On-Prem to Azure Data Migration Architecture

Architecture Components

1. **On-Premise Virtual Machine (VM):**
   The source data resides on an on-premises virtual machine. This VM has a file system that holds data in various formats like TXT, CSV, and Parquet.

2. **Azure Data Factory (ADF):**
   ADF is used to manage, schedule, and automate the data transfer from the on-prem system to the cloud. It acts as the central orchestrator for the end-to-end migration process.

3. **Azure Data Lake Storage Gen2 (ADLS Gen2):**
   This is the main storage layer in Azure where the data is organized into different zones. The raw zone holds the original data, while cleaned and transformed data goes into the preprocessed and final processed zones.

4. **Azure Synapse Analytics:**
   The final data, after all processing is complete, is loaded into Synapse Analytics. This serves as the central data warehouse for analytics and BI purposes.

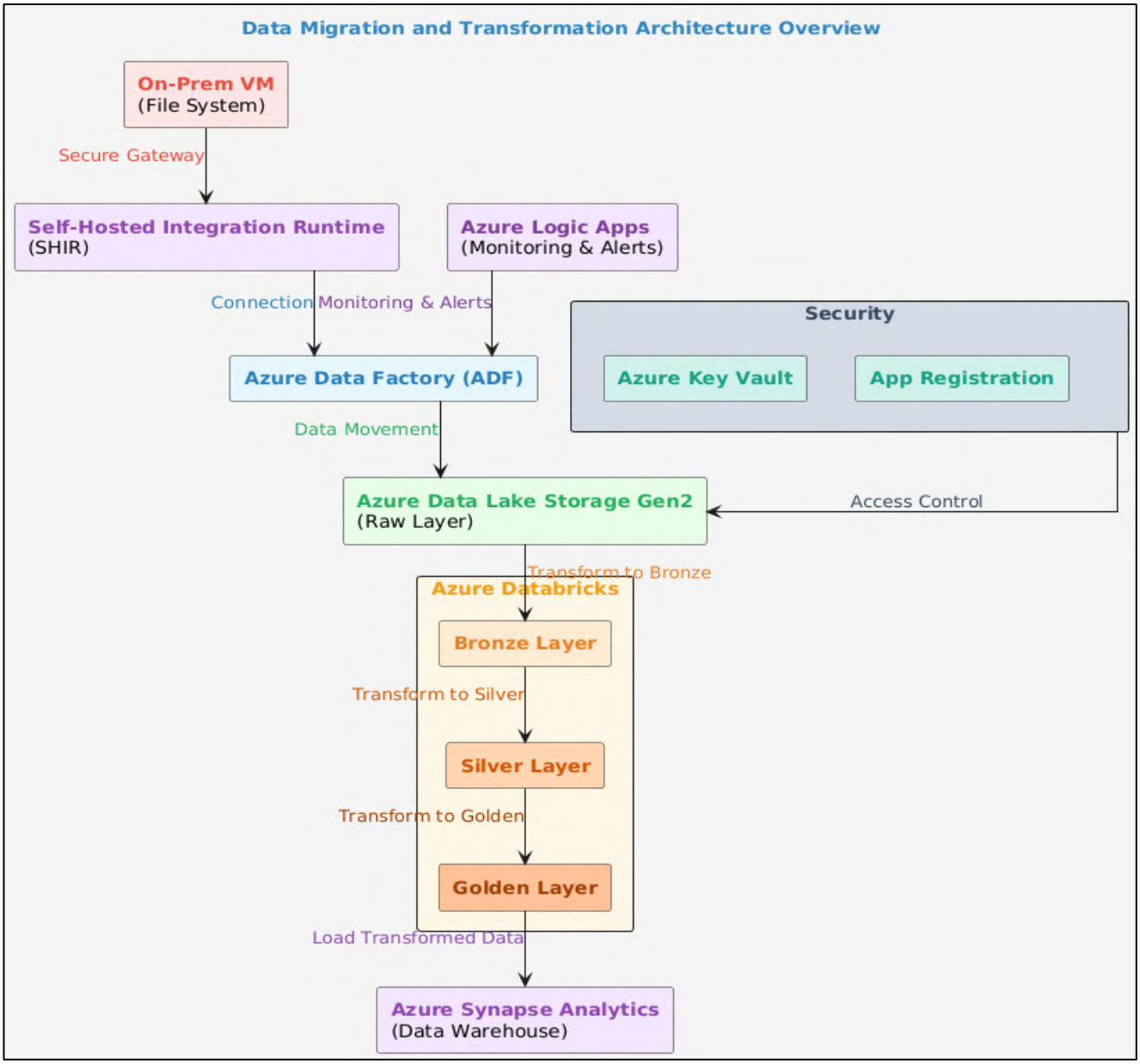
5. **Azure Databricks using PySpark:**
   Azure Databricks is used for building data transformation layers—Bronze, Silver, and Gold.

   - **Bronze Layer:** Cleans up the raw data by removing duplicates, null values, and unwanted records.
   - **Silver Layer:** Applies further transformations like joins, filters, and business logic.
   - **Gold Layer:** Produces fully structured, high-quality datasets that are ready for reporting or machine learning.

6. **Azure App Registration:**
   Used to securely connect Databricks to ADLS Gen2 through mounted access. This ensures secure and consistent connectivity.

7. **Azure Logic Apps:**
   Set up to monitor the data pipelines and send notifications or alerts when jobs fail or succeed.

8. **Azure Key Vault:**
   Secrets, credentials, and other sensitive configurations are stored securely in Key Vault, which is accessed by ADF and Databricks.


Data Migration and Transformation Architecture Overview
On-Prem VM
(File System)
Secure Gateway
Self-Hosted Integration Runtime
(SHIR)
Azure Logic Apps
(Monitoring & Alerts)
Connection
Monitoring & Alerts
Security
Azure Key Vault
App Registration
Azure Data Factory (ADF)
Data Movement
Azure Data Lake Storage Gen2
(Raw Layer)
Access Control
Transform to Bronze
Azure Databricks
Bronze Layer
Transform to Silver
Silver Layer
Transform to Golden
Golden Layer
Load Transformed Data
Azure Synapse Analytics
(Data Warehouse)

High-Level Architecture Flow

1. **On-Prem Data Availability:**
   We begin by preparing a virtual machine on-prem with files (TXT, CSV, Parquet) stored in a local directory.

2. **Connecting On-Prem VM to Azure:**
   In Azure Data Factory, we set up a Self-Hosted Integration Runtime (SHIR). This acts as a secure bridge between Azure and the on-prem system to enable data transfer.

3. **Transferring Data to ADLS Gen2:**
   ADF pipelines use various activities like Copy, Lookup, and Metadata to extract files from the VM and load them into the raw layer of ADLS Gen2.

4. **Processing with Azure Databricks:**
   In Databricks, PySpark scripts handle the transformation across three stages:

   - **Bronze:** Initial cleaning like removing nulls and duplicates.
   - **Silver:** Applying joins and business logic to transform the data.
   - **Gold:** Creating final datasets ready for consumption and analytics.

5. **Loading into Synapse Analytics:**
   The transformed data from the Gold layer is loaded into Synapse SQL Data Warehouse where analysts and BI tools can run queries.

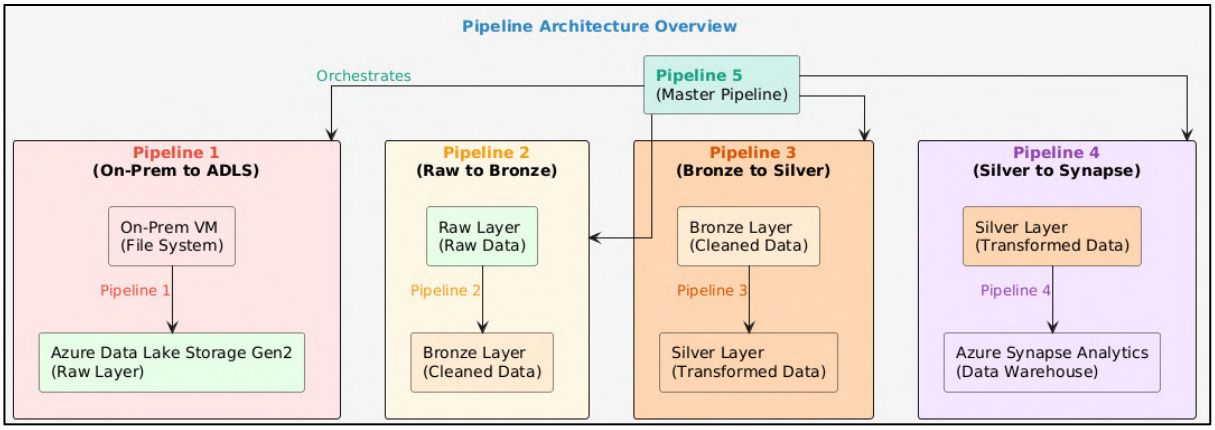
6. **Security Layer:**

   - App Registration enables token-based secure access to storage.
   - Key Vault holds credentials, which are accessed by ADF and Databricks as needed.

7. **Monitoring and Notifications:**
   Logic Apps are used to set up alert mechanisms that notify the team when a pipeline fails or completes successfully.

Pipeline Structure

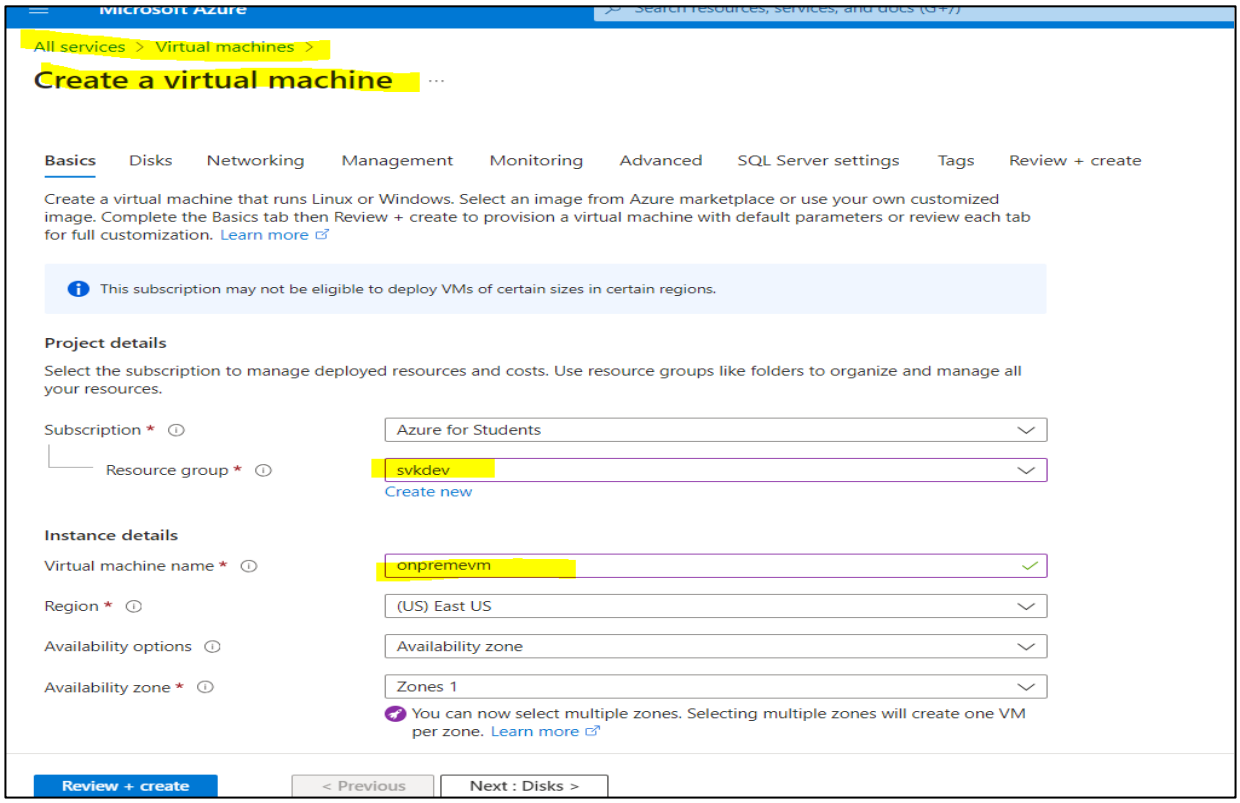
- **Pipeline 1:** Moves data from the on-prem VM to ADLS (raw layer).
- **Pipeline 2:** Transforms raw data to the Bronze layer in ADLS.
- **Pipeline 3:** Converts Bronze data to the Silver layer.
- **Pipeline 4:** Moves the refined Silver data into Synapse SQL Data Warehouse.
- **Pipeline 5:** A master pipeline that orchestrates the above pipelines in sequence.

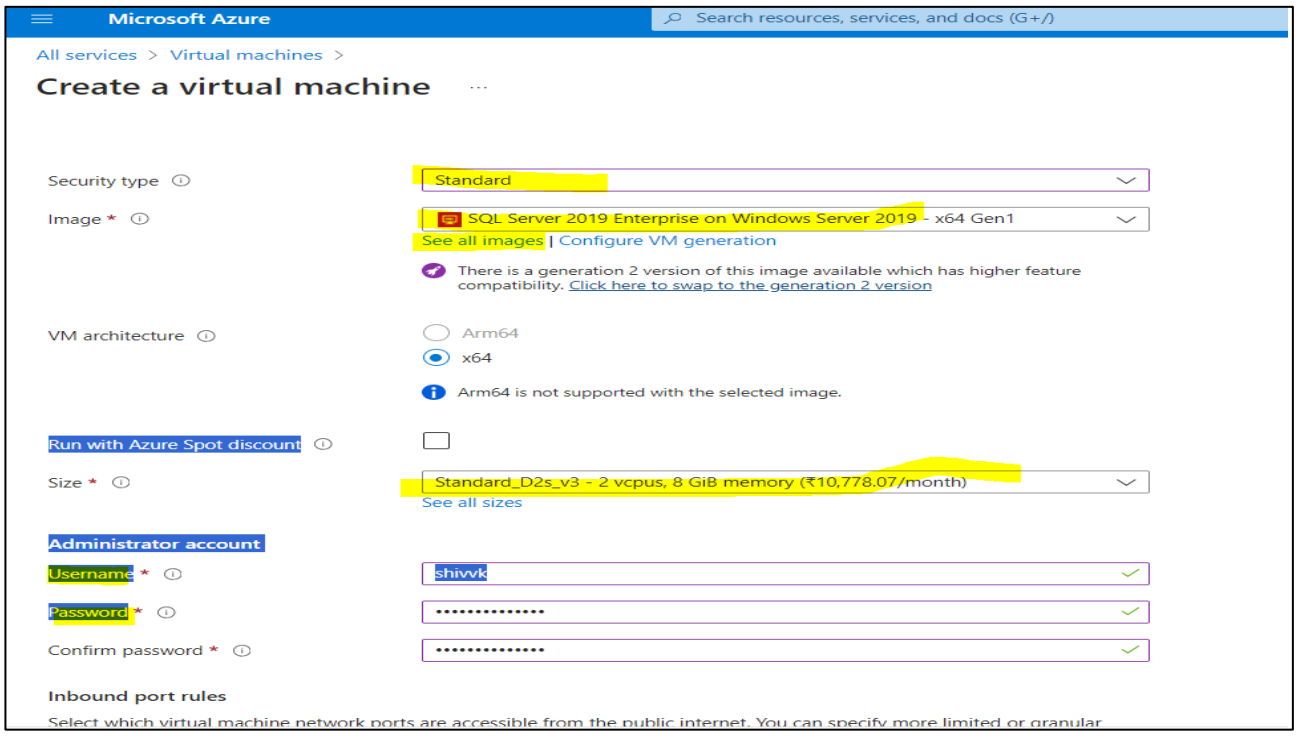


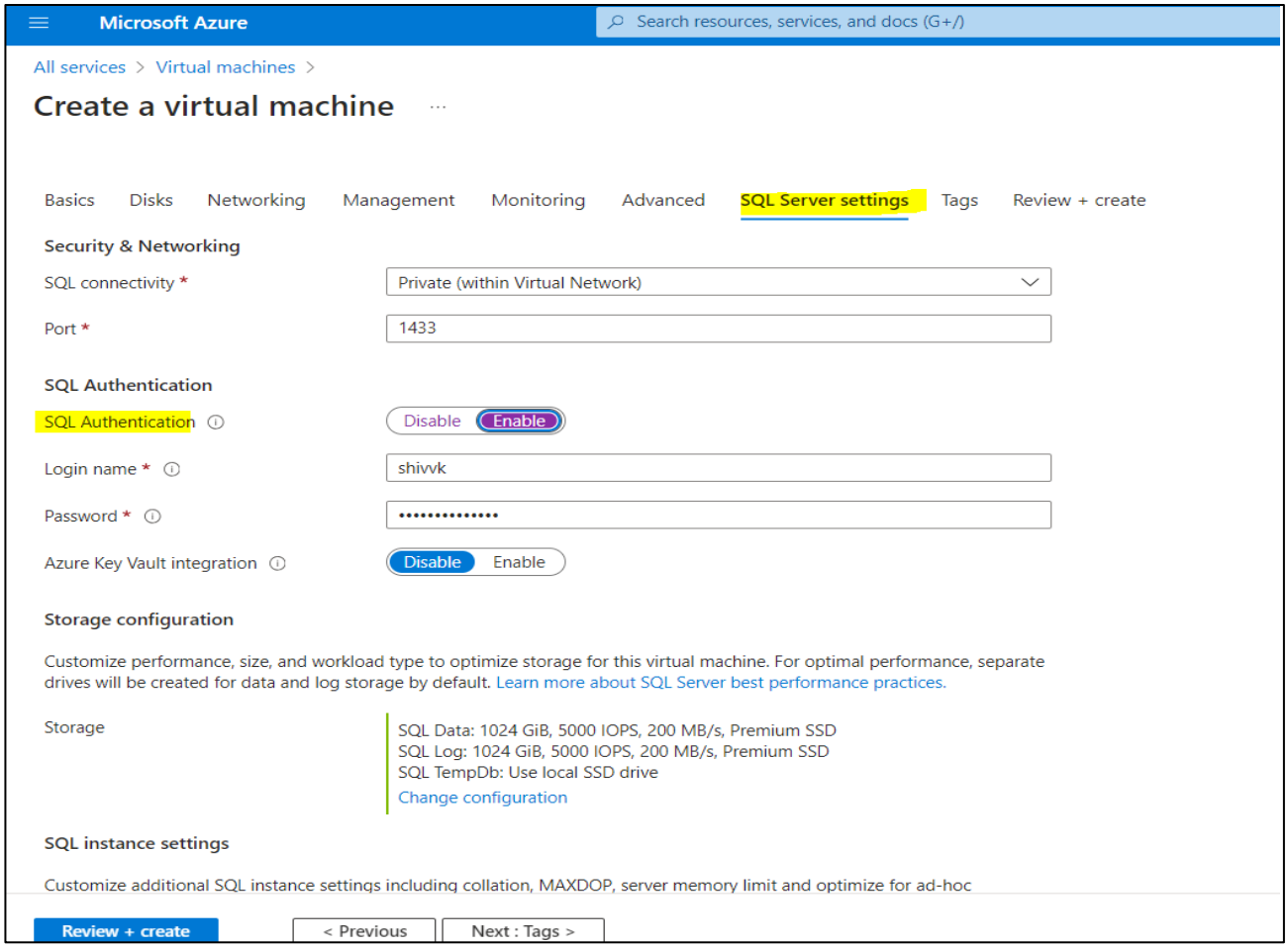
Step-by-Step Setup Instructions

Step 1 - Create the On-Prem VM:

Choose an image with SQL Server, then proceed through the VM creation wizard using the default options. Make sure to enable SQL Authentication.

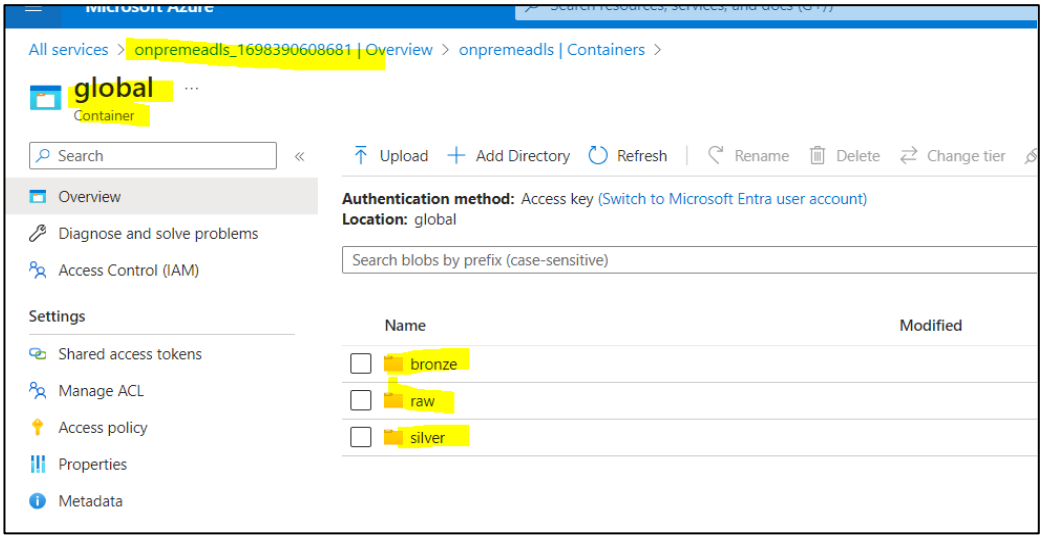





Microsoft Azure
Search resources, services, and docs (G+/)
All services > Virtual machines >
Create a virtual machine
Basics Disks Networking Management Monitoring Advanced SQL Server settings Tags Review + create
Security & Networking
SQL connectivity *
Private (within Virtual Network)
Port *
1433
SQL Authentication
SQL Authentication
Disable Enable
Login name *
shivvk
Password *
Azure Key Vault integration
Disable Enable
Storage configuration
Customize performance, size, and workload type to optimize storage for this virtual machine. For optimal performance, separate drives will be created for data and log storage by default. Learn more about SQL Server best performance practices.
Storage
SQL Data: 1024 GiB, 5000 IOPS, 200 MB/s, Premium SSD
SQL Log: 1024 GiB, 5000 IOPS, 200 MB/s, Premium SSD
SQL TempDb: Use local SSD drive
Change configuration
SQL instance settings
Customize additional SQL instance settings including collation, MAXDOP, server memory limit and optimize for ad-hoc
Review + create
< Previous
Next : Tags >

Step 2 - Set Up Azure Data Factory:

Create an ADF instance from the portal.

