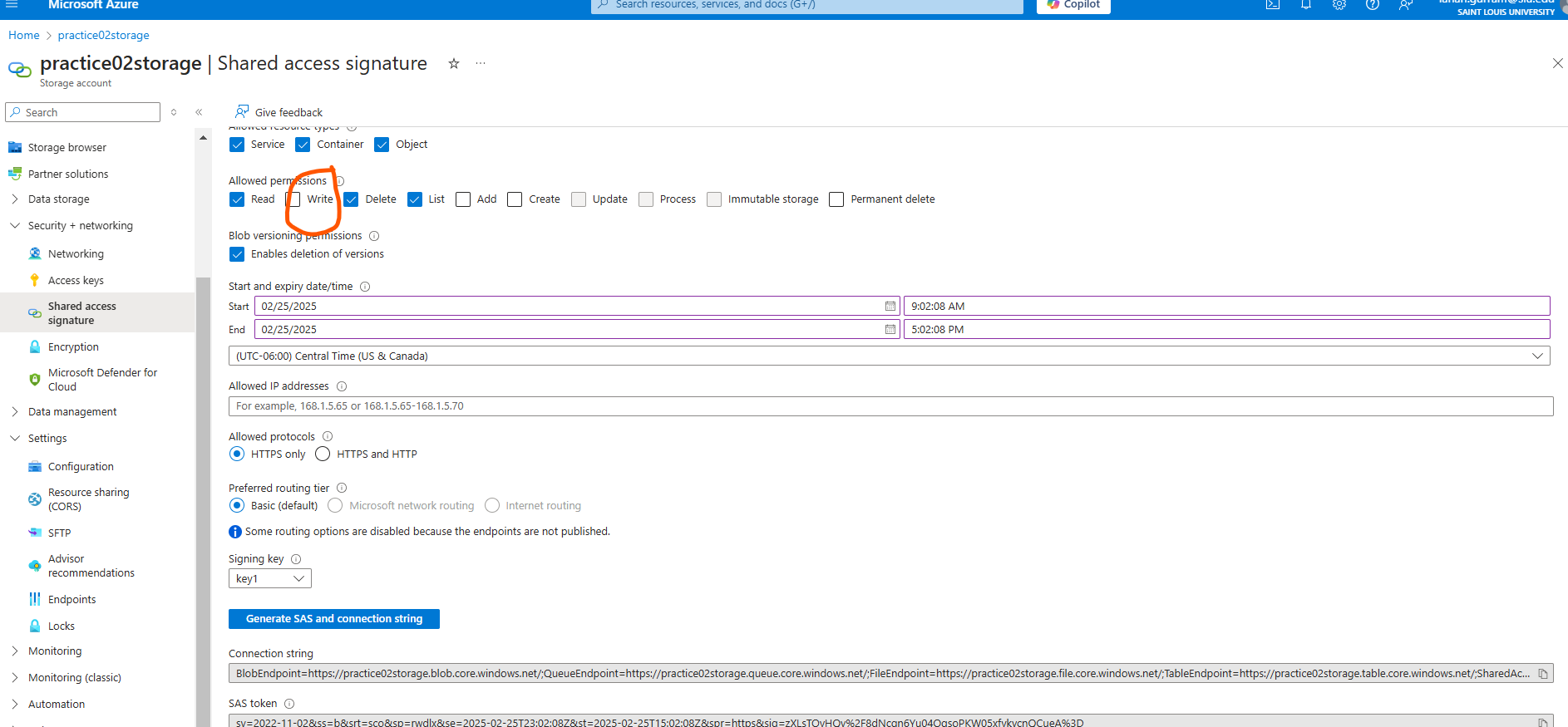
Now I will download it and will see :



