

Part 1:

Before creating all the resources we will create the resource group in which we will create all the required resources.

Go to Azure Portal and log in with your Azure account. In the left-hand menu, select "Resource groups". If you don't see it, use the search bar at the top of the page and search for "Resource groups". Click the "+ Create" button or "Add" at the top of the Resource groups page.

The screenshot shows the Microsoft Azure portal interface. The top navigation bar is blue with the Microsoft Azure logo and a search bar. Below the navigation bar, the page title is "Resource groups". A "Default Directory" dropdown is visible. At the top right, there are buttons for "+ Create", "Manage view", "Refresh", "Export to CSV", "Open query", and "Assign tags". Below these buttons are two filter options: "Subscription equals all" and "Location equals all".

Select the subscription under which the resource group will be created. Enter a unique name for your resource group. Choose a location (region) where your resources will reside (e.g., East US, West Europe).

The screenshot shows the "Create a resource group" wizard. The top navigation bar is blue with the Microsoft Azure logo and a search bar. The page title is "Create a resource group". There are three tabs: "Basics" (selected), "Tags", and "Review + create".
Project details:
Subscription: Pay-As-You-Go (136f20b5-00f7-4eb5-a52e-0843e7ad1034)
Resource group: trendytech-azure-project
Resource details:
Region: (Asia Pacific) Central India

At the bottom, there are buttons for "Review + create", "< Previous", and "Next : Tags >".

Click "Review + Create" and then "Create".

Azure Storage Account creation:

Search for "Storage accounts" and click "+ Create". Select subscription, resource group, region, and enter a unique storage account name as shown below.

The screenshot shows the "Create a storage account" wizard. Under "Subscription", "Pay-As-You-Go (136f20b5-00f7-4eb5-a52e-0843e7ad1034)" is selected. Under "Resource group", "trendytech-azure-project" is selected. In the "Instance details" section, the "Storage account name" is set to "ttadlsdevnew". The "Region" is "(Asia Pacific) Central India". The "Primary service" dropdown is set to "Select a primary service". Under "Performance", the "Standard" radio button is selected. Under "Redundancy", "Locally-redundant storage (LRS)" is chosen. At the bottom, there are "Previous", "Next", and "Review + create" buttons, where "Review + create" is highlighted in blue.

Enable ADLS Gen2: Go to the Advanced tab and enable Hierarchical namespace.

The screenshot shows the "Create a storage account" wizard with the "Advanced" tab selected. Under "Security", several checkboxes are checked: "Require secure transfer for REST API operations", "Enable storage account key access", and "Permitted scope for copy operations (preview)". Under "Hierarchical Namespace", the text states: "Hierarchical namespace, complemented by Data Lake Storage Gen2 endpoint, enables file and directory semantics, accelerates big data analytics workloads, and enables access control lists (ACLs). Learn more". The "Enable hierarchical namespace" checkbox is checked. At the bottom, there are "Review + create" and "Create" buttons, where "Review + create" is highlighted in blue.

Click Review + create, validate settings, and click Create.

Now create the containers “landing”, “bronze”, “silver”, “gold”, “configs” in this storage account as shown below.

The screenshot shows the Azure Storage Account Overview page for 'ttadlsdevnew'. In the left sidebar under 'Data storage', 'Containers' is selected. At the top right, there is a '+ Container' button. Below it, a search bar says 'Search containers by prefix' and a text input field contains '\$logs'. The main area displays a table with one row:

Name
\$logs

The screenshot shows the same Azure Storage Account Overview page as above, but now with six containers listed in the table:

Name	Last modified	Anonymous access level	Lease state
Slogs	12/10/2024, 3:56:08 PM	Private	Available
bronze	12/10/2024, 3:59:20 PM	Private	Available
configs	12/10/2024, 3:59:42 PM	Private	Available
gold	12/10/2024, 3:59:34 PM	Private	Available
landing	12/10/2024, 3:59:05 PM	Private	Available
silver	12/10/2024, 3:59:28 PM	Private	Available

Then in the configs container create the directory “emr” and then upload the file “load_config.csv” in it.

The screenshot shows the Azure Storage Account Container Overview page for the 'configs' container. The left sidebar shows 'Overview' is selected. The main area has a table header:

Name	Modified	Access tier	Archive status	Blob type
------	----------	-------------	----------------	-----------

Below the table, it says 'No results'.

Home > ttadlsdevnew_1733826328505 | Overview > ttadlsdevnew | Containers >

configs ...

Container

Search

Upload Add Directory Refresh Rename Delete Change tier Acquire lease Break lease

Overview Diagnose and solve problems Access Control (IAM) Settings

Authentication method: Access key (Switch to Microsoft Entra user account)
Location: configs / emr

Search blobs by prefix (case-sensitive)

Name	Modified	Access tier	Archive status	Blob type	Size
..	12/10/2024, 4:04:45 ...	Hot (Inferred)		Block blob	743 B
load_config.csv					

Steps to create Azure SQL database:

We will create 2 azure SQL db - trendytech-hospital-a, trendytech-hospital-b

In the search bar, type "SQL Database" and select "SQL Database" from the results. Choose your Subscription and Resource Group. Enter a Database Name. Also create a SQL Server.

Home > SQL databases >

Create SQL Database

Microsoft

Want to try Azure SQL Database for free? Create a free serverless database with the first 100,000 vCore seconds, 32GB of data, and 32GB of backup storage free per month for the lifetime of the subscription. [Learn more](#)

Apply offer (Preview)

SQL Database Hyperscale: Low price, high scalability, and best feature set. [Learn more](#)

Project details

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

Subscription * ⓘ Pay-As-You-Go (136f20b5-00f7-4eb5-a52e-0843e7ad1034)

Resource group * ⓘ trendytech-azure-project

Create new

Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name *	<input type="text" value="trendytech-hospital-a"/>
Server *	<input type="button" value="Select a server"/> <input type="button" value="Create new"/>
Compute + storage *	<input type="button" value="Please select a server first."/> Configure database

We will create the server as shown below. Choose a Compute + Storage tier (e.g., Basic, General Purpose).

Home > SQL databases > Create SQL Database >

Create SQL Database Server

Microsoft

Server details

Enter required settings for this server, including providing a name and location. This server will be created in the same subscription and resource group as your database.

Server name *	<input type="text" value="trendytech-sqlserver"/> .database.windows.net
Location *	<input type="button" value="(Asia Pacific) Central India"/>

Authentication

Azure Active Directory (Azure AD) is now Microsoft Entra ID. [Learn more](#)

Select your preferred authentication methods for accessing this server. Create a server admin login and password to access your server with SQL authentication, select only Microsoft Entra authentication [Learn more](#) using an existing Microsoft Entra user, group, or application as Microsoft Entra admin [Learn more](#), or select both SQL and Microsoft Entra authentication.

Authentication method

- Use Microsoft Entra-only authentication
- Use both SQL and Microsoft Entra authentication
- Use SQL authentication

Server admin login *

Password *

Confirm password *

OK

After this Go to Networking => In Network connectivity select “Public endpoint” option. Also set yes for the options “Allow Azure services and resources to access this server” and “Add current client IP address”

Basics **Networking** Security Additional settings Tags Review + create

Configure network access and connectivity for your server. The configuration selected below will apply to the selected server 'trendytech-sqlserver' and all databases it manages. [Learn more](#)

Firewall rules

The settings displayed below are read-only. They can be modified from the "Firewalls and virtual networks" blade for the selected server after database creation. [Learn more](#)

Allow Azure services and resources to access this server No Yes

Add current client IP address * No Yes

Private endpoints

Private endpoint connections are associated with a private IP address within a Virtual Network. The list below shows all the private endpoint connections for this server. Note that private endpoint connections are defined at the server level and they provide access to all databases in the server. [Learn more](#)

Note: Please note down this username and password for future reference.

Home > SQL databases >

Create SQL Database

Microsoft

Configure network access and connectivity for your server. The configuration selected below will apply to the selected server 'trendytech-server-new' and all databases it manages. [Learn more](#)

Network connectivity

Choose an option for configuring connectivity to your server via public endpoint or private endpoint. Choosing no access creates with defaults and you can configure connection method after server creation. [Learn more](#)

Connectivity method * Public endpoint No access Private endpoint

Firewall rules

Setting 'Allow Azure services and resources to access this server' to Yes allows communications from all resources inside the Azure boundary, that may or may not be part of your subscription. [Learn more](#)
Setting 'Add current client IP address' to Yes will add an entry for your client IP address to the server firewall.

Allow Azure services and resources to access this server * No Yes

Add current client IP address * No Yes

[Review + create](#) [< Previous](#) [Next : Security >](#)

Click Review + Create, validate the details, and then click Create as shown below.

Home > SQL databases >

Create SQL Database

Microsoft

Basics Networking Security Additional settings Tags Review + create

Create a SQL database with your preferred configurations. Complete the Basics tab then go to Review + Create to provision with smart defaults, or visit each tab to customize. [Learn more](#)

Project details

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

Cost summary

General Purpose (GP_S_Gen5_2)	0.00
Cost per GB (in INR)	x 41.6
Max storage selected (in GB)	
First 32 GB storage free	
First 100,000 vCore seconds free	
Overage billing!	Disabled
ESTIMATED STORAGE COST / MONTH	

Review + create Next : Networking >

Note: While creating database if you are not able to allow public access and add client ip address you can follow below steps:

After creating this database Go to Networking => For Public access (select option Selected networks and save this) =>

Home > Microsoft.SQLDatabase.newDatabaseExistingServer 906552b051fe4404 | Overview > trendytech-hospital-a (trendytech-sqlserver)

trendytech-sqlserver | Networking

SQL server

Search

- Overview
- Activity log
- Access control (IAM)
- Tags
- Quick start
- Diagnose and solve problems
- Settings
- Data management
- Security
- Networking**
- Microsoft Defender for Cloud

Feedback

Public access Private access Connectivity

Public network access

Public Endpoints allow access to this resource through the internet using a public IP address. An application requires proper authorization to access this resource. [Learn more](#)

Public network access

Disable

Selected networks

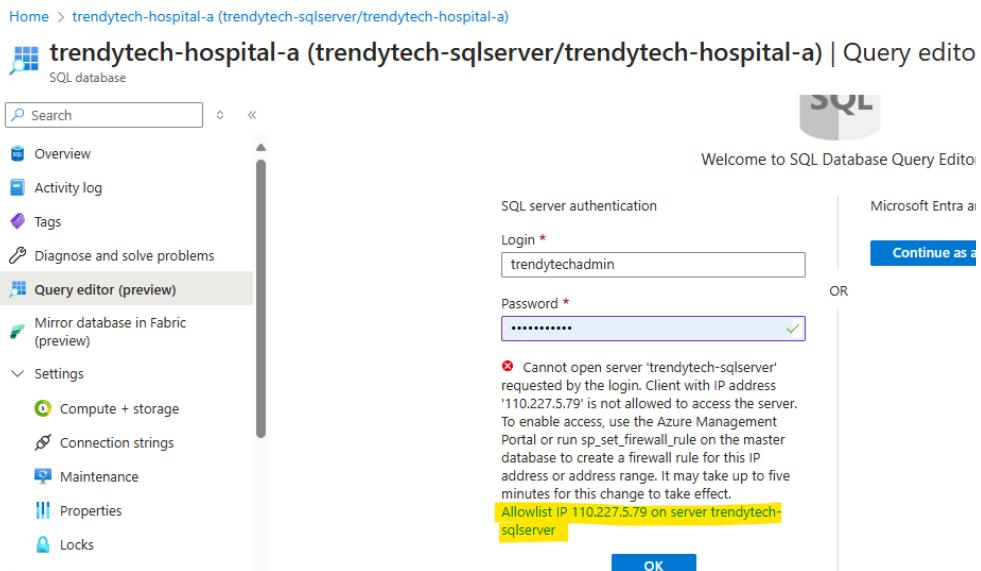
[Connections from the IP addresses configured in the Firewall are allowed. \[Learn more\]\(#\)](#)

Virtual networks

Allow virtual networks to connect to your resource using service endpoints. [Learn more](#)

+ Add a virtual network rule

Also while using query editor if you face below error click on “Allowing IP for current ip address” as shown below



Similarly we will create another database **trendytech-hospital-b**(We will use the same server i.e **trendytech-sqlserver** that we have created while creating **trendytech-hospital-a** database). Thus we have created 2 databases as shown below.

Name	Server	Replica type	Pricing tier	Location
trendytech-hospital-a	trendytech-sqlserver	--	General Purpose: S...	Central India
trendytech-hospital-b	trendytech-sqlserver	--	Free General Purpo...	Central India

Then we will create the tables in these databases and for creating tables in the database use below scripts which are present on github account:

For **trendytech-hospital-a =>**

Trendytech_hospital_A_table_creation_commands

For **trendytech-hospital-b =>**

Trendytech_hospital_B_table_creation_commands

Steps to create ADF:

In the search bar, type "Data Factory" and select "Data Factory" from the results. Click the "Create" button on the Data Factory page. Provide a globally unique name for your Data Factory instance. Choose V2 (Data Factory Version 2) for the latest features.

The screenshot shows the 'Create Data Factory' wizard in the Azure portal. The top navigation bar includes 'Home > Data factories >' followed by a yellow-highlighted 'Create Data Factory' button and a '...' button. Below the navigation is a horizontal menu with tabs: 'Basics' (underlined), 'Git configuration', 'Networking', 'Advanced', 'Tags', and 'Review + create'. A note below the tabs says 'One-click to create data factory with sample pipeline and datasets. Try it'. The main form area starts with 'Project details' which asks to 'Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.' It shows a 'Subscription' dropdown set to 'Pay-As-You-Go (136f20b5-00f7-4eb5-a52e-0843e7ad1034)' and a 'Resource group' dropdown set to 'trendytech-azure-project' with a 'Create new' link. The next section is 'Instance details' with fields for 'Name' (set to 'TT-health-projectdev'), 'Region' (set to 'Central India'), and 'Version' (set to 'V2'). At the bottom of the form are 'Previous' and 'Next' buttons, and a prominent blue 'Review + create' button.

Click "Review + Create" to validate the details. If validation passes, click "Create" to deploy the Data Factory.

Steps to create ADF pipeline:

Linked Services creation:

In the ADF interface, go to the Manage section on the left-hand panel. Under the Connections section, select Linked Services. Click on New to create a new Linked Service.

1. Azure SQL DB

Note down the server name for that sql database that we have created.

The screenshot shows the Azure portal's Overview page for a SQL database named 'trendytech-hospital-b'. The left sidebar includes links for Overview, Activity log, Tags, Diagnose and solve problems, Query editor (preview), Mirror database in Fabric (preview), Settings, Data management, and Integrations. The main content area displays the following details:

Essentials	
Resource group (...)	: trendytech-azure-project
Status	: Paused
Location	: Central India
Subscription (move)	: Pay-As-You-Go
Subscription ID	: 136f20b5-00f7-4eb5-a52e-0843e7ad1034
Tags (edit)	: Add tags
Server name	: trendytech-sqlserver.database.windows.net
Connection strings	: Show database connection strings
Pricing tier	: Free - General Purpose - Serverless: Gen5, 2 vC
Overage billing	: Disabled
Free monthly vCore ...	: 99,424 vCore seconds remaining
Earliest restore point	: 2024-12-10 17:18 UTC

In fully qualified domain names, mention the server name, mention the username and password for sql server and define the parameter db_name and using this parameter we will pass the database name as shown below.

New linked service

Azure SQL Database [Learn more](#)

Name *

Description

Connect via integration runtime * ⓘ

AutoResolveIntegrationRuntime

Version

Recommended Legacy

[Import from connection string](#)

Account selection method ⓘ

From Azure subscription Enter manually

Fully qualified domain name *

Database name *

@linkedService().db_name

Authentication type *

SQL authentication

User name *

trendytechadmin

Password Azure Key Vault

Password *

.....

Always encrypted ⓘ

Encrypt ⓘ Mandatory

Trust server certificate ⓘ

Host name in certificate

Additional connection properties [New](#)

Annotations [New](#)

Parameters [New](#) [Delete](#)

Name	Type	Default value
db_name	String	<input type="text"/>

> Advanced ⓘ

[Create](#) [Back](#) [Test connection](#) [Cancel](#)

And click on create to create the linked service.

2. ADLS GEN2

Select Azure Data Lake Storage1 as the data store. Provide the following details- Name of your Blob Storage account, Authentication. Then click Test Connection to verify and save the Linked Service.

To get the url for Azure Data Lake Storage go to Adls gen2 storage that we have created => Setting => Endpoints => and copy the URL as shown below

The screenshot shows the Azure Storage account settings for the account 'ttadlsdevnew'. The 'Endpoints' section is highlighted. Key details shown include:

- Queue service**: Resource ID: /subscriptions/136f20b5-00f7-4eb5-a52e-0843e7ad10; URL: https://ttadlsdevnew.queue.core.windows.net/
- Table service**: Resource ID: /subscriptions/136f20b5-00f7-4eb5-a52e-0843e7ad10; URL: https://ttadlsdevnew.table.core.windows.net/
- Data Lake Storage**: Resource ID: /subscriptions/136f20b5-00f7-4eb5-a52e-0843e7ad10; URL: https://ttadlsdevnew.dfs.core.windows.net/

Also copy the access key. Using these details create the linked service as shown below.

