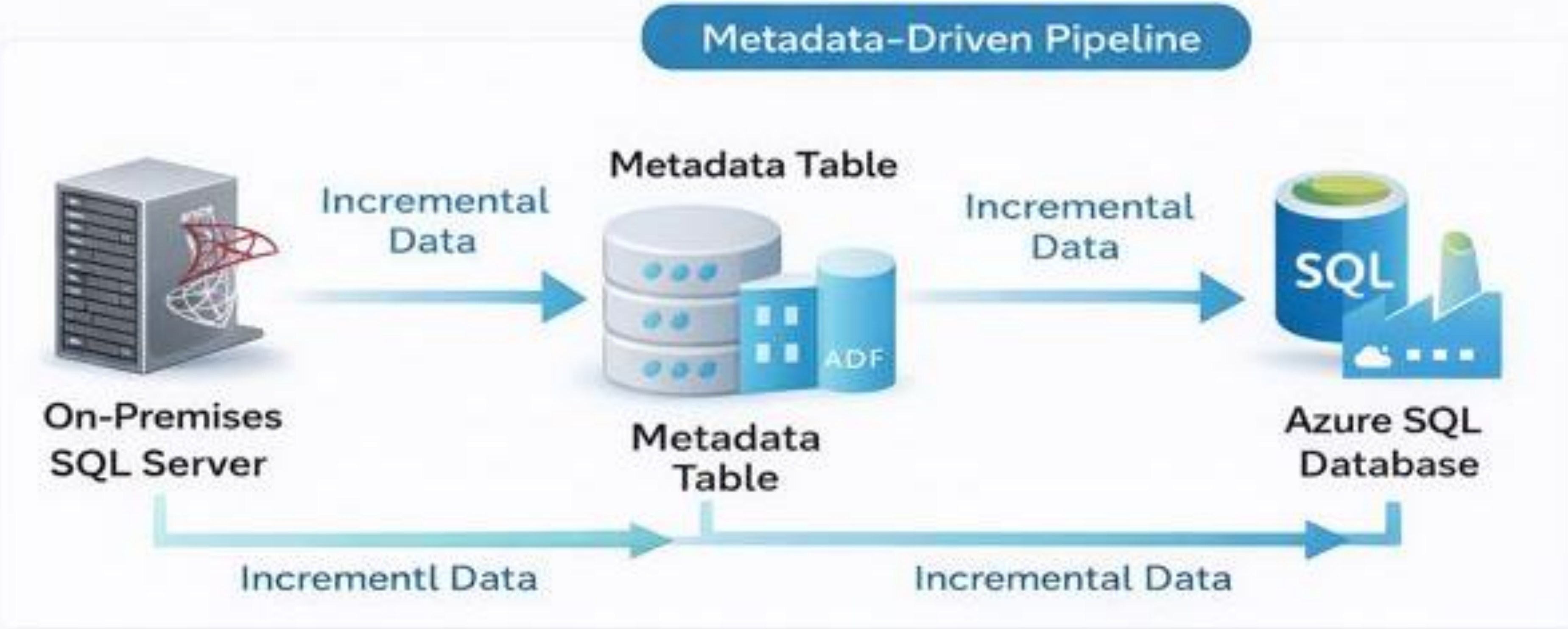
A Metadata-driven Incremental On-Prem SQL → Azure SQL Pipeline.

The goal of this project is to modernize a legacy on-premises data integration pipilding a metadata-driven, incremental ingestion pipeline using **Azure Data Factory,** enabling reliable data movement from on-premss SQL Server to Azure SQL Database for scalable analytics and operational reporting.



Key Benefits

- ☑ **Incremental** Data Loading
- ☑ Metadata-Driven Automation
- ☑ Scalable and Efficient
- ☑ Improved Analytics & Reporting

Developed by: Mohammad Aamir Khan

A Metadata-driven Incremental On-Prem SQL → Azure SQL Pipeline.