The above one is CSV dataset that is downloaded

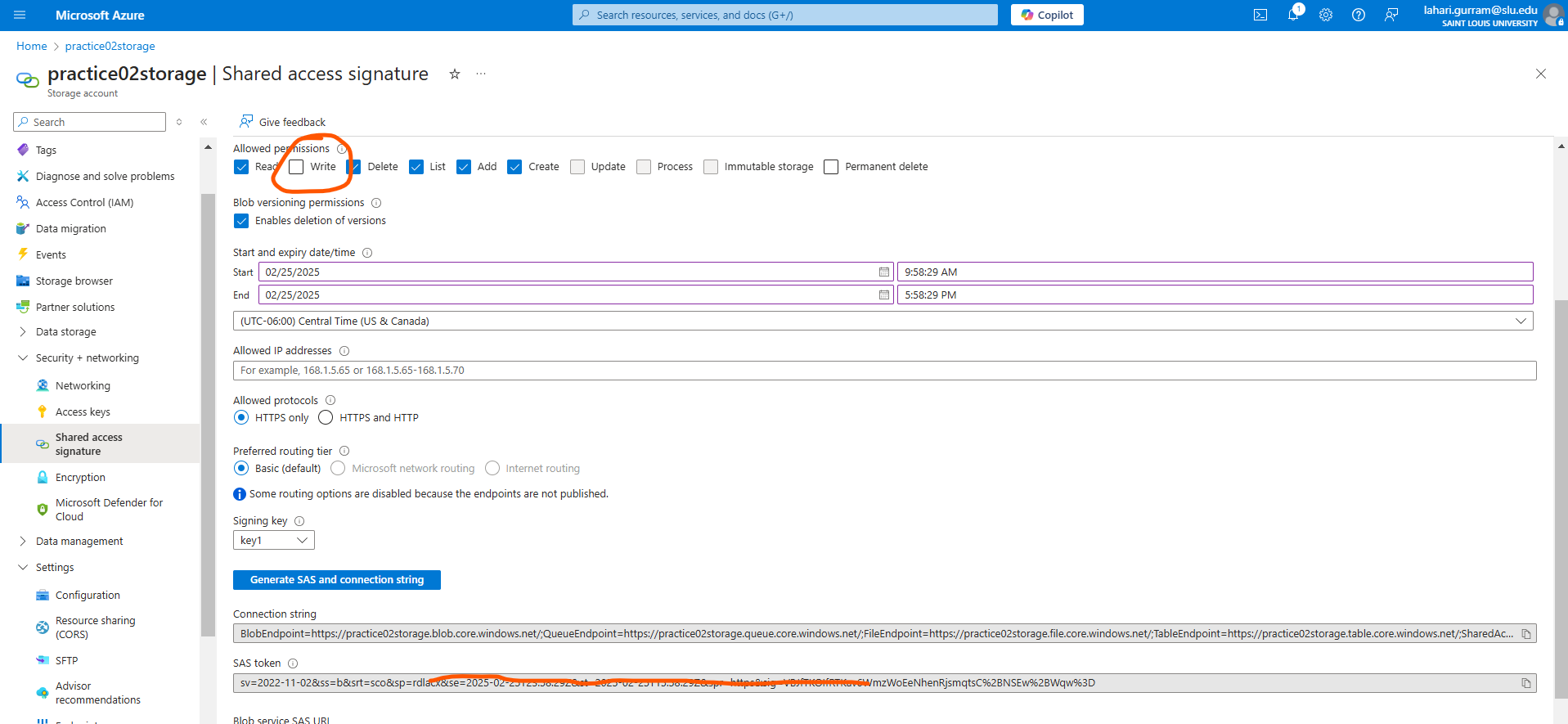
Now I will delete the file inside public and

Now I will disable the WRITE option In SAS and will check the status in the pipline