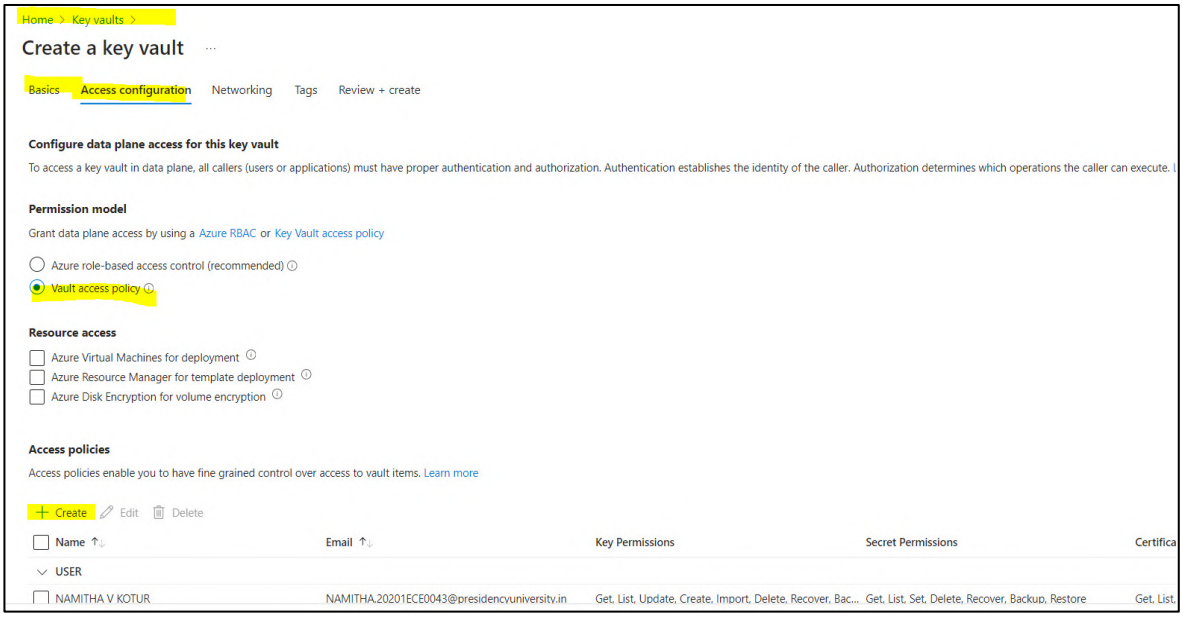


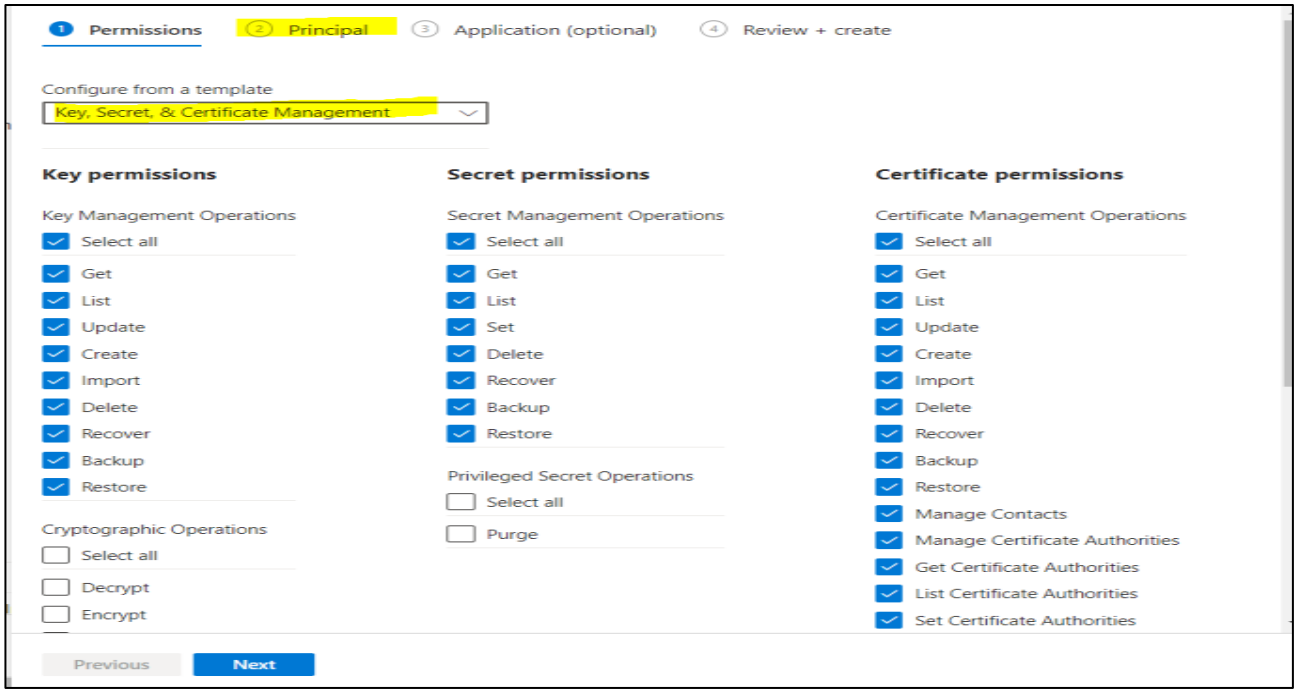
Step 3 - Provision ADLS Gen2:

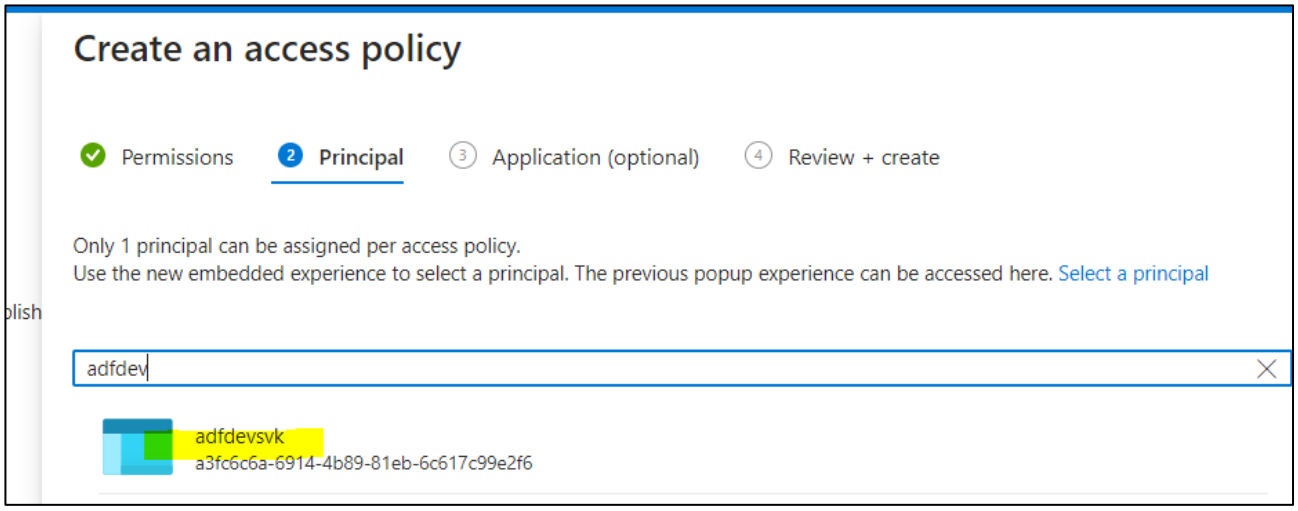
- Create a storage account.
- Add a container named global.
- Within that container, create folders for raw, bronze, and silver data.

Step 4 - Create a Key Vault:

- Assign access to ADF in the Key Vault's Access Policies section so it can retrieve secrets.
- Complete the setup by reviewing and creating the Key Vault.

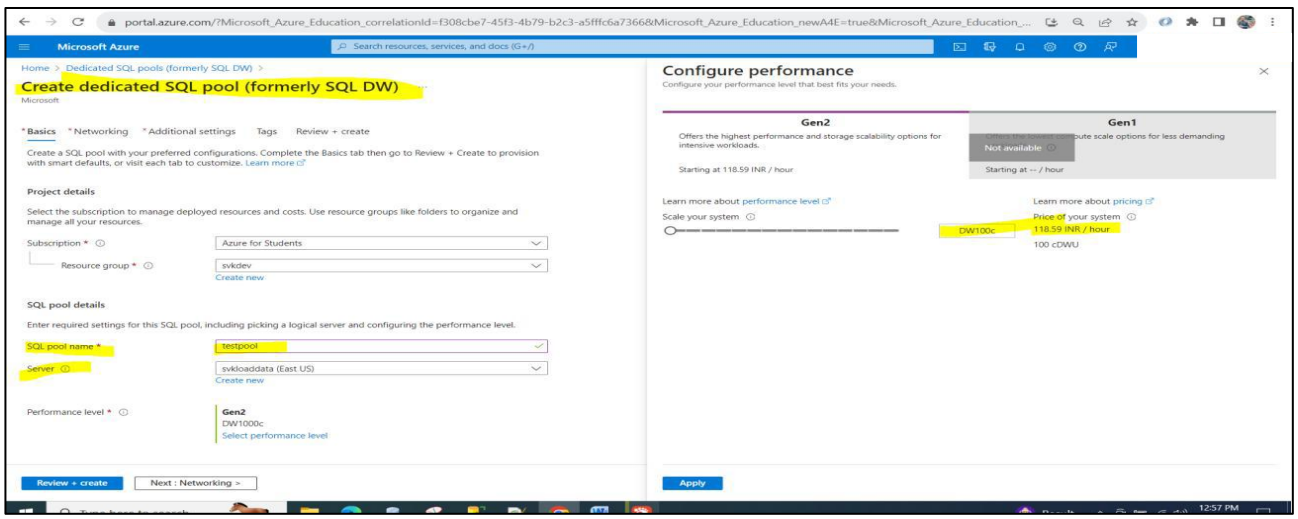






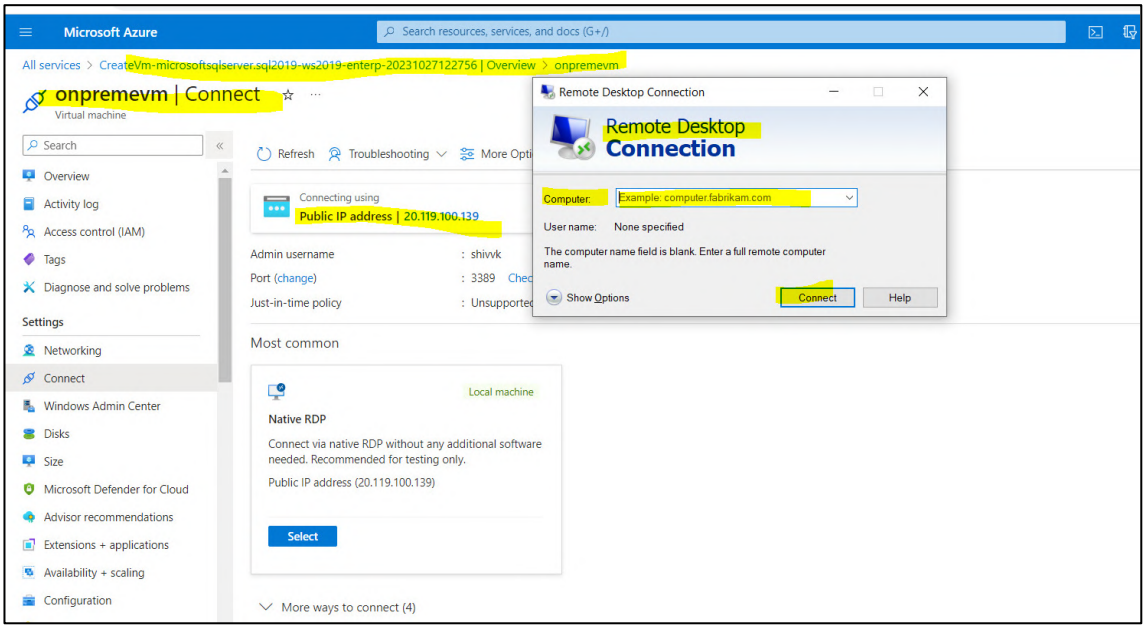
Step 5 - Deploy Dedicated SQL Pool (Synapse):

Create a Synapse SQL pool that will later serve as the target data warehouse.



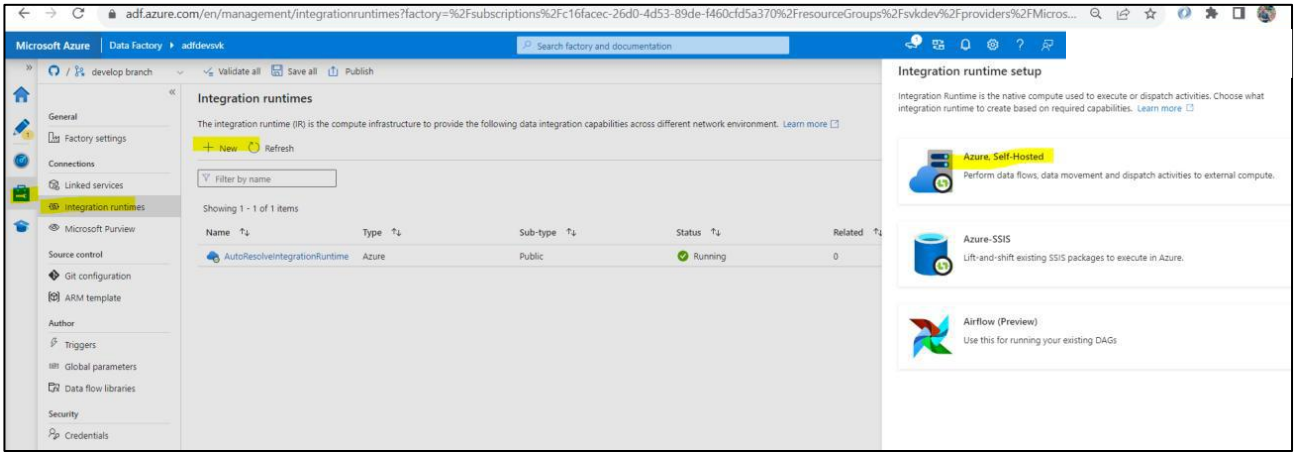
Step 6 - Remote into the VM:

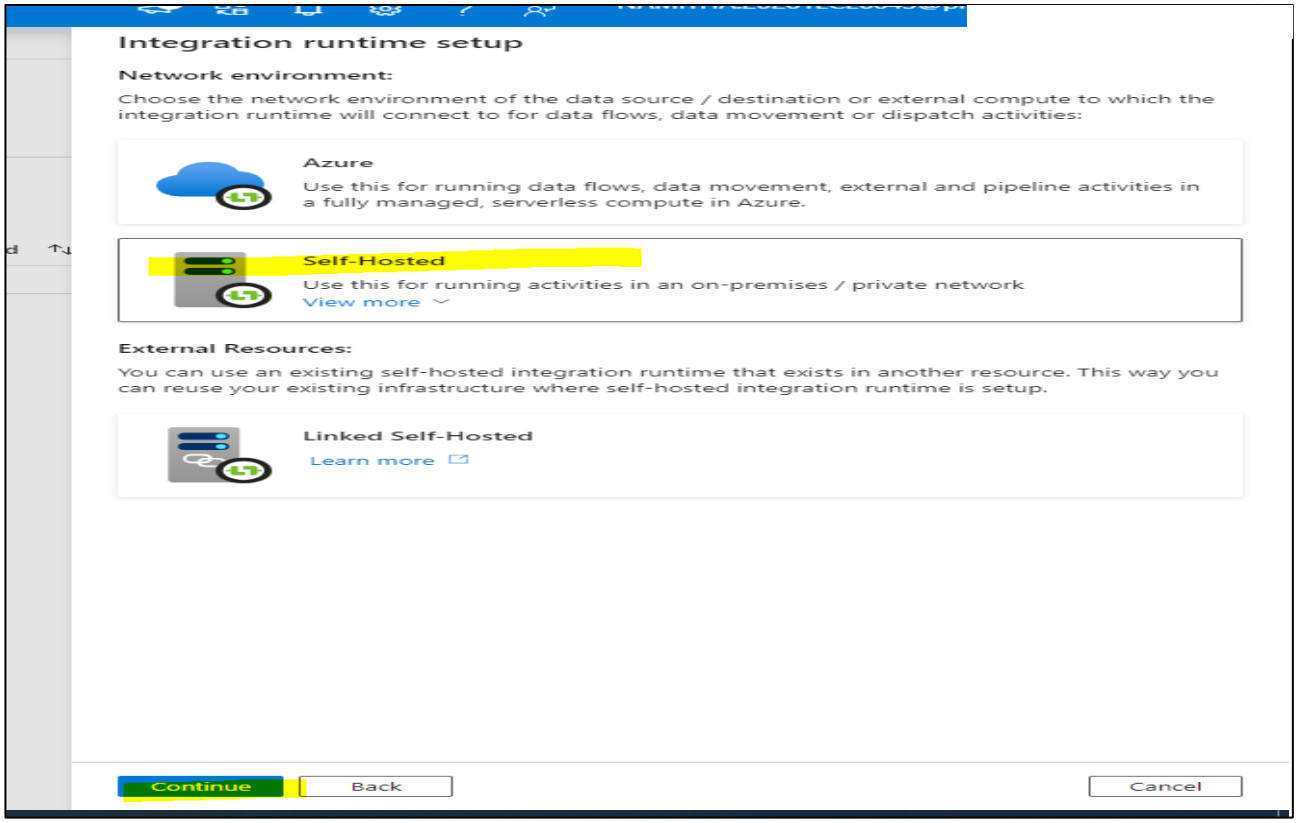
Use RDP to connect to the VM using its public IP and login credentials.



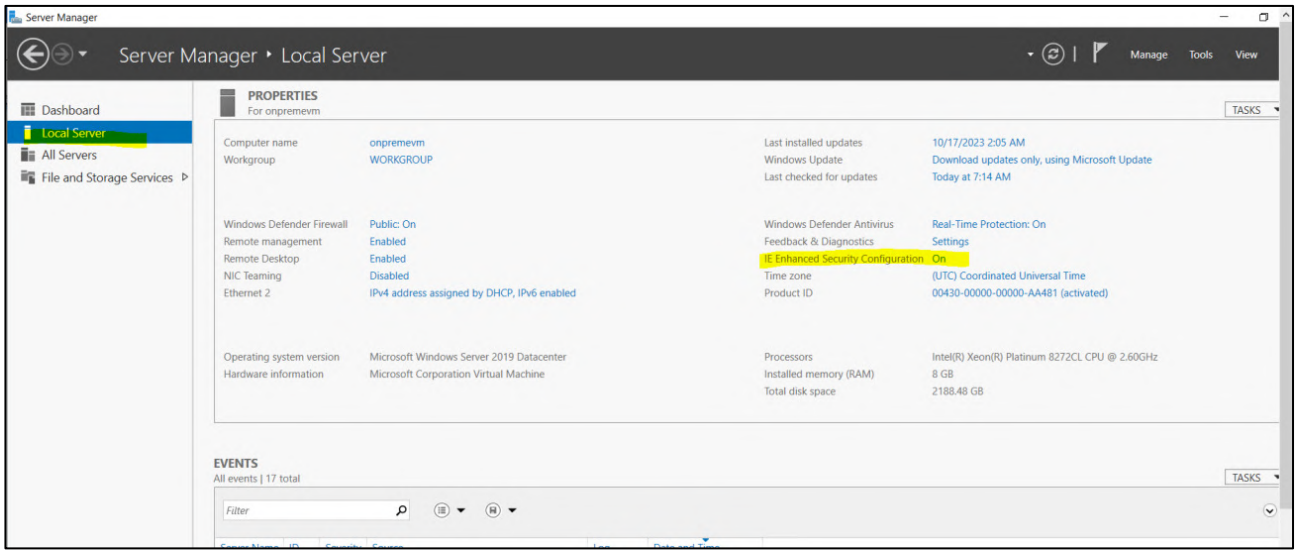
Step 7 -Install Self-Hosted Integration Runtime (SHIR):

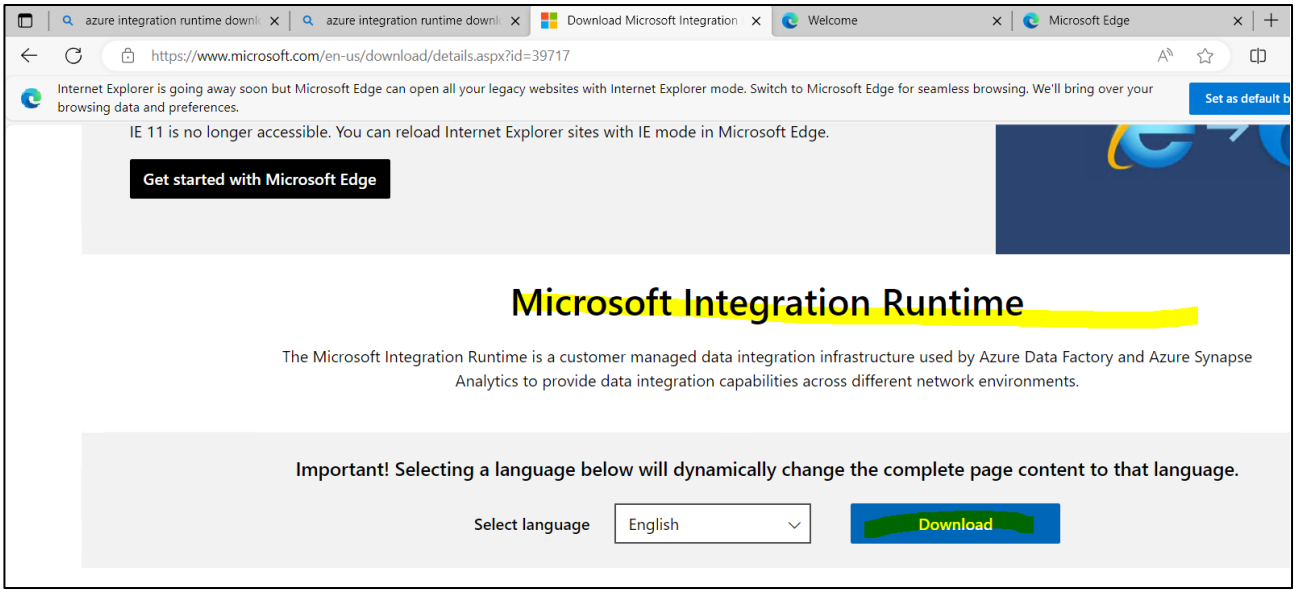
- In ADF, create a new SHIR.
- Download the integration runtime installer.
- Copy the registration key.
- Move to the VM and disable Enhanced Security in Internet Explorer.
- Install the integration runtime on the VM and register it using the copied key.