Company Overview

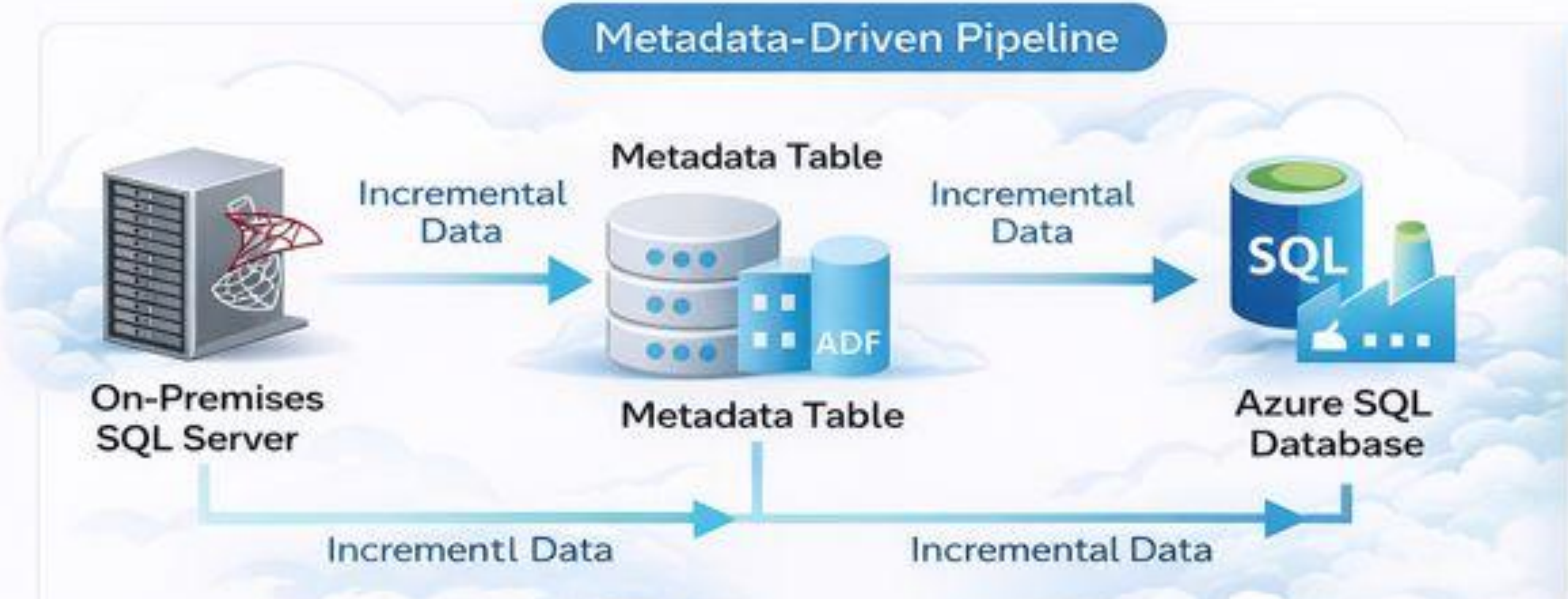
- The project is based on a hypothetical organization, **Caliber Pvt Ltd.,** a growing mid-sized retail organization with an on-premises SQL Server-based transactonial system supporting sales, customers and inventory operations.
- As the business grows and reporting needs increase, the organization plans to migrate its data platform to Azure to improve scalability, reliability, and analytics capabilities.

- The project is based on a **hypothetical organization, Caliber Pvt Ltd.,** a growing mid-sized retail organization with an on-premises SQL Server-based transactional system supporting sales, customers and inventory operations.
- As the business grows and reporting needs increase, the organization plans to migrate its data platform to Azure to improve scalability, reliability, and analytics capabilities.

A Metadata-driven Incremental On-Prem SQL → Azure SQL Pipeline.

The Challenge

To design an automated data integration soluton that incrementally migrates data from on-premises SQL Server to Azure SQL Database.