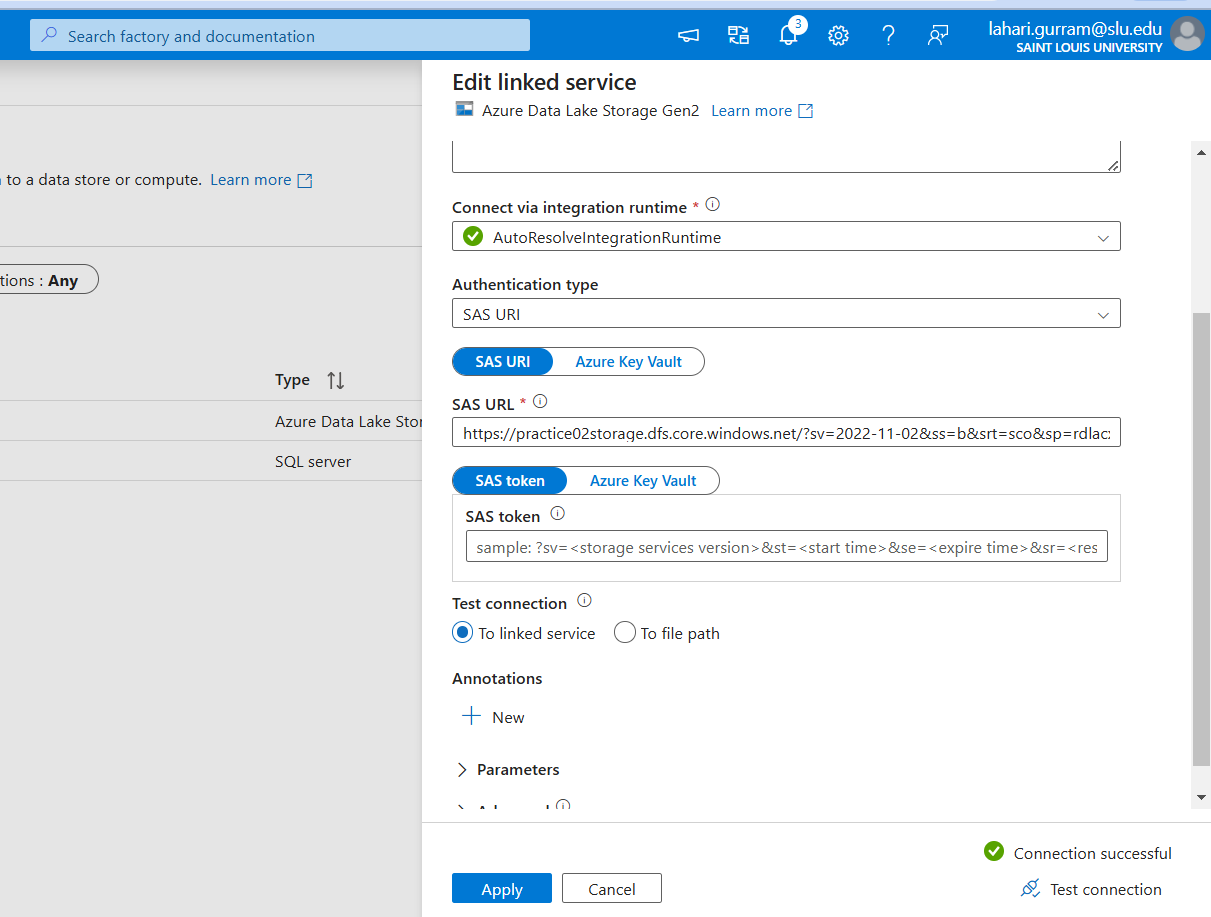


I disabled the WRITE permission and now I will see if the pipline actually works or will it specify any error :