The screenshot shows the 'New linked service' creation form for 'Azure Data Lake Storage Gen2'. The 'Storage account key' tab is selected. Key fields filled in include:

- Name ***: AzureDataLakeStorage1
- Description**: (empty)
- Connect via integration runtime * ⓘ**: AutoResolveIntegrationRuntime
- Authentication type**: Account key
- Account selection method ⓘ**: Enter manually
- URL ***: https://ttadlsdevnew.dfs.core.windows.net/
- Storage account key ***: (redacted)

At the bottom, there are 'Create', 'Back', 'Test connection', and 'Cancel' buttons.

3. Delta table - Audit_logs

Create the databricks workspace test and then upload the code notebook “Audit_table_DDL” and start your databricks cluster and create the schema audit table in it using below commands. (Notebook name - audit_table_ddl)

create schema if not exists audit;

```
CREATE TABLE IF NOT EXISTS audit.load_logs (
    id BIGINT GENERATED ALWAYS AS IDENTITY,
    data_source STRING,
    tablename STRING,
    numberofrowscopied INT,
    watermarkcolumnname STRING,
    loaddate TIMESTAMP
);
```

[Home](#) > [Azure Databricks](#) >

Create an Azure Databricks workspace ...

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

Subscription *	<input type="text" value="Pay-As-You-Go (136f20b5-00f7-4eb5-a52e-0843e7ad1034)"/>
Resource group *	<input type="text" value="trendytech-azure-project"/> Create new

Instance Details

Workspace name *	<input type="text" value="test"/>
Region *	<input type="text" value="Central India"/>
Pricing Tier *	<input type="text" value="Standard (Apache Spark, Secure with Microsoft Entra ID)"/>
Managed Resource Group name	<input type="text" value="Enter name for managed resource group"/>

[Review + create](#)

< Previous

Next : Networking >

The screenshot shows the Microsoft Azure Databricks workspace interface. The left sidebar contains navigation links for New, Workspace, Recents, Catalog, Workflows, Compute, Data Engineering, Job Runs, Machine Learning, Playground, Experiments, Features, Models, Serving, and Partner Connect. The main area displays the 'audit_table_ddl' notebook in Python mode. The notebook has two cells:

- Cell 1:** Contains the command `%sql` followed by `create schema if not exists audit;`. The output is 'OK'.
- Cell 2:** Contains the command `%sql` followed by the creation of a table `CREATE TABLE IF NOT EXISTS audit.load_logs (id BIGINT GENERATED ALWAYS AS IDENTITY, data_source STRING, tablename STRING, numberofrowsscopied INT, watermarkcolumnname STRING, loaddate TIMESTAMP);`.

At the top right, there is a search bar with placeholder text 'Search data, notebooks, recents, and more...', a 'Run all' button, and a 'CTRL + P' keybinding indicator.

Note: To get access token Click your profile icon (top-right corner of the workspace) => Select Settings => Developer (In user setting) => Generate Access Token as shown below

The screenshot shows the Microsoft Azure Databricks interface. On the left, there's a sidebar with various navigation items like Workspace, Recents, Catalog, Workflows, Compute, Data Engineering, Job Runs, Machine Learning, Playground, Experiments, Features, and Models. A pink callout box highlights the 'New' button. The main area has a yellow header bar labeled 'Settings'. Below it, there are several sections: 'Workspace admin' (with sub-links for Identity and access, Security, Compute, Development, Notifications, and Advanced), 'User' (with sub-links for Profile, Preferences, and Developer), and a 'Developer' section which is currently active. The 'Developer' section contains links for 'Access tokens' (with a 'Manage' button) and 'Editor settings' (with a 'General' link). At the bottom right, there's a 'Spark tips' section with a switch labeled 'On'.

Then generate the token and copy for Future use.

While creating linked service in source mention “AzureDatabricksDeltaLake”. Then in the domain mention the URL of databricks workspace. And mention the cluster id for the cluster that we have created. To get Databricks overview page to get Workspace URL and to get cluster id go to compute => Select the cluster => copy the cluster id as shown below

Microsoft Azure | databricks

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

+ New

Workspace

Recents

Catalog

Workflows

Compute

Data Engineering

Job Runs

Machine Learning

Playground

Experiments

Features

Models

Serving

Compute

Configuration Notebooks (1) Libraries Event log Spark UI Driver logs Metrics Apps Spark compute UI - Master

Multi node Single node

Access mode Single user access

Single user Arati Hatti

Performance

Databricks Runtime Version

15.4 LTS (includes Apache Spark 3.5.0, Scala 2.12)

Use Photon Acceleration

Node type

Standard_D3_v2 14 GB Memory, 4 Cores

Terminate after 120 minutes of inactivity

Refer below screenshot for more details.

New linked service

Azure Databricks Delta Lake Learn more

Name *

Description

Connect via integration runtime * ⓘ

AutoResolveIntegrationRuntime

Authentication method *

Access token

Account selection method

From Azure subscription Enter manually

Domain *

https://adb-934377009310382.2.azuredatabricks.net

Existing cluster ID *

1210-184039-eqjwfrv5

Connection successful

Create Back Test connection Cancel

Dataset creation:

In the ADF interface, click on the Author section (left-hand panel). Expand the Datasets option. Click on the “...” next to Datasets in order to create the dataset.

1. Azure SQL DB

We will select the linked service that we have created for the SQL database. To create the datasets for the tables in a parameterized way in the sql database , we will create the parameter db_name, schema_name, table_name.

Set properties

Name
generic_sql_ds

Linked service *
hosa_sql_ls

Table name
Select...  

Enter manually

Import schema
 From connection/store None

Then we will create parameters db_name, schema_name, table_name as shown below.

Azure SQL Database
generic_sql.ds

Connection Schema Parameters

+ New | Delete

Name	Type	Default value
db_name	String	Value
schema_name	String	Value
table_name	String	Value

And we will pass dynamic value for table name and schema name as shown below

generic_sql.ds

Azure SQL Database
generic_sql.ds

Connection Schema Parameters

Linked service * hosa_sql_ls

Test connection Edit + New Learn more ↗

Linked service properties ⓘ

Name	Value	Type
db_name	@dataset().db_name	string
Table	@dataset().schema_name	
	@dataset().table_name	

Enter manually

Preview data

2. Dataset for Flatfile in ADLS GEN2

Select source as ADLS gen2 and file format as delimited text. Also in order to make it generic we will create the parameter file_name, file_path and container.

Set properties

Name
generic_adls_flat_file_ds

Linked service *
AzureDataLakeStorage1

File path
File system / Directory / File name

First row as header

Import schema
 From connection/store From sample file None

OK Back Cancel

generic_sql_ds X generic_adls_flat_fil...

 DelimitedText
generic_adls_flat_file_ds

Connection Schema Parameters

+ New Delete

Name	Type	Default value
file_name	String	Value
file_path	String	Value
container	String	Value

The screenshot shows the Azure Data Studio interface with two tabs at the top: 'generic_sql_ds' and 'generic_adls_flat_fil...'. The current tab is 'generic_adls_flat_file_ds'. The interface displays a 'DelimitedText' file format icon and the name 'generic_adls_flat_file_ds'. Below this, there are three tabs: 'Connection', 'Schema', and 'Parameters', with 'Connection' selected. The 'Connection' tab contains the following configuration:

- Linked service: AzureDataLakeStorage1
- File path: @dataset().container / @dataset().file_path / @dataset().file_name
- Compression type: No compression
- Column delimiter: Comma (,)
- Row delimiter: Default (\r,\n, or \r\n)
- Encoding: Default(UTF-8)
- Quote character: Double quote ("")
- Escape character: Backslash (\)
- First row as header: checked

Now publish the changes.

3. Dataset for Parquet file in ADLS GEN2

In order to store data in ADLS gen2 in parquet format we will need the dataset.

While creating this dataset we will select source as ADLS gen2, fileformat as parquet and we will create parameters file_name, file_path and container.

Set properties

Name
generic_adls_parquet_ds

Linked service *
AzureDataLakeStorage1

File path
File system / Directory / File name

Import schema
 From connection/store From sample file None

OK

Back

Cancel

generic_sql_ds generic_adls_flat_file... generic_adls_parque... ●

Parquet
generic_adls_parquet_ds

Connection Schema Parameters

Linked service * AzureDataLakeStorage1 Test connection Edit + New Learn more

File path @dataset().container / @dataset().file_path / @dataset().file_name

Compression type snappy

4. Databricks Delta Lake for Delta lake

We will select the source as Azure Databricks Delta Lake. For this we will create the parameter schema_name and table_name.

generic_sql_ds generic_adls_flat_file... generic_adls_parque... AzureDatabricksDelta... ●

Azure Databricks Delta Lake
AzureDatabricksDeltaLakeDataset1

Connection Schema Parameters

Linked service * AzureDatabricksDeltaLake1_ls Test connection Edit + New Learn more

Database @dataset().schema_name

Table @dataset().table_name Preview data

Once all the dataset and linked service are created, publish all in order to save them.

Creation of Pipelines:

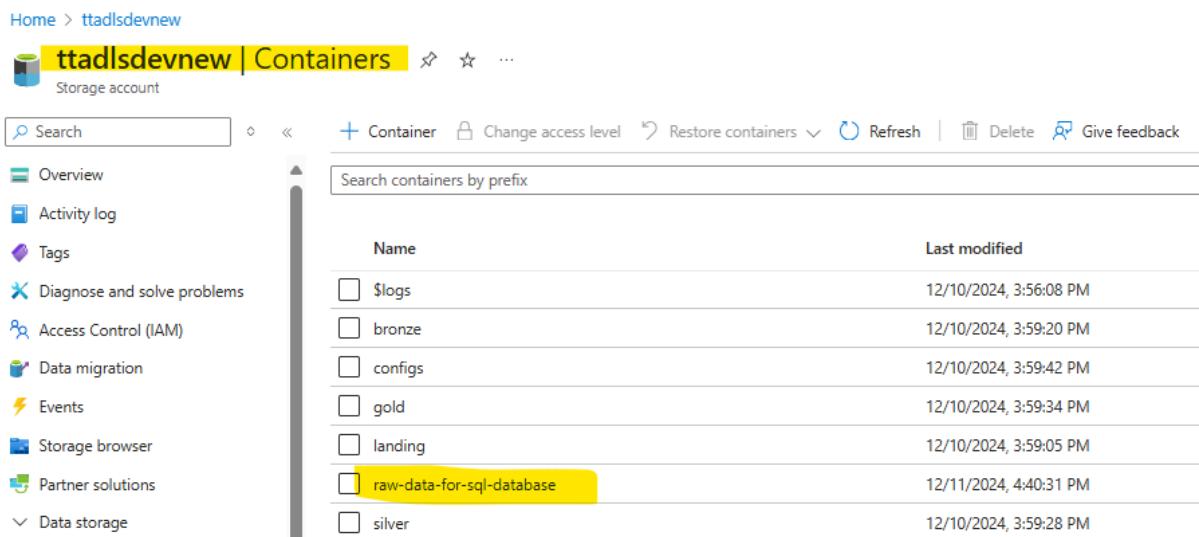
Background activity : Creation of pipeline to copy data into sql tables (pl_to_insert_data_to_sql_table_preprocessing).

Before proceeding with the main pipeline, we will create a simple pipeline in Azure Data Factory (ADF) to copy data from ADLS Gen2 storage into tables in an SQL database. This serves as a prerequisite to ensure that the SQL tables contain the data needed for the main pipeline.

Note: We will create a new container (raw-data-for-sql-database) in the given ADLS Gen2 storage (adlsdevnew) and upload our CSV files, which will serve as the source for the pipeline, along with a lookup file. Additionally, we will create a dataset for the lookup file to use in this pipeline. Using the copy activity in ADF, we will transfer the data into the following tables: Departments, Providers, Encounters, Patients, and Transactions, located in the SQL databases trendytech-hospital-a and trendytech-hospital-b.

Source: ADLS gen2 -adlsdevnew

We will create a new container (raw-data-for-sql-database) in the given ADLS Gen2 storage (adlsdevnew) and upload our CSV files, along with a lookup file.



The screenshot shows the Azure Storage Explorer interface for the 'ttadlsdevnew' storage account. On the left, there's a sidebar with various navigation options like Overview, Activity log, Tags, etc. The main area is titled 'Containers' and shows a list of containers. One container, 'raw-data-for-sql-database', is highlighted with a yellow background. The table lists the following data:

Name	Last modified
\$logs	12/10/2024, 3:56:08 PM
bronze	12/10/2024, 3:59:20 PM
configs	12/10/2024, 3:59:42 PM
gold	12/10/2024, 3:59:34 PM
landing	12/10/2024, 3:59:05 PM
raw-data-for-sql-database	12/11/2024, 4:40:31 PM
silver	12/10/2024, 3:59:28 PM

Folder: HospitalA, HospitalB for datafiles, Lookup for lookup file

The screenshot shows the Azure Storage Explorer interface. A container named "raw-data-for-sql-database" is selected. Inside the container, there are three blobs: "HospitalA", "HospitalB", and "Lookup". The "Lookup" blob was added on 12/13/2024 at 8:24:50.

Name	Modified	Access tier
HospitalA	12/11/2024, 2:56:39 ...	
HospitalB	12/11/2024, 2:56:48 ...	
Lookup	12/13/2024, 8:24:50 ...	

Sink: SQL DB - trendytech-hospital-a, trendytech-hospital-b:

Note: We have already created these databases so no need to create again.

The screenshot shows the Azure portal's "SQL databases" page. It lists two databases: "trendytech-hospital-a" and "trendytech-hospital-b". Both databases are associated with the "trendytech-server-new" server and have a replica type of "--".

Name	Server	Replica type
trendytech-hospital-a	trendytech-server-new	--
trendytech-hospital-b	trendytech-server-new	--

Pipeline creation Steps:

1. Creation of Linked Services:
=> For ADLS gen2 storage(source):

We will use the same linked service “AzureDataLakeStorage1” that we have created earlier for ADLS Gen2 storage.

=> For SQL DB(Sink):

We will use the same linked service “hosa_sql_ls” that we have created earlier for the database.

2. Creation of Datasets:

=> For source:

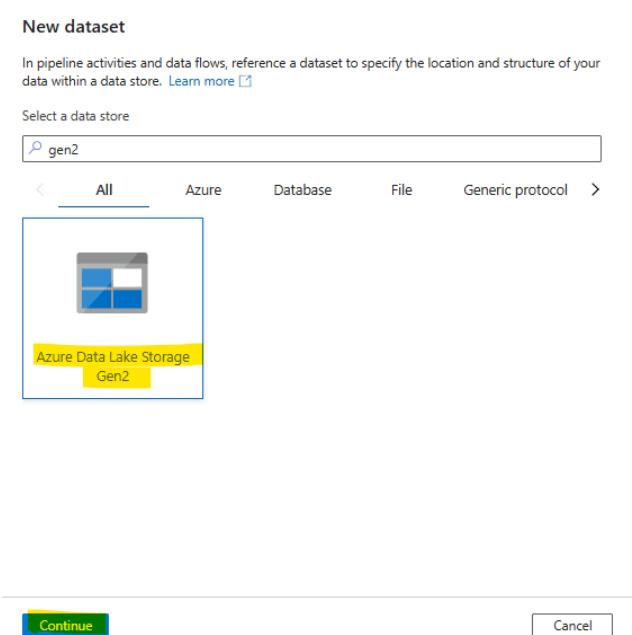
We will use the same generic dataset “generic_adls_flat_file_ds” that we have created earlier.

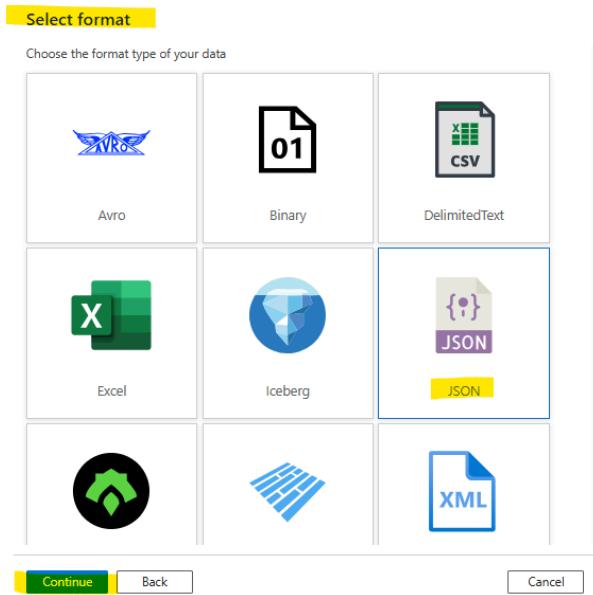
=> For sink:

We will use the same generic dataset “generic_sql_ds” that we have created earlier.

=>For Lookup we will create a new dataset as shown below.