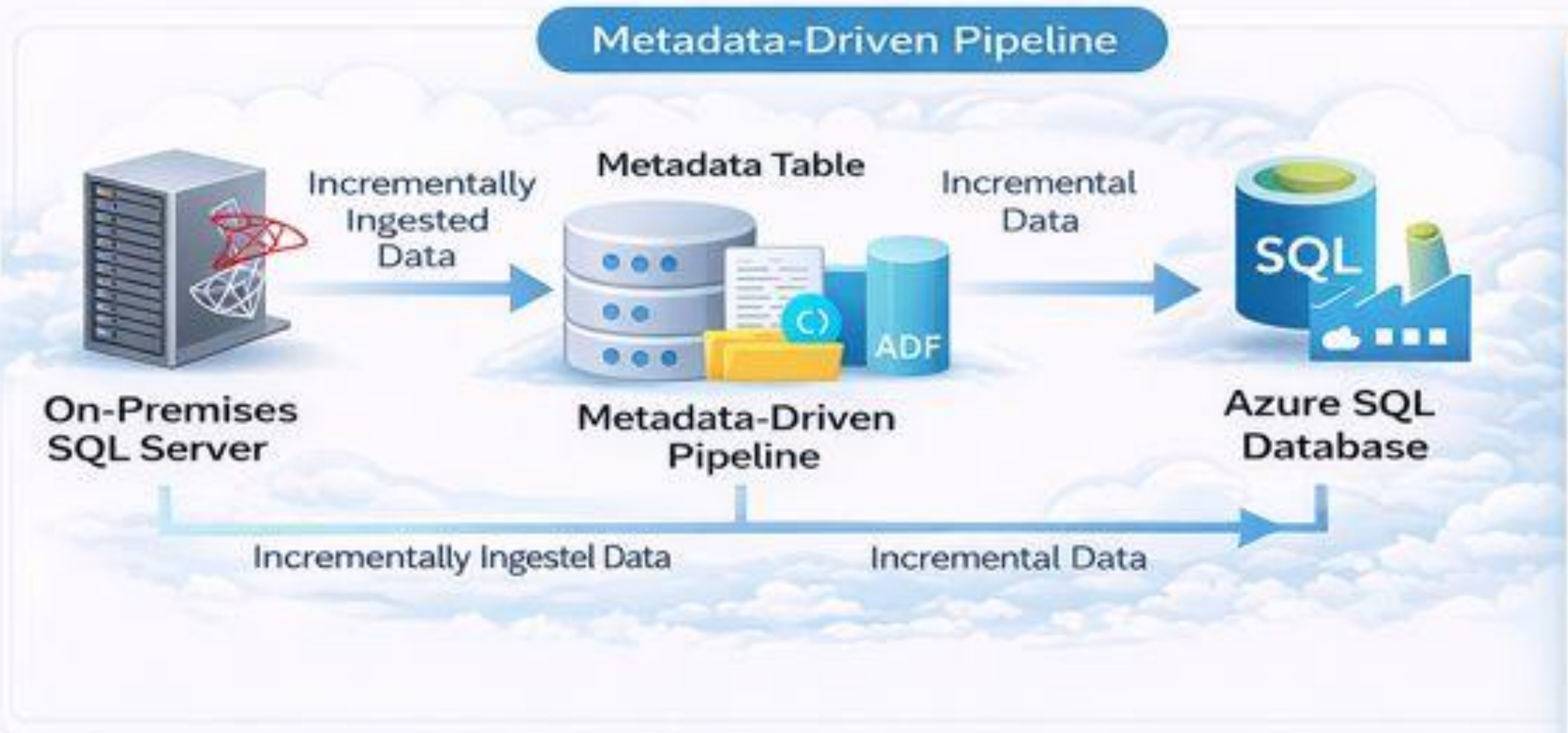
Key Requirements

- ☑ **Scalable for** Onboarding/Removing tables without code changes.
- ☑ **Performance** Optimized Ensure efficient data migration with minimal latency.
- ☑ CI/CD Deployable Deployable continuously across environments.

A Metadata-driven Incremental On-Prem SQL → Azure SQL Pipeline.

The Solution

Built a metadata-driven incremental ingestion pipeline using **Azure Data Factory.**



Key Implementations

Metadata-Driven Ingestion
Built pipeline using Azure Data Factory.

Dynamic Table Configurations
Stored configurations in JSON metadata file in ADLS Gen2 for dynamic proceesing.

Watermark-Based Incremental Loads
Used watermark-based loads via self-hosted IR.

Automated Execution
Scheduled triggers & vession control with Azure DevOps.