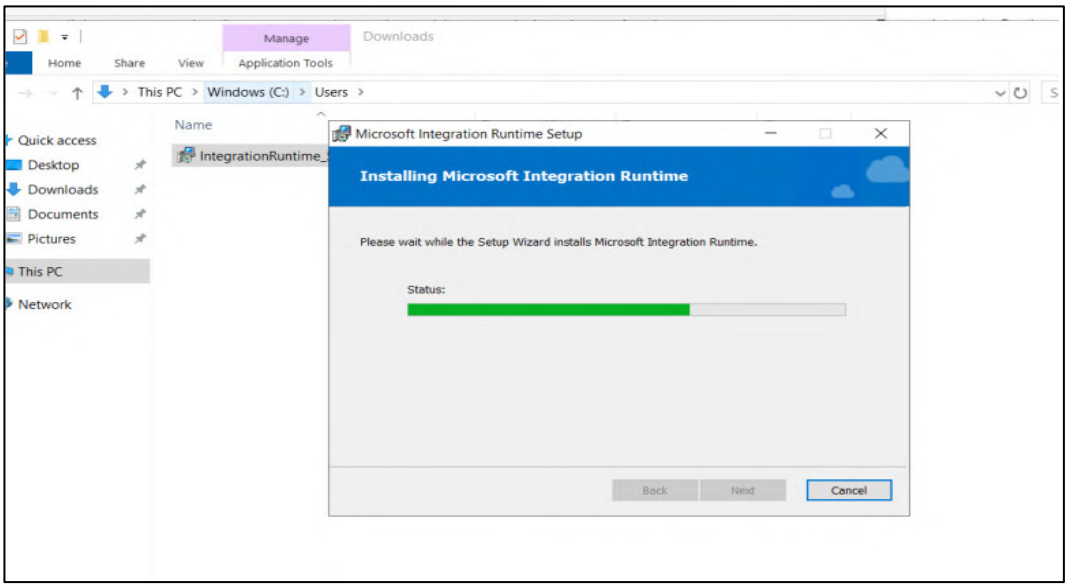


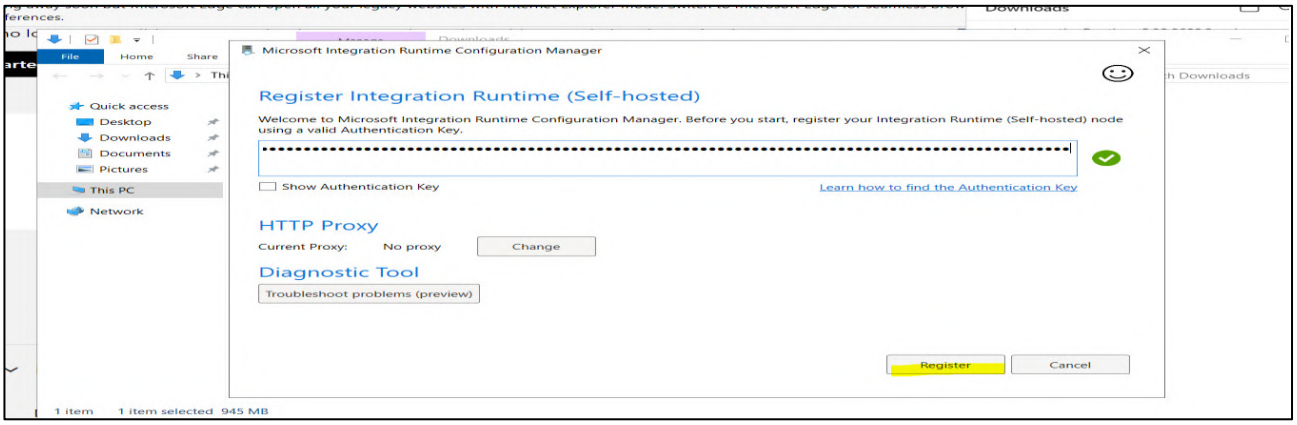

Integration runtime setup
Network environment:
Choose the network environment of the data source / destination or external compute to which the integration runtime will connect to for data flows, data movement or dispatch activities:
Azure
Use this for running data flows, data movement, external and pipeline activities in a fully managed, serverless compute in Azure.
Self-Hosted
Use this for running activities in an on-premises / private network
View more
External Resources:
You can use an existing self-hosted integration runtime that exists in another resource. This way you can reuse your existing infrastructure where self-hosted integration runtime is setup.
Linked Self-Hosted
Learn more
Continue
Back
Cancel


Integration runtime setup
Settings
Nodes
Auto update
Sharing
Links
Install integration runtime on Windows machine or add further nodes using the Authentication Key.
Name
SelfhostedIR
Option 1: Express setup
Click here to launch the express setup for this computer
Option 2: Manual setup
Step 1: Download and install integration runtime
Step 2: Use this key to register your integration runtime
Name
Authentication key
Key1
IR@fcfaac77-4cc0-428f-b6c2-d5c36af36cfa@adfdevsvk@ServiceEndpoin
Key2
IR@fcfaac77-4cc0-428f-b6c2-d5c36af36cfa@adfdevsvk@ServiceEndpoin
Close


Server Manager
Server Manager • Local Server
Manage
Tools
View
Dashboard
Local Server
All Servers
File and Storage Services
PROPERTIES
For onpremevm
TASKS
Computer name
onpremevm
Workgroup
WORKGROUP
Last installed updates
10/17/2023 2:05 AM
Windows Update
Download updates only, using Microsoft Update
Last checked for updates
Today at 7:14 AM
Windows Defender Firewall
Public: On
Remote management
Enabled
Remote Desktop
Enabled
NIC Teaming
Disabled
Ethernet 2
IPv4 address assigned by DHCP, IPv6 enabled
Windows Defender Antivirus
Real-Time Protection: On
Feedback & Diagnostics
Settings
IE Enhanced Security Configuration
On
Time zone
(UTC) Coordinated Universal Time
Product ID
00430-00000-00000-AA481 (activated)
Operating system version
Microsoft Windows Server 2019 Datacenter
Hardware information
Microsoft Corporation Virtual Machine
Processors
Intel(R) Xeon(R) Platinum 8272CL CPU @ 2.60GHz
Installed memory (RAM)
8 GB
Total disk space
2188.48 GB
EVENTS
All events | 17 total
TASKS
Filter

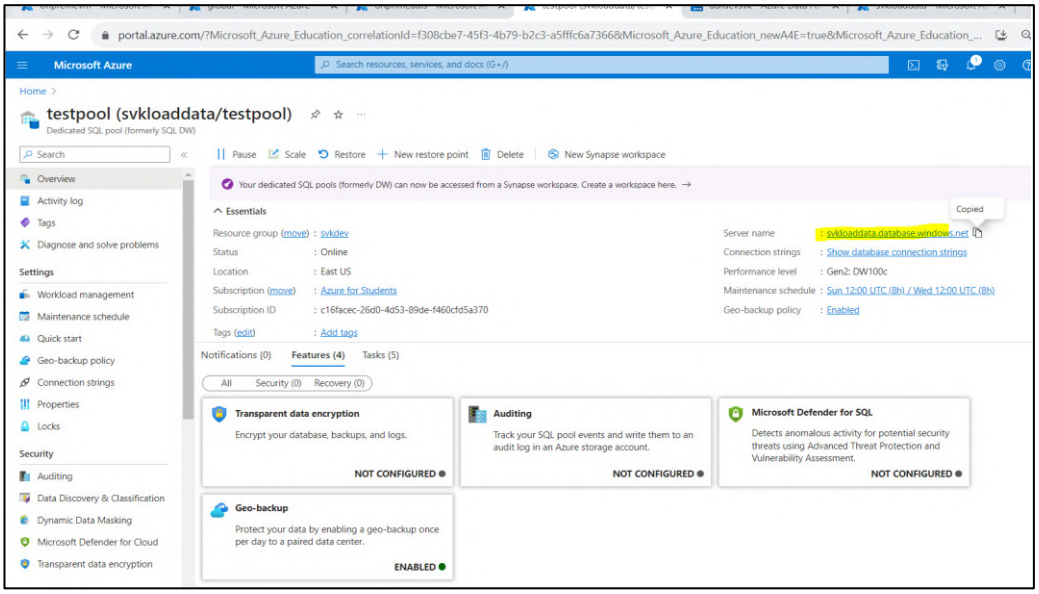

azure integration runtime downl
azure integration runtime downl
Download Microsoft Integration
Welcome
Microsoft Edge
https://www.microsoft.com/en-us/download/details.aspx?id=39717
Internet Explorer is going away soon but Microsoft Edge can open all your legacy websites with Internet Explorer mode. Switch to Microsoft Edge for seamless browsing. We'll bring over your browsing data and preferences.
Set as default b
IE 11 is no longer accessible. You can reload Internet Explorer sites with IE mode in Microsoft Edge.
Get started with Microsoft Edge
Microsoft Integration Runtime
The Microsoft Integration Runtime is a customer managed data integration infrastructure used by Azure Data Factory and Azure Synapse Analytics to provide data integration capabilities across different network environments.
Important! Selecting a language below will dynamically change the complete page content to that language.
Select language
English
Download


Manage
Downloads
Home
Share
View
Application Tools
This PC
Windows (C:)
Users
Quick access
Desktop
Downloads
Documents
Pictures
This PC
Network
Name
IntegrationRuntime_
Microsoft Integration Runtime Setup
Installing Microsoft Integration Runtime
Please wait while the Setup Wizard installs Microsoft Integration Runtime.
Status:
Back
Next
Cancel


Microsoft Integration Runtime Configuration Manager
Register Integration Runtime (Self-hosted)
Welcome to Microsoft Integration Runtime Configuration Manager. Before you start, register your Integration Runtime (Self-hosted) node using a valid Authentication Key.
Show Authentication Key
Learn how to find the Authentication Key
HTTP Proxy
Current Proxy:
No proxy
Change
Diagnostic Tool
Troubleshoot problems (preview)
Register
Cancel
1 item
1 item selected 945 MB

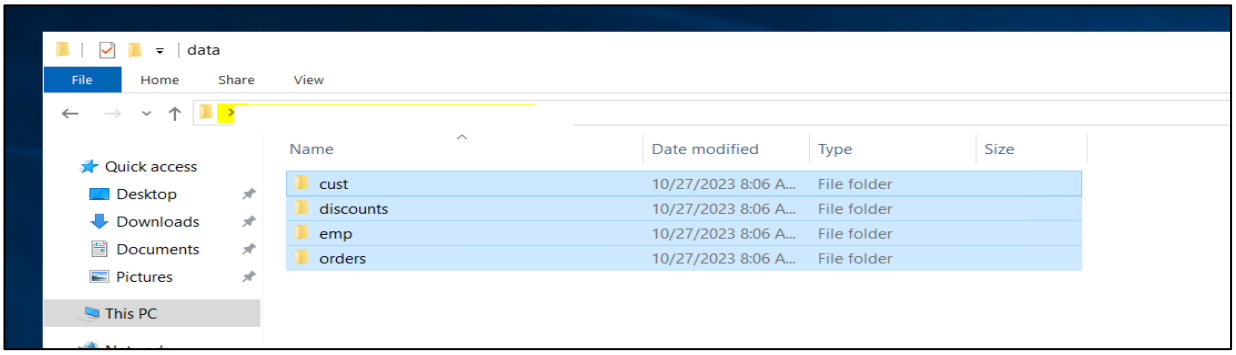
Step 8 - Connect Synapse Pool to SSMS:

Use SQL Server Management Studio (SSMS) to verify connectivity to your Synapse dedicated pool.



Step 9: - Prepare Files on the VM:

Upload your sample data files (preferably in .txt format since Excel isn't available) to the C: drive of the VM.



Step 10: - Disable Validation in Integration Runtime:

Go to the folder where the integration runtime is installed:

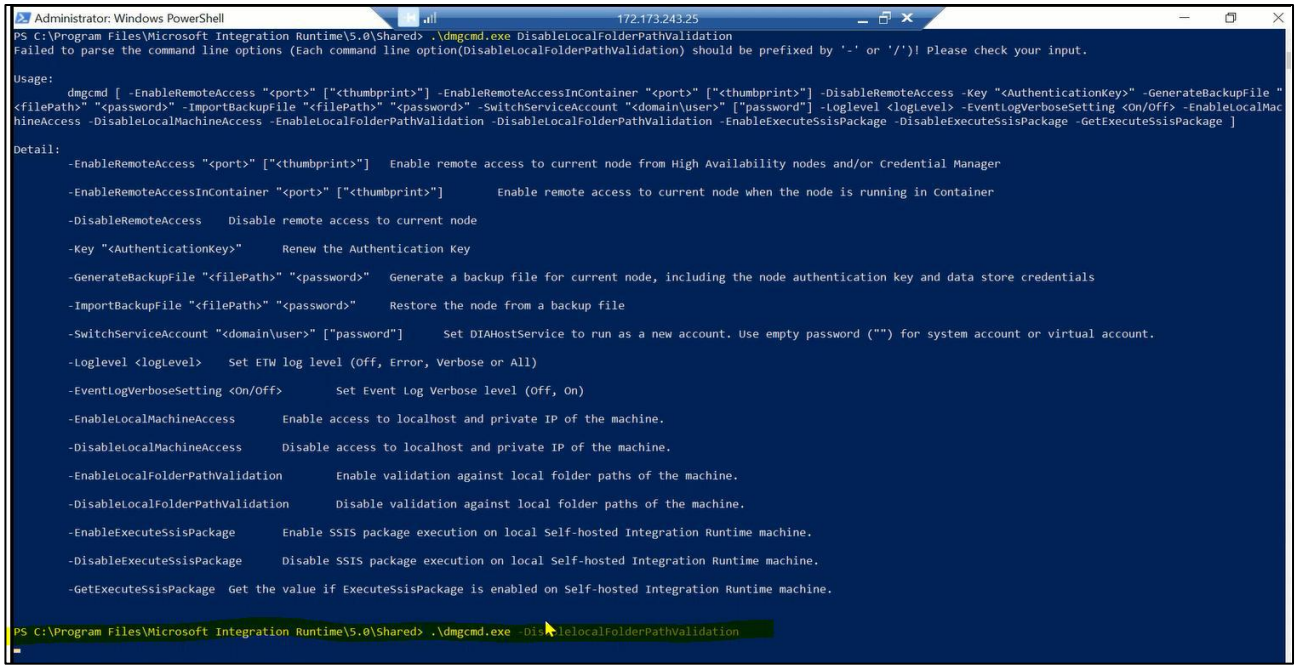
C:\Program Files\Microsoft Integration Runtime\5.0\Shared

- Open PowerShell inside the VM.
- Navigate to the directory using:

cd "C:\Program Files\Microsoft Integration Runtime\5.0\Shared"

- Run the following command to disable path validation:

.\dmgcmd.exe -DisableLocalFolderPathValidation


Administrator: Windows PowerShell
172.173.243.25
PS C:\Program Files\Microsoft Integration Runtime\5.0\Shared> .\dmgcmd.exe DisableLocalFolderPathValidation
Failed to parse the command line options (Each command line option(DisableLocalFolderPathValidation) should be prefixed by '-' or '/')! Please check your input.
Usage:
dmgcmd [-EnableRemoteAccess "<port>" ["<thumbprint>"] -EnableRemoteAccessInContainer "<port>" ["<thumbprint>"] -DisableRemoteAccess -Key "<AuthenticationKey>" -GenerateBackupFile "
<filePath>" "<password>" -ImportBackupFile "<filePath>" "<password>" -SwitchServiceAccount "<domain\user>" ["password"] -Loglevel <logLevel> -EventLogVerboseSetting <On/Off> -EnableLocalMac
hineAccess -DisableLocalMachineAccess -EnableLocalFolderPathValidation -DisableLocalFolderPathValidation -EnableExecuteSsisPackage -DisableExecuteSsisPackage -GetExecuteSsisPackage]
Detail:
-EnableRemoteAccess "<port>" ["<thumbprint>"] Enable remote access to current node from High Availability nodes and/or Credential Manager
-EnableRemoteAccessInContainer "<port>" ["<thumbprint>"] Enable remote access to current node when the node is running in Container
-DisableRemoteAccess Disable remote access to current node
-Key "<AuthenticationKey>" Renew the Authentication Key
-GenerateBackupFile "<filePath>" "<password>" Generate a backup file for current node, including the node authentication key and data store credentials
-ImportBackupFile "<filePath>" "<password>" Restore the node from a backup file
-SwitchServiceAccount "<domain\user>" ["password"] Set DIAHostService to run as a new account. Use empty password ("") for system account or virtual account.
-Loglevel <logLevel> Set ETW log level (Off, Error, Verbose or All)
-EventLogVerboseSetting <On/Off> Set Event Log Verbose level (Off, On)
-EnableLocalMachineAccess Enable access to localhost and private IP of the machine.
-DisableLocalMachineAccess Disable access to localhost and private IP of the machine.
-EnableLocalFolderPathValidation Enable validation against local folder paths of the machine.
-DisableLocalFolderPathValidation Disable validation against local folder paths of the machine.
-EnableExecuteSsisPackage Enable SSIS package execution on local Self-hosted Integration Runtime machine.
-DisableExecuteSsisPackage Disable SSIS package execution on local Self-hosted Integration Runtime machine.
-GetExecuteSsisPackage Get the value if ExecuteSsisPackage is enabled on Self-hosted Integration Runtime machine.
PS C:\Program Files\Microsoft Integration Runtime\5.0\Shared> .\dmgcmd.exe -DisableLocalFolderPathValidation

Step 11: Creating Linked Services in Azure Data Factory

1. **File System Linked Service (On-Prem VM):**

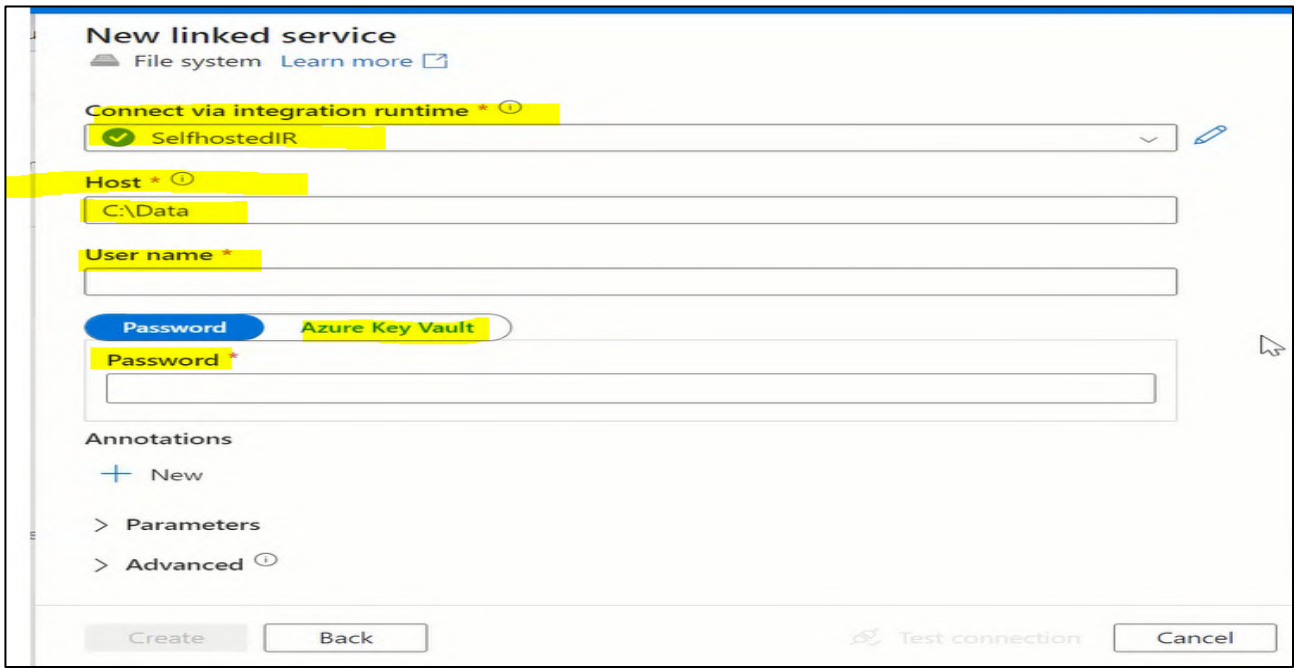
   Choose Self-Hosted Integration Runtime.

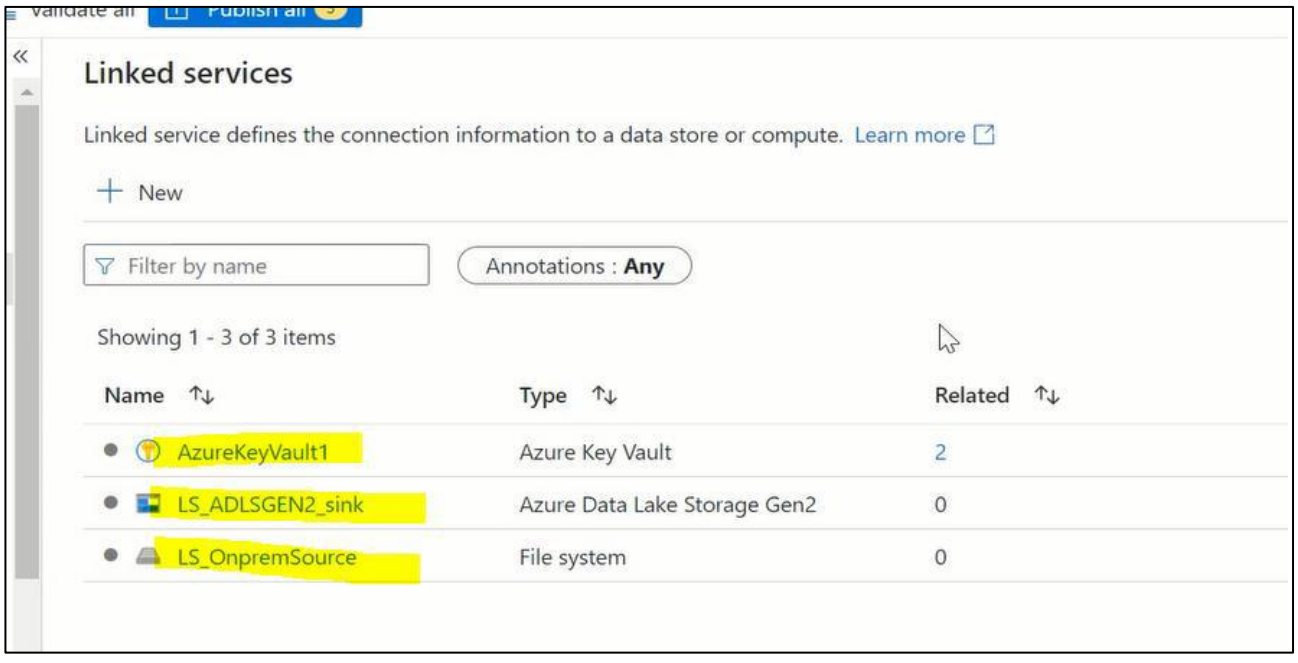
   In the host/path section, provide the exact directory path where the files were copied (e.g., C:\Data).