Select source as ADLS Gen2 storage, then in file format select json as our lookup file is a json file as shown below





Set properties

Name

Linked service * 

File path  

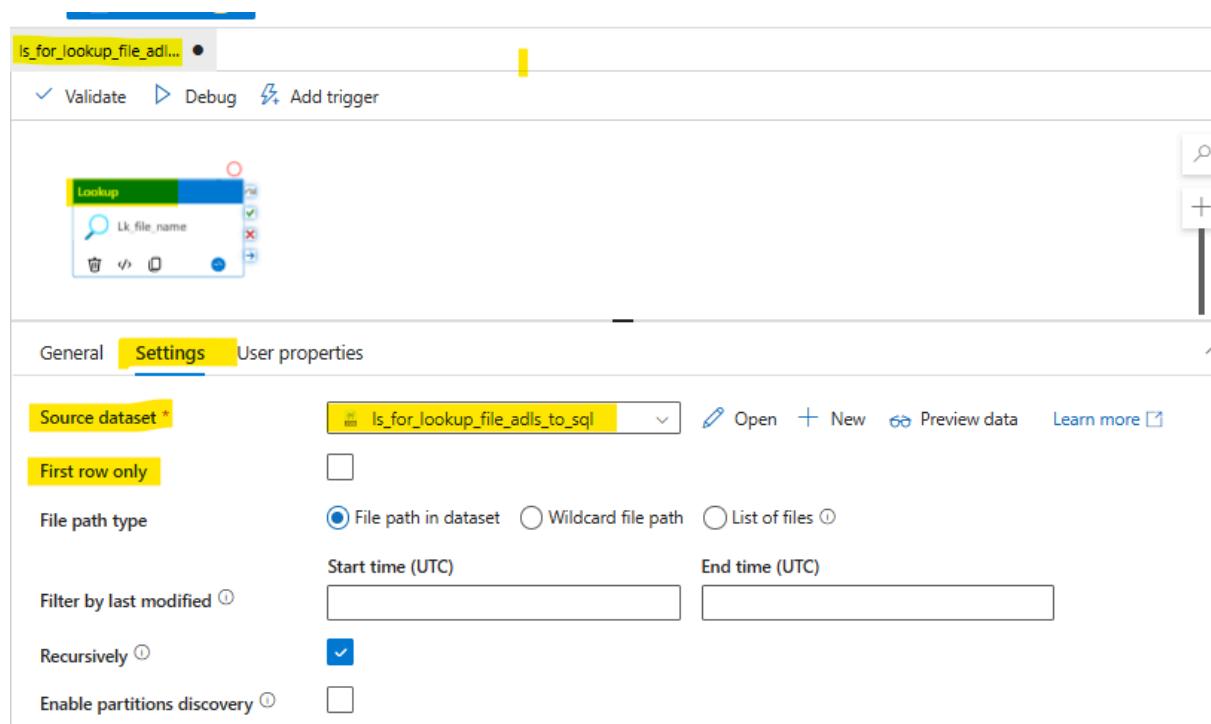
Import schema
 From connection/store From sample file None

OK  Back Cancel

Steps to Configure the Pipeline:

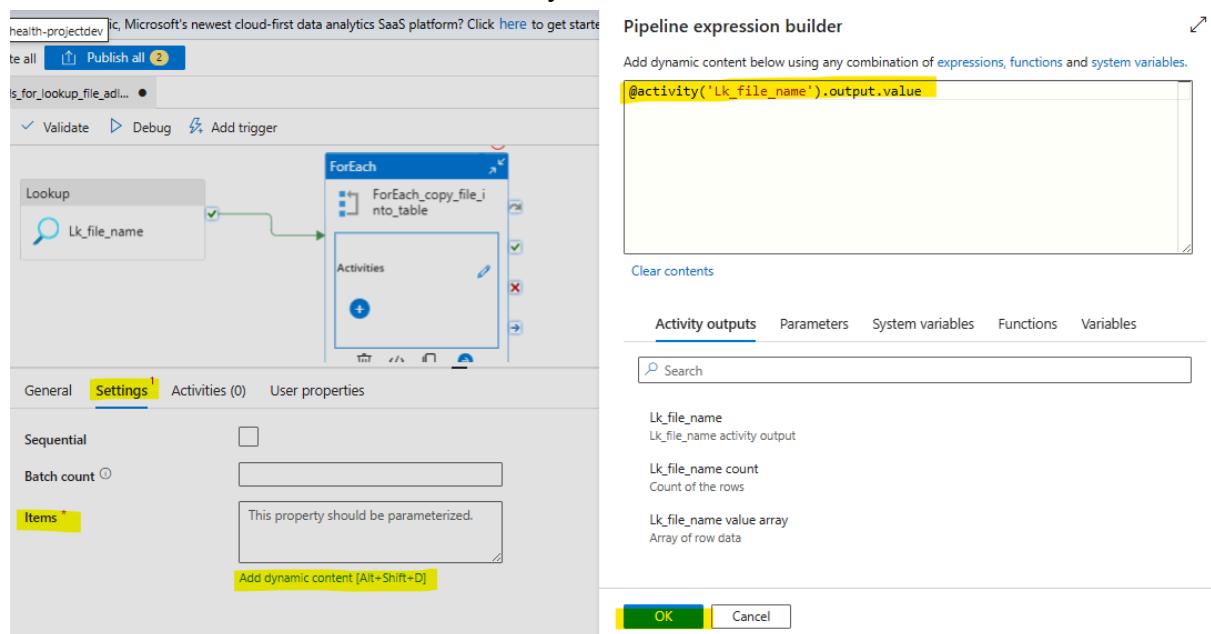
Add a Lookup Activity:

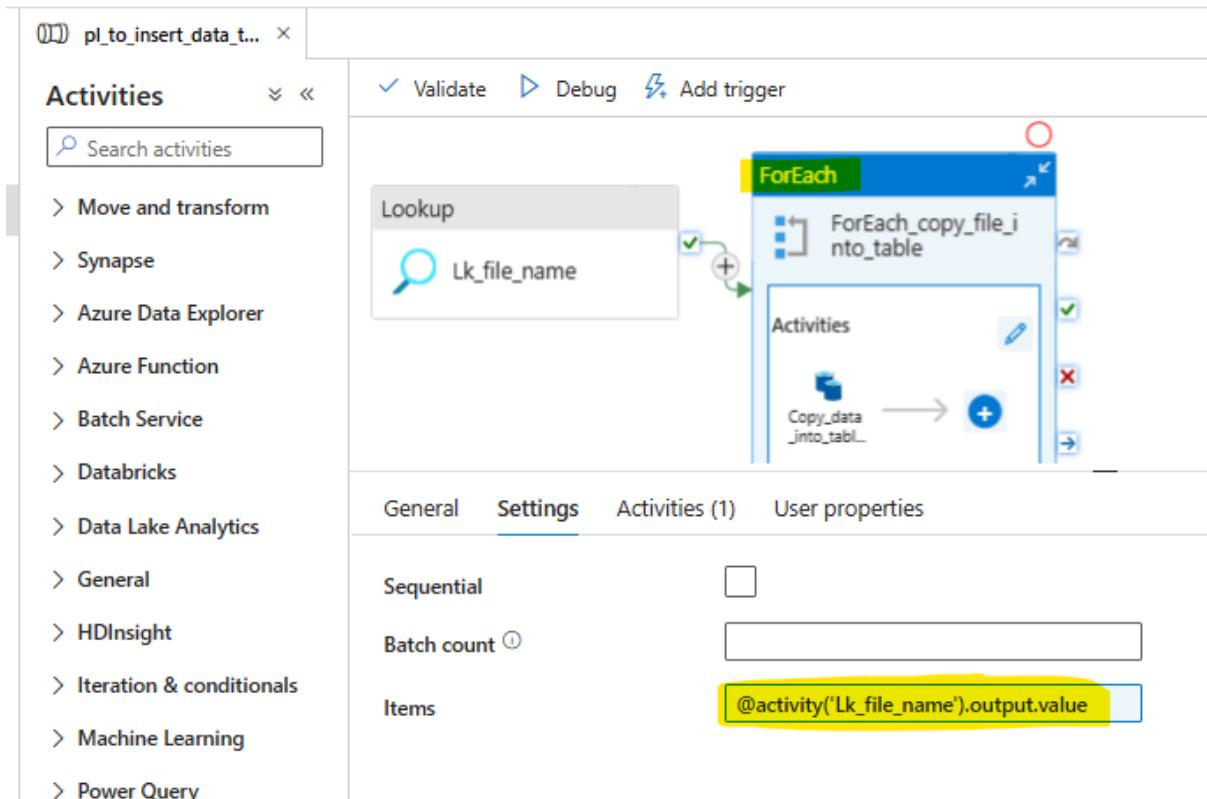
Drag a Lookup activity to the canvas. Point it to the mapping CSV dataset. Set First row only to false to read all rows. Refer below screenshot for more clarity.



Add a ForEach Activity:

Drag a ForEach activity and connect it to the Lookup activity.
Set its Items property to @activity('Lk_file_name').output.value
Refer below screenshot for more clarity.

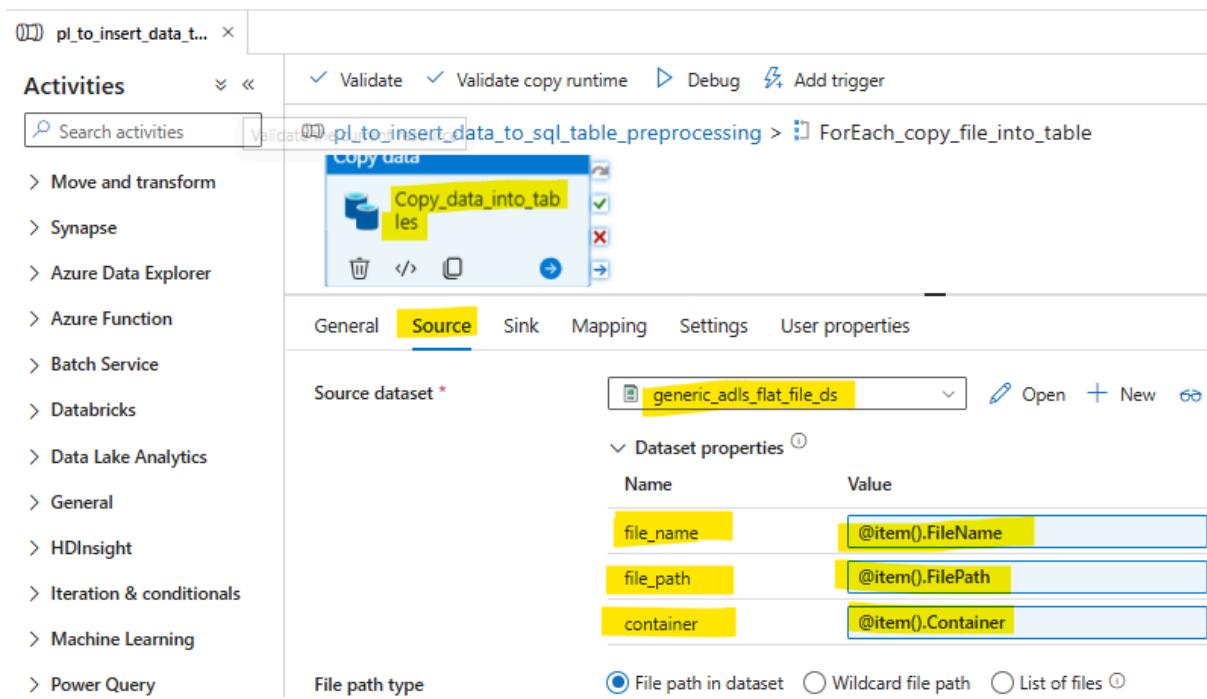




Configure the ForEach Activity:

Inside the ForEach activity, add a Copy Data activity.

Source: Use the source dataset.



Sink: Use the destination dataset.

The screenshot shows the Azure Data Factory pipeline editor. On the left, the 'Activities' sidebar is open, showing various options like Move and transform, Synapse, Azure Data Explorer, etc. The main area displays a pipeline named 'pl_to_insert_data_t...'. A 'Copy data' activity is selected, with its details pane open. The 'Sink' tab is active, showing a 'generic_sql_ds' dataset as the sink. Below it, 'Dataset properties' are listed:

Name	Type
db_name	string
schema_name	string
table_name	string

The values for 'db_name', 'schema_name', and 'table_name' are set to '@item().DatabaseName', '@item().SchemaName', and '@item().TableName' respectively, indicating dynamic mapping.

This pipeline will copy the data from the file into the tables in the sql database. On successfully running the pipeline we will get below output.

The screenshot shows the 'Output' tab for a pipeline run. The pipeline run ID is 'dc7ce2d3-a728-4619-9346-0c3c7d1f5034'. The status is 'Succeeded'. The table below lists the activities and their details:

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime
Copy_data_into_tables	Succeeded	Copy data	12/13/2024, 12:57:15 P	14s	AutoResolveIntegra
Copy_data_into_tables	Succeeded	Copy data	12/13/2024, 12:57:15 P	14s	AutoResolveIntegra
Copy_data_into_tables	Succeeded	Copy data	12/13/2024, 12:57:15 P	14s	AutoResolveIntegra
Copy_data_into_tables	Succeeded	Copy data	12/13/2024, 12:57:15 P	14s	AutoResolveIntegra
Copy_data_into_tables	Succeeded	Copy data	12/13/2024, 12:57:15 P	14s	AutoResolveIntegra
Copy_data_into_tables	Succeeded	Copy data	12/13/2024, 12:57:15 P	13s	AutoResolveIntegra
Copy_data_into_tables	Succeeded	Copy data	12/13/2024, 12:57:15 P	15s	AutoResolveIntegra
Copy_data_into_tables	Succeeded	Copy data	12/13/2024, 12:57:15 P	14s	AutoResolveIntegra
Copy_data_into_tables	Succeeded	Copy data	12/13/2024, 12:57:15 P	14s	AutoResolveIntegra

Pipeline to copy data from Azure Sql db to Landing Folder in ADLS Gen2

1. To read the config file we will use Lookup activity.

In this for source dataset will be for configs file and we will pass the parameter values as shown below.

The screenshot shows the Azure Data Factory pipeline editor with a pipeline named 'pl_emr_src_to_land...'. A 'Lookup' activity is selected, with its name set to 'lkp_EMR_configs'. The 'Source dataset' dropdown is set to 'generic_adls_flat_file_ds'. Under 'Dataset properties', there are three entries: 'file_name' with value 'load_config.csv', 'file_path' with value 'emr', and 'container' with value 'configs'. The 'File path type' is set to 'File path in dataset'. Below the dataset settings, there are fields for 'Start time (UTC)' and 'End time (UTC)'. The 'General' tab is selected at the bottom.

Then additionally we can preview the data.

2. In order to iterate through each row of configuration data we will use ForEach Activity.

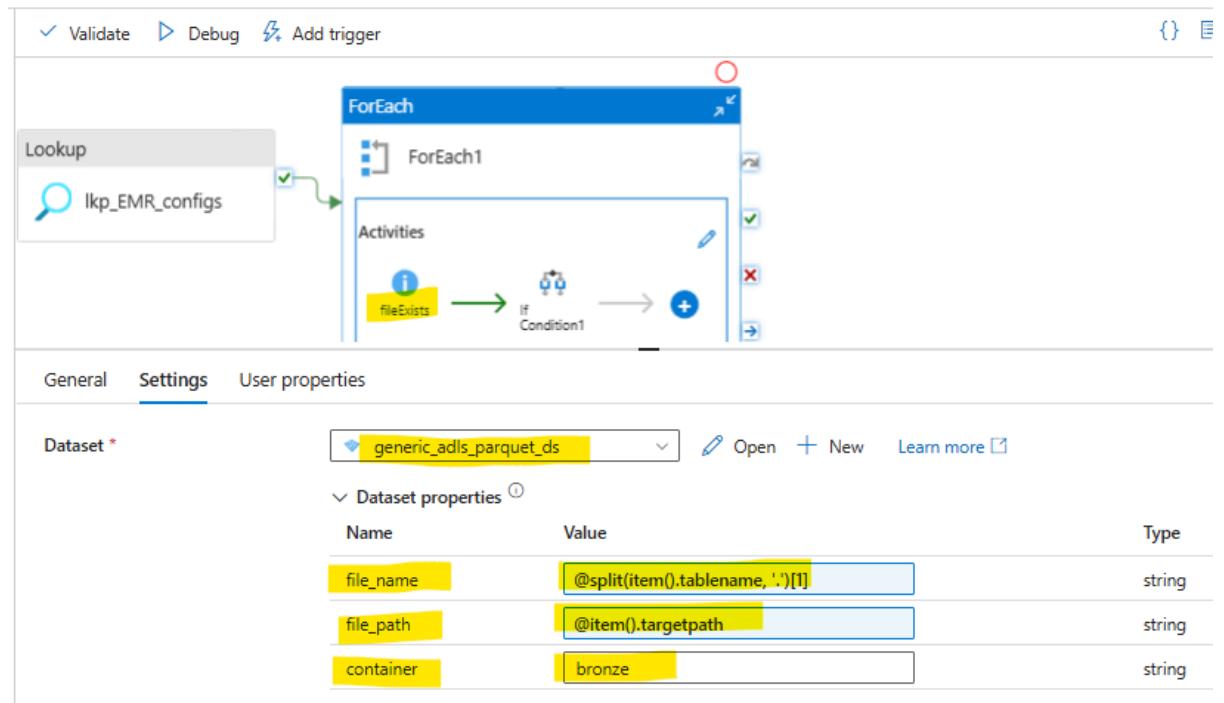
Processing Logic Within ForEach Activity:

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

The screenshot shows the Azure Data Factory pipeline editor with a 'ForEach' activity selected. It is connected to a 'Lookup' activity named 'lkp_EMR_configs'. The 'ForEach' activity has a single child activity labeled 'Activities'. The 'Settings' tab is selected at the bottom. Under 'Sequential', there is an unchecked checkbox. The 'Batch count' field is empty. The 'Items' field contains the expression '@activity('lkp_EMR_configs').output.value'. The 'General' tab is also visible at the bottom.

a. We will use get metadata activity in order to check whether file exists in Bronze container:

To file name we will use below logic - @split(item().tablename, '.')[1]
file_path is present in lookup file as targetpath
And container name we will explicitly mention as bronze as shown below



This will check if the file exists in the Bronze container. Based on the file's presence or absence, we will use an If Condition activity to determine the subsequent processing steps.

b. Use an If Condition activity based on the file's existence.