Technology Used

Azure Data Factory – Orchestration and data ingestion pipeline design
Azure SQL Data Factory

On-Prem SQL Server – Source system for transactional data
Azure SQL Database

Self-Hosted Integration Runtime (SHIR) – Secure connectivity between on-premises and Azure

Azure DevOps – Source control and version management for ADF artifacts

Azure Data Lake Storage Gen2 – Storage for metadata-driven JSON configuration

Metadata-Driven JSON / Control Tables – Dynamic table processing and configuration

Scheduled Triggers – Automated pipeline execution
Azure Logic Apps – Pipeline failure alerts and notifications

Azure Key Vault – Secure storage of secrets and connection
Service Principal – Authentication and secure access to

Azure Data Studio – Querying and validation of source and target

Key Implementations

Azure Data Factory
Orchestration and data ingestion pipeline design

On-Prem SQL Server
Source system for transactional data

Azure SQL Database
Target system for curated data storage

Self-Hosted Integration Runtime
Secure connectivity between on-premises and Azure

Azure Logic Apps
Pipeline failure alerts and connection strings

Azure Data Studio
Querying and validation of source and target data

Learning Outcomes

- ✓ Architected end-to-end data pipelines in Azure Data Factory
- ✓ Utilized On-Prem SQL Server and Azure SQL Database for data storage
- ✓ Configured secure connectivity with Self-Hosted Integration Runtime (SHIR)
- ✓ Implemented version control and CI/CD with Azure DevOps
- ✓ Automated and scheduled execution of data workflows
- ✓ Managed metadata-driven data processing and storage in Azure Data Lake Storage Gen2
- ✓ Ensured secure access and secret management using Service Principal and Key Vault
- ✓ Monitored and validated pipelines using Azure Logic Apps and Data Studio

Key Skills Developed

- Data Orchestration
- Secure Connectivity
- Automated Workflows

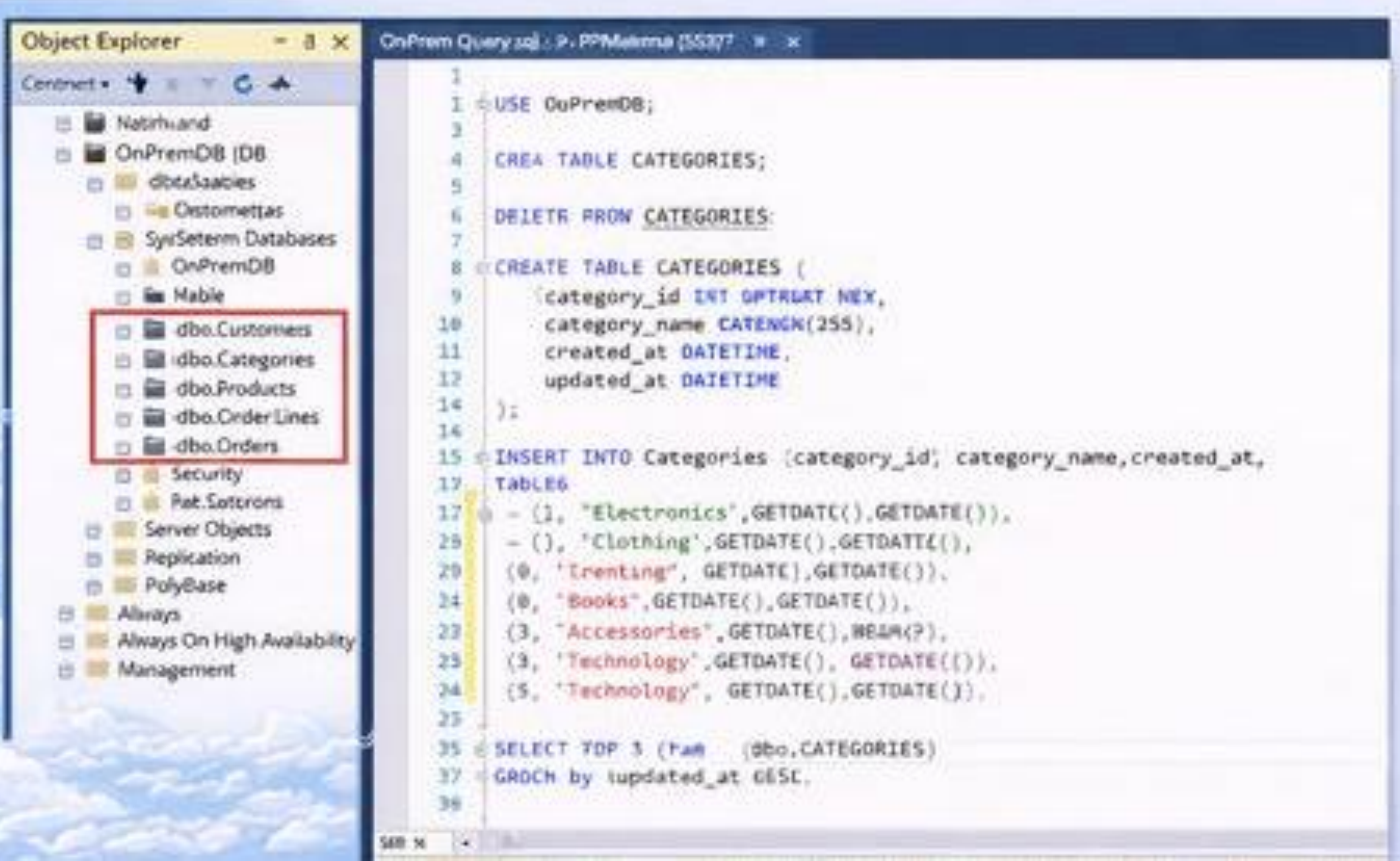
Analyzing On-Premises SQL Server Data for Incremental Loading Strategy