   For authentication:

   - Use the username of the VM (RDP username).
   - For the password, store it as a secret in Azure Key Vault, then reference it in the linked service using Key Vault integration.

   Click Test Connection to ensure it connects successfully. If the dmgcmd.exe validation is disabled correctly, the connection will succeed.





2. **ADLS Gen2 Linked Service:**

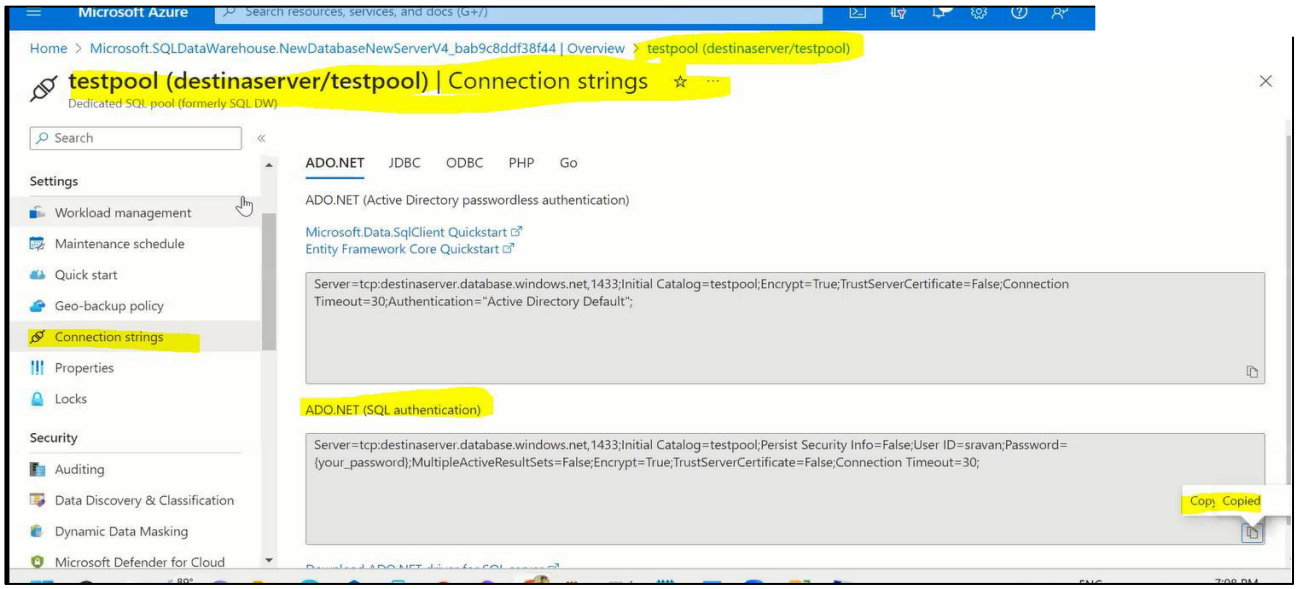
   Use Azure Key Vault to securely retrieve secrets.

   The URL format should be:
   https://<adlsaccountname>.dfs.core.windows.net

   To get the connection string:

   - Go to the ADLS Storage Account → Access Keys → Connection String → copy it.
   - Create a secret in Key Vault and store the connection string.

   If not already created, set up a Key Vault linked service in ADF to access the secret.

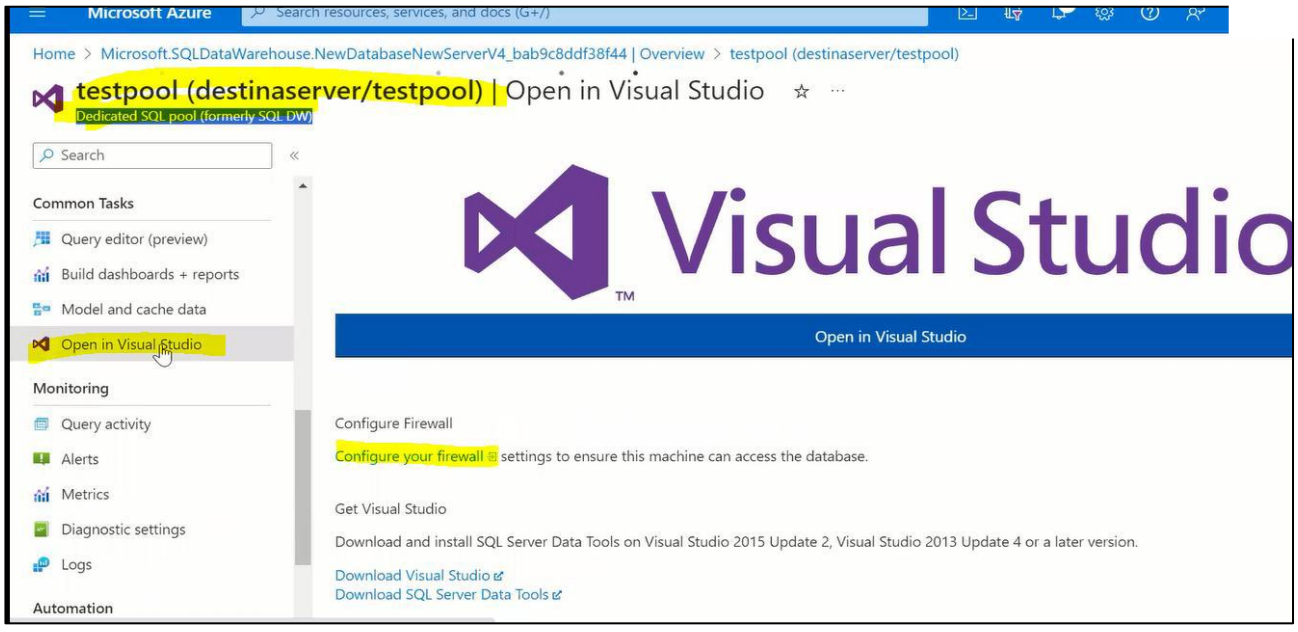


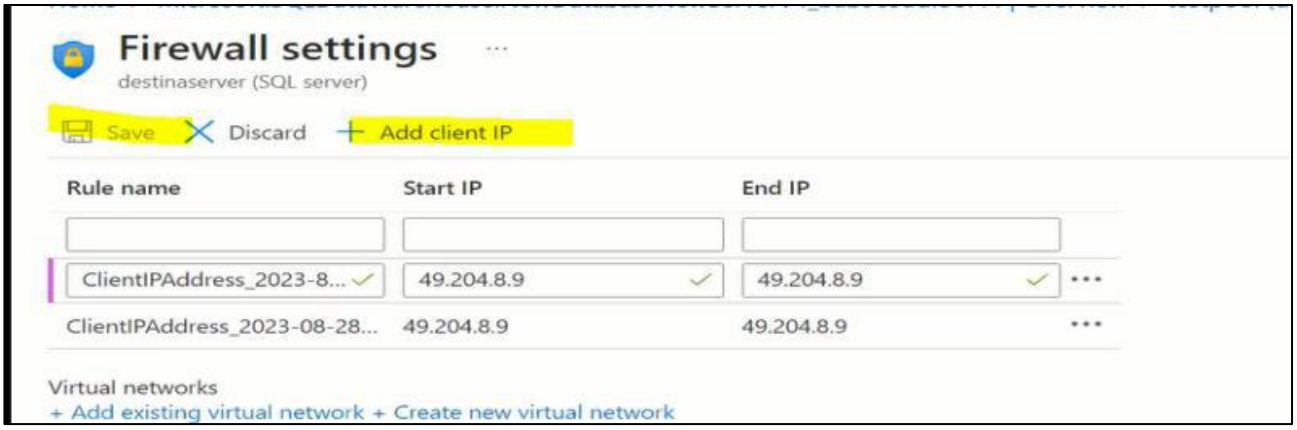
3. **Azure Synapse (SQL Pool) Linked Service:**

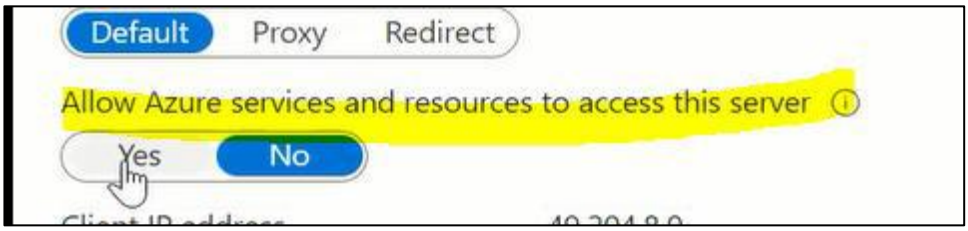
   Choose Azure Synapse Analytics as the type.

   Use the Key Vault secret to get the password dynamically via the connection string.

   The connection string should be of .NET format with the password field referencing Key Vault.

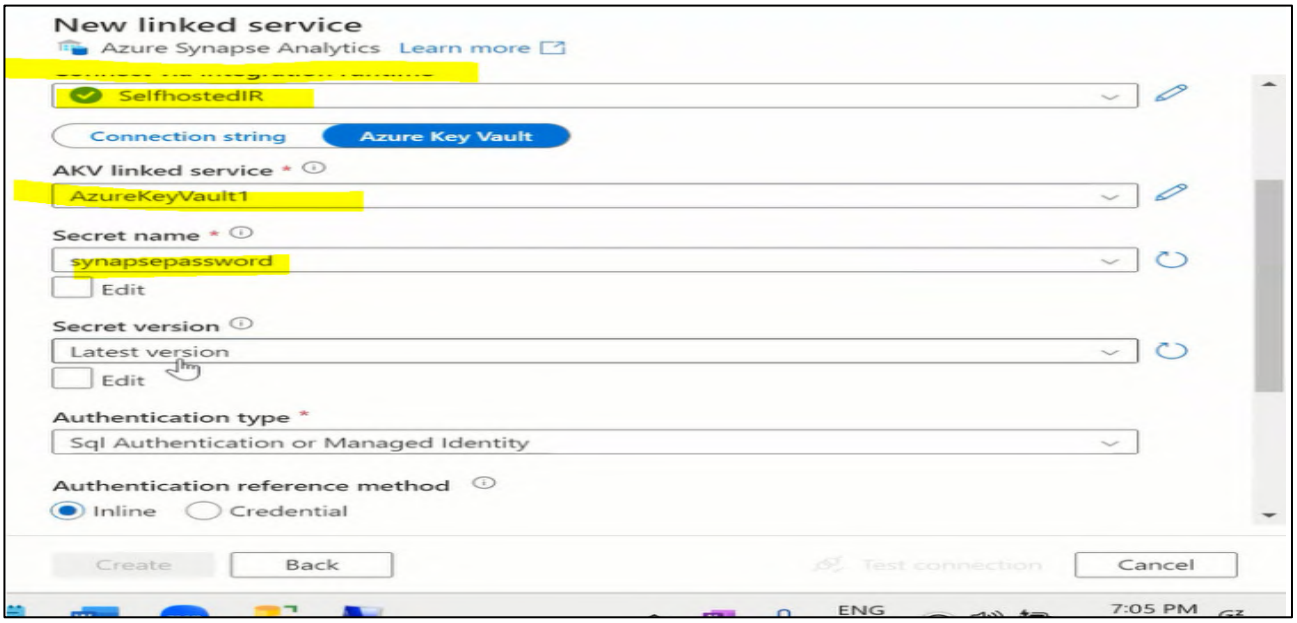






4. **Firewall Configuration:**

Ensure that the SQL Pool's firewall allows access from the Self-Hosted IR.



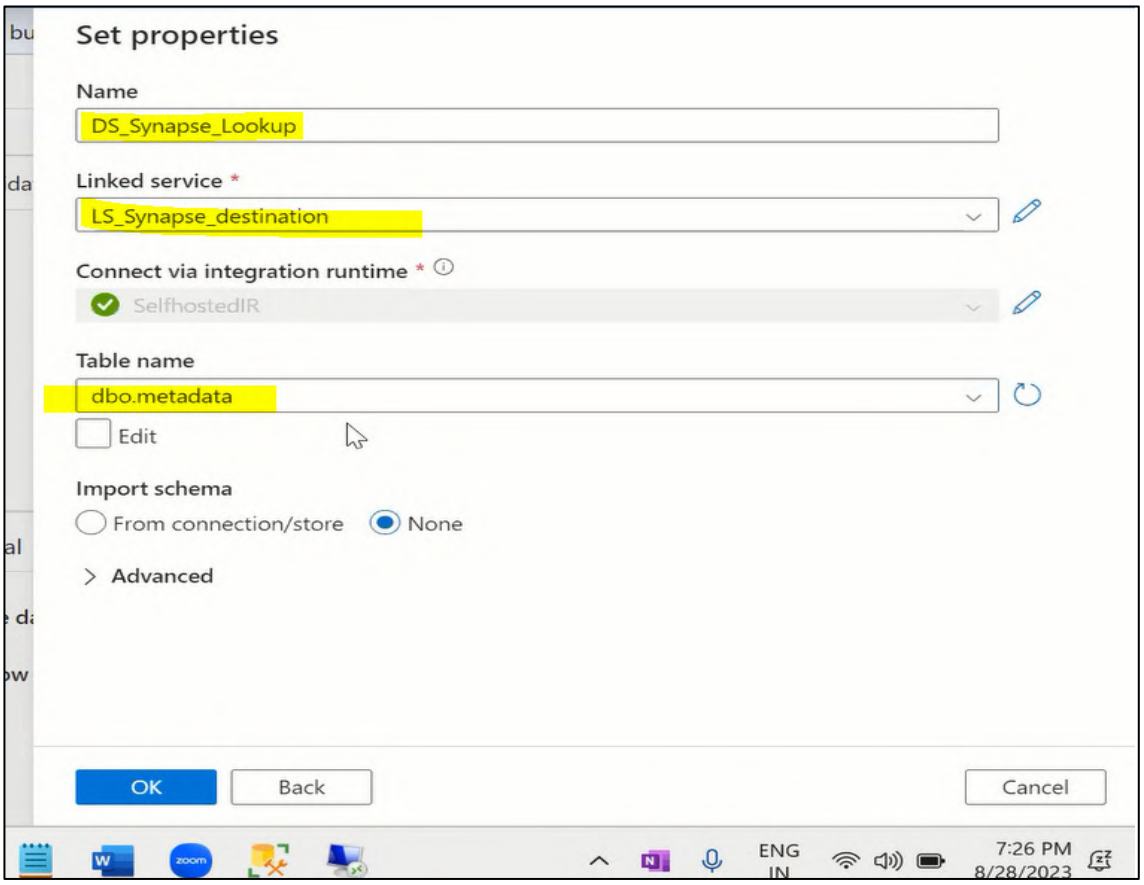
Step 12: Setting Up Metadata and Stored Procedures in Synapse (SSMS)

Run these scripts inside the testpool database using SSMS:

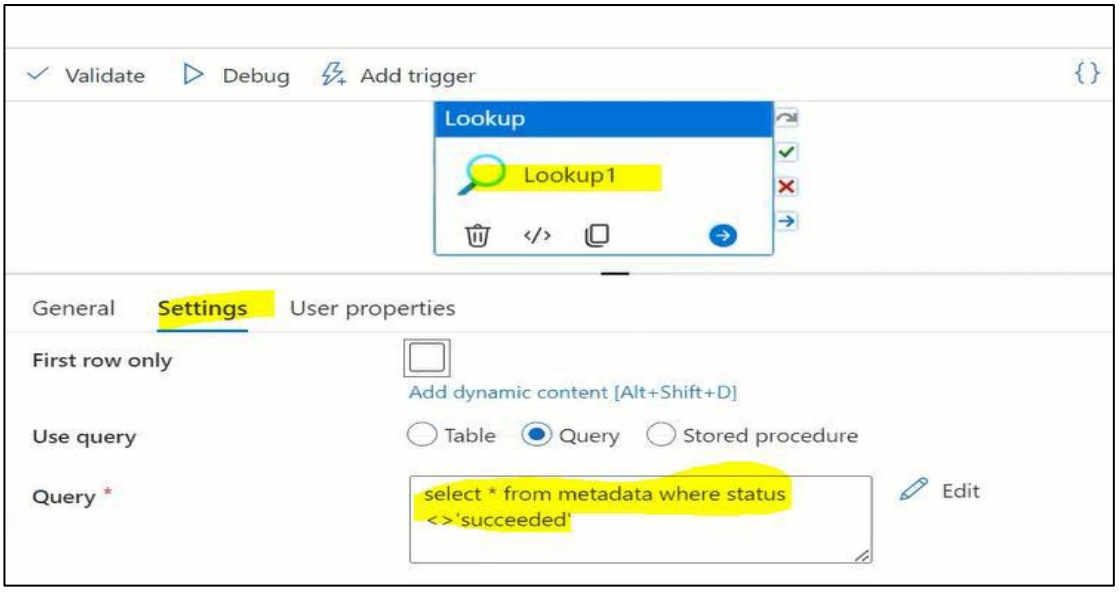
```
-- Create metadata table
CREATE TABLE metadata (
   sourcefoldername VARCHAR(50),
   storagepath VARCHAR(50),
   isactive INT,
   status VARCHAR(50)
);

-- Insert initial folder details
INSERT INTO metadata (sourcefoldername, storagepath, isactive, status)
VALUES
('cust', 'cust', 0, 'ready'),
('orders', 'orders', 0, 'ready'),
('emp', 'emp', 0, 'ready'),
('discounts', 'discounts', 0, 'ready');

-- Create stored procedure to update status
CREATE PROCEDURE metadata_usp (@status VARCHAR(50), @sourcefoldername VARCHAR(50))
AS
BEGIN
   UPDATE metadata
   SET status = @status
   WHERE sourcefoldername = @sourcefoldername;
END;

-- Create stored procedure to reset status
CREATE PROCEDURE reset_status_usp
AS
BEGIN
   UPDATE metadata
   SET status = 'ready';
END;
```

Step 13→ Create the pipeline in ADF

First activity is lookup→ to lookup metadata table from azure synapse



Take lookup activity→ setting→ create dataset→ create dataset as below



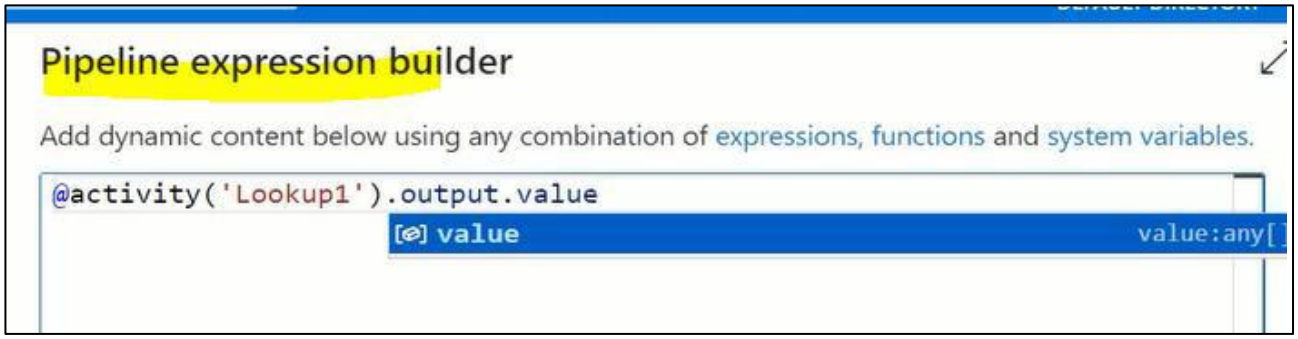
Preview data

Linked service: LS_Synapse_destination

Object:

	sourcefoldername	storagepath	isactive	status
1	orders	orders	0	ready
2	cust	cust	0	ready
3	discounts	discounts	0	ready
4	emp	emp	0	ready

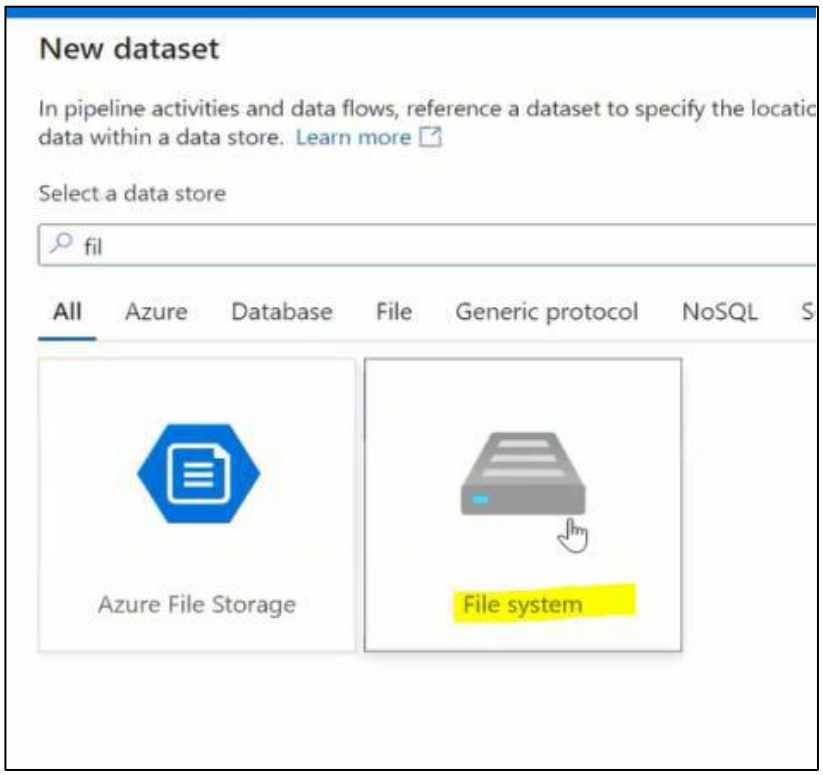
Step → take for each activity next to lookup, take the output of lookup as input to for each loop