Condition 1: File Exists (True) => Move the file to the Archive folder.

condition:

```
@and>equals(activity('fileExists').output.exists,true),equals(item().is_active, '1'))
```

The screenshot shows the Microsoft Fabric Pipeline expression builder interface. At the top, there's a toolbar with 'Validate all' and 'Publish all'. Below it is a navigation bar with 'Validate', 'Debug', 'Add trigger', and other options. The main area displays a flowchart titled 'Activities' with three nodes: 'fileExists', 'If Condition1', and 'If Condition2'. A tooltip for 'If Condition1' indicates it has two outputs. The 'Activities' node has a tooltip stating: '@and>equals(activity('fileExists').output.exists,true),equals(item().is_active, '1'))'. Below the flowchart, there are tabs for 'General', 'Activities (2)', and 'User properties'. Under 'Activities (2)', the 'Expression' field contains the same conditional logic. The 'Case' section shows two entries: 'True' leading to 'ArchiveFile' (1 Activity) and 'False' leading to 'Copy data2' (1 Activity). To the right, there's a sidebar for 'ForEach iterator' with sections for 'Current item' and 'ForEach1'. At the bottom are 'OK' and 'Cancel' buttons.

Source: Container: Bronze, Path: hosa, File: encounters

The screenshot shows the Microsoft Fabric Source configuration interface. At the top, there are validation and runtime check buttons. The main pane shows a pipeline path: 'pl_emr_src_to_landing' > 'ForEach1' > 'If Condition1 - True activities'. Below this, there are tabs for 'General', 'Source' (which is selected), 'Sink', 'Mapping', 'Settings', and 'User properties'. The 'Source dataset' dropdown is set to 'generic_adls_parquet_ds'. Under 'Dataset properties', there are three entries:

Name	Value	Type
file_name	@split(item().tablename, ':')[0]	string
file_path	@item().targetpath	string
container	bronze	string

Below these, 'File path type' is set to 'File path in dataset'. There are also fields for 'Start time (UTC)' and 'End time (UTC)', and a 'Filter by last modified' section. At the bottom, there's a 'Recursively' checkbox.

Target: Container: Bronze,

```
File_path -Path: hosa/archive/<year>/<month>/<day> =>
@concat(item().targetpath, '/archive',
formatDateTime(utcNow(), 'yyyy'), '/',
formatDateTime(utcNow(), '%M'), '/',
formatDateTime(utcNow(), '%d'))
```

File_name - @split(item().tablename, '.')[1]

The screenshot shows the Microsoft Fabric Pipeline expression builder interface. On the left, there's a pipeline diagram with a 'ForEach1' loop containing an 'ArchiveFile' activity. On the right, the 'Pipeline expression builder' window is open, showing the expression `@split(item().tablename, '.')[1]` highlighted in yellow. Below it, the 'Sink' tab is selected in the pipeline editor, and the 'generic_adls_parquet_ds' dataset is chosen. In the 'Sink dataset' section, the 'file_name' field is set to `@split(item().tablename, '.')[1]`. The 'file_path' field is set to `@concat(item().targetpath, '/archive/')`, which is also highlighted in yellow. Other fields like 'container' are set to 'bronze'. The 'Activity outputs' tab in the builder shows the 'fileExists' output, which is the dynamic content used in the pipeline.

c. Determine if it's a full load or incremental load using If condition.

`@equals(items().loadtype, 'Full')`

The screenshot shows the Microsoft Fabric Pipeline expression builder interface. The pipeline diagram on the left shows an 'Activities' block with an 'if' condition. The 'Expression' field in the Activities block is set to `@equals(items().loadtype, 'Full')`, which is highlighted in yellow. The 'Activity outputs' tab in the builder shows the 'items()' output, which is the dynamic content used in the pipeline. A warning message at the bottom of the builder window states: "'items' is not a recognized function".

If "If condition" holds true => Full Load => Copy all data from the database table. => Enter Log details in the audit table:

Folder and File Structure

Bronze Container:

Source Path: bronze/hosa

Target Path for Data Loads: bronze/<target-path>

The screenshot shows the Azure Data Factory interface with two 'Copy data' activities highlighted.

Top Activity (Source Tab):

- Activity Name: pl_emr_src_to_landing > ForEach1 > If Condition2 - True activities
- Source dataset: generic_sql_ds
- Dataset properties:
 - db_name: @item().database
 - schema_name: @split(item().tablename, '.')[0]
 - table_name: @split(item().tablename, '.')[1]

Bottom Activity (Sink Tab):

- Activity Name: pl_emr_src_to_landing > ForEach1 > If Condition2 - True activities
- Sink dataset: generic_adls_parquet_ds
- Dataset properties:
 - file_name: @split(item().tablename, '.')[1]
 - file_path: @item().targetpath
 - container: bronze

Query: @concat('select * "','"item().datasource," as datasource from
' , item().tablename)

The screenshot shows the Microsoft Fabric Data Flow interface. A 'Copy data' activity named 'Full_load_cp' is selected. In the 'Source' tab of the activity settings, the 'Query' section contains the expression: `@concat('select *','','item().datasource,' as datasource from ', item().tablename)'`. This query is used to select all columns from a table based on the current item's database, schema, and table names.

Enter Log details in the audit table:

Query: `@concat('insert into audit.load_logs(data_source,tablename,numberofrowscopied,watermarkcolumnname,loaddate) values (''',item().datasource,''', ''',item().tablename,''', ''',activity('Full_Load_CP').output.rowscopied,''', ''',item().watermark,''', ''',utcNow(),''')')`

The screenshot shows the Microsoft Fabric Data Flow interface. A 'Copy data' activity named 'Full_load_cp' is selected. In the 'Settings' tab of the activity settings, under 'Dataset properties', the 'schema_name' is set to 'aa' and the 'table_name' is set to 'aa'. In the 'Source' tab, the 'Query' section contains the expression: `@concat('insert into audit.load_logs(data_source,tablename,numberofrowscopied,watermarkcolumnname,loaddate) values (''',item().datasource,''', ''',item().tablename,''', ''',activity('Full_Load_CP').output.rowscopied,''', ''',item().watermark,''', ''',utcNow(),''')')`. This query inserts data into the audit table with specific column values derived from the current item's properties.

If condition is false => Incremental Load

(Fetch incremental data using the last fetched date) using Lookup=>
Incremental load using copy activity => Enter log details in the audit table:

Lookup:

The screenshot shows the Microsoft Fabric Pipeline expression builder interface. A pipeline step named "pl_emr_src_to_landing > ForEach1 > If Condition2 - False activities" is selected. The "Source dataset" is set to "AzureDatabricksDeltaLakeDataset". The "Query" field contains the following expression:

```
@concat('select coalesce(cast(max(loaddate) as date)', ''', '1900-01-01', '') as last_fetched_date from audit.load_logs where ', ' data_source='', item().datasource, '' and tablename='', item().tablename, '')'
```

The "Pipeline expression builder" pane on the right shows the expression with syntax highlighting and code completion suggestions for dynamic content.

Incremental load:

Source Path: bronze/hosa

The screenshot shows the Microsoft Fabric Pipeline editor. A pipeline step named "pl_emr_src_to_landing > ForEach1 > If Condition2 - True activities" is selected. The "Source" tab is active, showing a "Copy data" activity. The "Source dataset" is set to "generic_sql_ds". The "Dataset properties" section contains the following dynamic content:

Name	Value
db_name	@item().database
schema_name	@split(item().tablename, '.')[0]
table_name	@split(item().tablename, '.')[1]

Target Path for Data Loads: bronze/<target-path>

Copy data

Full_load_cp

Sink Mapping Settings User properties

Sink dataset * generic_adls_parquet_ds Open New Learn more

Dataset properties

Name	Value
file_name	@split(item().tablename, '.')[1]
file_path	@item().targetpath
container	bronze

Query: `@concat('select * ','+',item().datasource,'" as datasource from ',item().tablename,' where ',item().watermark,' >= ''',activity('Fetch_logs').output.firstRow.last_fetched_date,''))`

Source dataset * generic_sql_ds Open

Dataset properties

Name	Value
db_name	@item().database
schema_name	@split(item().tablename, '.')[0]
table_name	@split(item().tablename, '.')[1]

Use query

Query

Query timeout (minutes)

Isolation level

Pipeline expression builder

```
@concat('select * ','+',item().datasource,'" as datasource from ',item().tablename,' where ',item().watermark,' >= ''',activity('Fetch_logs').output.firstRow.last_fetched_date,''))
```

Activity outputs Parameters System variables Functions Variables

ArchiveFile
ArchiveFile activity output

Fetch_logs
Fetch_logs activity output

Fetch_logs first row
Data of the first row

OK Cancel

Lookup:

The screenshot shows the Microsoft Fabric Pipeline expression builder interface. On the left, there's a pipeline diagram with activities: 'Fetch_logs' followed by 'Copy data' (Incremental_Load_CP) and 'Lookup2'. The 'Lookup2' activity is selected. On the right, the 'Pipeline expression builder' window is open, displaying a query:

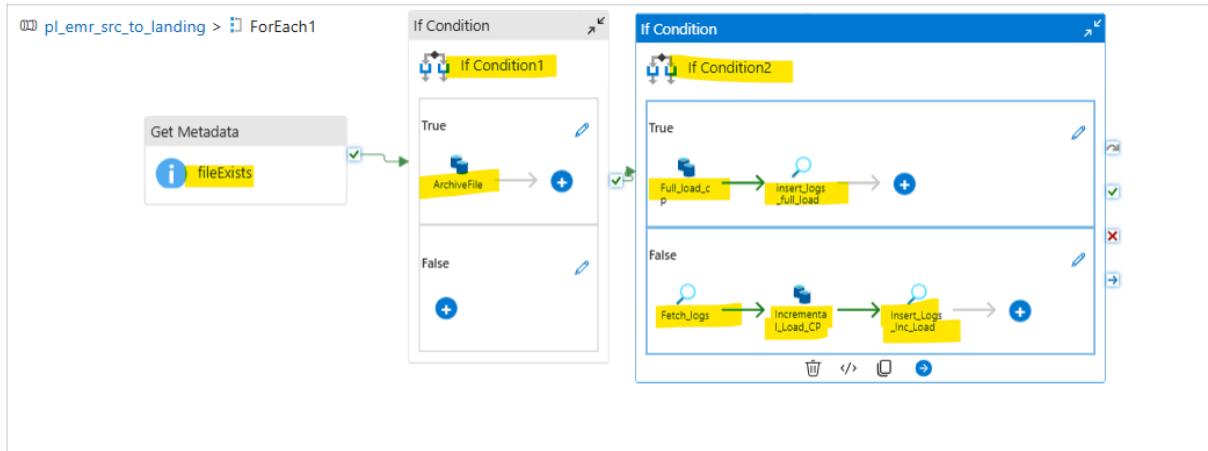
```
@concat('insert into audit.load_logs(data_source,tablename, numberofrowscopied,watermarkcolumnname,loaddate) values (''', item().datasource, '''', item().tablename, '''', item().activity (''Incremental_Load_CP'').output.rowscopied, '''', item().watermark, '''', utcNow(), ''')')
```

The 'Activity outputs' tab is selected, showing the output of the 'Fetch_logs' activity named 'ArchiveFile'. Below the builder are tabs for 'Parameters', 'System variables', 'Functions', and 'Variables', along with an 'OK' and 'Cancel' button.

This is our complete pipeline:

Before running the pipeline for each activity select the “sequential” option as shown below.

The screenshot shows the Microsoft Fabric pipeline settings for 'pl_emr_src_to_landing'. The 'Activities' pane on the left lists various activity types. The main area shows a 'ForEach' loop activity. Inside the 'ForEach' loop, there is a 'Lookup' activity named 'lkp_EMR_configs' followed by an 'Activities' container. The 'Activities' container contains three sequential steps: 'fileExists', 'If Condition1', and 'If Condition2'. In the bottom-left of the pipeline editor, the 'Settings' tab is selected, and the 'Sequential' option is chosen under the 'Items' section. The value '@activity(''lkp_EMR_configs'').output...' is also visible.



But limitation with this pipeline is it is sequential which will we resolve in part 2

Part 2:

In this section, we will focus on improving our data pipeline and governance.

Transition from a local Hive Metastore to Databricks Unity Catalog for centralized metadata management and improved data governance.

We will first create a new databricks workspace “tt-hc-adb-ws”, select the “Premium (+ Role-based access controls)” while creating the workspace.

To configure the Metastore for the Unity catalog follow below steps:

Step 1: Configure Storage for your Metastore - ADLS gen2 table -

Created a storage account for ADLS Gen2.- “ttunitycatalogsa”. Create a storage container that will hold your unity catalog metastore's metadata and managed data. container - unityroot

Name	Last modified
\$logs	12/16/2024, 2:36:18 PM
unityroot	12/16/2024, 2:36:49 PM

2. In Azure, create a databricks access connector that holds a managed identity and give it access to the storage container.

Go to “Access Connector for Azure Databricks” => create databricks access connector “ttdatabricksaccesscon”

The screenshot shows the Microsoft Azure portal interface. At the top, there's a search bar and a navigation bar with 'Microsoft Azure' and 'Search resources, services, and docs (G+/-)'. Below that, a breadcrumb trail shows 'All services > AccessConnectorCreate-20241216143854 | Overview > tt-databricks-access-connector'. The main title is 'tt-databricks-access-connector' with a subtitle 'Access Connector for Azure Databricks'. On the left, a sidebar has 'Overview' selected, followed by 'Activity log', 'Access control (IAM)', 'Tags', 'Settings', 'Automation', and 'Help'. The 'Overview' section displays resource details: Resource group (move) : [trendytech-azure-project](#), Location : East US, Subscription (move) : [Pay-As-You-Go](#), Subscription ID : 136f20b5-00f7-4eb5-a52e-0843e7ad1034, and Tags ([edit](#)) : [Add tags](#).

Then Go to your storage account and click on access control (IAM)=> click on add -> role assignment -> storage blob data contributor -> in managed identity select the databricks connector that we have created.

The screenshot shows the Microsoft Azure portal interface for adding a role assignment. The URL is 'Storage accounts > unitycatalogsanew | Access Control (IAM) > Add role assignment'. The 'Members' tab is selected. Under 'Role', 'Storage Blob Data Contributor' is chosen. Under 'Assign access to', 'Managed identity' is selected. The 'Members' section shows '+ Select members'. A modal window titled 'Select managed identities' lists 'Subscription *' (Pay-As-You-Go), 'Managed identity' (Access Connector for Azure Databricks (3)), and 'Select' and 'Search by name' buttons. The results show two entries: 'unity-catalog-access-connector /subscriptions/136f20b5-00f7-4eb5-a52e-0843e7ad1034' and 'unity-catalog-access-connector /subscriptions/136f20b5-00f7-4eb5-a52e-0843e7ad1034'. Under 'Selected members:', 'tt-databricks-access-connector /subscriptions/136f20b5-00f7-4eb5-a52e-0843e7ad1034' is listed. At the bottom, there are 'Review + assign', 'Previous', 'Next', 'Select', and 'Close' buttons.

Save it and then review and assign it.

Step 2: You will go to “<https://accounts.azuredatabricks.net>” in your browser. & there we will create the metastore.

Clicked on Catalog => create metastore -> region should be the same where you have the storage account and the workspaces -> storage account path:

Ex:

`unityroot@ttunitycatalogsa.dfs.core.windows.net/`

-> access connector id (mention the access connector id)

The screenshot shows the 'Create metastore' wizard in the Azure Data Bricks web interface. The left sidebar has 'Catalog' selected. The main area shows step 1: 'Create metastore' with a 'Name' field containing 'MetastoreForCentralIndia'. Step 2: 'Assign to workspaces' is shown below. The 'Region' dropdown is set to 'centralindia'. Below it, a note says 'Select the region for your metastore. You will only be able to assign workspaces in this region to this metastore.' The 'ADLS Gen 2 path (optional)' field contains 'unityroot@unitycatalogsanew.dfs.core.windows.net/'. A note below it says 'Optional location for storing managed tables data across all catalogs in the metastore. Once configured this path can't be removed.' The 'Access Connector Id' field contains '/subscriptions/136f20b5-00f7-4eb5-a52e-0843e7ad1034/resourceGroups/trendytech-azure-project/'. A 'Advanced options' button is at the bottom.

Also we will assign this metastore to our workspace.

Go to Catalog => select the recently created metastore => click on Assign to workspace => select the workspace that you want to assign.

The screenshot shows the Azure Databricks Catalog interface. On the left, there's a sidebar with options like Workspaces, Catalog (which is selected), Usage, User management, Cloud resources, Previews, and Settings. The main area is titled 'Catalog > metastoreforcentralindia'. It has tabs for Configuration and Workspaces, with 'Workspaces' selected. A search bar says 'Filter workspaces'. A button 'Assign to workspace' is visible. Below is a table with columns Name, Resource group, Region, Subscription, and Created. One row is highlighted: 'tt-hc-adb-ws' (Resource group: trendytech-azure-project, Region: centralindia, Subscription: 136f20b5-00f7-4eb5-a52e-08..., Created: last Friday at 4:55 PM). There's also an 'Open' button and a more options menu.