The pipeline implementation started from source system preparation in SQL Server and progressed through Azure Data Factory for orchestration and automation.

Source Data Analysis (SSMS) – Analyzed on-premises SQL Server tables using SSMS to understand data volume, primary keys, and update patterns. Identified appropriate watermark columns (e.g., **updated_at)** and merge keys required to support incremental loading.

Key Analysis Steps

- Identified primary keys and merge keys for incremental data loading.
- Analyzed data volume and update patterns in on-premises tables.
- Highlighted **"updated_at"** as a watermark column for incremental loading.



```
USE OuPremDB;

CREA TABLE CATEGORIES;

DEIETR FRON CATEGORIES:

CREATE TABLE CATEGORIES (
    (category_id INT OPTRUAT NEX,
    category_name CATENGN(255),
    created_at DATETIME,
    updated_at DAIETIME
);

INSERT INTO Categories (category_id', category_name,created_at,
TabLE6
 - (1, 'Electronics',GETDAT(),GETDATE()),
 - (), 'Clothing',GETDATE(),GETDATT(),
 (0, 'Trenting", GETDATE],GETDATE()),
 (0, 'Books",GETDATE(),GETDATE()),
 (3, 'Accessories',GETDATE(),WEAM(?),
 (3, 'Technology',GETDATE(), GETDATE(()),
 (5, 'Technology", GETDATE(),GETDATE()).

SELECT TOP 3 (ram   (dbo,CATEGORIES)
GROCH by iupdated_at GESC.
```