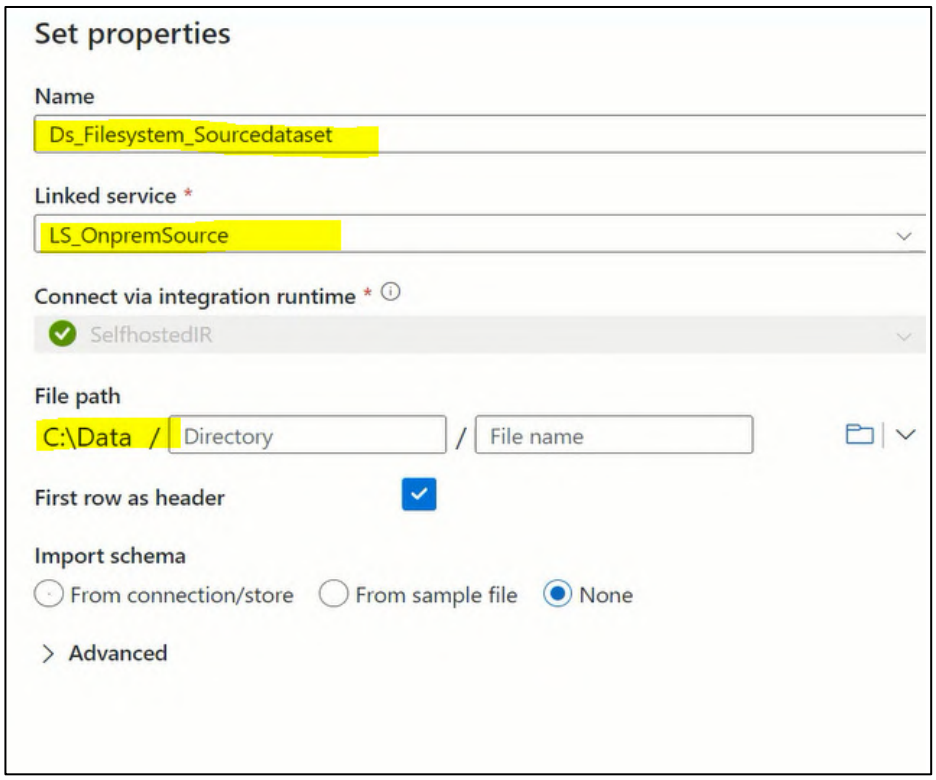


Inside for each → take copy activity → copy activity

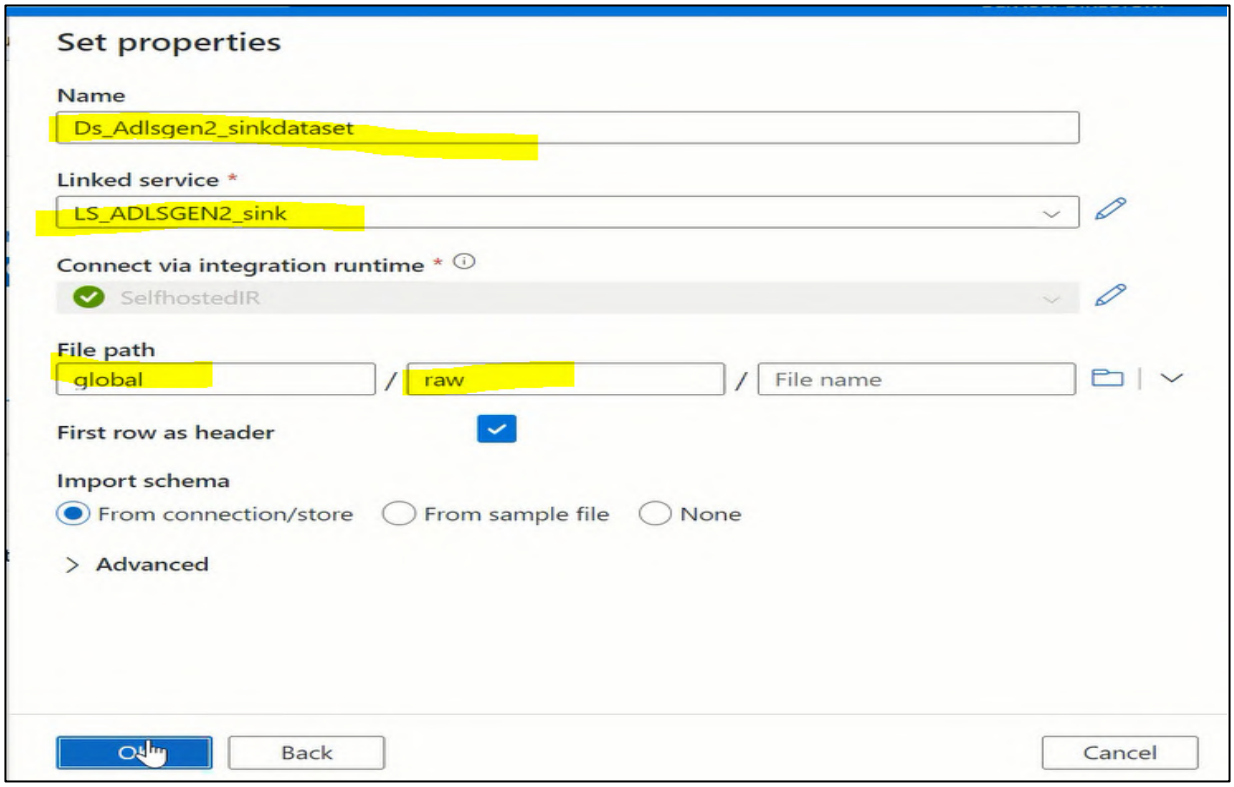
for copy activity source is file system → create the dataset →

file system→ csv file





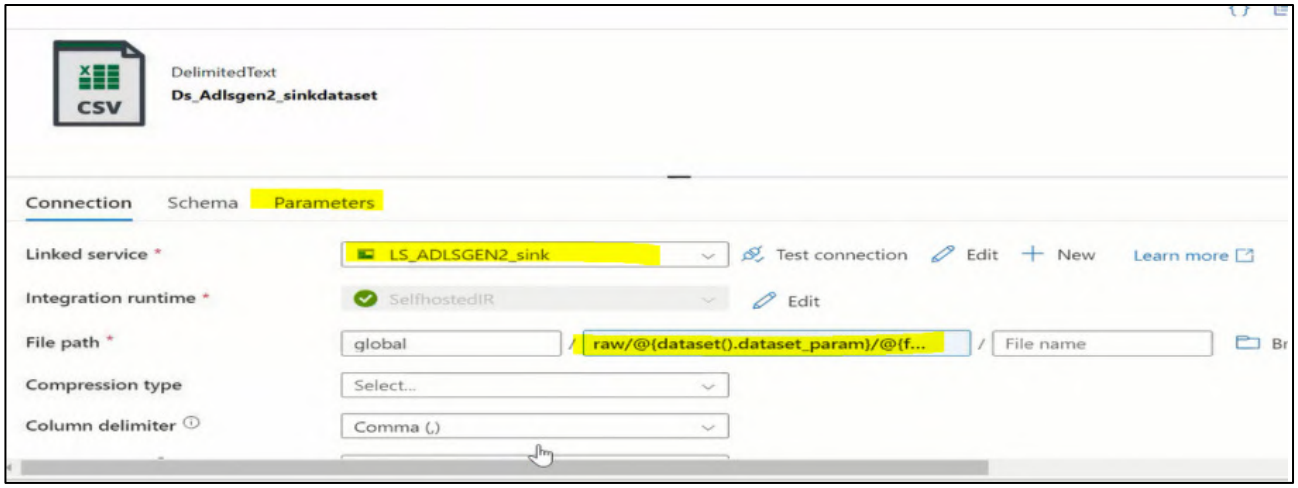
for sink use adls gen 2→ create dataset for the same



Here file path should be in sink raw, as we are going to put raw data in raw folder

Parameterize the directory → for file system dataset source

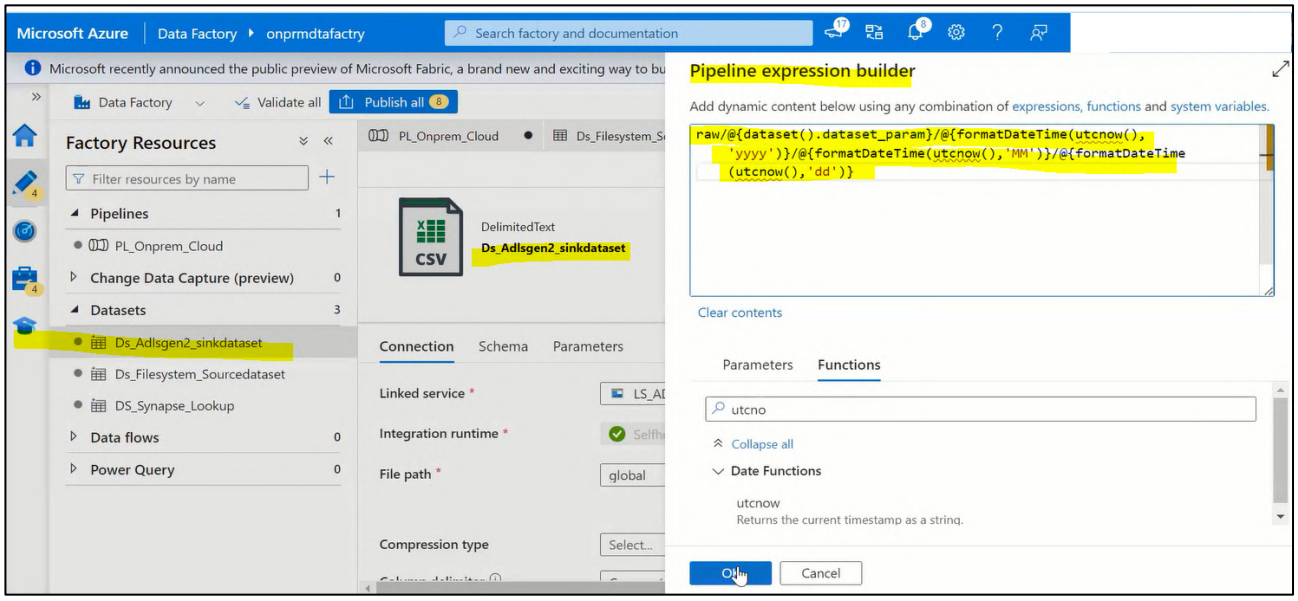
And in sink dataset also parameterize

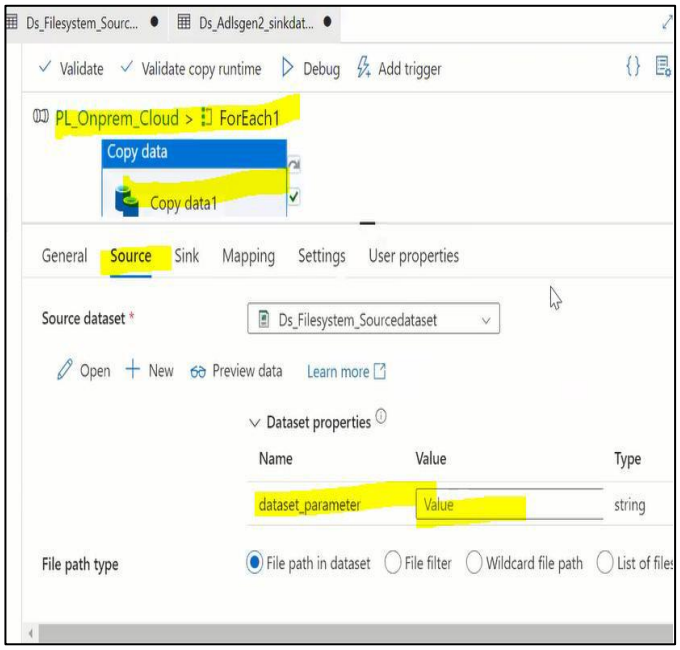


Add→ directory in form of

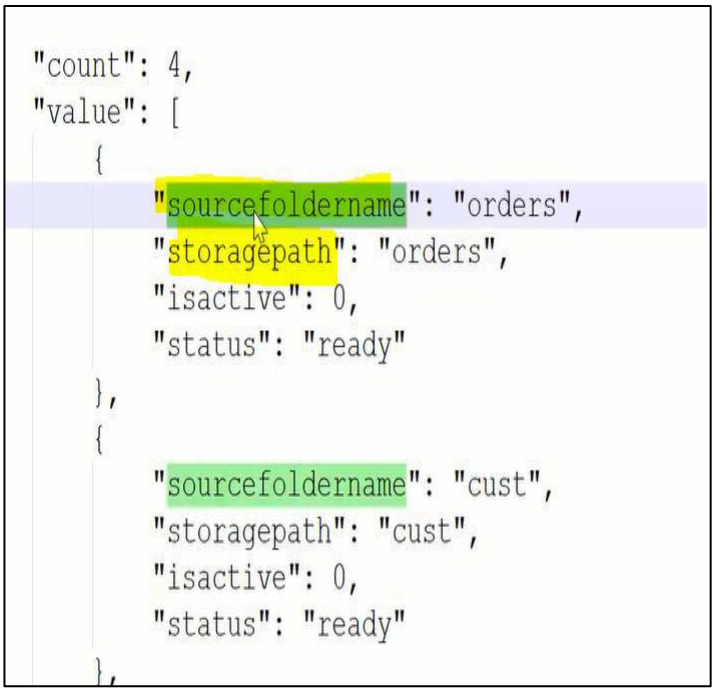
Raw/filename/yyyy/mm/dd

It should be created as folder.

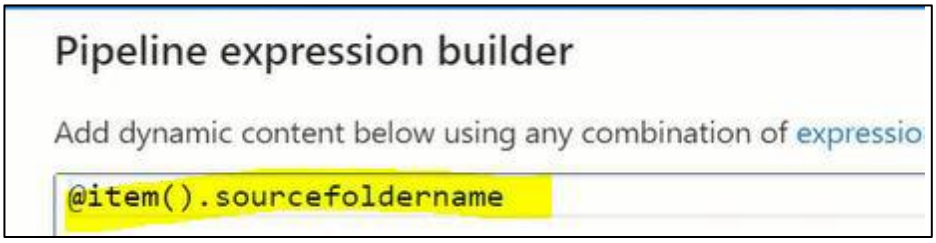


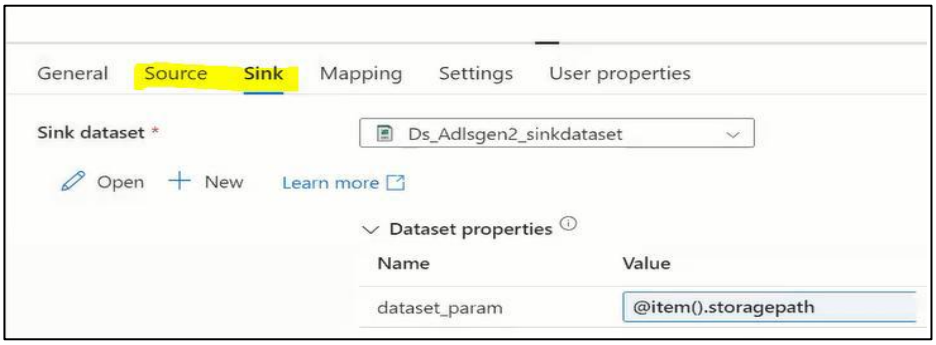


Take output from lookup activity



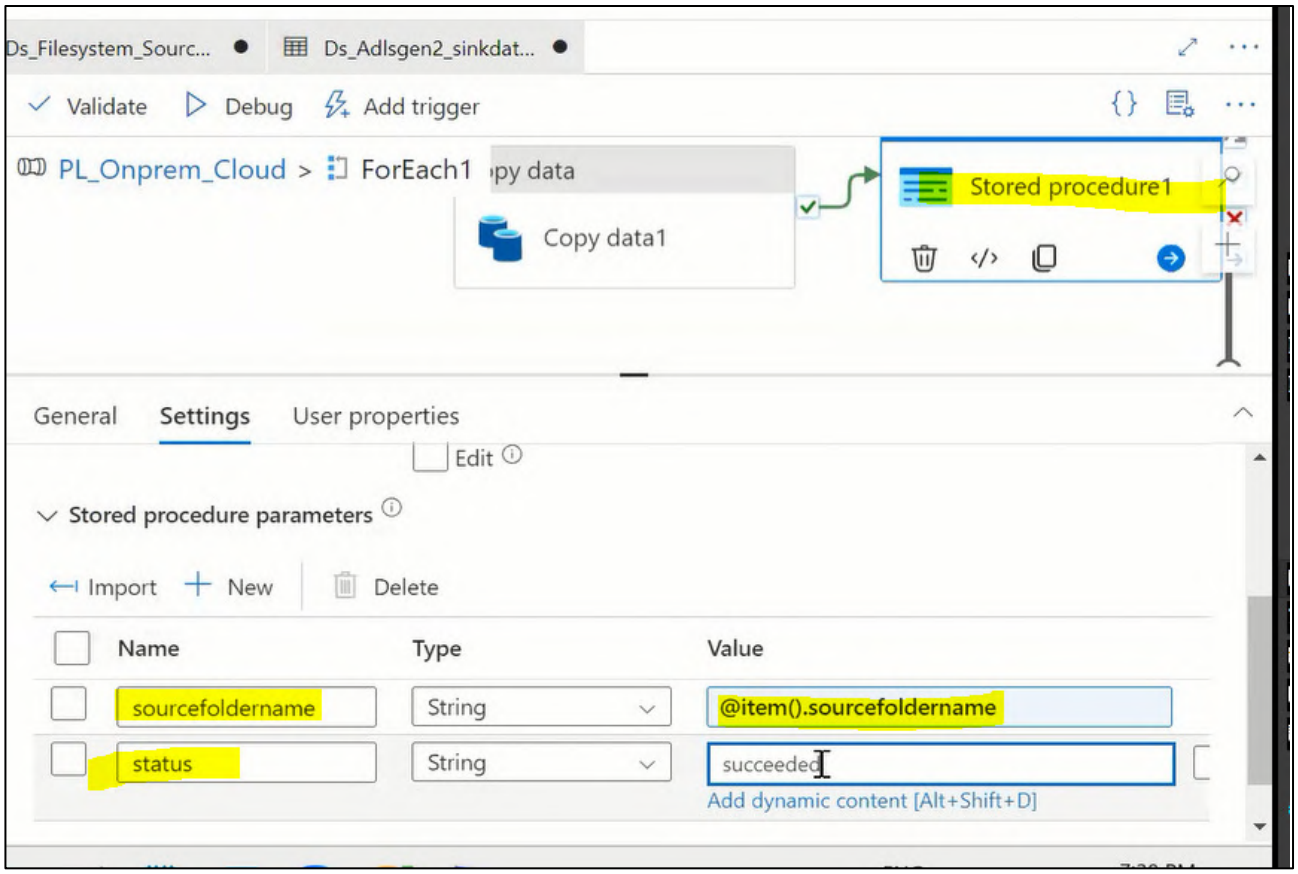
Sourcefoldername





Sink→ storage path

Following the copy activity take store proc activity

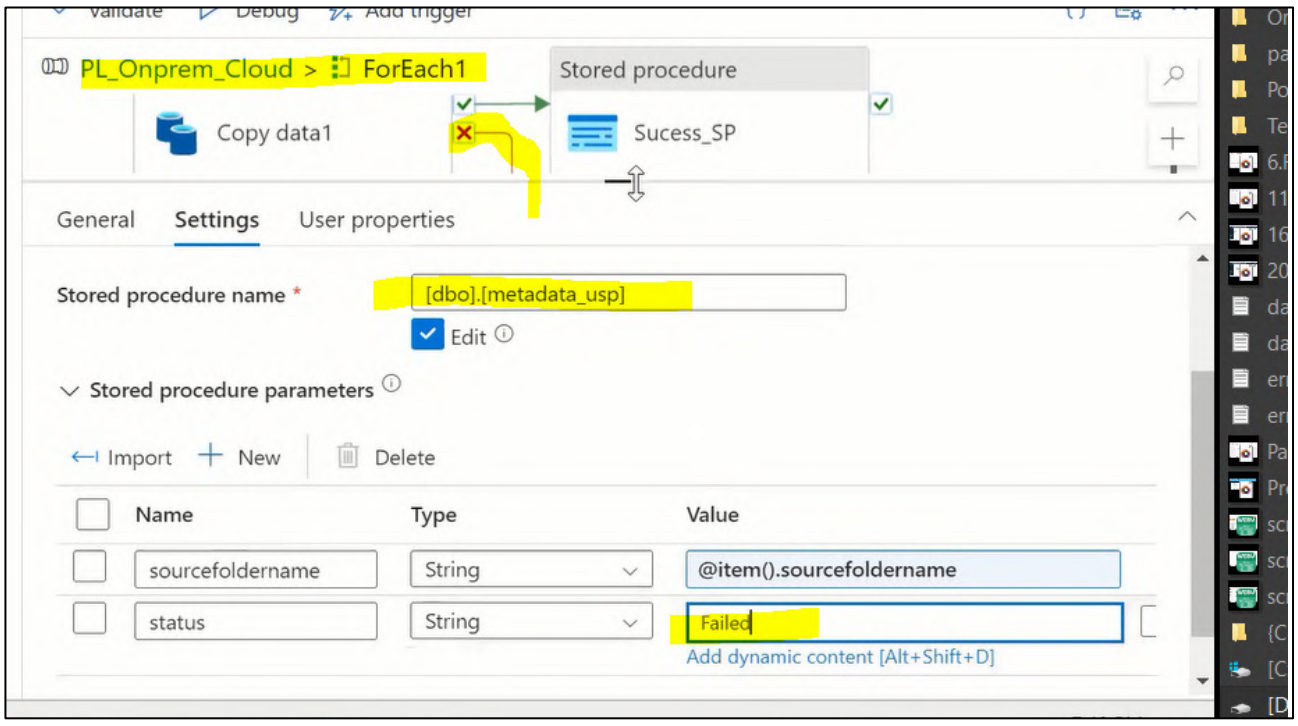


Use→metadat_usp as store proc,

And import parameter,

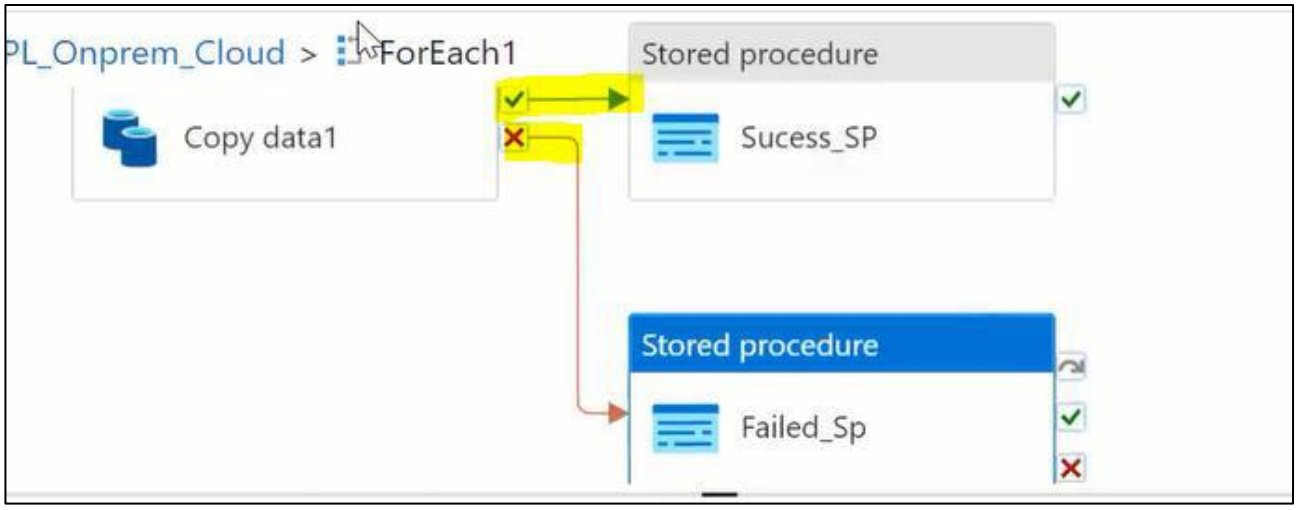
Add sourcefoldername as dynamically.

