

## **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:

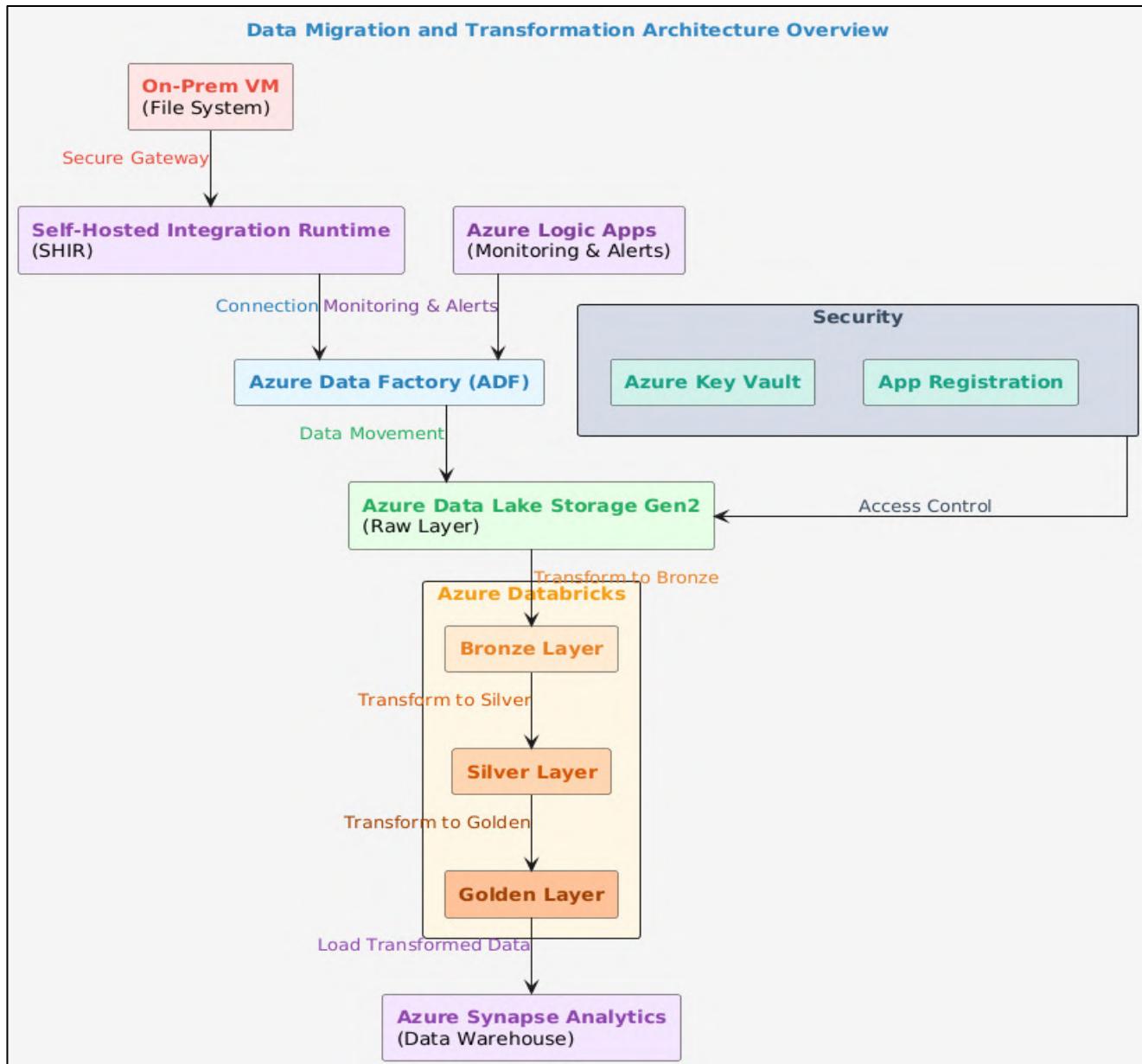
Used to securely connect Databricks to ADLS Gen2 through mounted access. This ensures secure and consistent connectivity.

### 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.



## 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:

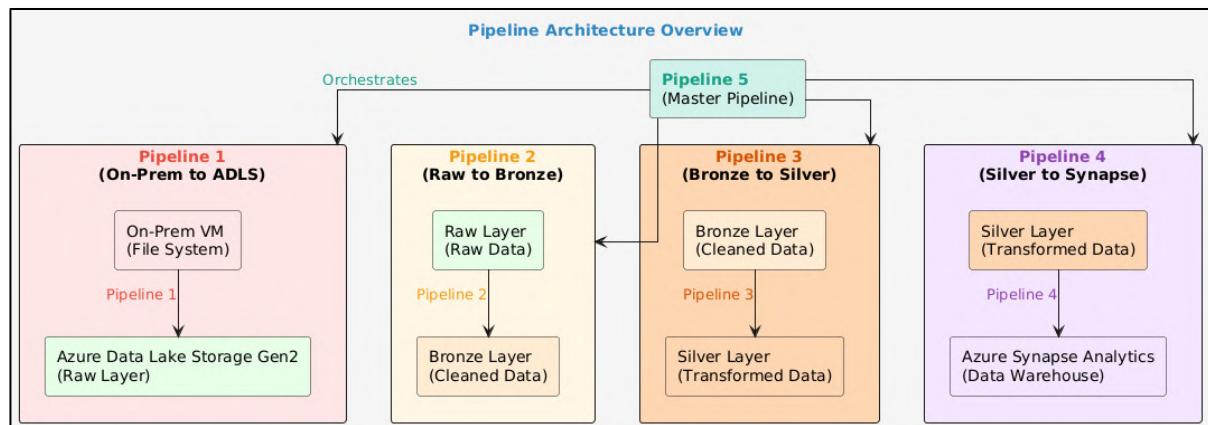
- App Registration enables token-based secure access to storage.
- Key Vault holds credentials, which are accessed by ADF and Databricks as needed.

### 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.

The screenshot shows the 'Create a virtual machine' wizard on the 'Basics' tab. Key fields filled in include:

- Subscription:** Azure for Students
- Resource group:** svkdev
- Virtual machine name:** onpremenvm
- Region:** (US) East US
- Availability options:** Availability zone
- Availability zone:** Zones 1

A note at the bottom states: "You can now select multiple zones. Selecting multiple zones will create one VM per zone." Below the form are navigation buttons: 'Review + create' (highlighted in blue), '< Previous', and 'Next : Disks >'.

The screenshot shows the 'Create a virtual machine' wizard on the 'Configuration' tab. Key fields filled in include:

- Security type:** Standard
- Image:** SQL Server 2019 Enterprise on Windows Server 2019 - x64 Gen1
- VM architecture:** x64 (radio button selected)
- Run with Azure Spot discount:** (checkbox)
- Size:** Standard\_D2s\_v3 - 2 vcpus, 8 GiB memory (₹10,778.07/month)
- Administrator account:**
  - Username:** shivvk
  - Password:** (redacted)
  - Confirm password:** (redacted)

Below the administrator account section is a note: "Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular".

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  Enable

Login name \* shivvk

Password \*  ······

Azure Key Vault integration  Disable

**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.

The screenshot shows the Azure portal interface for managing a storage account. The top navigation bar includes 'Search resources, services, and docs (S-7)'. Below it, the breadcrumb navigation shows 'All services > onpremeadls\_1698390608681 | Overview > onpremeadls | Containers > global Container'. The main content area displays the 'global' container settings. It shows the 'Authentication method' as 'Access key' and the 'Location' as 'global'. On the left, there's a sidebar with 'Overview', 'Diagnose and solve problems', 'Access Control (IAM)', 'Properties', and 'Metadata'. On the right, a table lists three items under 'Name': 'bronze', 'raw', and 'silver', each with a checkbox next to it. There are also 'Search blobs by prefix (case-sensitive)' and 'Change tier' buttons.

## 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.

The screenshot shows the 'Create a key vault' wizard in the Azure portal. The current step is 'Access configuration'. The 'Access configuration' tab is selected, while 'Basics', 'Networking', and 'Tags' are other tabs available. The 'Configure data plane access for this key vault' section contains a note about authentication and authorization. The 'Permission model' section shows 'Vault access policy' selected (radio button). The 'Resource access' section lists options for Azure VM deployment, Resource Manager template deployment, and Azure Disk Encryption. The 'Access policies' section allows creating new policies. A table at the bottom lists a user named 'NAMITHA V KOTUR' with email 'NAMITHA.20201ECE0043@presidencyuniversity.in' and permissions including 'Get, List, Update, Create, Import, Delete, Recover, Backup, Restore' and 'Get, List, Set, Delete, Recover, Backup, Restore'.