And we will assign “create catalog” permission to the user for this metastore if you are not getting the option to create a new Catalog.

The screenshot shows the Databricks Catalog Explorer. The left sidebar includes New, Workspace, Recents, Catalog (selected), Workflows, Compute, SQL, SQL Editor, Queries, and Dashboards. The main area shows the 'Catalog Explorer > metastoreforcentralindia' page. It displays details like 'centralindia' and 'External delta sharing: disabled'. The 'Permissions' tab is selected, showing 'Allowed JARs/Init Scripts' and buttons for 'Grant' and 'Revoke'. A table lists a principal (redacted) with the privilege 'CREATE CATALOG'. A 'CREATE CATALOG' button is at the bottom right.

Now we will create the catalog “tt_hc_adb_ws”.

The screenshot shows the 'Create a new catalog' dialog box. The sidebar on the left is identical to the previous one. The dialog box has a title 'Create a new catalog'. It contains instructions: 'A catalog is the first layer of Unity Catalog's three-level namespace and is used to organize your data assets. Learn more'. It has fields for 'Catalog name' (set to 'tt_hc_adb_ws'), 'Type' (set to 'Standard'), and 'Storage location' (with 'Select external location' dropdown and 'sub/path' input field). A note at the bottom says 'Location in cloud storage where data for managed tables will be stored. If not specified, the location will default to the metastore root location.' At the bottom are 'Cancel' and 'Create' buttons.

The screenshot shows the Databricks Catalog interface. On the left sidebar, under the 'Catalog' section, 'tt_hc_adb_ws' is selected. The main area displays a 'Quick access' list of catalogs, with 'tt_hc_adb_ws' highlighted in yellow.

Name	Owner	Created at
hive_metastore		
main	aratishatti15@gmail.co...	2024-12-16 15:33:23
samples	System user	2024-12-16 15:33:22
system	System user	2024-12-16 15:33:22
tt_hc_adb_ws	aratishatti15@gmail.com	2024-12-16 16:23:52

After creating this catalogue, set this catalogue as a default catalog for the workspace “tt-hc-adb-ws”. For this follow below steps:

Open your Databricks Workspace.

Navigate to Admin Settings → Workspace Settings.

Set the Default Catalog to your desired catalog (e.g., tt_hc_adb_ws).

Refer below screenshot for more clarity.

The screenshot shows the 'Settings' page in the Databricks Admin interface. Under the 'Advanced' tab, the 'Default catalog for the workspace' field is set to 'tt_hc_adb_ws'. A note indicates that this action permanently purges Spark driver logs and historical metrics snapshots for all clusters in the workspace.

Also to organize the notebook we will create the folder as shown below.

The screenshot shows the Databricks workspace navigation bar with 'Workspace > Users > trendytech.sumit501@outlook.com'. Below the navigation bar is a list of items, including a Git folder, several sub-folders (1. Set up, 2. API extracts, 3. Silver, 4. Gold), a Query, and a Notebook named 'keyvault test'.

Name	Type	Owner	Created at
Trendytech-Azure-Project	Git folder	Trendytech Insights	2024-12-02 22:02:28
1. Set up	Folder	Trendytech Insights	2024-11-25 00:14:09
2. API extracts	Folder	Trendytech Insights	2024-11-27 17:49:52
3. Silver	Folder	Trendytech Insights	2024-11-25 00:55:53
4. Gold	Folder	Trendytech Insights	2024-11-25 01:20:51
Gold queries	Query	Trendytech Insights	2024-12-03 00:27:54
keyvault test	Notebook	Trendytech Insights	2024-11-25 00:10:58

1. Set up:

Workspace > Users > [trendytech.sumit501@outlook.com](#) >

1. Set up ☆

Name	Type	Owner	Created at	⋮
1. audit_ddl	Notebook	Trendytech Insights	2024-11-25 00:14:14	⋮
2. adls_mount	Notebook	Trendytech Insights	2024-11-25 00:16:49	⋮

2. API extracts

Workspace > Users > [trendytech.sumit501@outlook.com](#) >

2. API extracts ☆

Name	Type	Owner	Created at	⋮
ICD Code API extract	Notebook	Trendytech Insights	2024-11-28 08:57:55	⋮
NPI API extract	Notebook	Trendytech Insights	2024-11-27 17:50:07	⋮

3. Silver

Workspace > Users > [trendytech.sumit501@outlook.com](#) >

3. Silver ☆

Name	Type	Owner	Created at	⋮
Claims	Notebook	Trendytech Insights	2024-11-25 00:55:54	⋮
CPT codes	Notebook	Trendytech Insights	2024-11-28 23:13:09	⋮
Departments_F	Notebook	Trendytech Insights	2024-11-25 00:55:55	⋮
Encounters	Notebook	Trendytech Insights	2024-11-25 00:55:55	⋮
ICD Code	Notebook	Trendytech Insights	2024-11-28 09:08:35	⋮
NPI	Notebook	Trendytech Insights	2024-11-26 14:42:54	⋮
Patient	Notebook	Trendytech Insights	2024-11-25 00:55:54	⋮
Providers_F	Notebook	Trendytech Insights	2024-11-25 00:55:55	⋮
Transactions	Notebook	Trendytech Insights	2024-11-25 00:55:53	⋮

4. Gold

Workspace > Users > [trendytech.sumit501@outlook.com](#) >

4. Gold ☆

Name	Type	Owner	Created at	⋮
dim_cpt_code	Notebook	Trendytech Insights	2024-11-28 23:36:42	⋮
dim_department	Notebook	Trendytech Insights	2024-11-25 01:42:29	⋮
dim_icd_code	Notebook	Trendytech Insights	2024-11-28 09:11:38	⋮
dim_npi	Notebook	Trendytech Insights	2024-11-28 09:15:54	⋮
dim_patient	Notebook	Trendytech Insights	2024-11-25 01:20:59	⋮
dim_provider	Notebook	Trendytech Insights	2024-11-25 01:29:07	⋮
fact_transaction	Notebook	Trendytech Insights	2024-11-26 14:22:04	⋮

Note: After creating the databricks workspace, enable the DBFS.

Create the catalog “tt-hc-adb-ws” as shown below.

The screenshot shows the Databricks interface with the 'Catalog' tab selected. A search bar at the top right contains the placeholder 'Search data, notebooks, recents, and more...'. Below it are several buttons: 'Delta Sharing', 'Clean Rooms', 'External Data', 'Browse DBFS', and 'Add dbfs'. On the left, a sidebar lists navigation options: 'New', 'Workspace', 'Recents', 'Catalog' (which is highlighted in yellow), 'Workflows', 'Compute', 'SQL', 'SQL Editor', 'Queries', 'Dashboards', 'Genie', and 'Alerts'. The main area is titled 'Catalog' and shows 'Arati Hatti's Cluster 14 GB, 4 Cores'. It includes a search bar 'Type to search...' and a 'Quick access' section with tabs for 'Recents', 'Favorites', and 'Catalogs' (which is also highlighted in yellow). A large button labeled 'Create catalog' is visible. A table lists existing catalogs:

Name	Owner
hive_metastore	
main	aratihatti15_gmail.com
samples	System user
system	System user

We will create the audit database using as shown below

The screenshot shows the Databricks SQL Editor. The title bar says '1. audit_ddl' and 'SQL'. The editor window displays the following SQL code:

```
%sql  
CREATE SCHEMA IF NOT EXISTS tt_hc_adb_ws.audit;  
  
CREATE TABLE IF NOT EXISTS tt_hc_adb_ws.audit.load_logs (  
    data_source STRING,  
    tablename STRING,  
    numberofrowscopied INT,  
    watermarkcolumnname STRING,  
    loaddate TIMESTAMP  
);
```

The code was run successfully, indicated by a green checkmark and the message '2 minutes ago (23s)'. The status bar at the bottom right shows 'Last edit was 3 days ago'.

Note: Before proceeding ahead we will create the key vault in azure and will store key for our ADLS gen2 storage in it. Refer below steps for more clarification.

Key Vault creation in Azure:

Steps to Store Key in Key Vault and Create Secret Scope in Databricks:

1. Store the Storage Account Access Key in Azure Key Vault

Create an Azure Key Vault (if not already created):

Go to the Azure Portal. => Search for "Key Vault" and click Create.=> Fill in the necessary details (Resource Group, Key Vault Name, Region) and click Review + Create.

Refer below screenshot for more details.

The screenshot shows the 'Create a key vault' wizard in the Azure portal. The 'Project details' section is visible, including fields for 'Subscription' (Pay-As-You-Go), 'Resource group' (trendytech-azure-project), and 'Key vault name' (kv-healthcare-project). The 'Instance details' section shows 'Region' (Central India) and 'Pricing tier' (Standard). The 'Recovery options' section is present but not filled. The URL in the browser bar is 'Home > Key vaults > Create a key vault'.

In Access configuration select option vault access policy option and select define permission for your username.

The screenshot shows the 'Create a key vault' wizard in the Azure portal, focusing on 'Access policies'. It highlights the 'Vault access policy' option under 'Permission model'. The 'Resource access' section includes checkboxes for Azure Virtual Machines, Azure Resource Manager, and Azure Disk Encryption. The 'Access policies' section shows a table with columns for Name, Email, Key Permissions, and Secret Permissions. A row for a user named 'aratishatti15@gmail.com' is selected, showing permissions for Get, List, Update, Create, Import, Delete, Recover, and Set operations. Navigation buttons at the bottom include 'Previous', 'Next', and 'Review + create'.

Then create the key vault.

Add the Storage Account Access Key to the Key Vault, to add follow below steps:

First get the storage account key, to get the storage account access key follow below steps:

- => Go to your Azure Storage Account => Security & Networking> Access Keys.
- => Copy the key under the "Key1" or "Key2" section.

The screenshot shows the 'Access keys' page for a storage account named 'ttadlsdevnew'. The left sidebar has a 'Security + networking' section with 'Access keys' highlighted. The main area displays two keys: 'key1' (last rotated 10/12/2024) and 'key2'. Each key has a 'Show' button to reveal its value. A note at the top says: 'Access keys authenticate your applications' requests to this storage account. Keep your keys in a secure location like Az Key Vault, and replace them often with new keys. The two keys allow you to replace one while still using the other.' Below the keys, there's a 'Connection string' field with a 'Show' button.

To store keys in the key vault, navigate to your Key Vault.

Click Secrets > + Generate/Import. => Enter the following details:

Name: Provide a descriptive name for the secret (e.g., adls-access-key).

Value: Copy the access key of your storage account and paste it here.

=> Click Create.

The screenshot shows the 'Create a secret' form in the Azure Key Vault. The 'Name' field is set to 'access-key-ttadlsdevnew'. The 'Secret value' field contains a redacted key value. Other fields include 'Content type (optional)', 'Set activation date', 'Set expiration date', 'Enabled' (set to Yes), and 'Tags' (0 tags). At the bottom are 'Create' and 'Cancel' buttons.

2. Assign Key Vault Access to Databricks

To allow Databricks to access the Key Vault:

Go to the Key Vault => Navigate to your Key Vault in the Azure Portal. => Click on Access Policies > + Add Access Policy. => Grant Permissions:

Template: Select Secret Management.

Home > kv-heathcare-project | Access policies >

Create an access policy

kv-heathcare-project

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 Select all Get List Update Create Import Delete Recover	Secret Management Operations Select all Get List Set Delete Recover Backup Restore	Certificate Management Operations Select all Get List Update Create Import Delete Recover

Previous Next

Principal: Add your Azure Databricks Managed Identity.

You can find the Managed Identity from the Databricks Resource in Azure Portal under Managed Identity.

Microsoft Azure

tt-hc-adb-ws

Azure Databricks Service

Overview

Status: Active

Resource group: [trendytech-azure-project](#)

Location: Central India

Subscription: Pay-As-You-Go

Managed Resource Group: [databricks-rg-tt-hc-adb-ws-tyz24a7ak5](#)

URL: <https://adb-360847858525250.10.azure.databricks.net>

Pricing Tier: Premium (+ Role-based access control)

Now select this managed resource group and check the managed identity and define policy for it.

Home > tt-hc-adb-ws >

databricks-rg-tt-hc-adb-ws-tyz24a7ak54oe

Resource group

Search Create Manage view Delete resource group Refresh Export to CSV Open query Assign

Overview Essentials

Activity log Access control (IAM) Tags Resource visualizer Events Settings Cost Management Monitoring

Resources Recommendations (1)

Filter for any field... Type equals all Location equals all Add filter

Showing 1 to 5 of 5 records. Show hidden types ⓘ

Name	Type
dbmanagedidentity	Managed Identity
dbstoragesonvyaont3lc	Storage account

Save the Policy:

Home > kv-heathcare-project | Access policies >

Create an access policy ...

kv-heathcare-project

Permissions Principal Application (optional) 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](#)

Selected item

No item selected

Previous

Next

Click Add, then Save the changes.

3. Create a Secret Scope in Databricks

Once the secret is stored in Key Vault, create a secret scope in Databricks.

Open Databricks Workspace:

Go to your Azure Databricks Workspace.

While creating the secret scope, edit the url till .net and add "#secrets/createScope"

Create a Secret Scope Linked to Key Vault:

Enter the following:

Scope Name: Give a name to the scope (e.g., adlsdevnew).

Scope Backing: Select Azure Key Vault.

DNS Name: Enter the Key Vault URL.

To get these details go to key vault => Setting => Properties

Refer below screenshot for more clarity.

The screenshot shows the Azure Key Vault properties page for a vault named 'kv-heathcare-project'. The 'Properties' tab is active. Key details visible include:

- Vault URI:** https://kv-heathcare-project.vault.azure.net/
- Name:** kv-heathcare-project
- Sku (Pricing tier):** Standard
- Location:** centralindia
- Resource ID:** /subscriptions/136f20b5-00f7-4eb5-a52e-0843e7ad1034/resourceGroups/kv-heathcare-project/providers/Microsoft.KeyVault/vaults/kv-heathcare-project
- Subscription ID:** 136f20b5-00f7-4eb5-a52e-0843e7ad1034
- Subscription Name:** Pay-As-You-Go
- Directory ID:** 978c93b0-7633-4a7a-91af-ecf633ade5c8
- Directory Name:** Default Directory
- Soft-delete:** Soft delete has been enabled on this key vault

Now provide these details in scope creation and create the scope.

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 [?](#)
tt-hc-kv

Manage Principal [?](#)
Creator

Azure Key Vault [?](#)

DNS Name
https://kv-heathcare-project.vault.azure.net/

Resource ID
/subscriptions/136f20b5-00f7-4eb5-a52e-0843e7ad1034/resourceGroups/trendytec

Verify the Secret Scope:

Similarly first we store credentials for sql database (Username and password) and databricks also (In case of Databricks we will store access token).

To get access token in Databricks got to Setting => Developer => Access token => Generate Access token

Name	Type	Status	Expiration date
db-hc-adb-ws-at	Application Configuration	✓ Enabled	
azure-sql-db-password	Application Configuration	✓ Enabled	
azure-sql-db-username	Application Configuration	✓ Enabled	
access-key-ttadlsdevnew	Application Configuration	✓ Enabled	

Now for ADF and ADLS gen we will create application and will provide permission (You can Provide all the permission) as shown by Sumit Sir in the video

The screenshot shows the 'Access policies' section of a Key Vault named 'tt-health-care-kv'. The left sidebar includes options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and Access policies (which is selected). The main area displays a table of access policies:

Name	Email	Key Permissions	Secret Permissions	Certificate Permissions
AzureDatabricks		All	Get, List	All
tt-hc-adf-app		All	All	All
tt-healthcare-adf-dev		Get, Get, List, Update, Cre...	Get, List, Set, Delete, Reco...	

To perform role assignments in Azure Databricks and Azure Data Factory (ADF), follow the steps outlined below.

For Azure Databricks:

To assign the role : Go to resource => IAM => Add => Role Assignment.

Define Contributor role for the application that is created for databricks as shown below.

The screenshot shows the 'Add role assignment' page for an Azure Databricks workspace named 'tt-hc-adb-ws'. The top navigation bar shows 'Home > Azure Databricks > tt-hc-adb-ws | Access control (IAM) >'. The main area has tabs for 'Role' (selected), 'Members' (with a required asterisk), 'Conditions', and 'Review + assign'. Below this, it says 'A role definition is a collection of permissions. You can use the built-in roles or you can create your own custom roles. Learn more ⓘ'. It lists 'Job function roles' and 'Privileged administrator roles' (selected). A warning message '⚠️ Can a job function role with less access be used instead?' is present. The search bar shows 'contributor' and the results table shows one result:

Name	Description	Type
Contributor	Grants full access to manage all resources, but does not allow you to assign roles in Azure RBAC, manage assignments in ...	BuiltinRole

At the bottom, there are buttons for 'Review + assign', 'Previous', and 'Next'.

The screenshot shows the 'Add role assignment' step in the Azure Databricks IAM interface. The 'Members' tab is selected. Under 'Assign access to', 'User, group, or service principal' is chosen. A search bar at the top right shows 'tt-'. Below it, a list of applications is shown, with 'tt-databricks' highlighted. The 'Selected members' section shows 'tt-databricks' selected. At the bottom, there are 'Review + assign', 'Previous', 'Next', 'Select', and 'Close' buttons.

Note: In condition give select “Allow user to assign all roles (highly privileged)”

Also for your username define the owner role if it is not present as shown below.