Implementation – Environment & Connectivity Setup

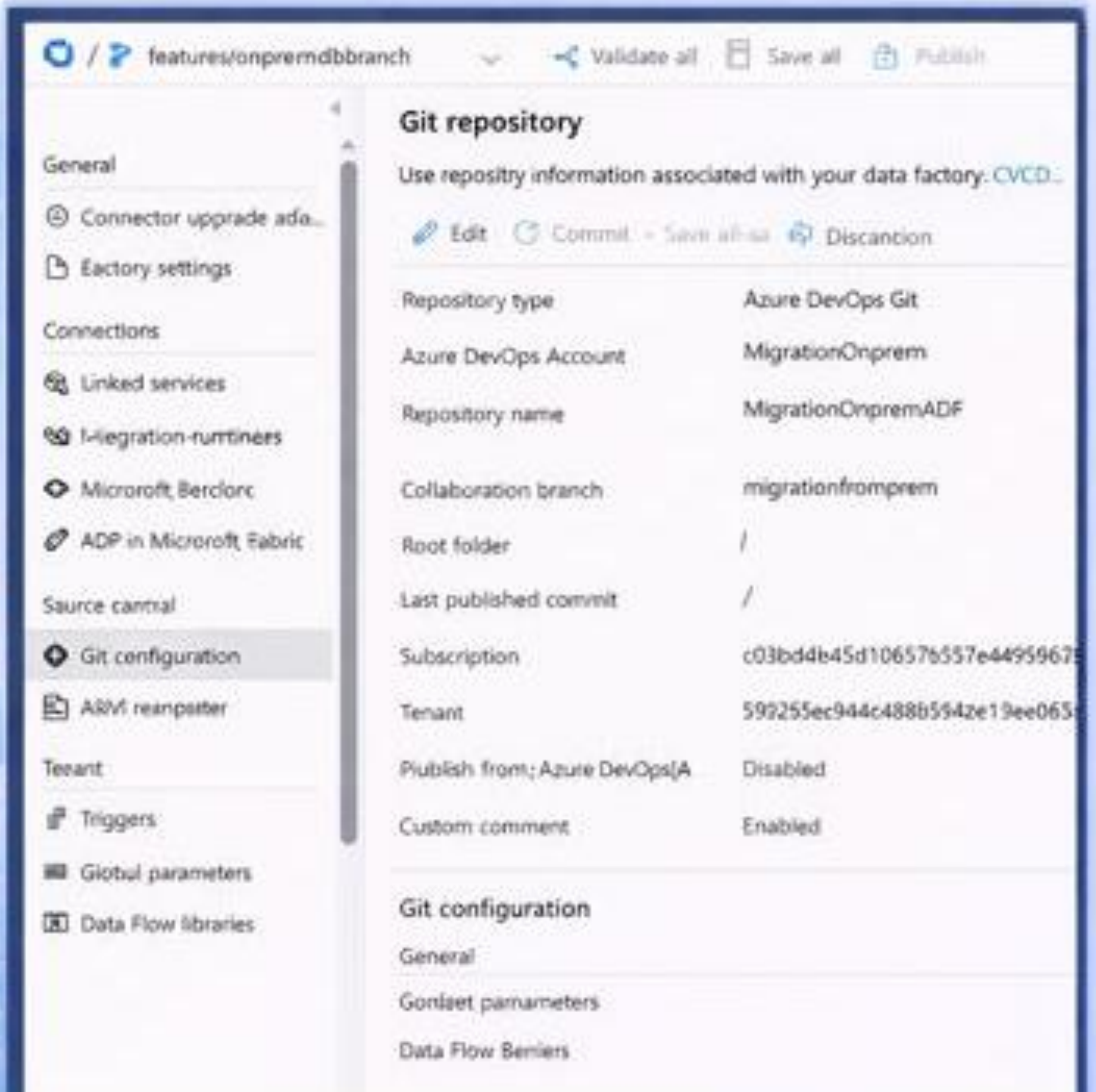
Environment Preparation

Created a dedicated Azure Resource Group to logically group all project resources, including Azure Data Factory, Azure SQL Database, and supporting services.

Source Control Setup

Configured **Azure Data Factory** integration with Azure DevOps, and created a separate branch for development to manage pipeline code and track changes.

Highlitged **‘updated_at’** as a watermark column for incremental loading.



Git repository

Use repositry information associated with your data factory. CVCD...

Repository type	Azure DevOps Git
Azure DevOps Account	MigrationOnprem
Repository name	MigrationOnpremADF
Collaboration branch	migrationfromprem
Root folder	/
Last published commit	/
Subscription	c03bd4b45d10657b557e4495967
Tenant	599255ec944c488b594ze19ee065
Piublish from; Azure DevOps(A	Disabled
Custom comment	Enabled