1 Permissions    2 Principal    3 Application (optional)    4 Review + create

Configure from a template  
Key, Secret, & Certificate Management

Key permissions	Secret permissions	Certificate permissions
Key Management Operations <input checked="" type="checkbox"/> Select all <input checked="" type="checkbox"/> Get <input checked="" type="checkbox"/> List <input checked="" type="checkbox"/> Update <input checked="" type="checkbox"/> Create <input checked="" type="checkbox"/> Import <input checked="" type="checkbox"/> Delete <input checked="" type="checkbox"/> Recover <input checked="" type="checkbox"/> Backup <input checked="" type="checkbox"/> Restore	Secret Management Operations <input checked="" type="checkbox"/> Select all <input checked="" type="checkbox"/> Get <input checked="" type="checkbox"/> List <input checked="" type="checkbox"/> Set <input checked="" type="checkbox"/> Delete <input checked="" type="checkbox"/> Recover <input checked="" type="checkbox"/> Backup <input checked="" type="checkbox"/> Restore	Certificate Management Operations <input checked="" type="checkbox"/> Select all <input checked="" type="checkbox"/> Get <input checked="" type="checkbox"/> List <input checked="" type="checkbox"/> Update <input checked="" type="checkbox"/> Create <input checked="" type="checkbox"/> Import <input checked="" type="checkbox"/> Delete <input checked="" type="checkbox"/> Recover <input checked="" type="checkbox"/> Backup <input checked="" type="checkbox"/> Restore <input checked="" type="checkbox"/> Manage Contacts <input checked="" type="checkbox"/> Manage Certificate Authorities <input checked="" type="checkbox"/> Get Certificate Authorities <input checked="" type="checkbox"/> List Certificate Authorities <input checked="" type="checkbox"/> Set Certificate Authorities
Cryptographic Operations <input type="checkbox"/> Select all <input type="checkbox"/> Decrypt <input type="checkbox"/> Encrypt	Privileged Secret Operations <input type="checkbox"/> Select all <input type="checkbox"/> Purge	

[Previous](#) [Next](#)

## Create an access policy

1 Permissions    2 Principal    3 Application (optional)    4 Review + create

Only 1 principal can be assigned per access policy.  
Use the new embedded experience to select a principal. The previous popup experience can be accessed here. [Select a principal](#)

adfdev

adfdevsvk  
a3fc6c6a-6914-4b89-81eb-6c617c99e2f6

## Step 5 - Deploy Dedicated SQL Pool (Synapse):

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

Microsoft Azure

Home > Dedicated SQL pools (formerly SQL DW) > Create dedicated SQL pool (formerly SQL DW)

Configure performance

Configure your performance level that best fits your needs.

**Gen2** Offers the highest performance and storage scalability options for intensive workloads.  
Starting at 118.59 INR / hour

**Gen1** Offers lower compute scale options for less demanding workloads.  
Not available  
Starting at -- / hour

Learn more about pricing  
Price of your system  
DW100c 118.59 INR / hour  
100 cDWU

**Project details**  
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription: Azure for Students  
Resource group: svkdev

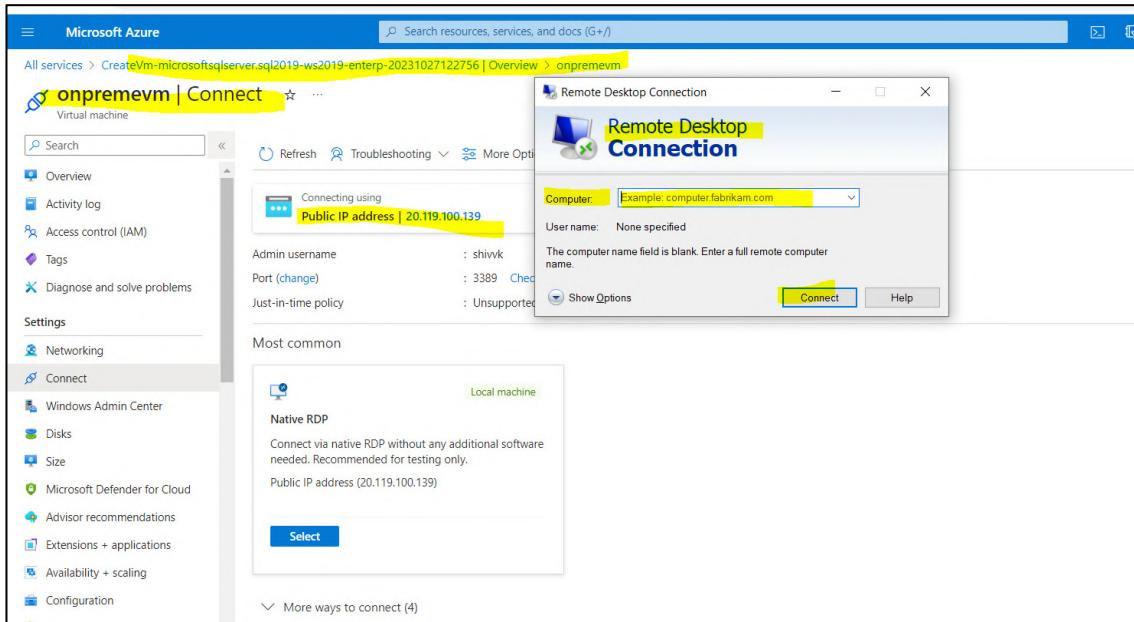
**SQL pool details**  
Enter required settings for this SQL pool, including picking a logical server and configuring the performance level.

SQL pool name: testpool  
Server: svkloodata (East US)  
Performance level: Gen2 DW1000c  
Select performance level

[Review + create](#) [Next : Networking >](#) [Apply](#)

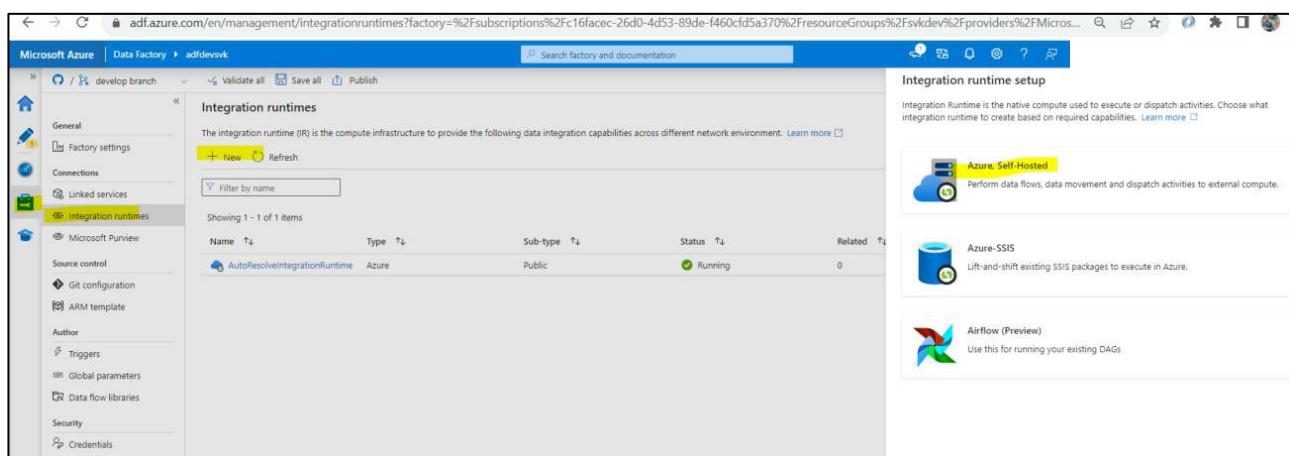
## Step 6 - Remote into the VM:

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** [Yellow Box]  
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](#)

**Buttons:** 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](#) [Yellow Box]

**Option 2: Manual setup**  
**Step 1: Download and install integration runtime**  
Step 2: Use this key to register your integration runtime

<b>Name</b>	<b>Authentication key</b>
Key 1	IR@fcfaac77-4cc0-428f-b6c2-d5c36af36cfa@adfdevsvk@ServiceEndpoint
Key 2	IR@fcfaac77-4cc0-428f-b6c2-d5c36af36cfa@adfdevsvk@ServiceEndpoint

**Buttons:** Close

**Server Manager**

Server Manager • Local Server

**Local Server** [Yellow Box]

**PROPERTIES** For onpremenvm

Computer name	onpremenvm	Last installed updates	10/17/2023 2:05 AM
Workgroup	WORKGROUP	Windows Update	Download updates only, using Microsoft Update
		Last checked for updates	Today at 7:14 AM
Windows Defender Firewall	Public: On	Windows Defender Antivirus	Real-Time Protection: On
Remote management	Enabled	Feedback & Diagnostics	Settings
Remote Desktop	Enabled	<b>IE Enhanced Security Configuration</b> On	Time zone
NIC Teaming	Disabled		(UTC) Coordinated Universal Time
Ethernet 2	IPv4 address assigned by DHCP, IPv6 enabled	Product ID	00430-00000-0000-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