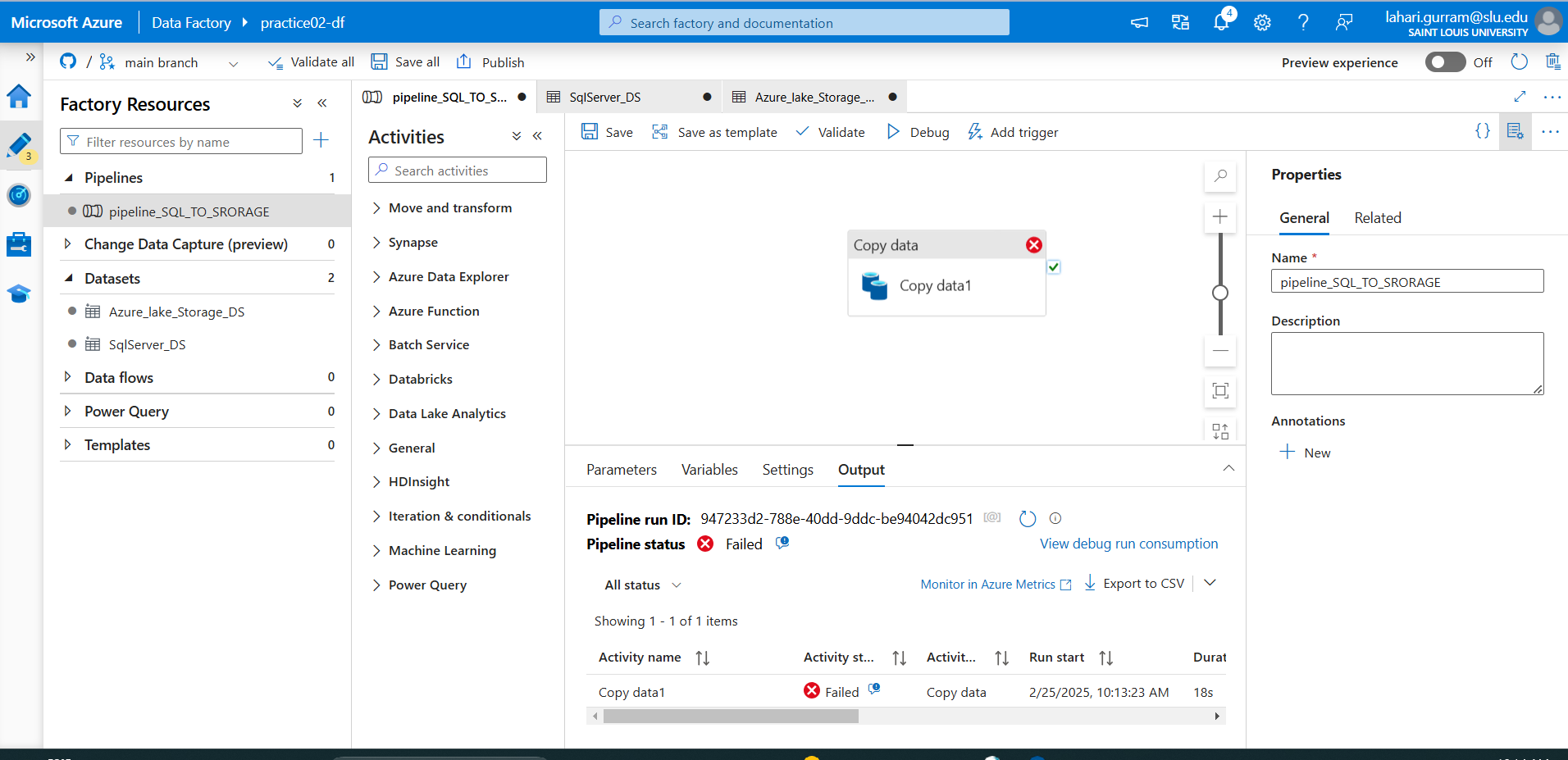
Now I am doing (changing the SAS string and updating it in linked service ) to see changes :



I changed the SAS token WHERE THE WRITE PERMISSION IS DISABLED IN AZURE LAKE STORAGE and updated it in linked sevice:



Now let me check if I can actually see if I can upload the file :



The pipeline is actually failed( which is expected because I disabled write permissions for that SAS ) SECURITY:

ErrorCode=AdlsGen2ForbiddenError,'Type=Microsoft.DataTransfer.Common.Shared.HybridDeliveryException,Message=ADLS Gen2 failed for forbidden: Storage operation 'AcquireLease' on container 'public' and path 'data\_4ed2a0c9-bcce-492c-a770-6ef2c1112dc8\_145f4803-ac19-48b4-bd2c-9679a6cb216e.txt' get failed with 'Operation returned an invalid status code 'Forbidden''. Possible root causes: (1). It's possible because the IP address of the self-hosted integration runtime machines are not allowed by your Azure Storage firewall settings. (2). If the self-hosted integration runtime use proxy server, it's possible because the IP address of the proxy server is not allowed by your Azure Storage firewall settings.. Account: ''. FileSystem: 'public'. Path: 'data\_4ed2a0c9-bcce-492c-a770-6ef2c1112dc8\_145f4803-ac19-48b4-bd2c-9679a6cb216e.txt'. ErrorCode: 'AuthorizationPermissionMismatch'. Message: 'This request is not authorized to perform this operation using this permission.'. RequestId: '505e1244-801f-000e-2da0-87db66000000'. TimeStamp: 'Tue, 25 Feb 2025 16:16:36 GMT'..,Source=Microsoft.DataTransfer.ClientLibrary,''Type=Microsoft.DataTransfer.Common.Shared.HybridDeliveryException,Message=Operation returned an invalid status code 'Forbidden',Source=,''Type=Microsoft.Azure.Storage.Data.Models.ErrorSchemaException,Message=Operation returned an invalid status code 'Forbidden',Source=Microsoft.DataTransfer.ClientLibrary,'