Status hard code succeeded indicating the copy activity finished

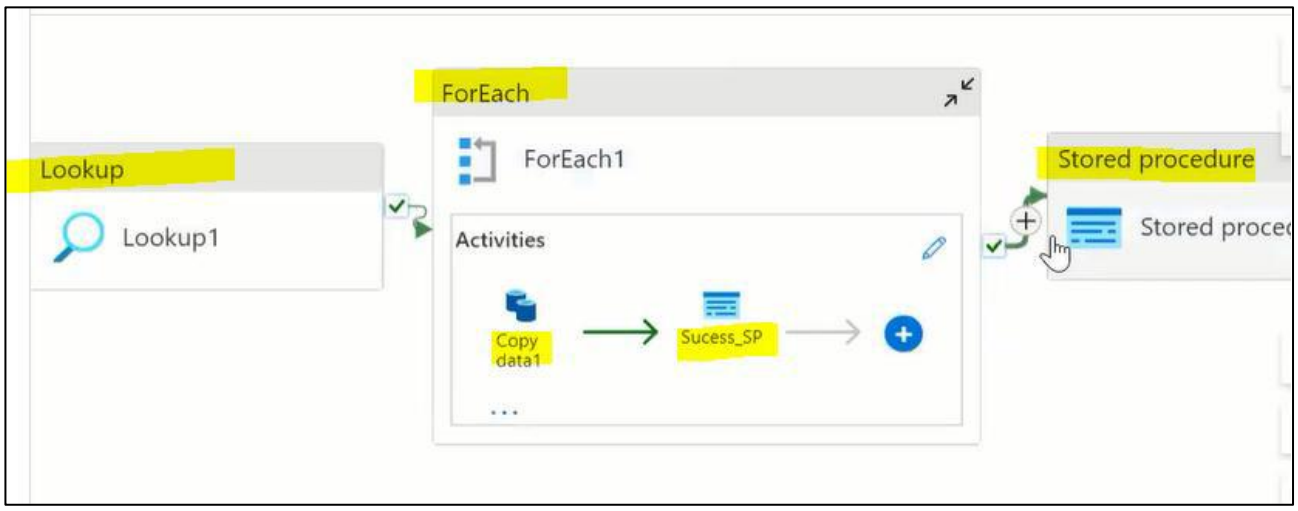


Take another store procedure activity for failure

Use the same store procedure and hardcode status as failed



Next take the store procedure outside the for each activity



Here we reset the sp on success

CREATE PROCEDURE reset_status_usp

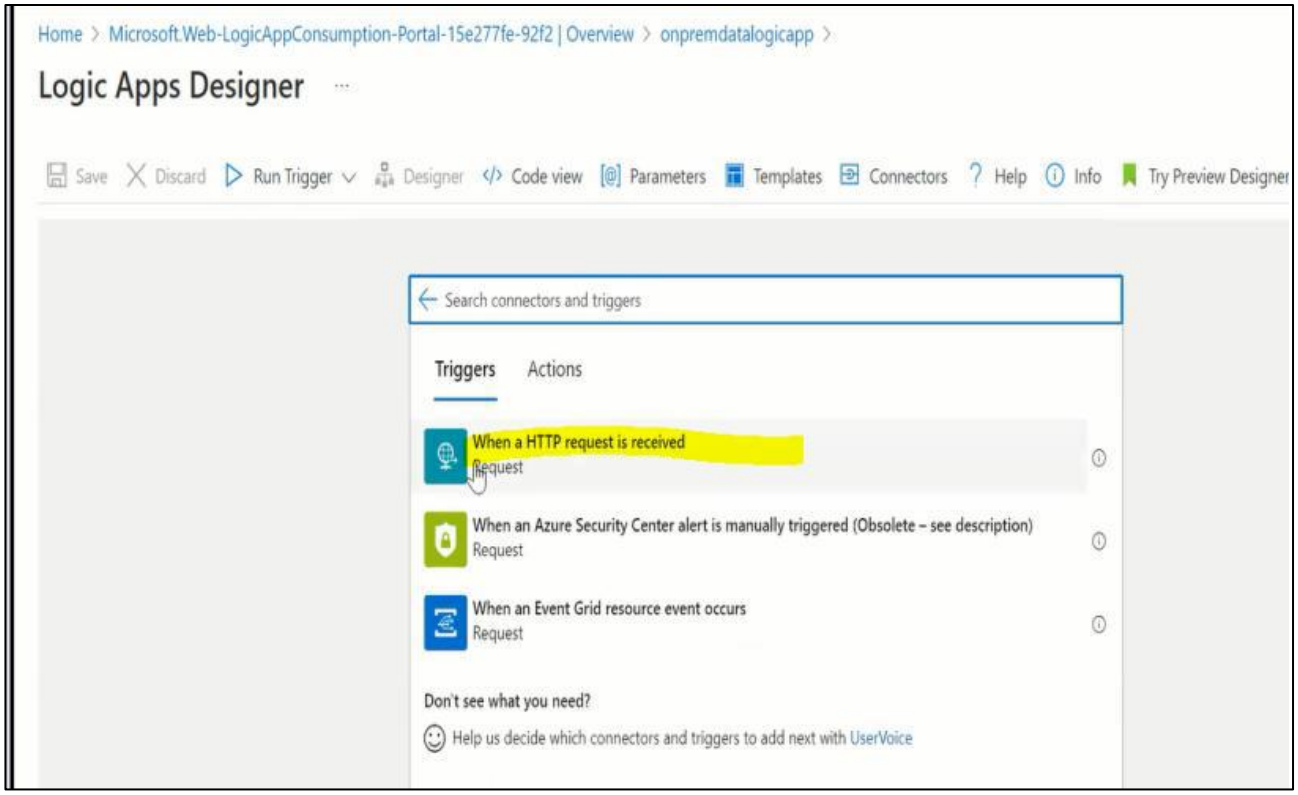
AS

BEGIN

UPDATE metadata

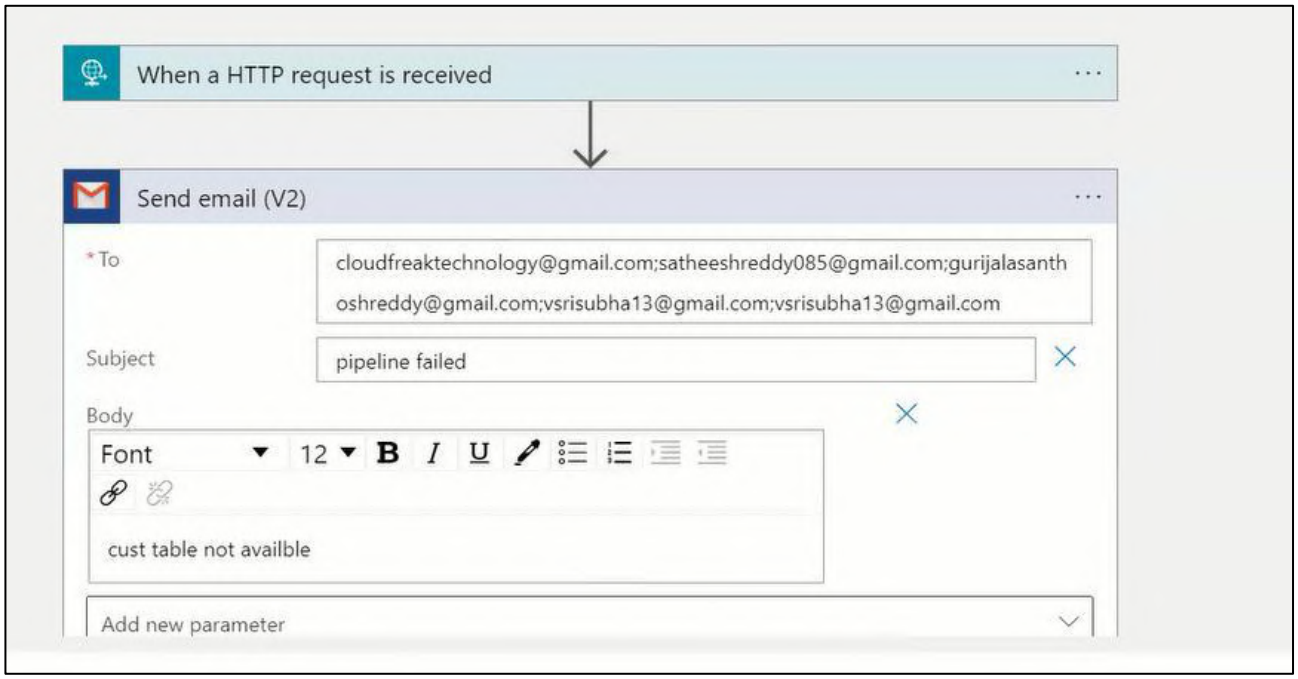
SET status = 'ready';

END;

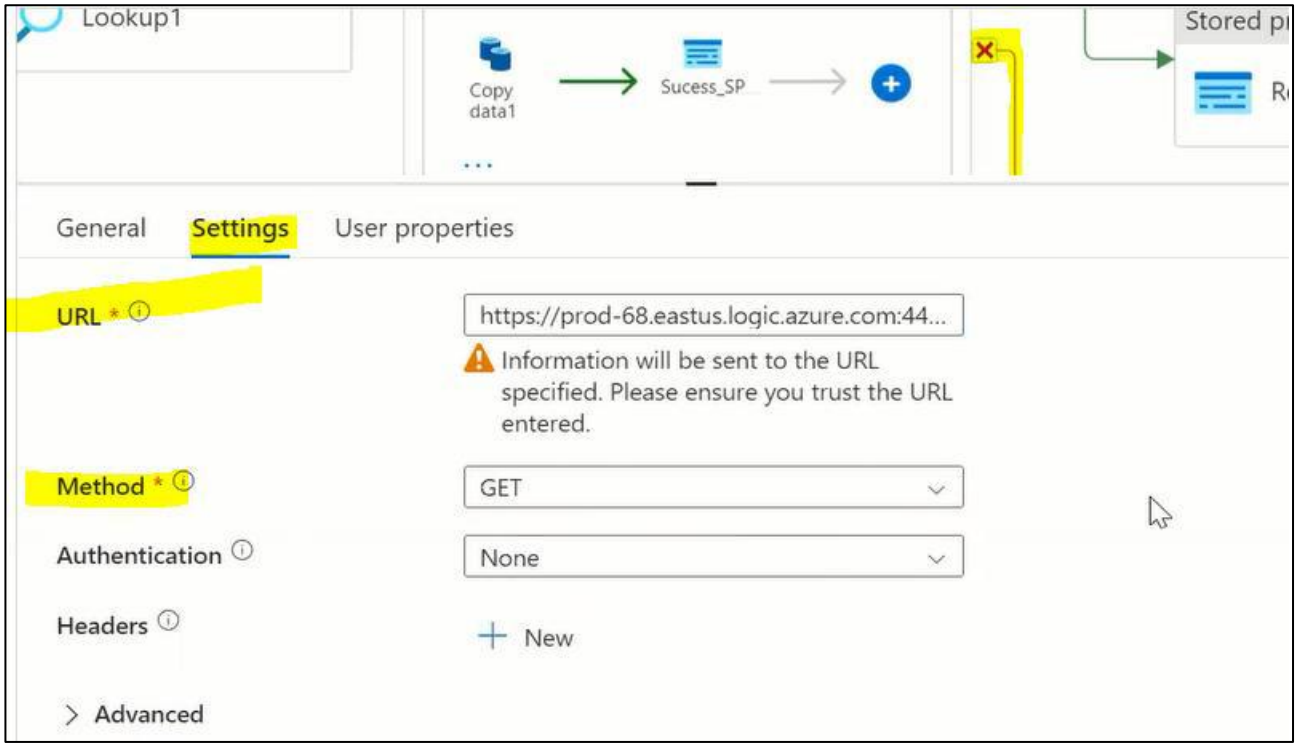


Step → Create a logic app

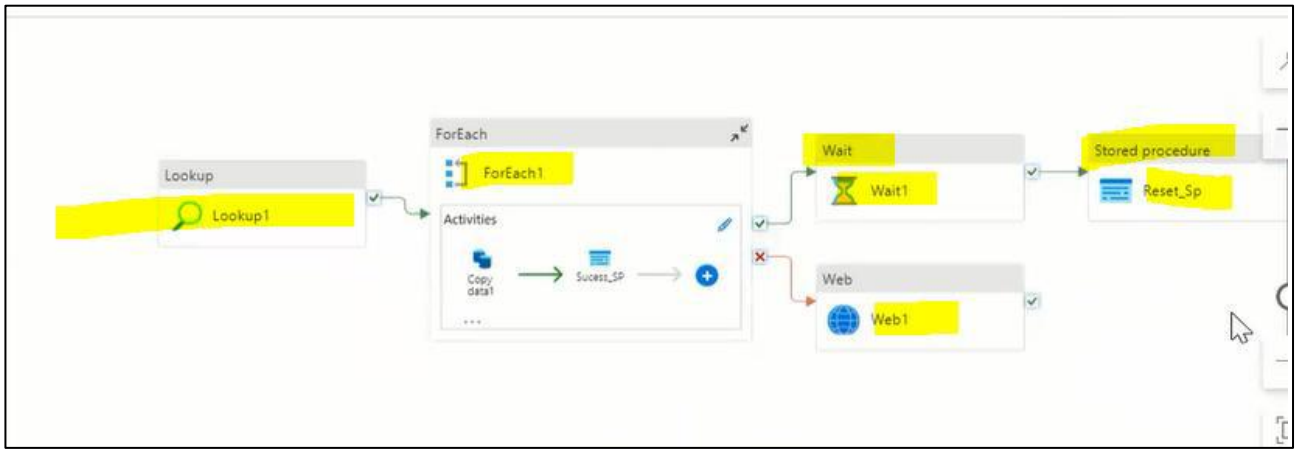
Create logic app→go to resource→ create blank new→ search for hhtp→select request→ add new parameter→method →GET→ add next step → gmail→send email→name = gmail →sign in → add the to email→ subject → save → url will be generated in http→ copy url

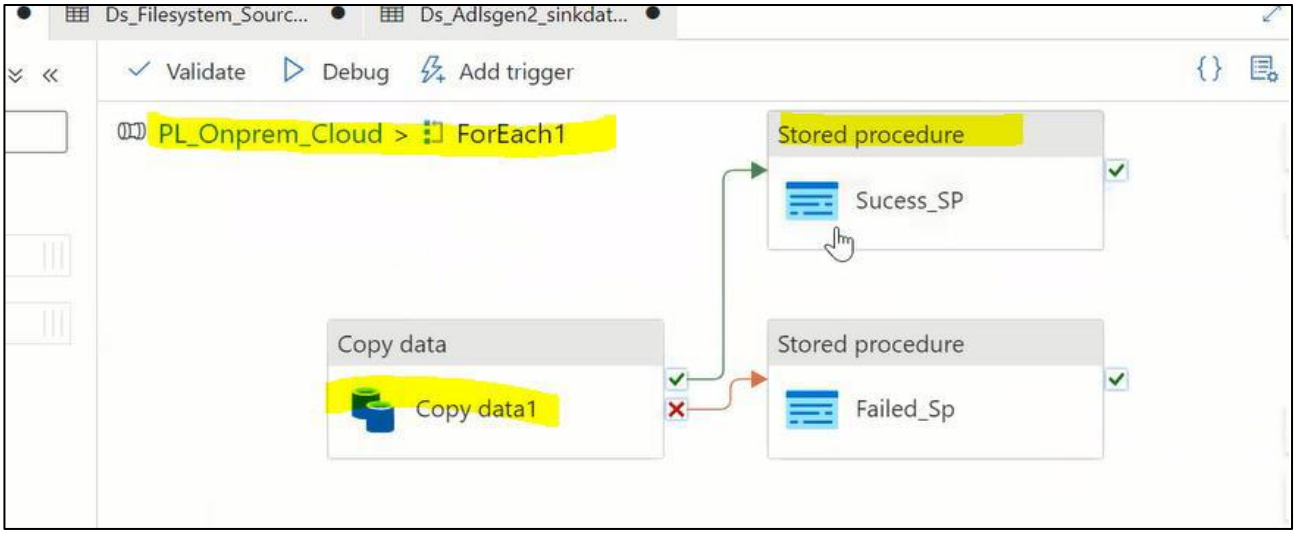


Step→ go to ADF→ on failed of for each take a web activity

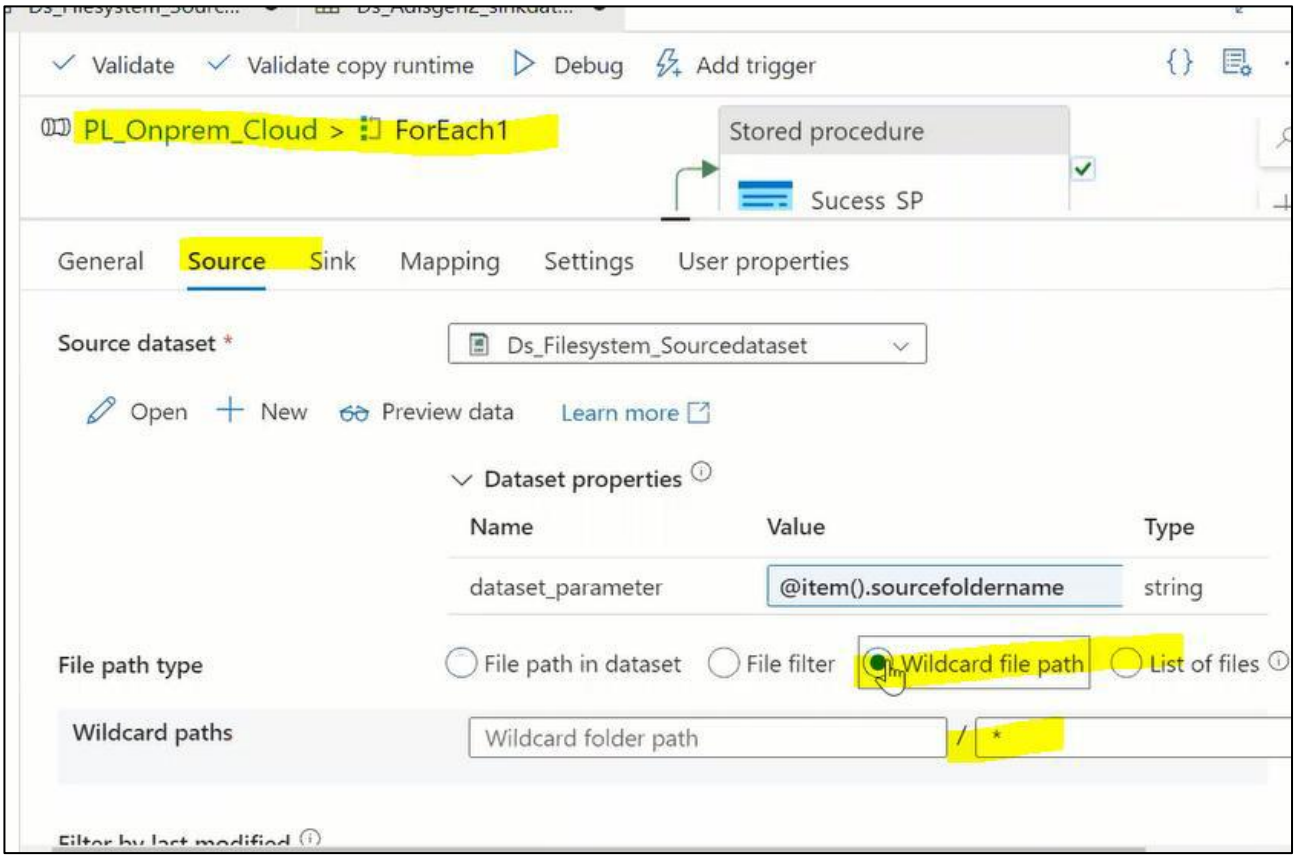


Small change→ add the wait activity after for each loop → 30 secs





Here add the wild card and * as there are folder in source in RDP where the files are added



If any failure like the change in file name or meatadata occurs email is triggered.