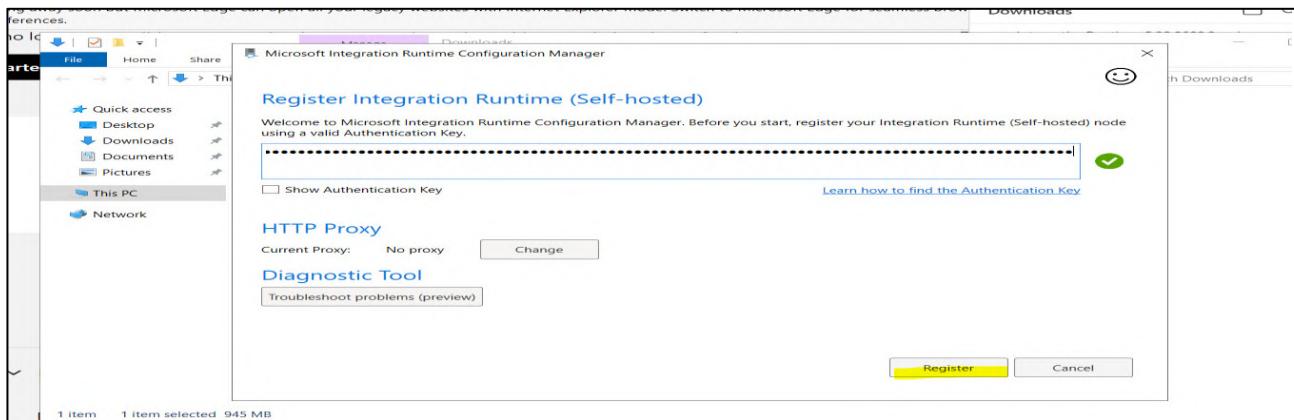
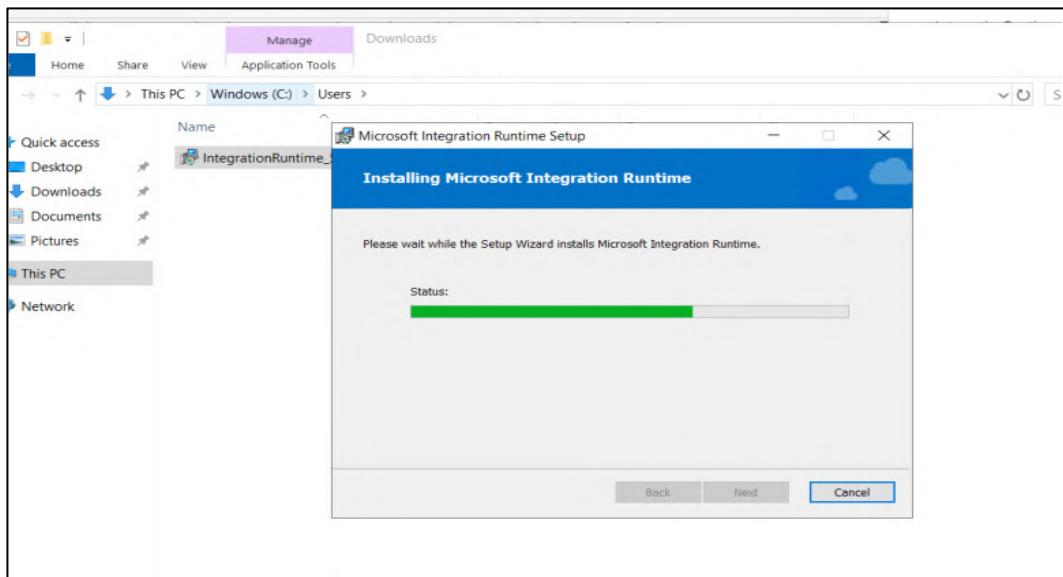
**EVENTS**  
All events | 17 total

**Filter**

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



## Step 8 - Connect Synapse Pool to SSMS:

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

The screenshot shows the Azure portal interface for managing a Dedicated SQL pool. The main title is "testpool (svkloaddata/testpool)". On the left, there's a sidebar with options like Overview, Activity log, Tags, Diagnose and solve problems, Settings (Workload management, Maintenance schedule, Quick start, Geo-backup policy, Connection strings, Properties, Locks), Security (Auditing, Data Discovery & Classification, Dynamic Data Masking, Microsoft Defender for Cloud, Transparent data encryption), and Notifications. The main content area has tabs for Overview, Features (4), and Tasks (5). Under Features, there are four cards: Transparent data encryption (NOT CONFIGURED), Auditing (NOT CONFIGURED), Geo-backup (ENABLED), and Microsoft Defender for SQL (NOT CONFIGURED). A prominent yellow box highlights the "Server name" field, which contains "svkloaddata.database.windows.net". Above this field, a tooltip says "Copied".

## 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.

The screenshot shows a Windows File Explorer window with the path "C:\data". The left pane shows "Quick access" with links to Desktop, Downloads, Documents, Pictures, and This PC. The right pane displays a list of four file folders: "cust", "discounts", "emp", and "orders", all of which were modified on 10/27/2023 at 8:06 AM. The "Type" column indicates they are all "File folder".

## 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:
    dmcmd [ -EnableRemoteAccess "<ports>" ["<thumbprint>"] -EnableRemoteAccessInContainer "<port>" ["<thumbprint>"] -DisableRemoteAccess -Key "<AuthenticationKey>" -GenerateBackupFile "<filePath>" "<password>" -ImportBackupFile "<filePath>" "<password>" -SwitchServiceAccount "<domain\user>" ["<password>"] -LogLevel <logLevel> -EventLogVerboseSetting <on/off> -EnableLocalMachineAccess -DisableLocalMachineAccess -EnableLocalFolderPathValidation -DisableLocalFolderPathValidation -EnableExecuteSsisPackage -DisableExecuteSsisPackage -GetExecuteSsisPackage ]
```

Detail:

```
-EnableRemoteAccess "<ports>" [<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 -Di 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 dmvcmd.exe validation is disabled correctly, the connection will succeed.

New linked service

File system [Learn more](#)

Connect via integration runtime \* ⓘ

SelfhostedIR

Host \* ⓘ

C:\Data

User name \*

Password [Azure Key Vault](#)

Annotations

+ New

> Parameters

> Advanced ⓘ

Create Back [Test connection](#) Cancel

Validate all [Publish all](#)

### Linked services

Linked service defines the connection information to a data store or compute. [Learn more](#)

+ New

Filter by name Annotations : Any

Showing 1 - 3 of 3 items

Name ↑↓	Type ↑↓	Related ↑↓
AzureKeyVault1	Azure Key Vault	2
LS_ADLSGEN2_sink	Azure Data Lake Storage Gen2	0
LS_OnpremSource	File system	0

## 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.

Microsoft Azure Search resources, services, and docs (G+/-)

Home > Microsoft.SQLDataWarehouse.NewDatabaseNewServerV4\_bab9c8ddf38f44 | Overview > testpool (destinserver/testpool)

testpool (destinserver/testpool) | Connection strings

Dedicated SQL pool (formerly SQL DW)

Search

Settings

- Workload management
- Maintenance schedule
- Quick start
- Geo-backup policy
- Connection strings**
- Properties
- Locks

Security

- Auditing
- Data Discovery & Classification
- Dynamic Data Masking
- Microsoft Defender for Cloud

ADO.NET JDBC ODBC PHP Go

ADO.NET (Active Directory passwordless authentication)

Microsoft.Data.SqlClient Quickstart Entity Framework Core Quickstart

Server=tcp:destinserver.database.windows.net,1433;Initial Catalog=testpool;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;Authentication="Active Directory Default";

ADO.NET (SQL authentication)

Server=tcp:destinserver.database.windows.net,1433;Initial Catalog=testpool;Persist Security Info=False;User ID=sravan;Password=your\_password;MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;

Copied

## 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.

Microsoft Azure Search resources, services, and docs (G+/-)

Home > Microsoft.SQLDataWarehouse.NewDatabaseNewServerV4\_bab9c8ddf38f44 | Overview > testpool (destinserver/testpool)

testpool (destinserver/testpool) | Open in Visual Studio

Dedicated SQL pool (formerly SQL DW)

Search

Common Tasks

- Query editor (preview)
- Build dashboards + reports
- Model and cache data
- Open in Visual Studio**

Monitoring

- Query activity
- Alerts
- Metrics
- Diagnostic settings
- Logs

Automation

Visual Studio

Configure Firewall

Configure your firewall settings to ensure this machine can access the database.

Get Visual Studio

Download and install SQL Server Data Tools on Visual Studio 2015 Update 2, Visual Studio 2013 Update 4 or a later version.

Download Visual Studio

Download SQL Server Data Tools

Open in Visual Studio

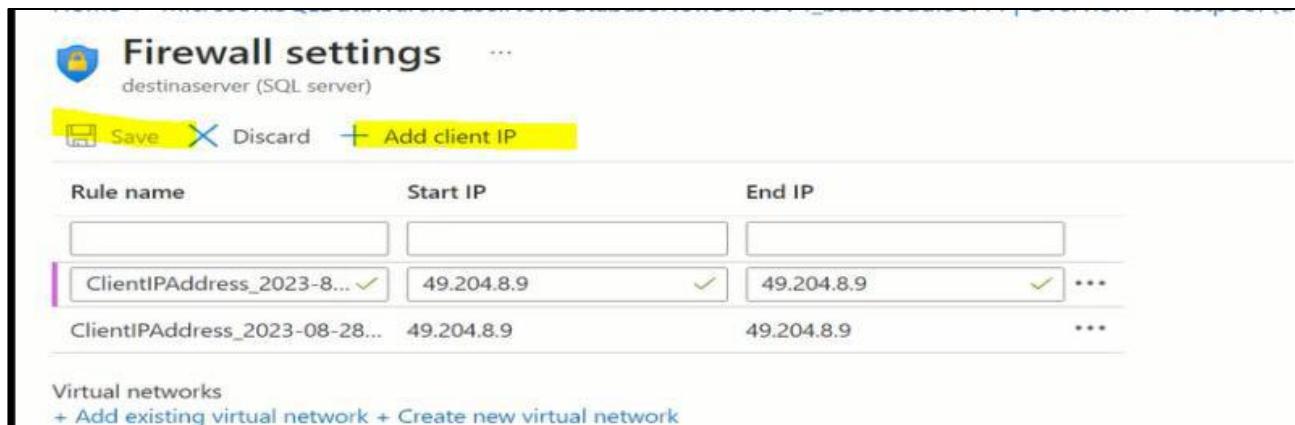
**Firewall settings** ...