The screenshot shows the 'Access control (IAM)' blade in the Azure portal. It displays the number of role assignments for the current subscription (2). The 'Privileged' role is selected. The table below lists the assigned users and their roles:

Name	Type	Role	Scope	Condition
Trendytech Insights	User	Owner	Subscription (Inherited)	None
tt-hc-adb	App	Owner	This resource	Add

For ADF:

To assign the role : Go to resource => IAM => Add => Role Assignment.

Define Contributor role for the application that is created for databricks as shown below.

Home > tt-healthcare-adf-dev

 tt-healthcare-adf-dev | Access control (IAM) ⚡ ...

Data factory (V2)

Overview	Number of role assignments for this subscription	Privileged
16	4000	3

[View assignments](#)

 Tags

 Diagnose and solve problems

> Settings

> Getting started

> Monitoring

> Automation

> Help

 Search by name or email

Type : All Role : All Scope : All scopes Group by : Role

3 items (1 Users, 2 Service Principals)

Name	Type	Role	Scope
Owner (3)			
Trendytech Insights	User	Owner ⓘ	Subscription (Inherited)
tt-adf-healthcare-app	App	Owner ⓘ	This resource
tt-hc-adf-app	App	Owner ⓘ	This resource

We will use the code in the `adls_mount` notebook to mount the containers in our storage account.

4 minutes ago (54s) 1 Python

```

storageAccountName = "ttadlsdevnew"
storageAccountAccessKey = dbutils.secrets.get('tt-hc-kv', 'access-key-ttadlsdevnew')
mountPoints=["gold","silver","bronze","landing","configs"]
for mountPoint in mountPoints:
    if not any(mount.mountPoint == f"/mnt/{mountPoint}" for mount in dbutils.fs.mounts()):
        try:
            dbutils.fs.mount(
                source = "wasbs://{}@{}.blob.core.windows.net".format(mountPoint, storageAccountName),
                mount_point = f"/mnt/{mountPoint}",
                extra_configs = {'fs.azure.account.key.' + storageAccountName + '.blob.core.windows.net': storageAccountAccessKey}
            )
            print(f"{mountPoint} mount succeeded!")
        except Exception as e:
            print("mount exception", e)

```

```

gold mount succeeded!
silver mount succeeded!
bronze mount succeeded!
landing mount succeeded!
configs mount succeeded!

```

Now we will create a linked service for key vault and Databricks as shown below.

New linked service

Azure Key Vault

Name *

Description

Azure key vault selection method ⓘ
 From Azure subscription Enter manually

Azure subscription ⓘ

Azure key vault name *

Authentication method

Managed identity name: TT-health-projectdev
 Managed identity object ID: 8d35fbef-2451-4cce-9d24-31bf7d57b55d
[Grant Data Factory service managed identity access to your Azure Key Vault](#)

Connection successful

[Create](#) [Back](#)

Before creating the linked services, grant access permissions to the Data Factory service principal under the access policies in Key Vault and provide key and secret access permission.

[+ Create](#) [Refresh](#) | [Delete](#) [Edit](#)

Access policies enable you to have fine grained control over access to vault items. [Learn more](#)

Search	Permissions : All	Type : All		
Showing 1 to 6 of 6 records.				
<input type="checkbox"/> Name ↑↓	<input type="checkbox"/> Email ↑↓	Key Permissions	Secret Permissions	Certificate Permissions
APPLICATION				
<input type="checkbox"/> AzureDatabricks		Get, List		
<input type="checkbox"/> dbmanagedentity		Get, List, Update, Create, Import, Del... Get, List, Set, Delete, Recover, Backup... Get, List, Update, Create, Import, D...		
<input type="checkbox"/> tt-adls-healthcare		Get, List, Update, Create, Import, Del... Get, List, Set, Delete, Recover, Backup... Get, List, Update, Create, Import, D...		
<input type="checkbox"/> TT-health-projectdev		Get, List, Update, Create, Import, Del... Get, List, Set, Delete, Recover, Backup...		
<input type="checkbox"/> tt-healthcare-adf		Get, List, Update, Create, Import, Del... Get, List, Set, Delete, Recover, Backup... Get, List, Update, Create, Import, D...		
USER				
<input type="checkbox"/> Arati Hatti	aratihatti15_gmail.com#EXT#@arati...	Get, List, Update, Create, Import, Del... Get, List, Set, Delete, Recover, Backup... Get, List, Update, Create, Import, D...		

After this you will be able to access secrets while creating linked services.

Note: While creating Linked service for Databricks select the existing cluster option as shown below.

Edit linked service

Azure Databricks [Learn more](#)

Authentication type *

Access Token

Access token Azure Key Vault

AKV linked service * ⓘ
tt_hc_kv_ls

Secret name *

db-hc-adb-ws-at

Edit

Secret version ⓘ

Latest version

Edit

Select cluster

New job cluster Existing interactive cluster Existing instance pool

Choose from existing clusters * ⓘ

Sumit Hatt's Personal Compute Cluster

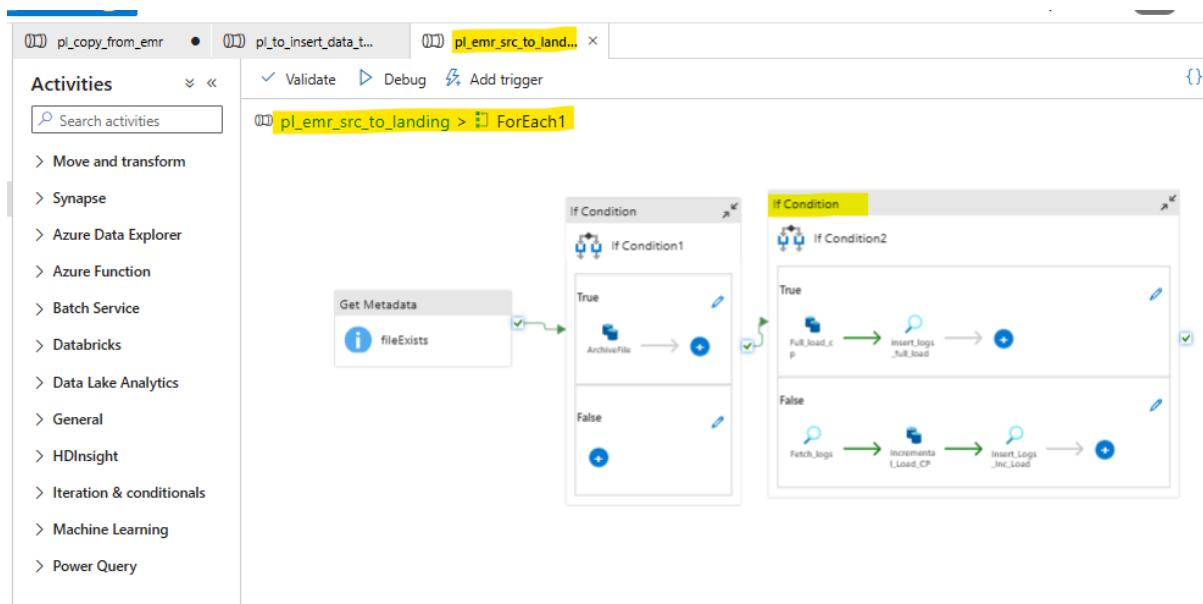
Annotations

We will update the linked services for ADLS Gen2, sql database and now instead of providing the storage account key directly, we will select the secret scope for respective ADLS gen storage, SQL DB, etc. as shown by Sumit sir in the video.

How to implement active and inactive Flag.

1. Creating new pipeline - pl_copy_from_emr

To create the pipeline “pl_copy_from_emr” and we will copy the “2nd If condition (having condition @equals(item().loadtype, 'Full'))” for copying the data into this pipeline.



In the pipeline `pl_emr_src_to_landing`, when using `@equals(item().loadtype, 'Full')`, it works because the copy activity is directly inside the For Each loop, and `item()` refers to the current element being processed during the loop's iteration.

However, when you move the copy activity to a child pipeline (triggered using the Execute Pipeline activity) and place the If condition in the parent pipeline's For Each loop, the child pipeline no longer has direct access to `item()`. Instead, you need to pass the loop data (e.g., `loadtype`) as a parameter to the child pipeline.

This is where `@equals(pipeline().parameters.Load_Type, 'Full')` comes into play. The parameter `Load_Type` is explicitly passed to the child pipeline when it's invoked, and the condition uses this parameter to determine the action. By referencing the parameter, the child pipeline can work independently while still receiving the necessary context from the parent pipeline.

So, in this pipeline update the Expression as shown below and also we have to add the parameter as shown below

```
@equals(pipeline().parameters.Load_Type, 'Full' )
```

The screenshot shows the Pipeline expression builder interface. On the left, there's a preview of a pipeline step named 'pl_copy_from_emr' with options to 'Validate', 'Debug', and 'Add trigger'. Below this is a detailed view of an 'If Condition' block. The condition is set to 'True' and points to a 'Full_load_cp' activity followed by an 'insert_logs_full_Load' activity. The 'Activities (5)' tab is selected, showing five activities: 'Full_load_cp', 'insert_logs_full_Load', 'Fetch_logs', 'Incremental_Load', and 'Insert_Logs_Inc_Load'. The 'Expression' field contains the expression '@equals(pipeline().parameters.Load_Type, 'Full')'. On the right, there's a sidebar with tabs for 'Activity outputs', 'Parameters', 'System variables', 'Functions', and 'Variables'. A search bar at the top of the sidebar contains the text 'Search'. Below it, under 'Activity outputs', are listed 'Fetch_logs' (Fetch_logs activity output) and 'Fetch_logs first row' (Data of the first row). Under 'Variables', there are entries for 'Full_load_cp' (Full_load_cp activity output) and 'Insert_Logs_Inc_Load'. At the bottom right of the builder are 'OK' and 'Cancel' buttons.

Parameter:

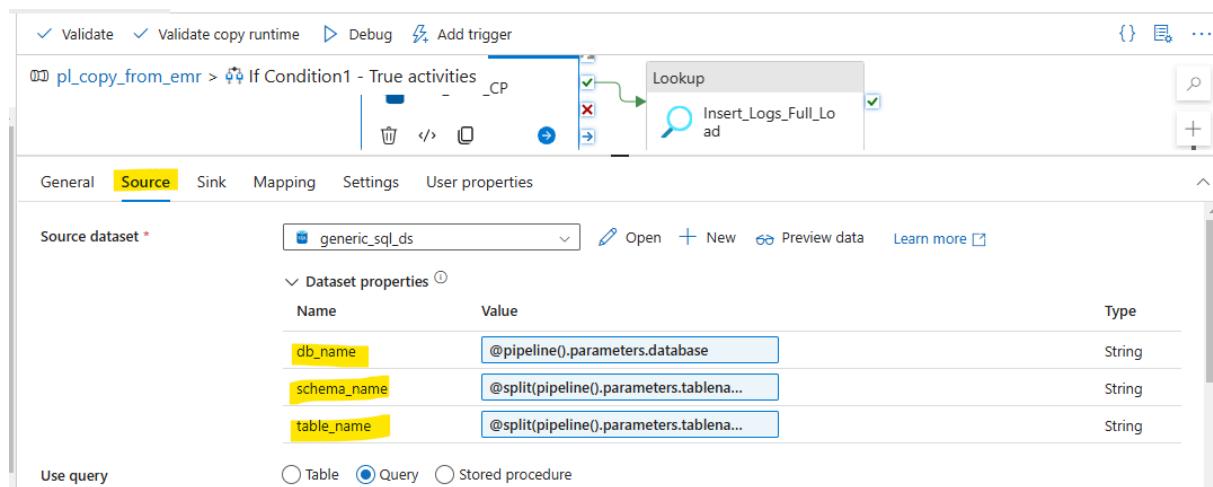
The screenshot shows the 'Parameters' section of a pipeline configuration. At the top, there are buttons for 'Validate', 'Debug', and 'Add trigger'. Below this is a list of parameters with tabs for 'Parameters', 'Variables', 'Settings', and 'Output'. The 'Parameters' tab is selected. It shows a table with columns for 'Name', 'Type', and 'Default value'. There are six parameters listed: 'Load_Type' (String type), 'database' (String type), 'tablename' (String type), 'datasource' (String type), 'targetpath' (String type), and 'watermark' (String type). Each parameter has a 'Value' input field and a delete icon. At the bottom of the table are 'New' and 'Delete' buttons.

Name	Type	Default value
Load_Type	String	Value
database	String	Value
tablename	String	Value
datasource	String	Value
targetpath	String	Value
watermark	String	Value

We will update the values for the source and sink variables for both the full load and incremental load copy activities. Since the variable values are the same for both, we will use the same variable, as demonstrated below.

Source:

```
db_name - @pipeline().parameters.database
schema_name - @split(pipeline().parameters.tablename, '.') [0]
table_name - @split(pipeline().parameters.tablename, '.') [1]
```

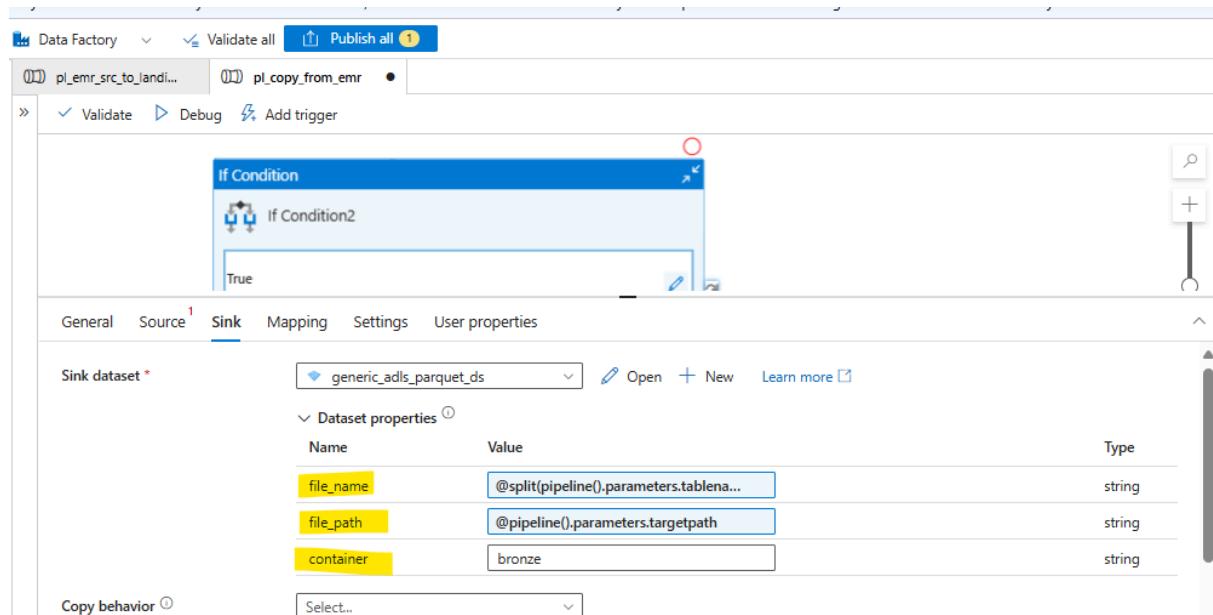


Sink:

```
file_name = @split(pipeline().parameters.tablename, '.') [1]
```

```
file_path = @pipeline().parameters.targetpath
```

Container -bronze



Also we have to update the queries for Full load as shown below

Full Load:

```
@concat('select *,''',pipeline().parameters.datasource,'' as
datasource from ',pipeline().parameters.tablename)
```

The screenshot shows the Microsoft Data Factory Pipeline expression builder interface. On the left, a pipeline diagram is visible with an 'If Condition' activity followed by another 'If Condition' activity. The main area displays a dynamic query being built:

```
@concat('select *',pipeline().parameters.datasource,' as
datasource from ',pipeline().parameters.tablename)
```

The 'Source' tab is selected in the pipeline editor. Under 'Source dataset', it is set to 'generic_sql_ds'. The 'Query' field contains the above dynamic SQL query. To the right, the 'Pipeline expression builder' window is open, showing the parameters used in the query:

- Parameters: Load_Type, database, datasource, tablename, targetpath
- System variables: None
- Functions: None
- Variables: None

Additionally, we will update the lookup query to insert logs into the audit table for Full load as shown below.

```
@concat('insert into
audit.load_logs(data_source,tablename,numberofrowscopied,watermarkcolumn
nname,loaddate) values (''',pipeline().parameters.datasource,'',
'',pipeline().parameters.tablename,'',''',activity('Incremental_Load_
CP').output.rowscopied,'',''',pipeline().parameters.watermark,'','',
utcNow(),''')')
```

The screenshot shows the Microsoft Data Factory Pipeline expression builder interface. A pipeline diagram is visible with an 'If Condition2 - True activities' section containing a 'Full_load_cp' activity. The main area displays a dynamic query being built:

```
@concat('insert into audit.load_logs(data_source,tablename,
numberofrowscopied,watermarkcolumnname,loaddate) values ('',
pipeline().parameters.datasource,'', '',
parameters.tablename,'','.',activity('Full_load_cp').output,
rowscopied,'',pipeline().parameters.watermark,'','',
utcNow(),''')')
```

The 'Settings' tab is selected in the pipeline editor. Under 'Source dataset', it is set to 'AzureDatabricksDeltaLakeDataset1'. The 'Query' field contains the above dynamic SQL query. To the right, the 'Pipeline expression builder' window is open, showing the activity output used in the query:

- Activity outputs: Full_load_cp
- Parameters: None
- System variables: None
- Functions: None
- Variables: None

Incremental load:

For activity “Fetch logs” we have to update the query instead of item().

```
@concat('select coalesce(cast(max(loaddate) as date), ''1900-01-01'') as last_fetched_date from audit.load_logs where', ' data_source=''', pipeline().parameters.datasource, ''' and tablename=''', pipeline().parameters.tablename, '')
```

The screenshot shows the Microsoft Data Factory Pipeline expression builder interface. On the left, there's a pipeline editor with a step named 'pl_copy_from_emr'. On the right, the 'Pipeline expression builder' window is open, displaying the query: '@concat('select coalesce(cast(max(loaddate) as date), ''1900-01-01'') as last_fetched_date from audit.load_logs where', ' data_source=''', pipeline().parameters.datasource, ''' and tablename=''', pipeline().parameters.tablename, '')'. The 'Parameters' tab is selected, showing variables like Load_Type, database, datasource, tablename, and targetpath. The 'OK' button is at the bottom right.