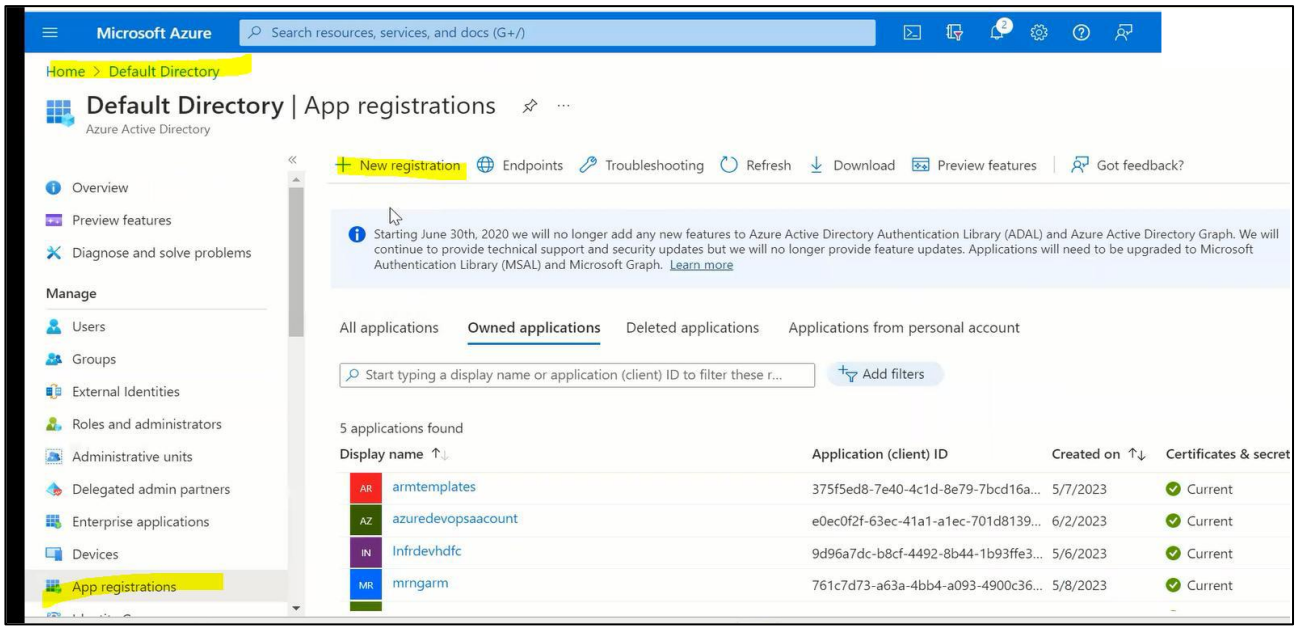
Step 14→ perform data quality checks and move cleansed data from Raw layer to bronze layer.

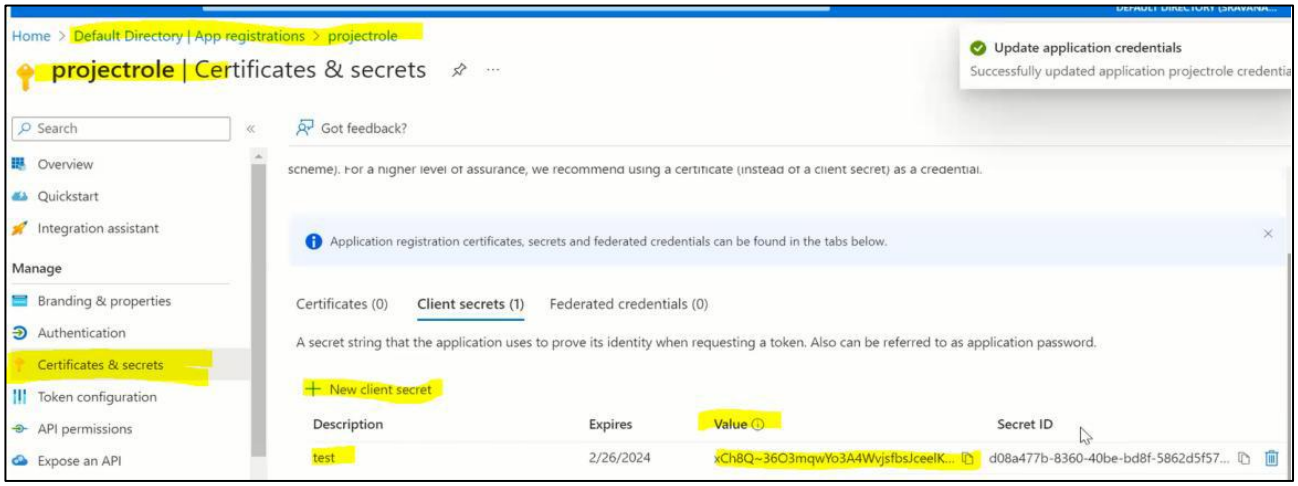
Data quality checks are performed in data bricks →Create Azure databricks service in Azure.

Data source is ADLS container → create a mount point to connect to container (since we are using only one container and inside that container we are moving data from raw folder to bronze folder only 1 mount point is enough)

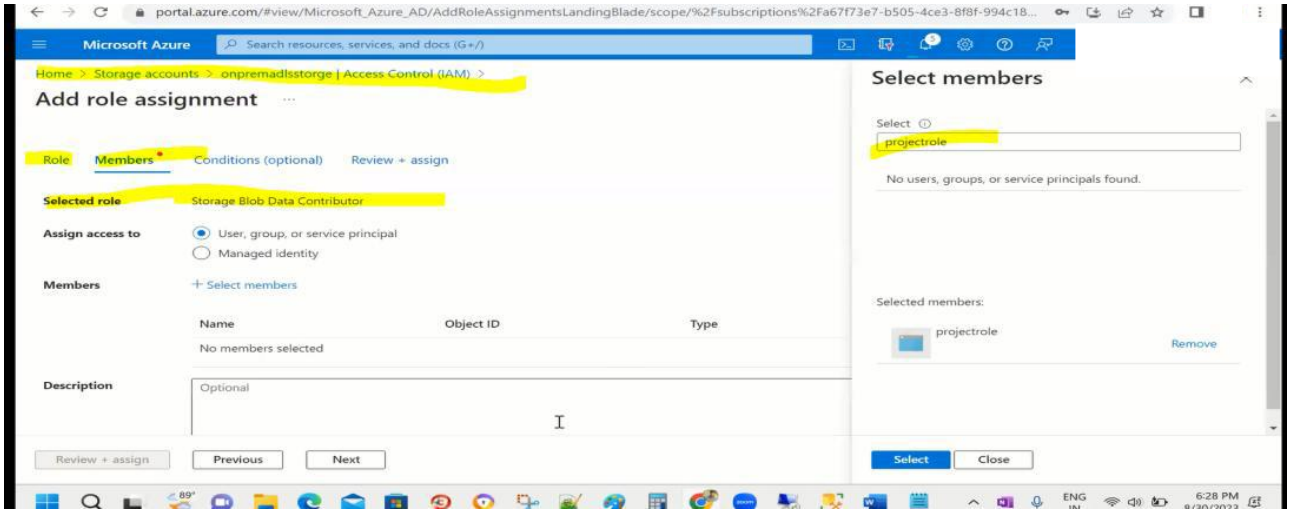
For mount ADLS → required service→ ADB, Azure key vault and SPN→ i.e App registration(where we create a new client network and from there extract client secret value, client ID, tenant ID and create the key vault secrets for the same)

Create app registration → go to app directory →app registration→once created→ go to secrets and certificate→new client secret → copy the secret value and keep as it will be encrypted once web is closed





Step→ go to your Adls storage account→I am role→ create a role for storage Blob data contributor →assign that to app regist that i.e created above → review and assign



Step 15→ go to ADB →create notebook→

Create dbutilities widgets (once executes the widgets box appears at top through which we can filter the particular data and run the entire notebook)

dbutils.widgets.text(processeddate,'')

dbutils.widgets.text(foldername,'')

step→ create the ADLS mount point

configs = {"fs.azure.account.auth.type": "OAuth",

"fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",

"fs.azure.account.oauth2.client.id": dbutils.secrets.get(scope="adlsgenkey",key="appid"),

"fs.azure.account.oauth2.client.secret": dbutils.secrets.get(scope="adlsgenkey",key="apppwd"),

"fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/f5ea40f2-c7b8-4658-8d25-0aac8535e48c/oauth2/v2.0/token",

"fs.azure.createRemoteFileSystemDuringInitialization": "true"}

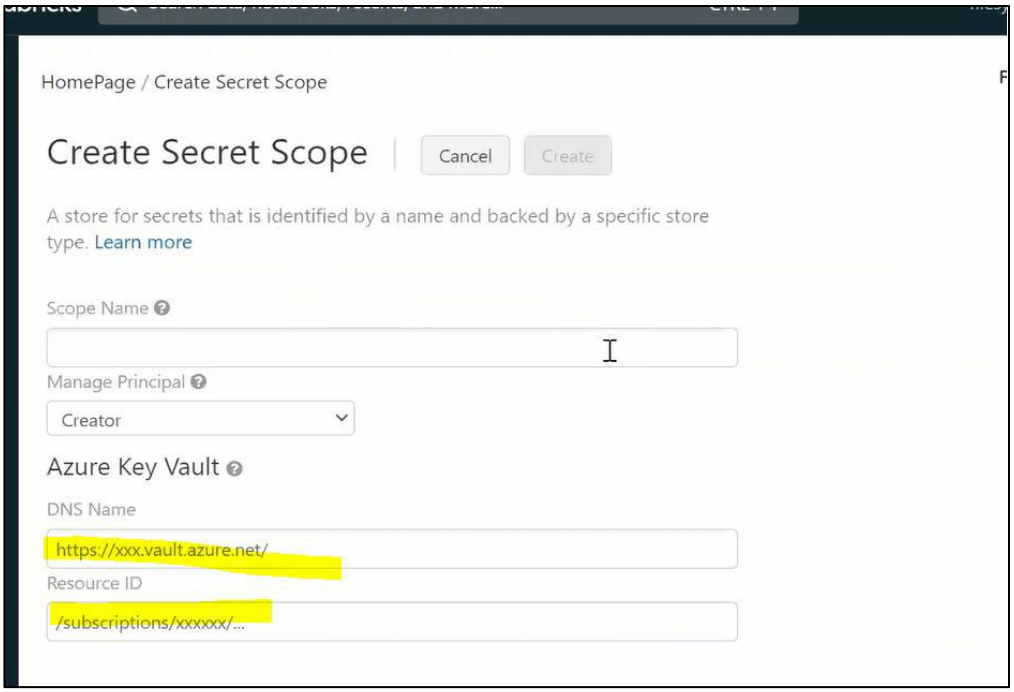
dbutils.fs.mount(

source = "abfss://global@adlsgenstorageaccountny.dfs.core.windows.net/",

mount_point = "/mnt/global",

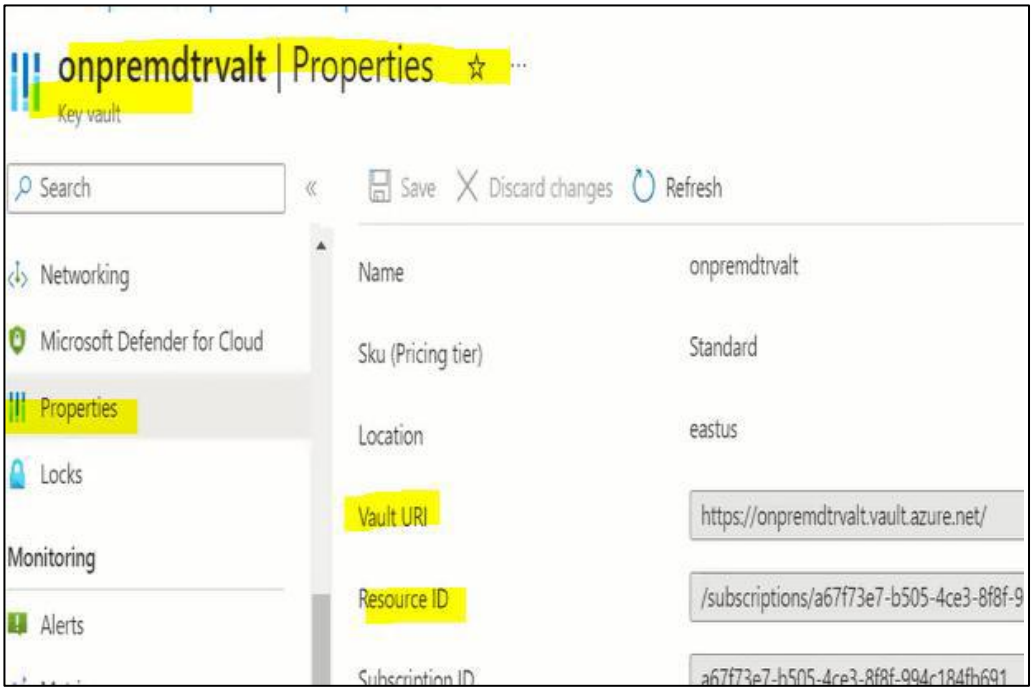
extra_configs = configs)

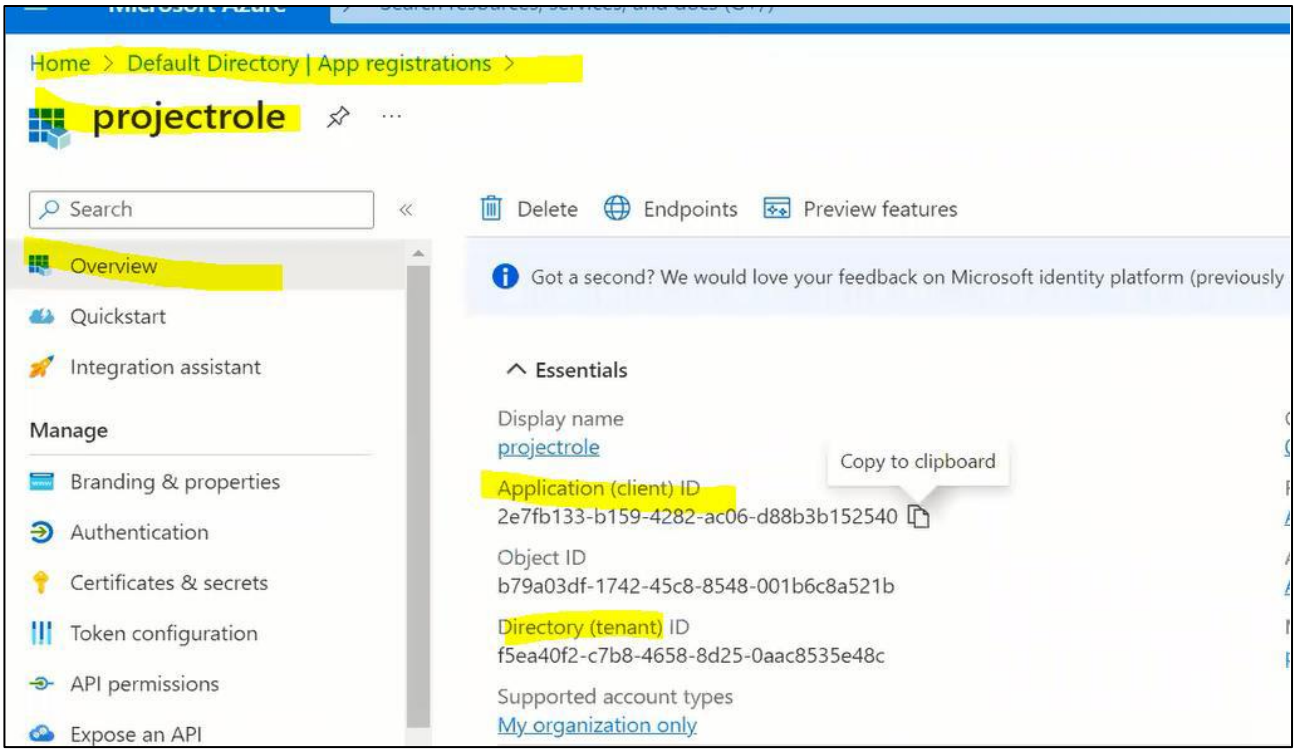
create the scope by adding attend of url as →#secrets/createScope



Give the scope name → adlsgenkey

DNS name and rescource ID → from azure key vault





in mount point→

client.secret":→ i.e apppwd the secret value copied from app registration

client id→ application id

tenant id copy and paste in

fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/<tenantID > /oauth2/v2.0/token

source = "abfss://<containername>@<storageaccountname>.dfs.core.windows.net/",

once execute comment all (select all in text and Ctrl+/ shortcut to comment all together)

Step 16→Aim is to move data from raw to bronze

```
src_path="/mnt/global/raw/"
dest_path="/mnt/global/bronze/"
dbutils.widgets.text(processeddate,'')
dbutils.widgets.text(foldername,'')
foldername = dbutils.widgets.get(' foldername')
pdate = dbutils.widgets.get(' processeddate')
print(foldername )
print(cdate)
src_final_path=src_path+foldername+"/"+pdate
print(src_final_path)
dest_final_path=dest_path+foldername+"/"+pdate
print(dest_final_path)
#following is the code for cleaning the data
try:
# Read data from source path
df = spark.read.format("csv").option("header", True).load(src_final_path)
# Count the number of rows in the source DataFrame
src_count = df.count()
print("Source count:", src_count)
# Remove duplicates
df1 = df.dropDuplicates()
# Count the number of rows in the destination DataFrame
dest_count = df1.count()
print("Destination count:", dest_count)
# Write the cleaned data to the destination path
df1.write.mode("overwrite").format("csv").option("header", True).save(dest_final_path)
# Exit the notebook with success message and counts
print("Success: Source count = " + str(src_count) + ", Destination count = " + str(dest_count))
except Exception as e:
# Handle exceptions and exit with an error message
dbutils.notebook.exit("Error: " + str(e))
```

```
1  src_path="/mnt/global/raw"
```

Cmd 3

```
1  dbutils.widgets.text('processeddate','') # 2023/08/30
2  dbutils.widgets.text('foldername','')    # cust
```

Command took 0.07 seconds -- by sravanazure72710@gmail.com at 8/30/2023, 6:45:22

Cmd 4

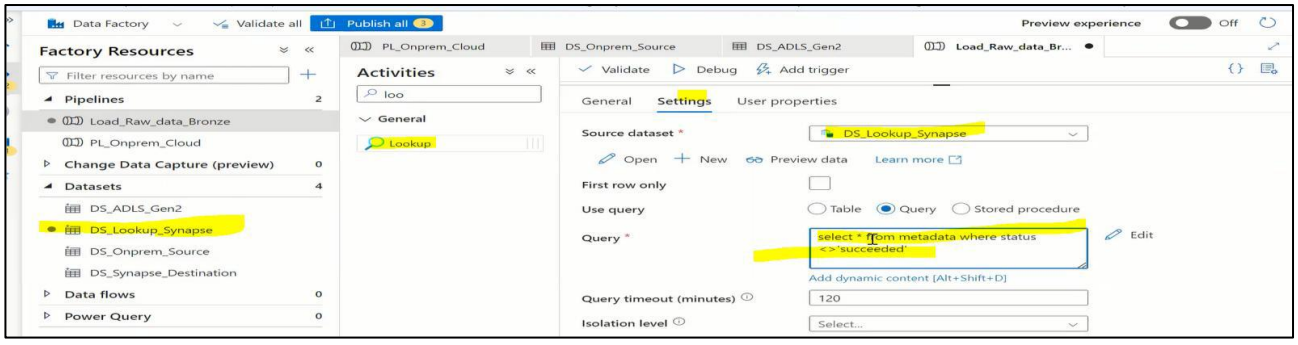
```
1  foldername=dbutils.widgets.get('foldername','')
2  pdate=dbutils.widgets.get('processeddate','')
```

Cmd 5

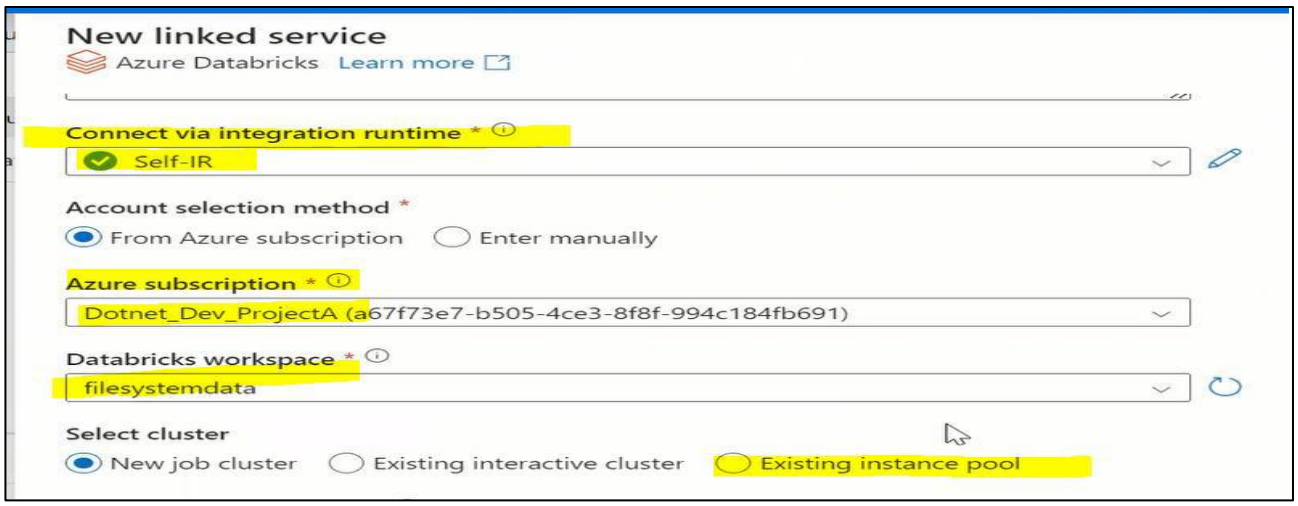
```
1  src_final_path =src_path+pdate+"/"+pdate
```