destinaserver (SQL server)

Save Discard + Add client IP

Rule name	Start IP	End IP
ClientIPAddress_2023-8...	49.204.8.9	49.204.8.9
ClientIPAddress_2023-08-28...	49.204.8.9	49.204.8.9

Virtual networks  
+ Add existing virtual network + Create new virtual network

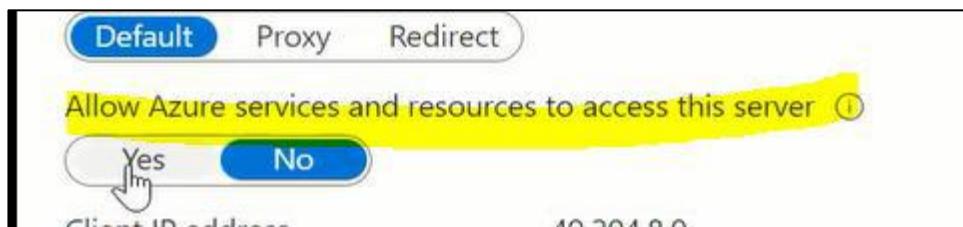


Default Proxy Redirect

Allow Azure services and resources to access this server ⓘ

Yes No

Client IP address 49.204.8.9



#### 4. Firewall Configuration:

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

New linked service

Azure Synapse Analytics Learn more

SelfhostedIR

Connection string Azure Key Vault

AKV linked service \* ⓘ

AzureKeyVault1

Secret name \* ⓘ

synapsepassword

Edit

Secret version ⓘ

Latest version

Edit

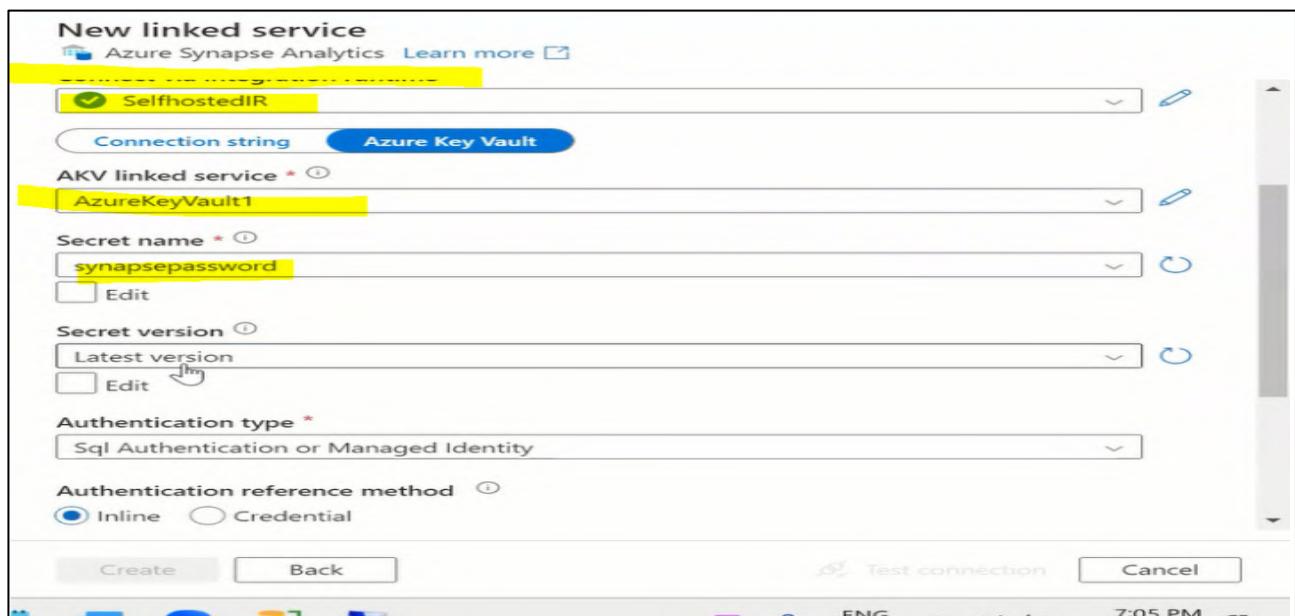
Authentication type \*

Sql Authentication or Managed Identity

Authentication reference method ⓘ

Inline Credential

Create Back Test connection Cancel



## 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 (
    sourcefilename VARCHAR(50),
    storagepath VARCHAR(50),
    isactive INT,
    status VARCHAR(50)
);
```

```
-- Insert initial folder details
```

```
INSERT INTO metadata (sourcefilename, 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
```

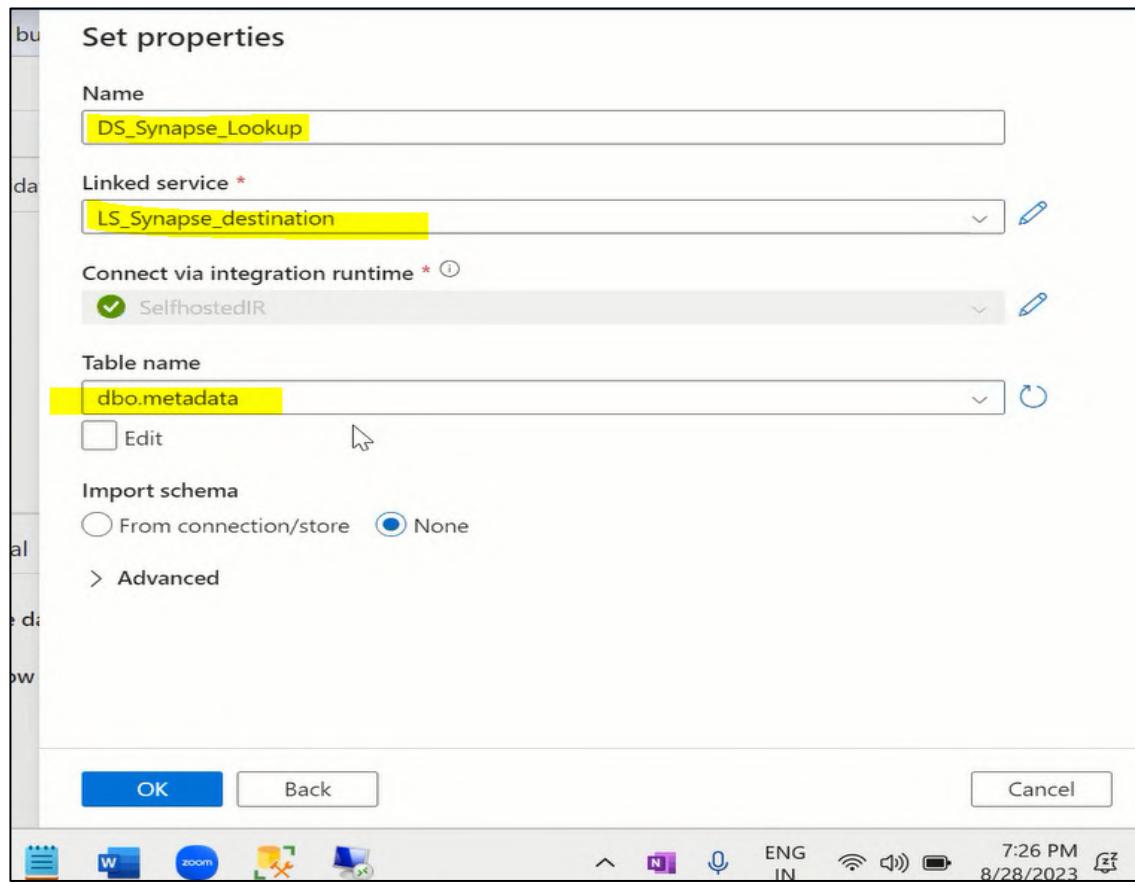
```
CREATE PROCEDURE metadata_usp (@status VARCHAR(50), @sourcefilename VARCHAR(50))
AS
BEGIN
    UPDATE metadata
    SET status = @status
    WHERE sourcefilename = @sourcefilename;
END;
```

```
-- Create stored procedure to reset status
```

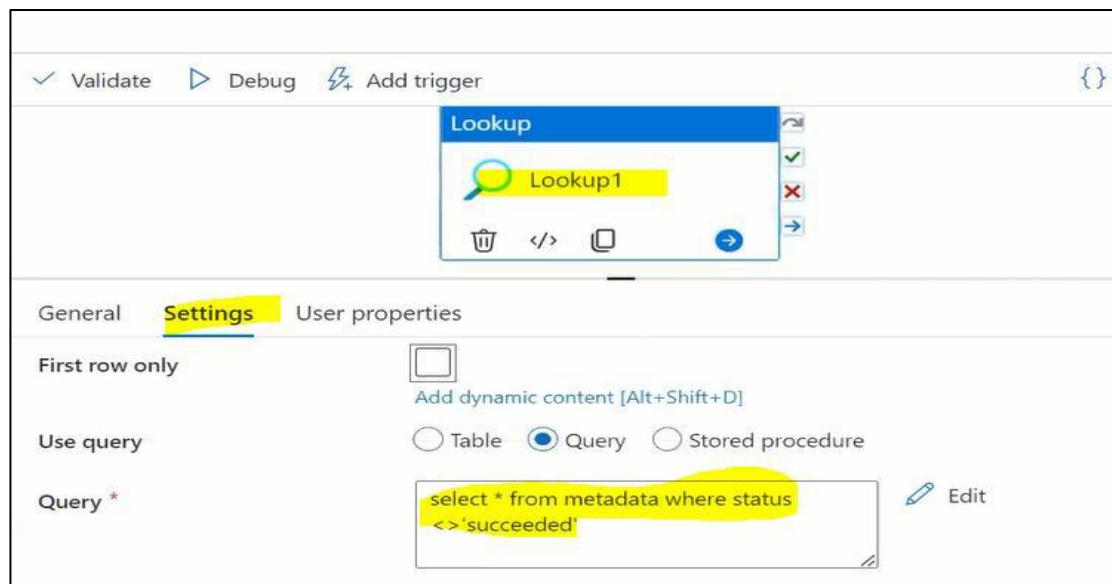
```
CREATE PROCEDURE reset_status_usp
AS
BEGIN
    UPDATE metadata
    SET status = 'ready';
END;
```

## 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

## Pipeline expression builder

Add dynamic content below using any combination of [expressions](#), [functions](#) and [system variables](#).