WHICH indicates:

The error you're encountering, AdlsGen2ForbiddenError, indicates that the request to perform an operation (such as acquiring a lease) on the file in **Azure Data Lake Storage Gen2 (ADLS Gen2)** failed because of insufficient permissions or network configuration issues.

Here’s a breakdown of the possible causes and solutions:

**Possible Causes:**

**1. IP Address Not Allowed by Storage Firewall**

**Cause**: The IP address of your **self-hosted integration runtime** (SHIR) or **proxy server** is not allowed by your **Azure Storage firewall**.

**Solution**:

* **Check Firewall Settings**: In your **Storage Account**, go to **Networking** → **Firewalls and virtual networks**.
  + Make sure that your **self-hosted integration runtime’s IP addresses** are included in the allowed list.
  + If you're using a **proxy server**, ensure that the **proxy server’s IP address** is allowed as well.
  + Alternatively, you can enable **trusted Microsoft services** in the firewall settings, which will bypass the IP filtering for services like ADF.
    - Under **Firewalls and virtual networks**, check the box **Allow trusted Microsoft services to access this storage account**.

**2. Authorization Permission Mismatch (SAS or Managed Identity Issues)**

**Cause**: The **permission** associated with the authentication method used (e.g., **SAS Token**, **Managed Identity**, etc.) does not allow the operation you're trying to perform (e.g., acquiring a lease).

**Solution**:

* **Check Permissions**:
  + **If Using SAS Token**: Ensure the SAS Token has **write** or **lease** permissions, as acquiring a lease is a privileged operation.
  + **If Using Managed Identity**: Make sure the **Managed Identity** (whether for ADF or SHIR) has the **necessary permissions** to perform the action on the ADLS Gen2 container.
    - Go to the **Azure Storage Account** → **Access Control (IAM)** → **Add Role Assignment** → Assign the **Storage Blob Data Contributor** role (or appropriate role) to the **Managed Identity** of your ADF or SHIR.

**3. Self-Hosted Integration Runtime Configuration (Proxy Server)**

**Cause**: If your **Self-Hosted Integration Runtime (SHIR)** is behind a **proxy server**, it might not be able to connect to the ADLS Gen2 storage because the proxy's IP is not allowed by the firewall.

**Solution**:

* **Check Proxy Configuration**: Make sure that the **proxy server**’s IP is added to the allowed list in the **Storage Account** firewall.
* **Self-Hosted Integration Runtime**: Ensure that the SHIR is correctly configured to access the Azure storage account and that the firewall settings are not blocking its communication.

**Summary of Actions to Take:**

1. **Verify IP Firewall Settings**:
   * Ensure the **self-hosted integration runtime** or **proxy server IP** is allowed by the storage firewall.
   * If needed, enable **"Allow trusted Microsoft services"** to bypass firewall restrictions for services like ADF.
2. **Check Permissions for SAS or Managed Identity**:
   * Ensure that the **SAS token** used has **write and lease permissions** or that the **Managed Identity** has the correct **role assignments** (e.g., **Storage Blob Data Contributor**).
3. **Review Self-Hosted Integration Runtime (SHIR) Configuration**:
   * If you're using a **proxy server**, make sure the **proxy’s IP** is allowed in the firewall settings.

After performing these checks, try the operation again to see if the error persists.

Let me know if you need further assistance! 😊

So after diabling the write permissions we are unable to copy data from database to the lake storage .