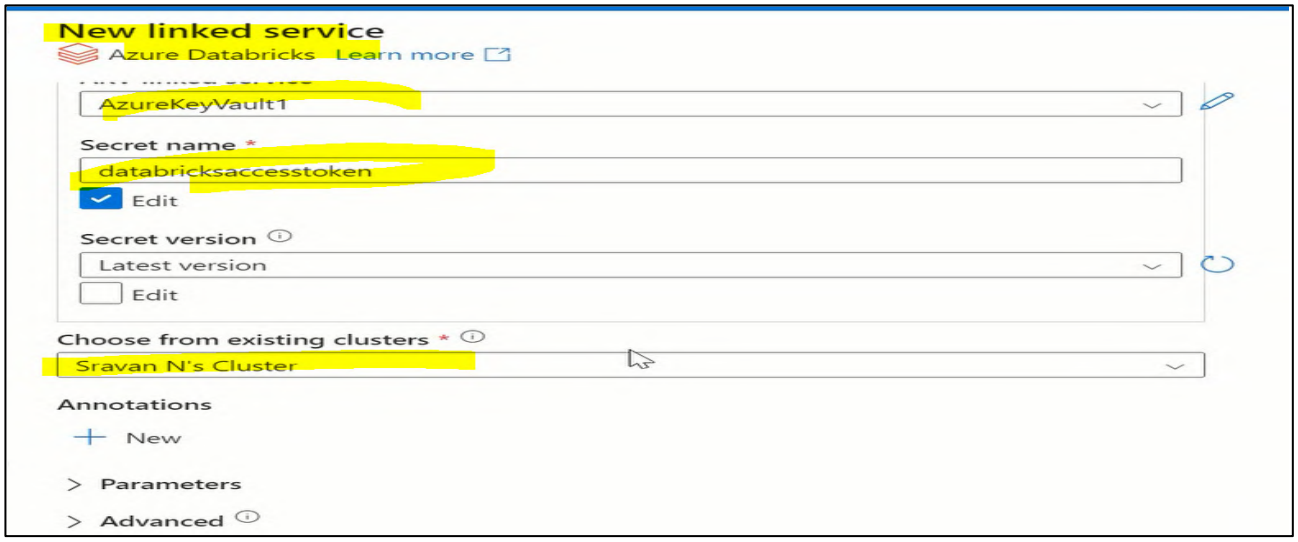
Step 17→ go to ADF to create the pipeline for the movement of data from raw to bronze

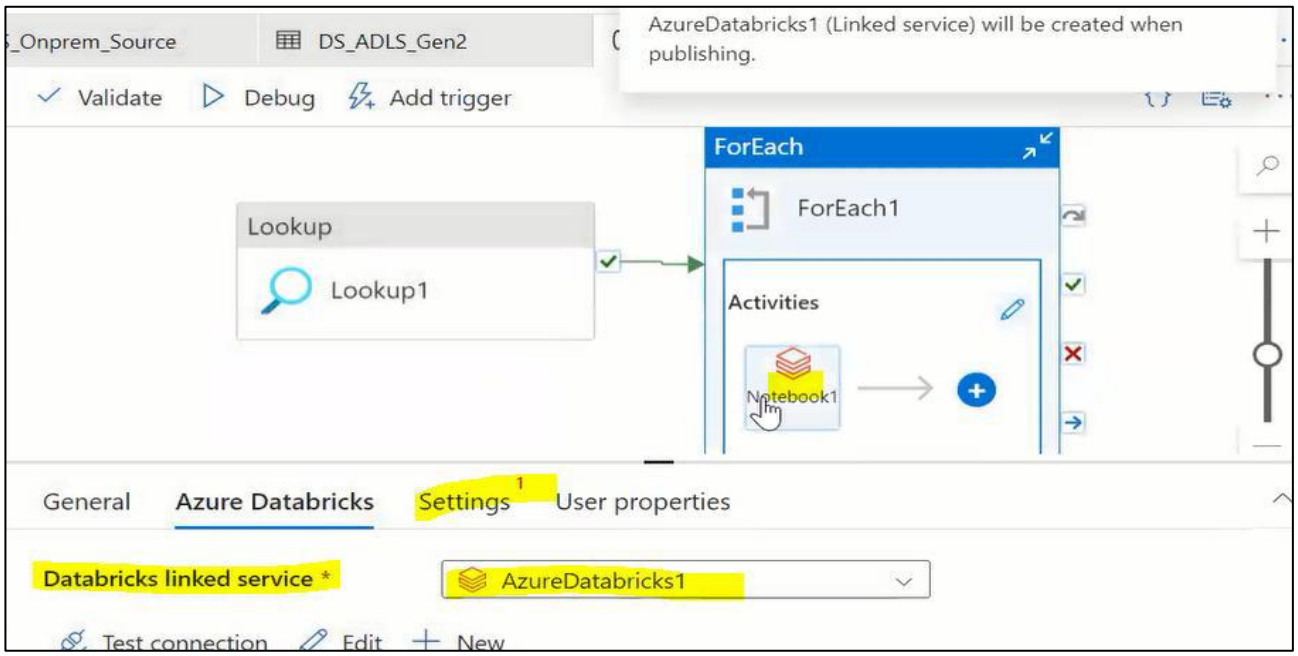
Take the lookup activity →create the source dataset for synapse →



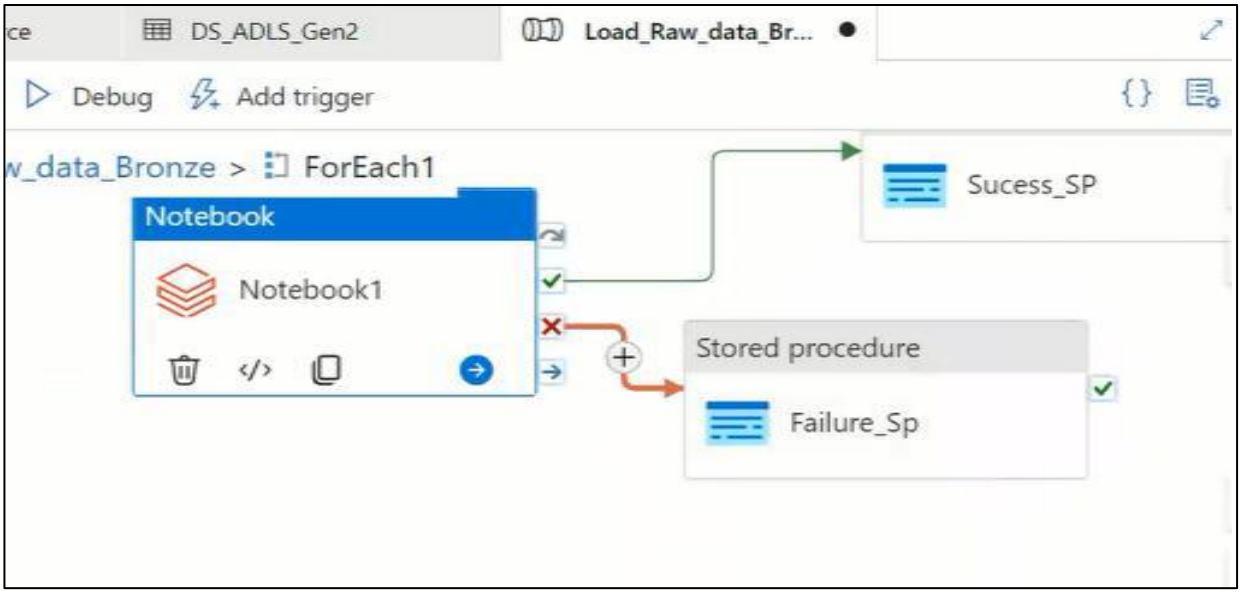
Next take→ foreach activity→ take output of lookup as input→ go inside for each → take notebook activity → create linked service→ dataset for notebook

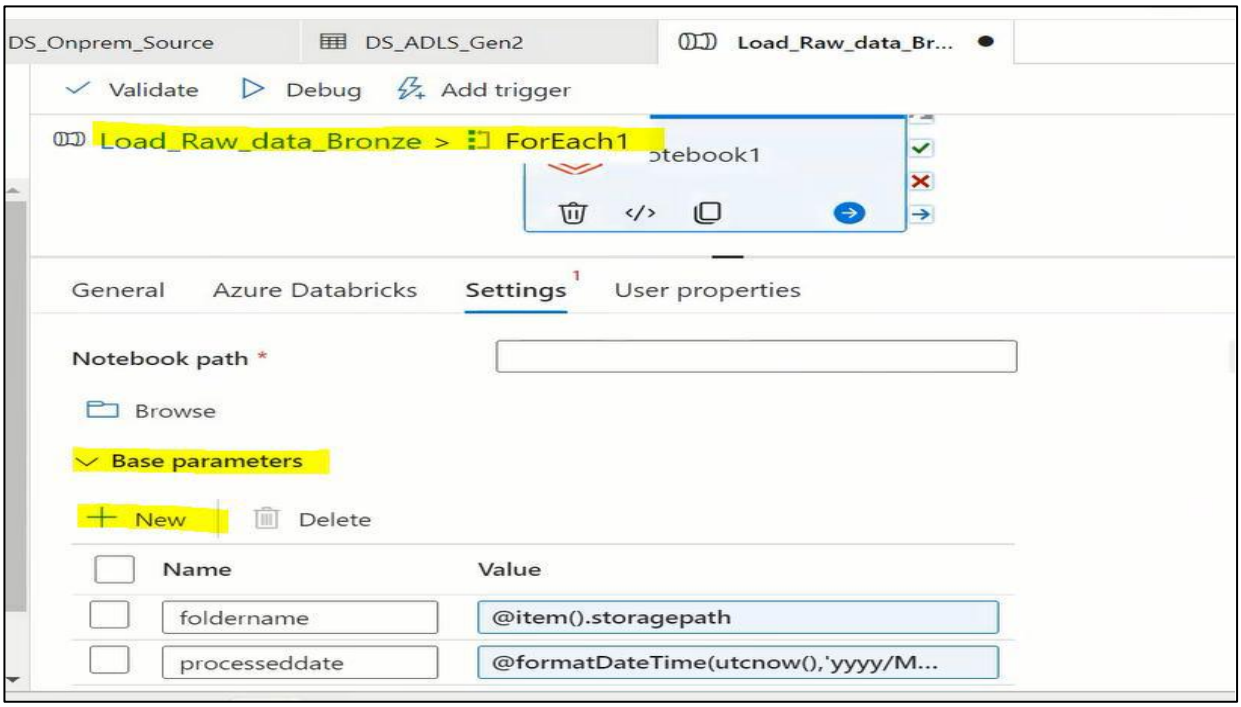


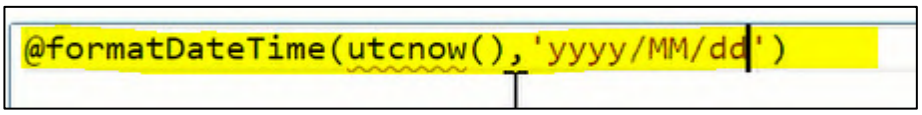




Under setting→ we have to pass 2 base parameters (as we have take processeddate and foldername as widgets in notebook)



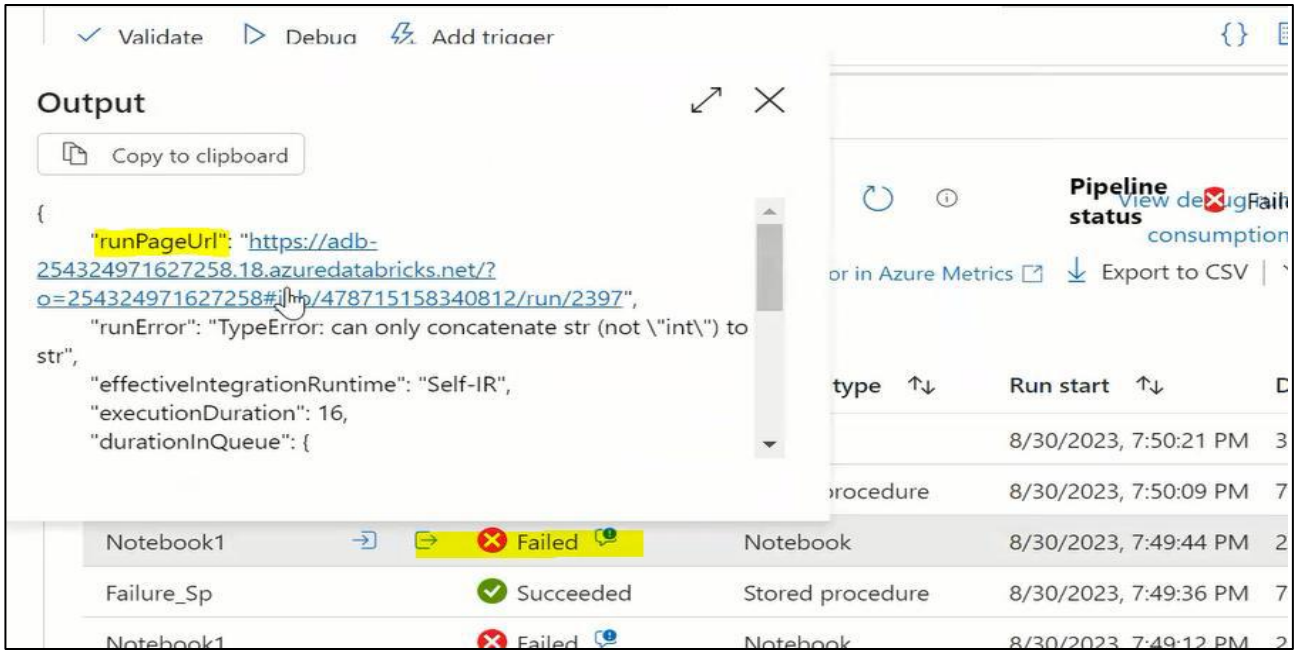


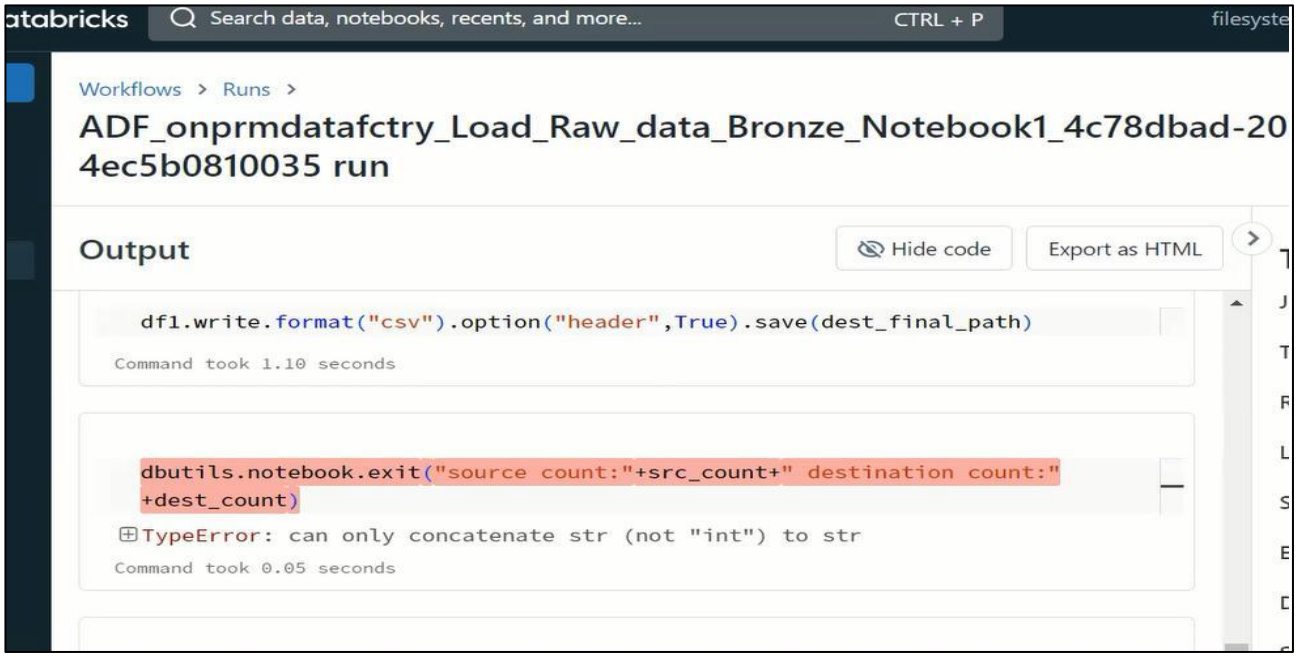


Simialry to last pipeline→following the notebook activity take 2 store proc , 1 for success and other for failure add the parameter

Outside the for each take reset store proc and web activity for failure where we call the logic app

If any errors in notebook, pipeline fails and in output of the activity we get the notebook url, click on that and it directly goes to ADB, and one with highlighted one is having error fix it and run the pipeline again.





cannot edit here directly→ so go to the main development branch and edit the code.

Step 18→ to do transformation and move data from bronze layer to Silver layer

Below is the scripts to perform transformation in notebook from Bronze to silver, here only join transfprmtion for cust table is performed

```
# Set source and destination paths

src_path = "/mnt/global/bronze/"

dest_path = "/mnt/global/silver/"

# Input widgets for folder name and processing date

dbutils.widgets.text('foldername', '')

dbutils.widgets.text('pdate', '')

try:

# Get user input for folder name and processing date

foldername = dbutils.widgets.get('foldername')

pdate = dbutils.widgets.get('pdate')

print("Folder Name:", foldername)

print("Processing Date:", pdate)

# Create source and destination paths based on user input

src_final_path = src_path + foldername + "/" + pdate

print("Source Path:", src_final_path)

# Destination path for writing processed data

dest_final_path = dest_path + 'dim' + foldername

print("Destination Path:", dest_final_path)

# Load data from the source path

df = spark.read.format("csv").option("header", True).load(src_final_path)

src_count = df.count()

print("Source Count:", src_count)

# Display the DataFrame

df.show()

# Create a sample DataFrame (df11) - replace this with your actual data

df11 = spark.createDataFrame([(2, '78654345'), (3, '67865467')], ['cid', 'cphone'])

df11.show()

# Join dataframes if foldername is 'cust', otherwise use df as is

from pyspark.sql.functions import col

if foldername == 'cust':

df1 = df.alias('a').join(df11.alias('b'), col('a.cid') == col('b.cid'), "inner").select('a.*', 'b.cphone')

df1.show()
```

```
else:
df1 = df
# Count rows in the destination DataFrame
dest_count = df1.count()

# Write processed data to the destination path
df1.coalesce(1).write.mode("overwrite").format("csv").option("header", True).save(dest_final_path)
print("Processing completed successfully.")
print("Source Count:", src_count)
print("Destination Count:", dest_count)
dbutils.notebook.exit("Processing completed successfully.")
except Exception as e:
print("Error:", str(e))
dbutils.notebook.exit("Error: " + str(e))
```

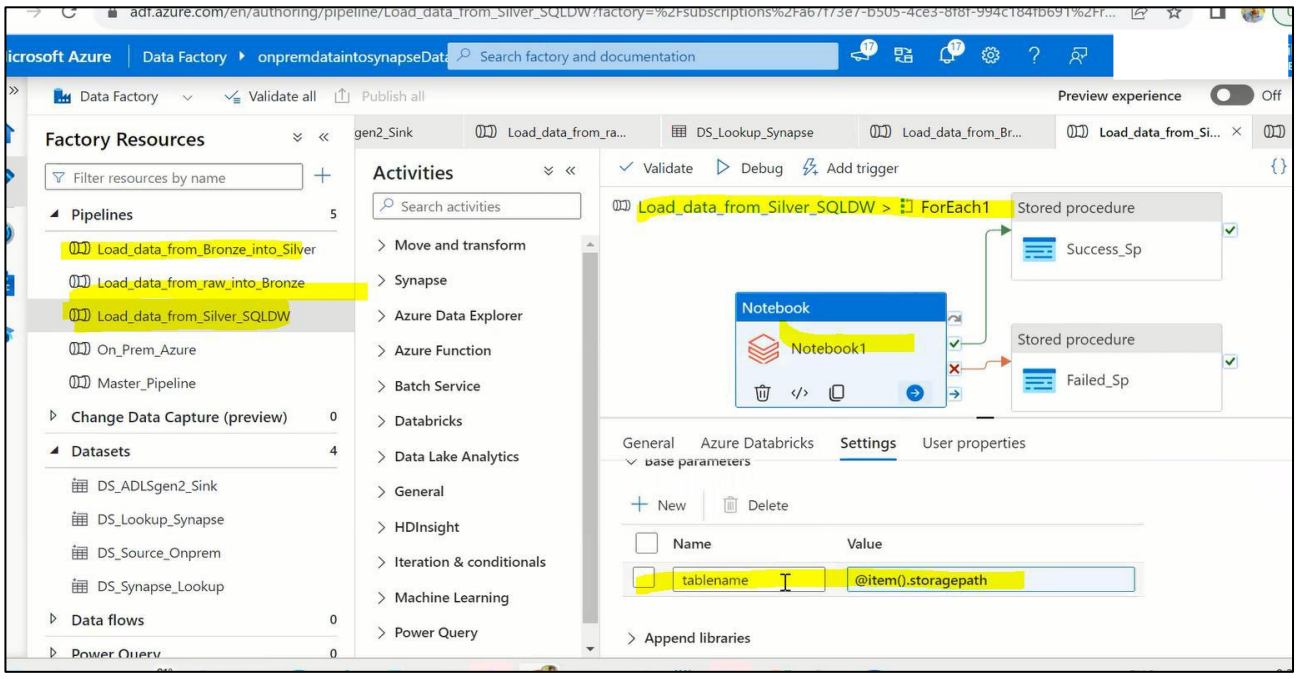
Create a pipeline similar to above raw to Bronze, change the notebook and provide the base parameters properly

Step 19→ move data from Silver layer to Sql DW

```
print("Source Count:", src_count)
print("Destination Count:", dest_count)
# Load SQL data into the data warehouse
dbutils.widgets.text('foldername', '')
foldername = dbutils.widgets.get('foldername')
print("Folder Name:", foldername)
# Set source and destination paths for SQL data
src_path = "/mnt/global/silver/" + 'dim' + foldername
dest_path = "dim" + foldername
print("Source Path:", src_path)
print("Destination Path:", dest_path)
# Read data from the source path
df = spark.read.format("csv").option("header", True).load(src_path)
src_count = df.count()
print("Source Count:", src_count)
# Set Azure Storage account key
```

```
spark.conf.set("fs.azure.account.key.onpremdatasynasegen.dfs.core.windows.net",
"o82RdY56QpidiJOBzA0+c0xBYomGajKVXZ8oZKRr+TtVSjYOTI5+i6IVTmOFL5E73Ha5wJHe7aQ1+AStdI
FwNA==")

# Write data to SQL Data Warehouse (using JDBC connection from key vault)

df.write \

.mode("overwrite") \

.format("com.databricks.spark.sqldw") \

.option("url", dbutils.secrets.get(scope="adlsgenkey", key="sqljdbcpwd")) \

.option("dbtable", dest_path) \

.option("tempDir", "abfss://global@onpremdatasynasegen.dfs.core.windows.net/tmp/synapse") \

.option("forwardSparkAzureStorageCredentials", "true") \

.save()

# Display source count

print("Source Count:", src_count)

dbutils.notebook.exit("Source Count: " + str(src_count) + ", Destination Count: " + str(dest_count)
```

Create a pipeline similar to above here we give only 1 base parameter



Step 20→Create a master pipeline to execute all the pipeline using execute pipeline activity