```
@activity('Lookup1').output.value
```

[?] value

value: any[ ]

Inside for each → take copy activity → copy activity

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

file system → csv file

## New dataset

In pipeline activities and data flows, reference a dataset to specify the location of data within a data store. [Learn more](#) [?]

Select a data store



fil

All

Azure

Database

File

Generic protocol

NoSQL

S



Azure File Storage



File system

**Set properties**

Name  
Ds\_FileSystem\_Sourcedataset

Linked service \*  
LS\_OnpremSource

Connect via integration runtime \* ⓘ  
 SelfhostedIR

File path  
C:\Data /  Directory /  File name | ▼

First row as header

Import schema  
 From connection/store  From sample file  None

> Advanced

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

**Set properties**

Name  
Ds\_Adlsgen2\_sinkdataset

Linked service \*  
LS\_ADLSGEN2\_sink

Connect via integration runtime \* ⓘ  
 SelfhostedIR

File path  
global | raw |  File name | ▼

First row as header

Import schema  
 From connection/store  From sample file  None

> Advanced

OK Back Cancel

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

DelimitedText  
Ds\_AdlsGen2\_sinkdataset

**Connection** Schema **Parameters**

Linked service \* LS\_ADLSGEN2\_sink Test connection Edit + New Learn more

Integration runtime \* SelfhostedIR Edit

File path \* global raw/@{dataset().dataset\_param}/@(f...)/File name Br

Compression type Select...

Column delimiter Comma (,)

Add → directory in form of

Raw/filename/yyyy/mm/dd

It should be created as folder.

Microsoft Azure | Data Factory > onprmdatfactory Search factory and documentation

Validate all Publish all

Factory Resources Pipelines Datasets

PL\_Onprem\_Cloud Ds\_Filesystem\_S...

Ds\_AdlsGen2\_sinkdataset

Pipeline expression builder

Add dynamic content below using any combination of expressions, functions and system variables.

```
raw/@{dataset().dataset_param}/@{formatDateTime.UtcNow('yyyy')}/@{formatDateTime.UtcNow('MM')}/@{formatDateTime.UtcNow('dd')}
```

Clear contents

Parameters Functions

utcnow

Collapse all Date Functions

utcnow Returns the current timestamp as a string.

Ok Cancel

Validate Validate copy runtime Debug Add trigger

PL\_Onprem\_Cloud > ForEach1

Copy data Copy data1

General Source Sink Mapping Settings User properties

Source dataset \* Ds\_Filesystem\_Sourcedataset

Open New Preview data Learn more

Dataset properties

Name	Value	Type
dataset_parameter	Value	string

File path type

- File path in dataset (radio button selected)
- File filter
- Wildcard file path
- List of files

Take output from lookup activity

```
"count": 4,  
"value": [  
    {  
        "sourcefoldername": "orders",  
        "storagepath": "orders",  
        "isactive": 0,  
        "status": "ready"  
    },  
    {  
        "sourcefoldername": "cust",  
        "storagepath": "cust",  
        "isactive": 0,  
        "status": "ready"  
    },  
]
```

Sourcefoldername

## Pipeline expression builder

Add dynamic content below using any combination of [expressions](#)

```
@item().sourcefoldername
```

Sink dataset \*

General Source **Sink** Mapping Settings User properties

Sink dataset \*

Dataset properties (1)

Name	Value
dataset_param	<input type="text" value="@item().storagepath"/>

**Sink → storage path**

## Pipeline expression builder

Add dynamic content below using any combination of [expressions](#)

```
@item().storagepath
```

Following the copy activity take store proc activity

The screenshot shows a pipeline named 'PL\_Onprem\_Cloud' with a 'ForEach1' loop. Inside the loop, there is a 'Copy data1' activity followed by a 'Stored procedure1' activity. The 'Stored procedure1' activity is highlighted with a yellow box. Below it, the 'Settings' tab is selected, showing 'Stored procedure parameters'. There are two parameters: 'sourcefilename' (String type) with value '@item().sourcefilename' and 'status' (String type) with value 'succeeded'. A tooltip 'Add dynamic content [Alt+Shift+D]' is visible.

Use → metadata\_usp as store proc,

And import parameter,

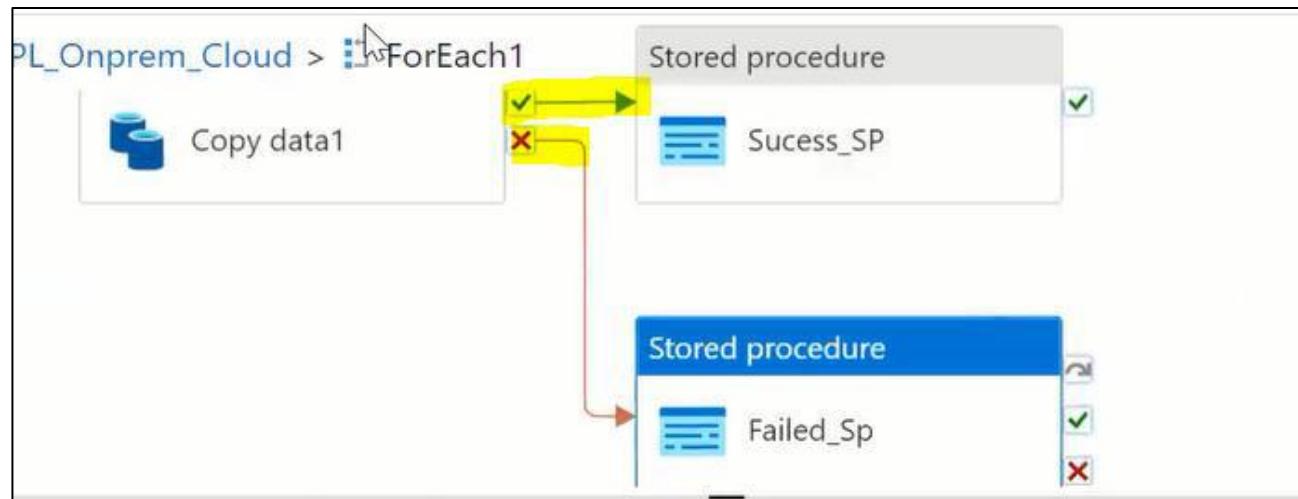
Add sourcefilename as dynamically.

Status hard code succeeded indicating the copy activity finished

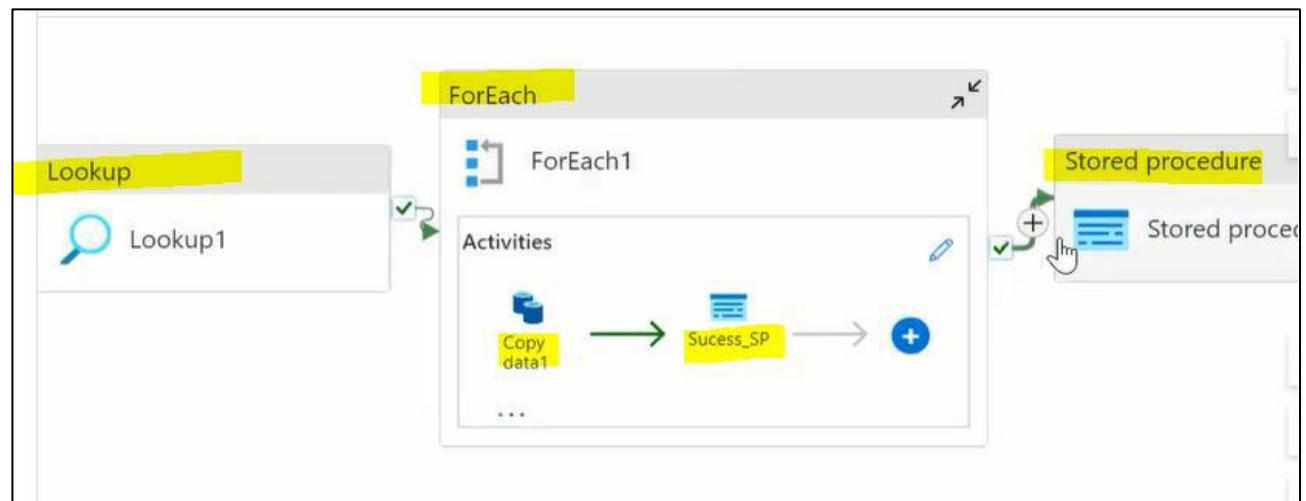
The screenshot shows a pipeline named 'PL\_Onprem\_Cloud' with a 'ForEach1' loop. Inside the loop, there is a 'Copy data1' activity followed by a 'Sucess\_SP' stored procedure activity. The 'Sucess\_SP' activity is highlighted with a yellow box. Below it, the 'Settings' tab is selected, showing 'Stored procedure name' set to '[dbo].[metadata\_usp]'. In the 'Stored procedure parameters' section, there are two parameters: 'sourcefilename' (String type) with value '@item().sourcefilename' and 'status' (String type) with value 'Failed'. A tooltip 'Add dynamic content [Alt+Shift+D]' is visible.

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;
```