Also we have to update the queries for incremental load as shown below

```
@concat('select *,''',pipeline().parameters.datasource,''' as datasource from ',pipeline().parameters.tablename,' where ',pipeline().parameters.watermark,' >= ''',activity('Fetch_logs').output.firstRow.last_fetched_date,'''')
```

The screenshot shows the Microsoft Data Factory Pipeline expression builder interface. On the left, there's a pipeline editor with a step named 'pl_copy_from_emr'. On the right, the 'Pipeline expression builder' window is open, displaying the query: '@concat('select *,'',pipeline().parameters.datasource,''' as datasource from ',pipeline().parameters.tablename,' where ',pipeline().parameters.watermark,' >= ''',activity('Fetch_logs').output.firstRow.last_fetched_date,'''')'. The 'Activity outputs' tab is selected, showing 'Fetch_logs' and 'Fetch_logs first row'. The 'OK' button is at the bottom right.

Additionally, we will update the lookup query to insert logs into the audit table for Incremental load as shown below.

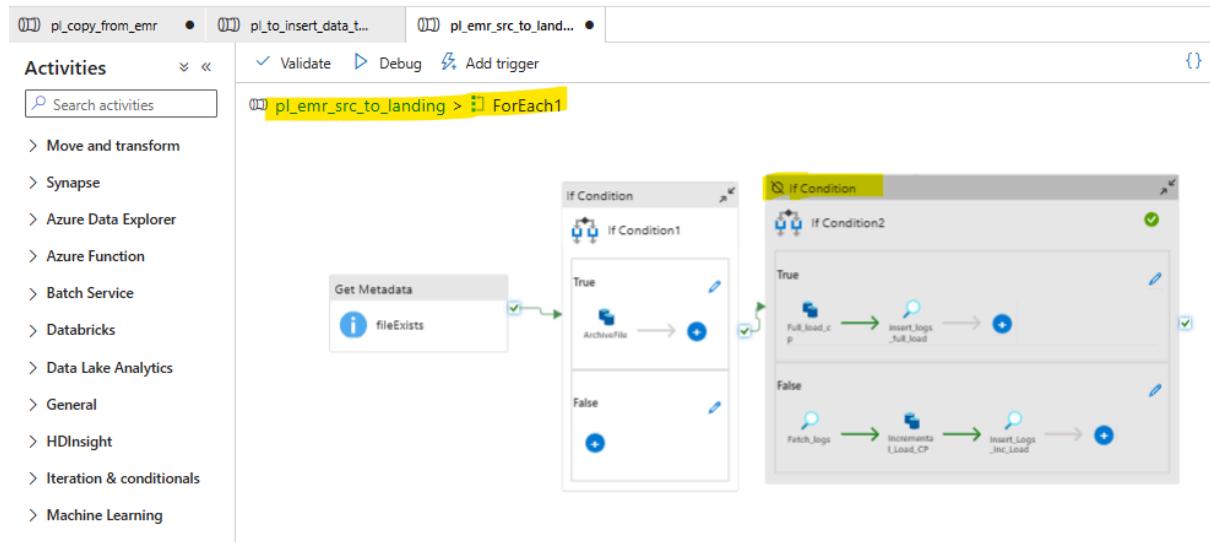
```
@concat('insert into
audit.load_logs(data_source,tablename,numberOfRowsCopied,watermarkcolumn
name,loaddate) values (''',pipeline().parameters.datasource,'',
'',pipeline().parameters.tablename,'','',activity('Incremental_Load_
CP').output.rowscopied,'','',pipeline().parameters.watermark,'', '',
utcNow(),'''')')
```

In order to make pipeline -pl_emr_src_to_landing parallel, for “For each” Activity in the pipeline “pl_emr_src_to_landing” we will remove sequential options and will add batch count as 5.

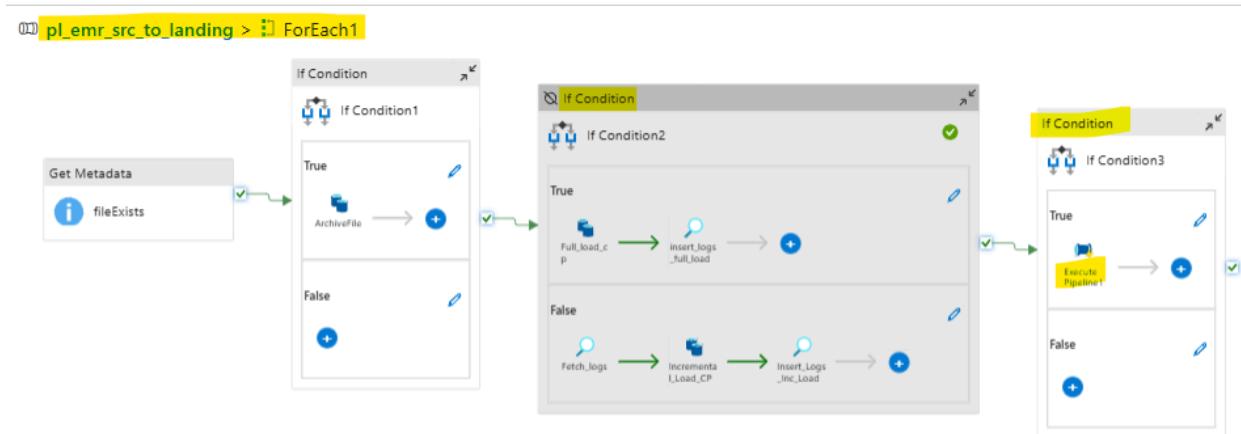
Once this pipeline is updated, publish these changes.

2. Updating the pipeline - pl_emr_src_to_landing

In this pipeline “pl_emr_src_to_landing” deactivate this activity as shown below



Now in this pipeline “pl_emr_src_to_landing” we will add one more if condition with logic @equals(item().is_active,'1') as shown below and in this we will add the execute pipeline in which we will attach the recently created pipeline “pl_copy_from_emr” as shown below



Also for this execute pipeline we will pass the parameter value as shown below.

pl_copy_from_emr

Validate Debug Add trigger

pl_emr_src_to_landing > ForEach1 > If Condition3 - True activities

Execute Pipeline

Execute Pipeline1
pl_copy_from_emr

General Settings User properties

Invoked pipeline * pl_copy_from_emr Open New

Wait on completion

Parameters

Name	Type	Value	Default value
Load_Type	String	@item().loadtype	
database	string	@item().database	
tablename	string	@item().tablename	
datasource	string	@item().datasource	
targetpath	string	@item().targetpath	
watermark	string	@item().watermark	

Steps to move NPI and ICD from API to bronze layer:

For this, we will use the provided code notebook and implement the logic in it.

2. API extracts

=> ICD Code API extract

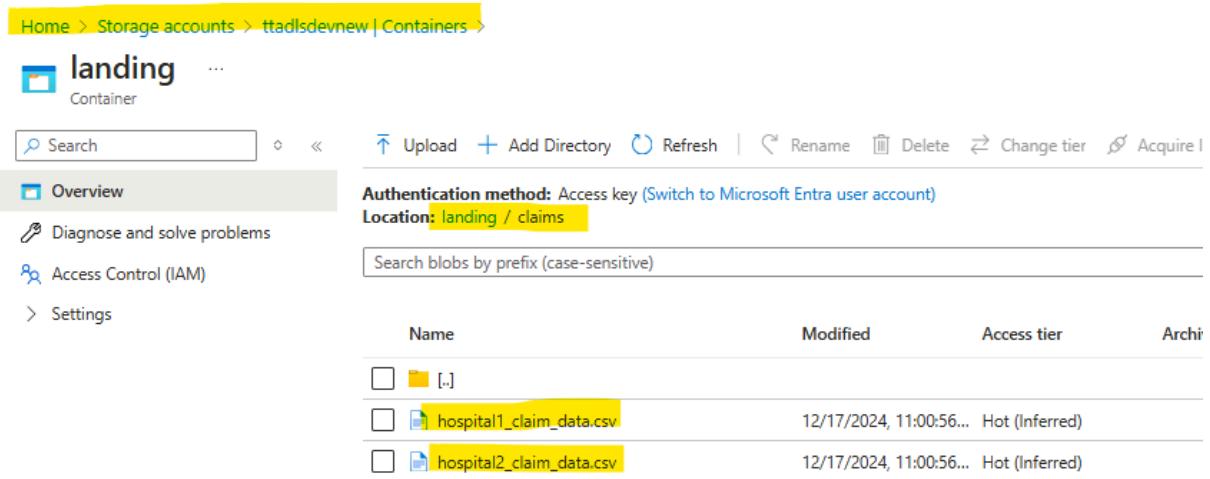
=> NPI API extract

Note: Run this code notebooks in order to fetch data before going ahead.

Steps to move Claims and CPT data from landing to bronze layer:

As part of the background activity, we will manually place the claims and CPT data files into their respective folders within the landing folder.

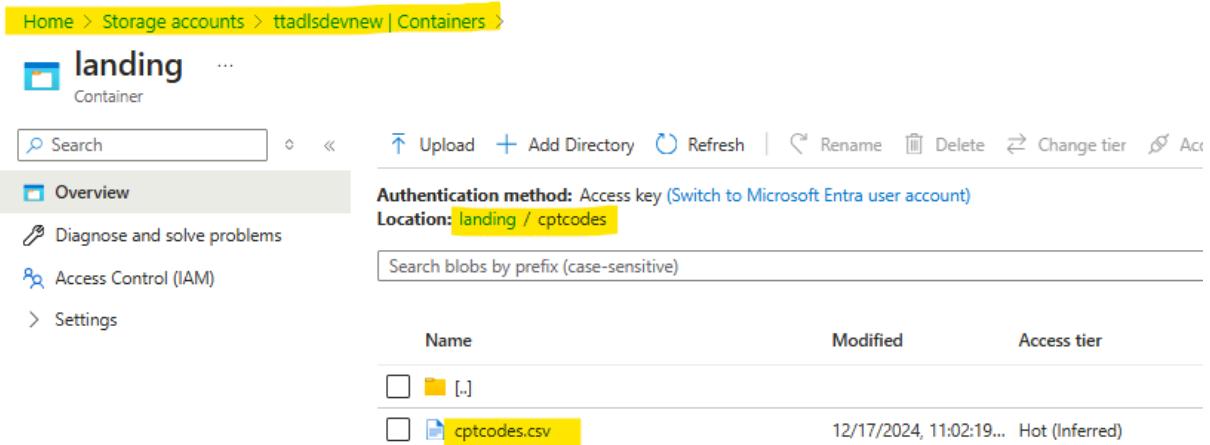
Claims data:



The screenshot shows the Azure Storage Explorer interface for a storage account named 'ttadlsdevnew'. A container named 'landing' is selected. Inside 'landing', there is a folder named 'claims'. The 'Overview' tab is active. The location is listed as 'landing / claims'. There are two CSV files in the folder: 'hospital1_claim_data.csv' and 'hospital2_claim_data.csv', both modified on 12/17/2024 at 11:00:56... and assigned to the 'Hot (Inferred)' access tier.

Name	Modified	Access tier
hospital1_claim_data.csv	12/17/2024, 11:00:56...	Hot (Inferred)
hospital2_claim_data.csv	12/17/2024, 11:00:56...	Hot (Inferred)

CPT codes data:



The screenshot shows the Azure Storage Explorer interface for a storage account named 'ttadlsdevnew'. A container named 'landing' is selected. Inside 'landing', there is a folder named 'cptcodes'. The 'Overview' tab is active. The location is listed as 'landing / cptcodes'. There is one CSV file in the folder: 'cptcodes.csv', modified on 12/17/2024 at 11:02:19... and assigned to the 'Hot (Inferred)' access tier.

Name	Modified	Access tier
cptcodes.csv	12/17/2024, 11:02:19...	Hot (Inferred)

Now using the logic present in the code notebooks we will move this data to the bronze layer in parquet format.

3. Silver :
- => Claims
- => CPT codes

Moving data from Bronze layer to Silver.

In this layer, we implement the following logic to transform and refine the data:

Data Cleaning (clean):

=> Removed invalid, null, or duplicate records to ensure data quality.

=> Standardized the data format to align with the Common Data Model for consistency and compatibility across systems.

=> Implement SCD Type 2 logic to maintain historical changes in the data, enabling tracking of changes over time.

Delta Table:

=> Stored the transformed data in Delta tables to support ACID transactions, incremental loads, and versioned data.

This transformation ensures that data from the Bronze layer is cleansed, standardized, and enriched before moving to the Silver layer for further analytics or downstream processing.

We have documented the logic for this transformation in the notebook located in the Silver folder within our Databricks workspace. We have shared the code notebook on Github to which you can refer.

Note: For both claims and CPT codes, we have used the same notebook. This notebook contains the code for moving data from the landing layer to the bronze layer, followed by the necessary transformations to clean and move the data from bronze to the silver layer.

Path: Notebooks → Silver → claims, CPT codes

Moving data from Silver to Gold layer:

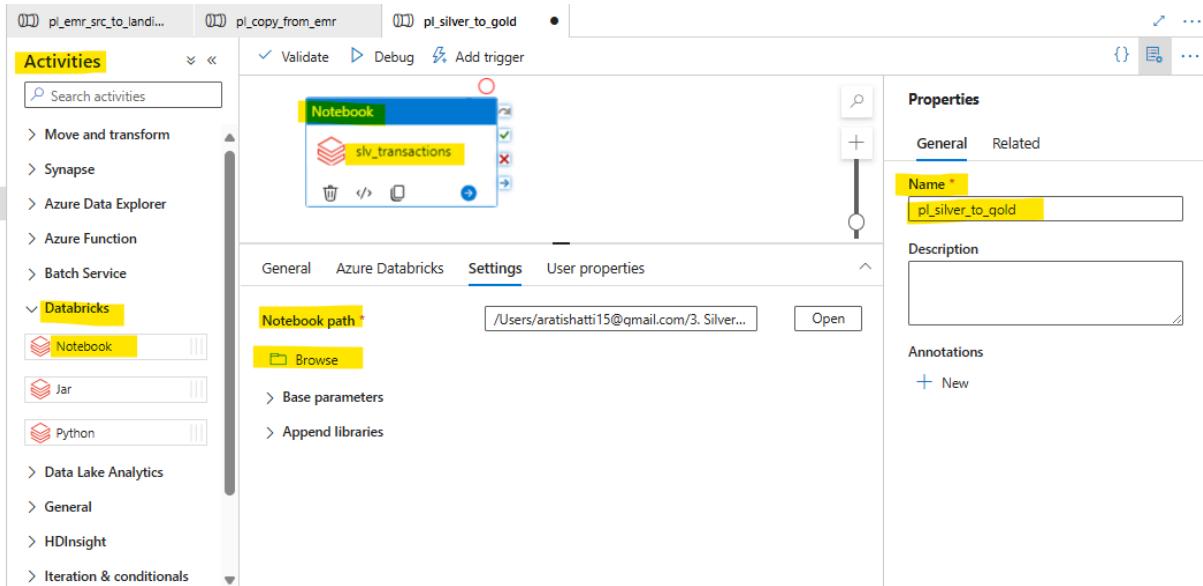
We have created notebooks to implement the required logic, and all these notebooks are located in the "Gold" folder. We have shared the code notebook on Github to which you can refer.

Pipeline to Move data from Silver to Gold layer:

We will create a pipeline “pl_silver_to_gold” to move the data from the Silver layer to the Gold layer.

First create the pipeline “pl_silver_to_gold”, and add the activity Databricks notebook as shown below.

Ex: Here, we have added a notebook activity and named it slv_transaction. Using the Browse Path option, we selected the Transactions notebook located in the Silver folder. Refer below screenshot for more information.



Similarly, we will add the remaining notebooks as demonstrated by Sumit Sir in the video. And our complete pipeline will be as shown below.

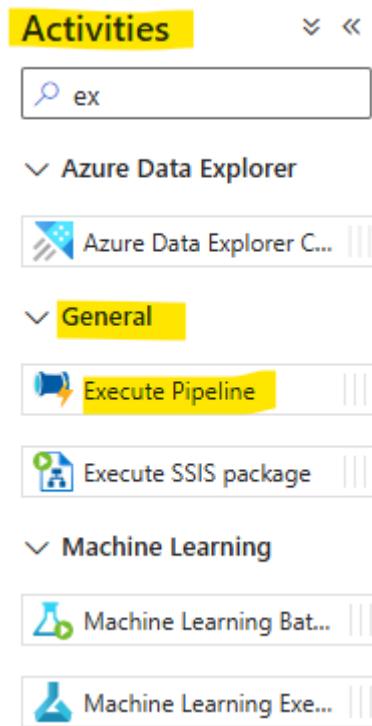


Once this pipeline is created, publish these changes.