## Logic Apps Designer

Save Discard Run Trigger Designer Code view Parameters Templates Connectors Help Info Try Preview Designer

Search connectors and triggers

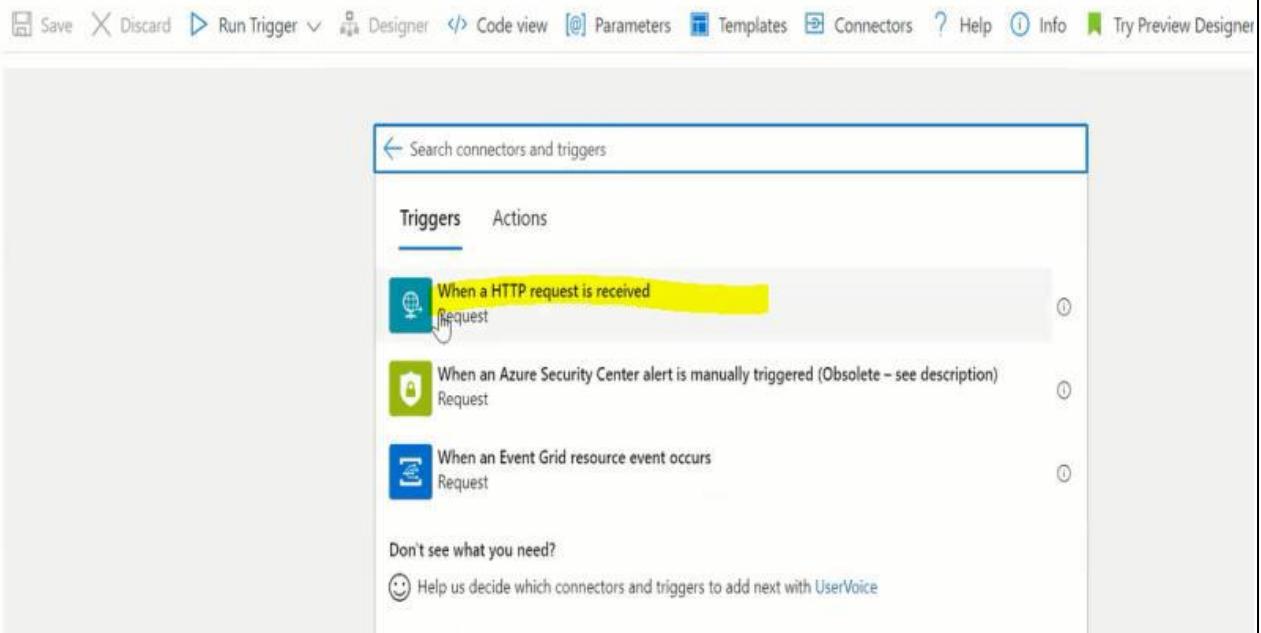
Triggers Actions

When a HTTP request is received Request

When an Azure Security Center alert is manually triggered (Obsolete – see description) Request

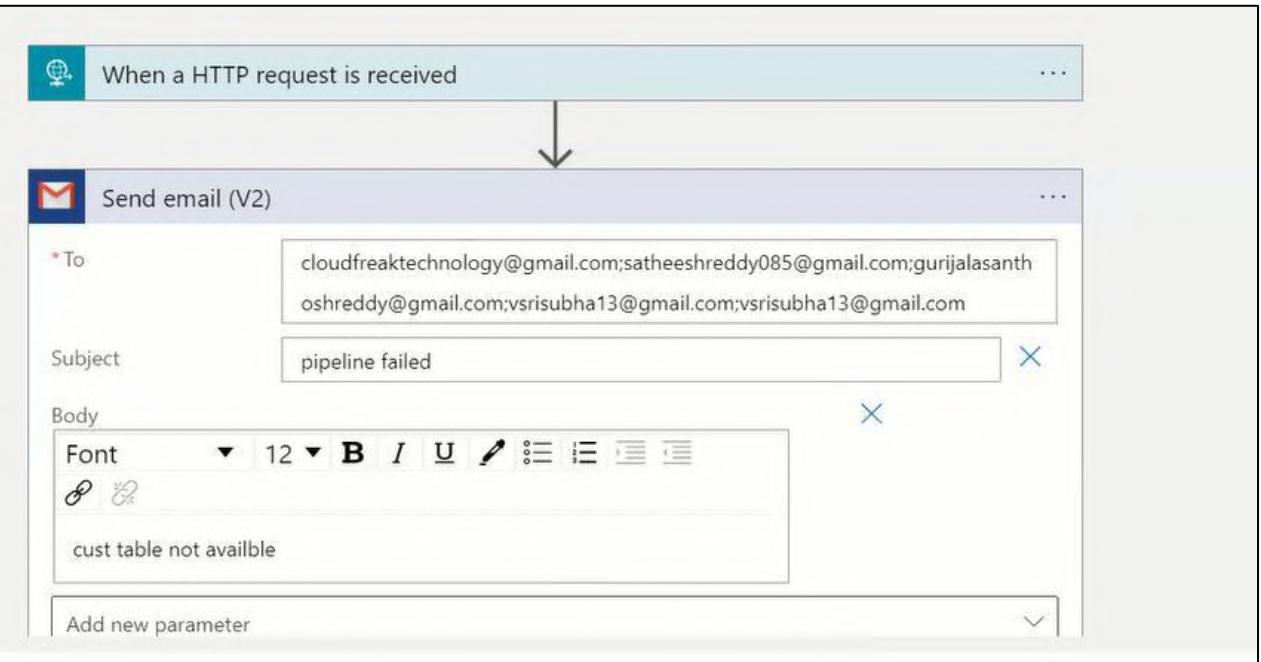
When an Event Grid resource event occurs Request

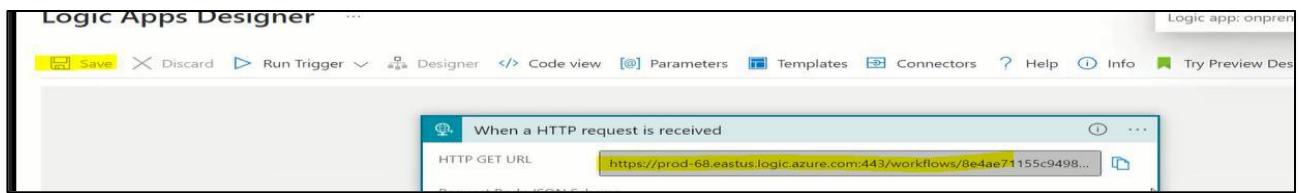
Don't see what you need? Help us decide which connectors and triggers to add next with UserVoice



### Step → Create a logic app

Create logic app → go to resource → create blank new → search for http → 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

**General**    **Settings**    User properties

**URL \*** ⓘ https://prod-68.eastus.logic.azure.com:44...

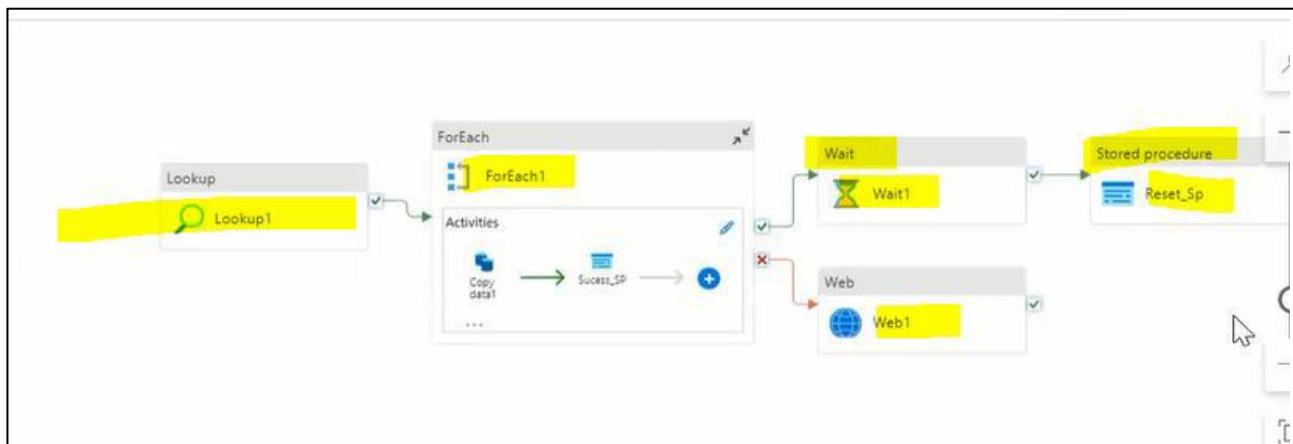
**Method \*** ⓘ GET

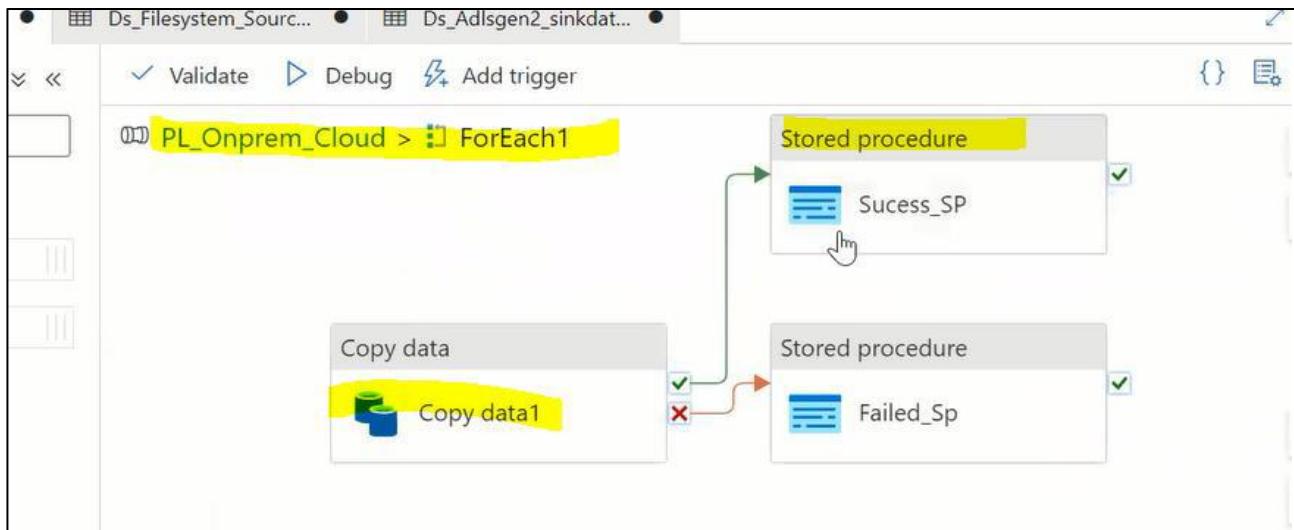
**Authentication** ⓘ None

**Headers** ⓘ + New

> Advanced

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

Name	Value	Type
dataset_parameter	@item().sourcefilename	string

If any failure like the change in file name or metadata 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

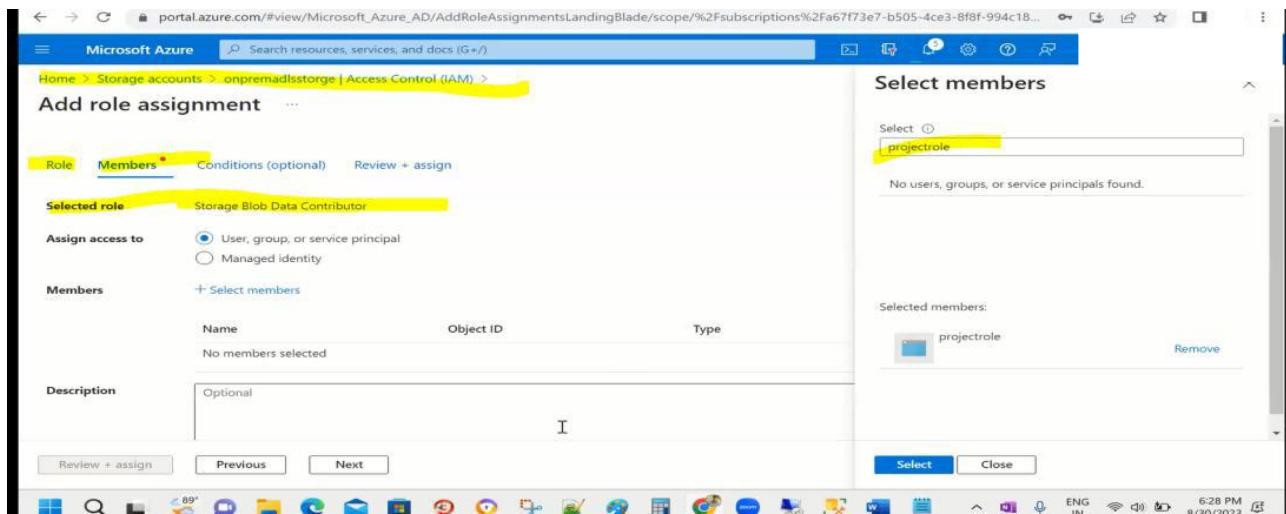
The screenshot shows the Microsoft Azure portal interface. The left sidebar has 'Default Directory' selected under 'Azure Active Directory'. The main area is titled 'Default Directory | App registrations'. It shows a list of 'Owned applications' with the following details:

Display name	Application (client) ID	Created on	Certificates & secret
armtemplates	375f5ed8-7e40-4c1d-8e79-7bcd16a...	5/7/2023	Current
azuredevopsaaccount	e0ec0f2f-63ec-41a1-a1ec-701d8139...	6/2/2023	Current
Infrdevhdhc	9d96a7dc-b8cf-4492-8b44-1b93ffe3...	5/6/2023	Current
mrgarm	761c7d73-a63a-4bb4-a093-4900c36...	5/8/2023	Current

The screenshot shows the Microsoft Azure portal interface. The left sidebar has 'Certificates & secrets' selected under 'App registrations'. The main area is titled 'projectrole | Certificates & secrets'. A message box says 'Successfully updated application projectrole credential'. The 'Client secrets' tab is selected, showing a table with one entry:

Description	Expires	Value	Secret ID
test	2/26/2024	xCh8Q~36O3mqwYo3A4WvjsfbJceelK... d08a477b-8360-40be-bd8f-5862d5f57...	d08a477b-8360-40be-bd8f-5862d5f57...