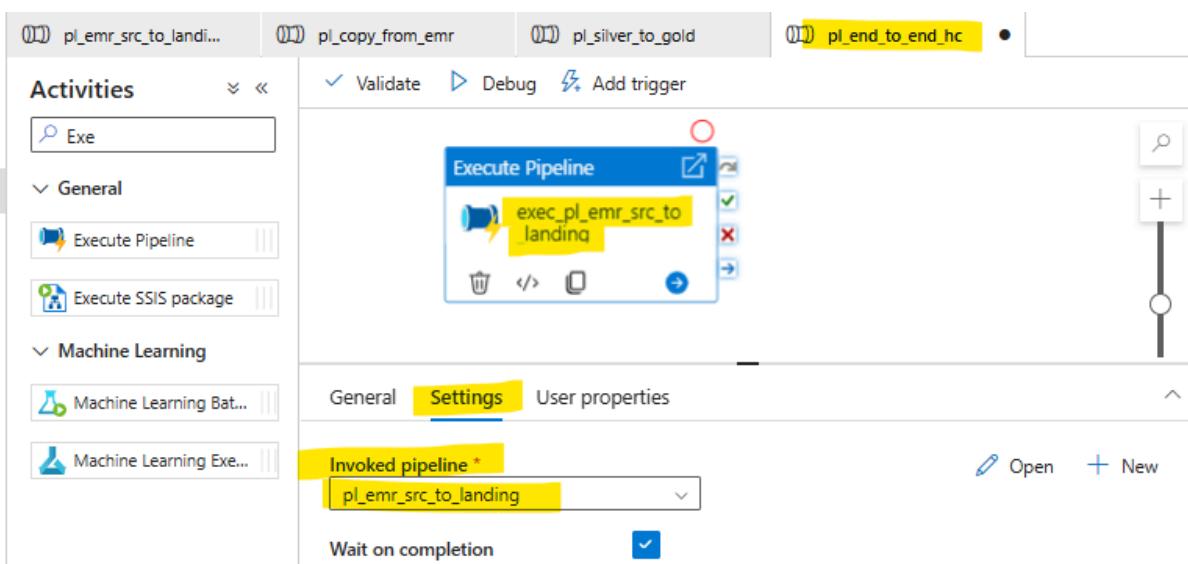
Final Pipeline:

The final pipeline (pl_end_to_end_hc) will include two execution pipelines:

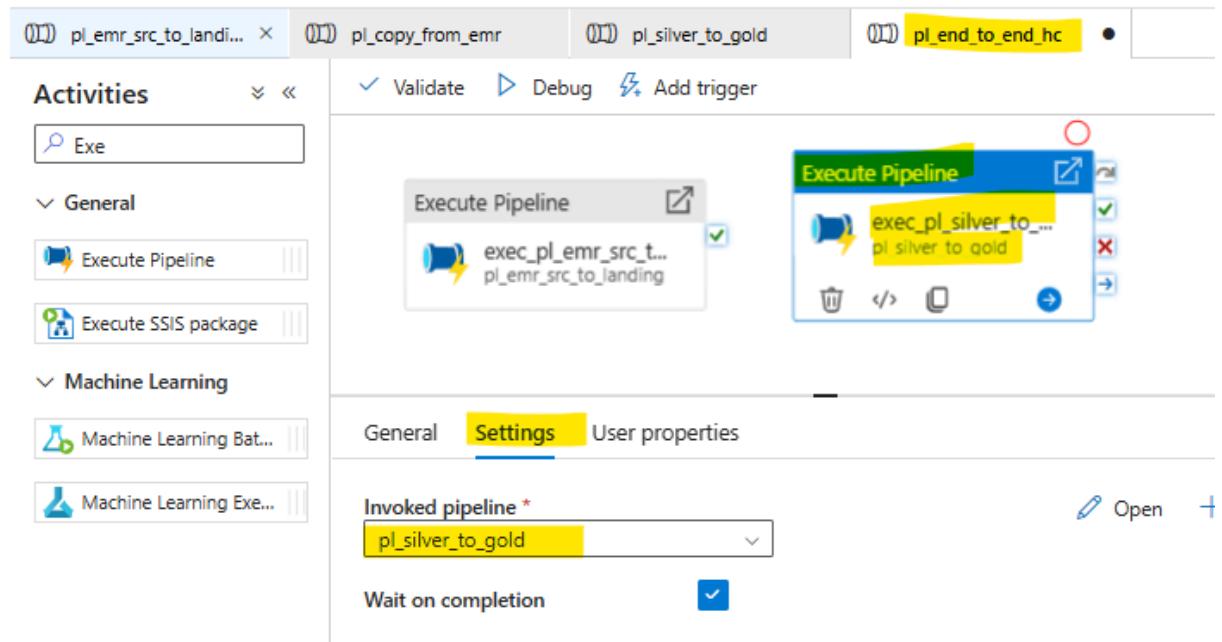
In this we will use an activity Execution pipeline.



=> The first execution pipeline (exec_pl_emr_src_to_landing) will move data from the SQL database to the landing folder.



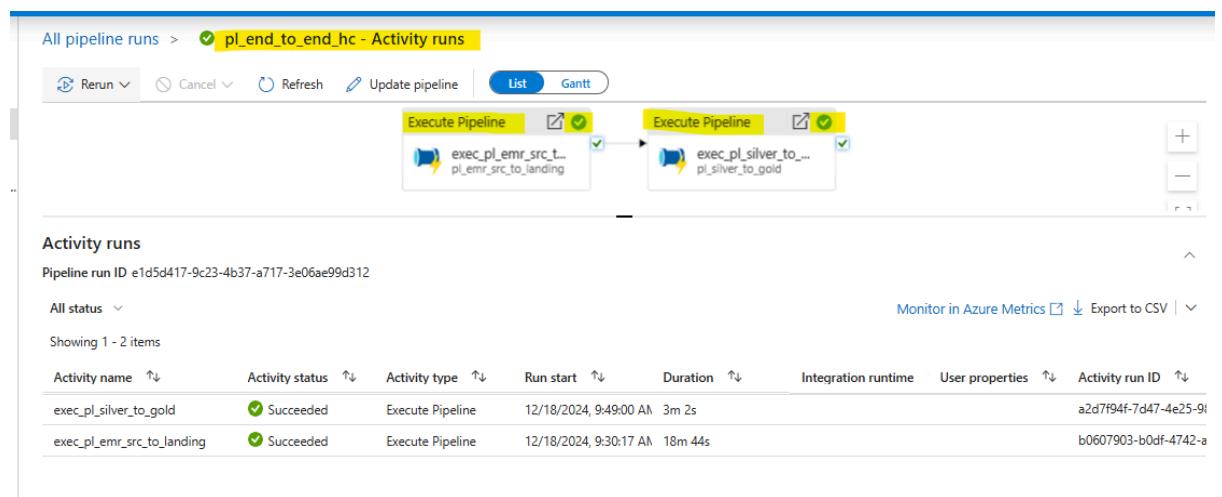
=> The second execution pipeline (exec_pl_silver_to_gold) will move the cleaned and transformed data from the Silver layer to the Gold layer.



Before running these pipelines make sure to create the silver and gold schema using below commands in catalog tt_hc_adb_ws.

```
CREATE SCHEMA IF NOT EXISTS tt_hc_adb_ws.silver;
CREATE SCHEMA IF NOT EXISTS tt_hc_adb_ws.gold;
```

Upon successful execution of the pipeline, the results will be displayed as shown below.



To link a project on Github follow below steps.

In Azure Databricks: Get the username and token for your databricks

In Github:

Get the username of your github account.

Generate personal access token:

Go to Github => Settings => Developer Settings => Personal Access token => Tokens (Classic) => select all the scopes => generate token => Copy the token

You can provide below permissions:

repo – Full access to repositories.

read:org – Access organizational-level data.

workflow – Trigger GitHub Actions workflows.

read:packages (Optional, for package access).

The screenshot shows the GitHub 'Personal access tokens' page. A sidebar on the left lists 'GitHub Apps', 'OAuth Apps', and 'Personal access tokens' (which is highlighted with a yellow box). Below these are 'Fine-grained tokens' and 'Tokens (classic)'. A 'Preview' button is visible next to the 'Tokens (classic)' section. The main area is titled 'New personal access token (classic)'. It contains a note about personal access tokens functioning like OAuth tokens and can be used over HTTPS or via Basic Authentication. A 'Note' field contains the value 'github', which is also highlighted with a yellow box. A question 'What's this token for?' is present. An 'Expiration *' field is set to '30 days', with a note below stating 'The token will expire on Fri, Jan 17 2025'. A 'Select scopes' section follows, with a note that scopes define access for personal tokens. A link to 'Read more about OAuth scopes' is provided. A table lists several scopes with checkboxes, where 'repo' and its sub-scopes ('repo:status', 'repo_deployment', 'public_repo', 'repo:invite', 'security_events') are selected. Descriptions for each scope are provided.

Scope	Description
<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events

Create the token and note it once created.

Now go to databricks => setting => Linked accounts => Git provider (Github) then select personal access token => then provide the username and the token for github

The screenshot shows the Databricks 'Settings' page. On the left, a sidebar lists 'Workspace admin', 'Appearance', 'Identity and access', 'Security', 'Compute', 'Development', 'Notifications', 'Advanced', 'User' (with 'Profile', 'Preferences', 'Developer', and 'Linked accounts' listed), and 'Notifications'. The 'Linked accounts' item is highlighted with a yellow box. The main area is titled 'Linked accounts' with a sub-section 'Git integration'. It includes sections for 'With co-versioned repo' (describing Databricks Git folders and Repos), 'With individual notebooks' (describing individual notebook version control integration with GitHub, Bitbucket Cloud, or Azure DevOps Services using AAD authentication only), and 'Set your Git provider and credentials' (with a note about API support). A dropdown menu for 'Git provider' is set to 'GitHub', which is also highlighted with a yellow box. Below this, there are two options: 'Link Git account' (using GitHub single sign-on) and 'Personal access token' (selected, indicated by a blue circle), which is also highlighted with a yellow box. A 'Git provider username or email' field contains a redacted value. A 'Token' field is present with a note about generating a GitHub personal access token and its requirements. A 'Save' button is at the bottom.

Now create the repository for healthcare project on github account:

To create a repository on GitHub:

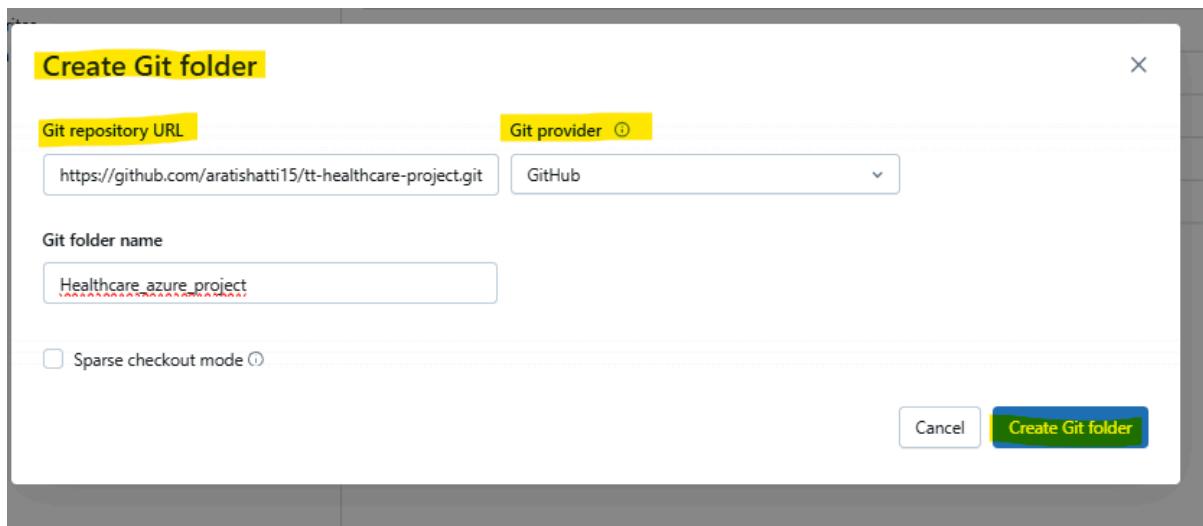
1. Go to [GitHub](#).
2. Click on **New** under "Repositories" in your GitHub profile.
3. Provide:
 - o Repository Name: e.g., tt-healthcare-project.
 - o Description (optional).
 - o Choose **Public** or **Private** visibility.
4. Click **Create repository**.

The screenshot shows a GitHub repository page for 'tt-healthcare-project'. The repository is public. On the left, there's a 'Set up GitHub Copilot' section with a 'Get started with GitHub Copilot' button. On the right, there's a 'Add collaborators to this repository' section with a 'Invite collaborators' button. Below these sections, there's a 'Quick setup — if you've done this kind of thing before' section with options for 'Set up in Desktop', 'HTTPS', and 'SSH', and a link to the repository's URL: <https://github.com/aratishatti15/tt-healthcare-project.git>. At the bottom, there's a note: 'Get started by creating a new file or uploading an existing file. We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#)'.

Cloning GitHub Repository into Databricks

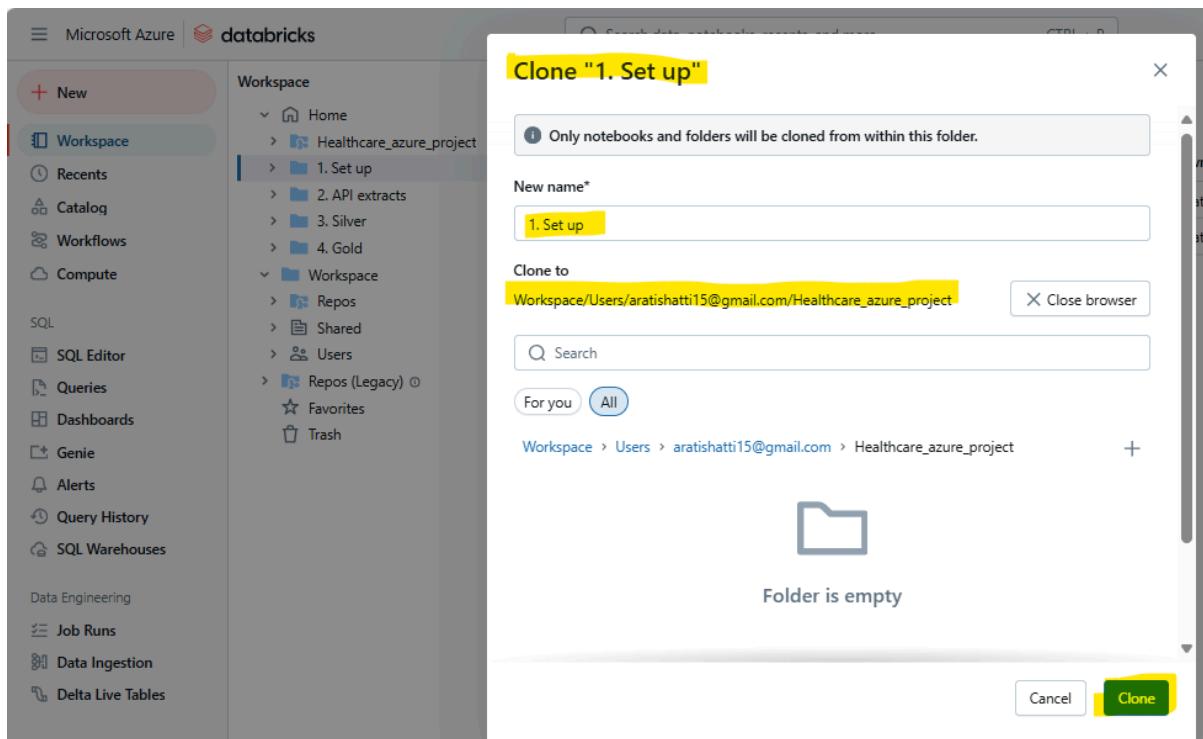
Go to databricks => Workspace => Repos => select option create git folder

The screenshot shows the Databricks workspace interface. The left sidebar has a 'Workspace' section with links for Home, Workspace, and Repos (Legacy). The main area is titled 'Repos' and shows a message: 'You can now create Git folders (previously Repos) outside the Repos folder for simplified UI and organization. Go to home folder and create Git folder.' It also says 'Alternatively, you can create a Repo here. Your user folder under /Repos will be visible afterward.' There is a 'Create Git Folder' button. A table below lists repositories, showing one entry: 'This folder is empty.'



Now you can clone existing folders and files into this git folder.

Right click on file/folder => clone => then using the browse option select this git folder.



Similarly clone all the required files and folders.

To use Git in Databricks, click on the three dots next to the Git-linked folder and select the Git option. This allows you to pull and push changes, as well as create branches, directly from the Databricks interface.

Healthcare_azure_project

main

Name	Type	Owner
1. Set up	Folder	Arati Hatti
2. API extracts	Folder	Arati Hatti
3. Silver	Folder	Arati Hatti
4. Gold	Folder	Arati Hatti

Git...

Import

Download as

Copy URL/path

Rename

Move

Add to favorites

Move to Trash

Share

Create