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 dbutils 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@adlsgenstorageaccountn.dfs.core.windows.net/",
    mount_point = "/mnt/global",
    extra_configs = configs)
```

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

HomePage / Create Secret Scope

## Create Secret Scope

A store for secrets that is identified by a name and backed by a specific store type. [Learn more](#)

Scope Name ?

Manage Principal ?

Creator

Azure Key Vault ?

DNS Name

<https://xxx.vault.azure.net/>

Resource ID

/subscriptions/xxxxxx/...

[Create](#)

Give the scope name → adlsgenkey

DNS name and resource ID → from azure key vault

onpremdtrvalt | Properties ☆ ...

Key vault

Search Save Discard changes Refresh

Networking

Name: onpremdtrvalt

Microsoft Defender for Cloud

Sku (Pricing tier): Standard

Properties

Location: eastus

Vault URL: <https://onpremdtrvalt.vault.azure.net/>

Locks

Resource ID: /subscriptions/a67f73e7-b505-4ce3-8f8f-9

Monitoring

Subscription ID: a67f73e7-b505-4ce3-8f8f-904r184fh691

Alerts

Home > Default Directory | App registrations >

**projectrole**

Search

Delete Endpoints Preview features

Overview Quickstart Integration assistant

Manage

- Branding & properties
- Authentication
- Certificates & secrets
- Token configuration
- API permissions
- Expose an API

Display name: **projectrole** Copy to clipboard

Application (client) ID: **2e7fb133-b159-4282-ac06-d88b3b152540** Copy

Object ID: **b79a03df-1742-45c8-8548-001b6c8a521b**

Directory (tenant) ID: **f5ea40f2-c7b8-4658-8d25-0aac8535e48c**

Supported account types: **My organization only**

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 | I

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 →

The screenshot shows the Azure Data Factory Pipeline Editor. On the left, the 'Factory Resources' sidebar lists 'Pipelines' (Load\_Raw\_data\_Bronze), 'Datasets' (DS\_AdLS\_Gen2, DS\_Lookup\_Synapse, DS\_Onprem\_Source, DS\_Synapse\_Destination), and 'Data flows'. In the main area, a pipeline named 'PL\_Onprem\_Cloud' is selected. A 'Lookup' activity is being configured under the 'Activities' tab. The 'General' tab is active, showing the 'Source dataset' as 'DS\_Lookup\_Synapse' and the 'Query' as 'select \* from metadata where status <> 'succeeded''. The 'Settings' tab is also visible.

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

**New linked service**

Azure Databricks [Learn more](#)

Connect via integration runtime \* ⓘ

Self-IR

Account selection method \*

From Azure subscription  Enter manually

Azure subscription \* ⓘ

Dotnet\_Dev\_ProjectA (a67f73e7-b505-4ce3-8f8f-994c184fb691)

Databricks workspace \* ⓘ

filesystemdata

Select cluster

New job cluster  Existing interactive cluster  Existing instance pool

**New linked service**

Azure Databricks [Learn more](#)

AzureKeyVault1

Secret name \* ⓘ

databricksaccesstoken

Edit

Secret version ⓘ

Latest version

Edit

Choose from existing clusters \* ⓘ

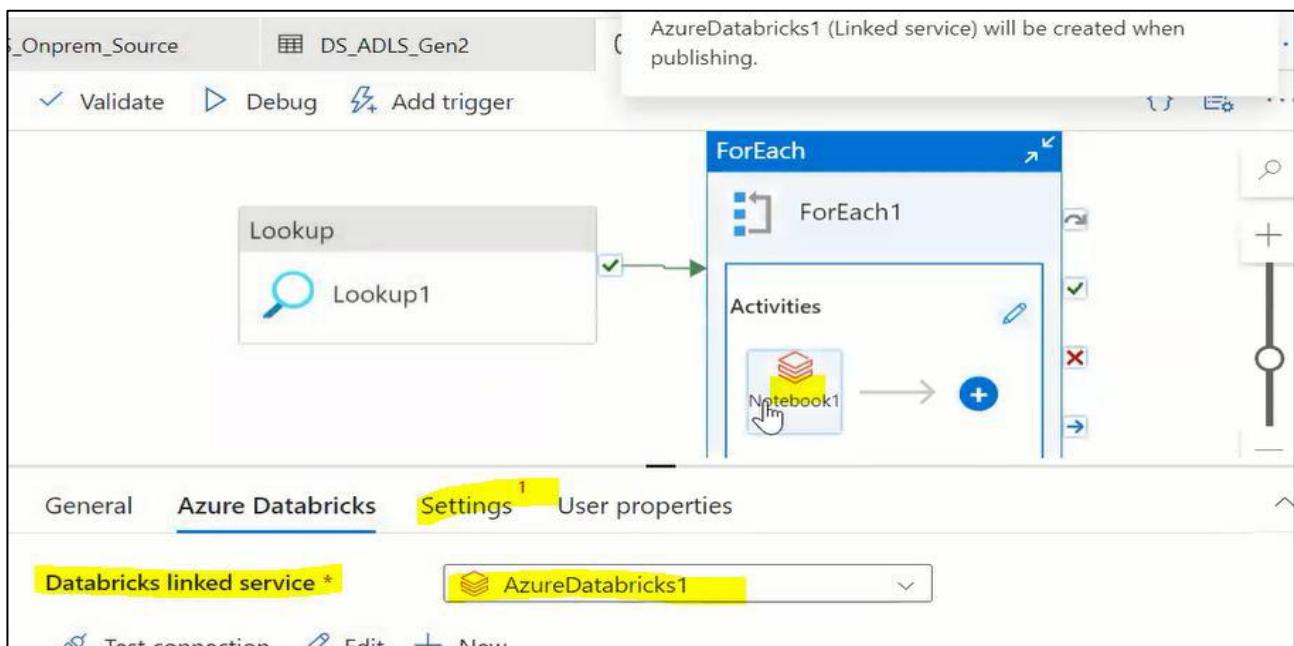
Sravan N's Cluster

Annotations

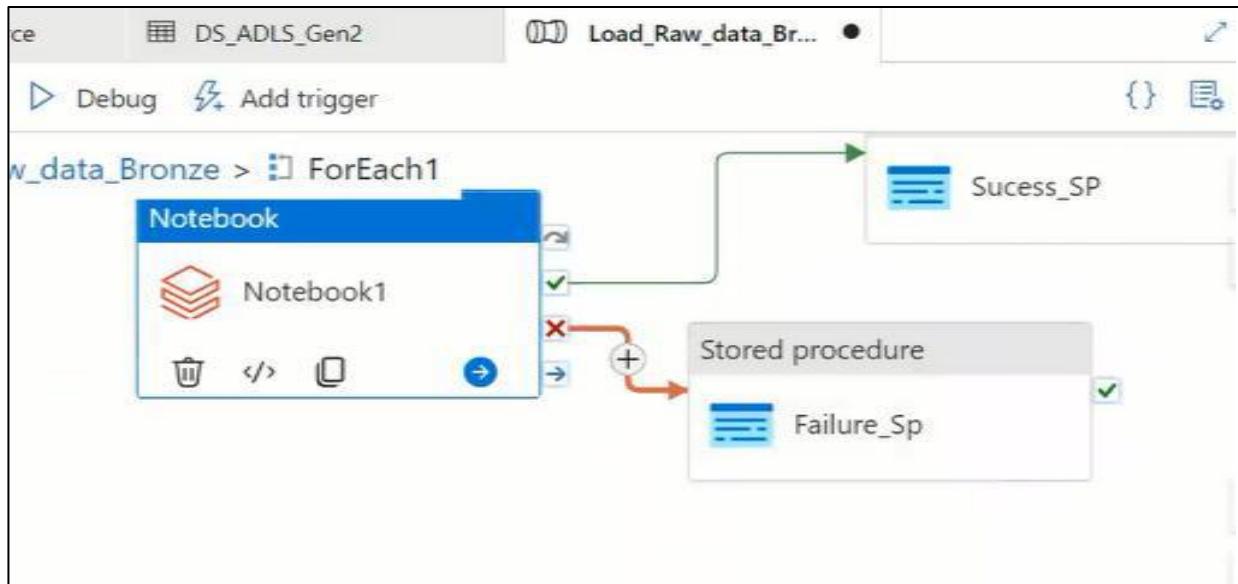
+ New

> Parameters

> Advanced ⓘ



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



Name	Value
foldername	@item().storagepath
processeddate	@formatDateTime(utcnow(), 'yyyy/MM/dd')

Similarly 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.

✓ Validate ▶ Debug ⚙ Add trigger

Output

Copy to clipboard

```
{  
  "runPageUrl": "https://adb-  
254324971627258.18.azuredatabricks.net/?  
o=254324971627258#jnp/478715158340812/run/2397",  
  "runError": "TypeError: can only concatenate str (not \"int\") to  
str",  
  "effectiveIntegrationRuntime": "Self-IR",  
  "executionDuration": 16,  
  "durationInQueue": {  
    "Notebook1": 8333,  
    "Failure_Sp": 0,  
    "Notebook1": 8333  
  }  
}
```

Pipeline status

or in Azure Metrics Export to CSV

type	Run start	Count
Notebook	8/30/2023, 7:50:21 PM	3
Procedure	8/30/2023, 7:50:09 PM	7
Notebook	8/30/2023, 7:49:44 PM	2
Stored procedure	8/30/2023, 7:49:36 PM	7
Notebook	8/30/2023, 7:49:12 PM	2

atabricks Search data, notebooks, recents, and more... CTRL + P filesystem

Workflows > Runs >  
**ADF\_onprmdatfctry\_Load\_Raw\_data\_Bronze\_Notebook1\_4c78dbad-204ec5b0810035 run**

Output

df1.write.format("csv").option("header",True).save(dest\_final\_path)  
Command took 1.10 seconds

dbutils.notebook.exit("source count:"+src\_count+" destination count:"+dest\_count)  
TypeError: can only concatenate str (not "int") to str  
Command took 0.05 seconds

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 transformation 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()

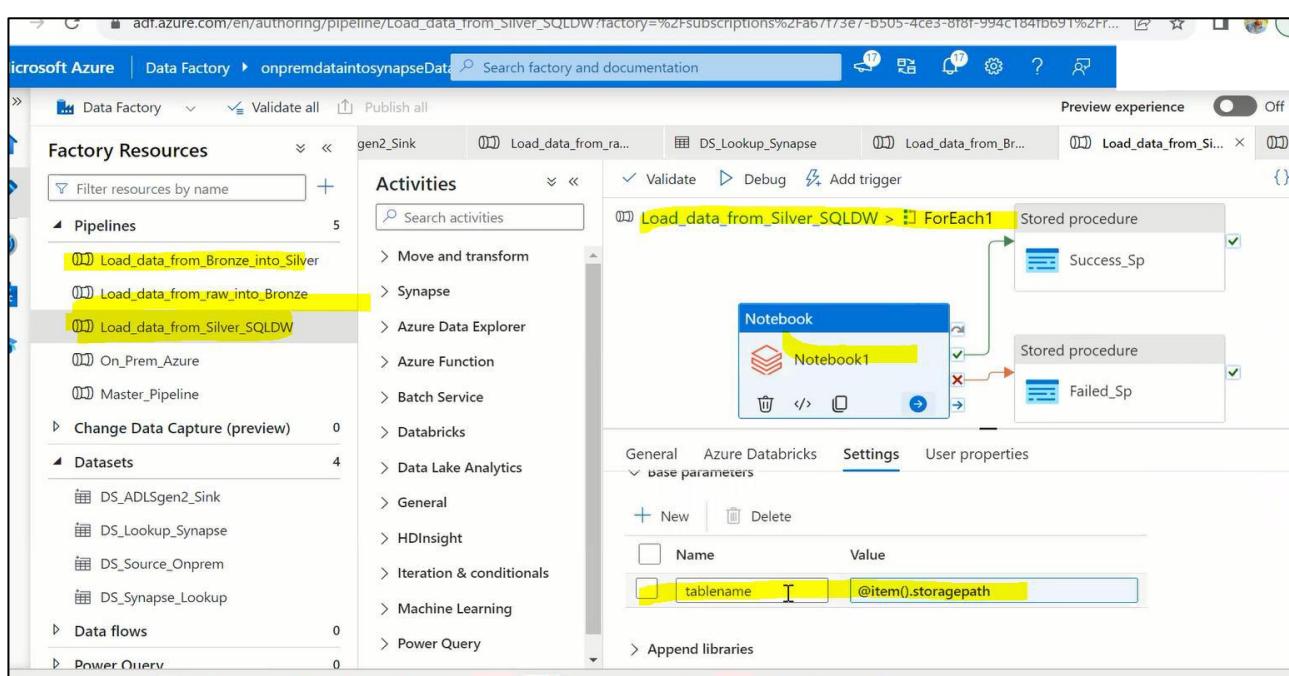
# Display source count

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