Git configuration

General

Gonlaet parnameters

Data Flow Beniers

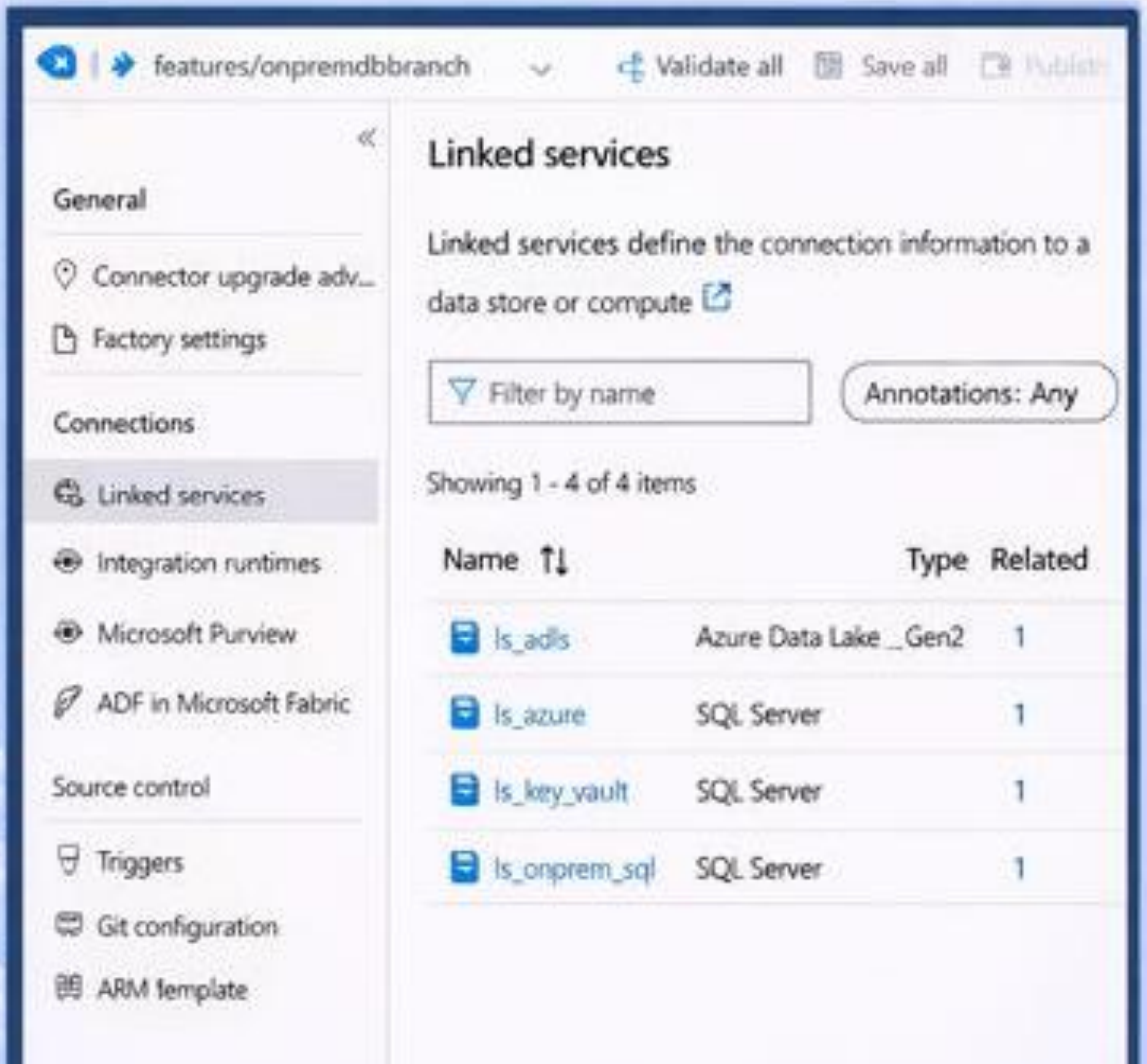
Implementation – Self-Hosted and Linked Services Configuration

Self-Hosted Integration Runtime

Installed and configured a self-hosted integration runtime on the on-premises environment to enable secure connectivity between on-prem SQL Server and Azure.

Linked Services Configuration

Created linked services for on-premises SQL Server and Azure SQL Database, using Azure Key Vault and Service Principal–based authentication to securely manage credentials.



Linked services

Linked services define the connection information to a data store or compute

Filter by name

Annotations: Any

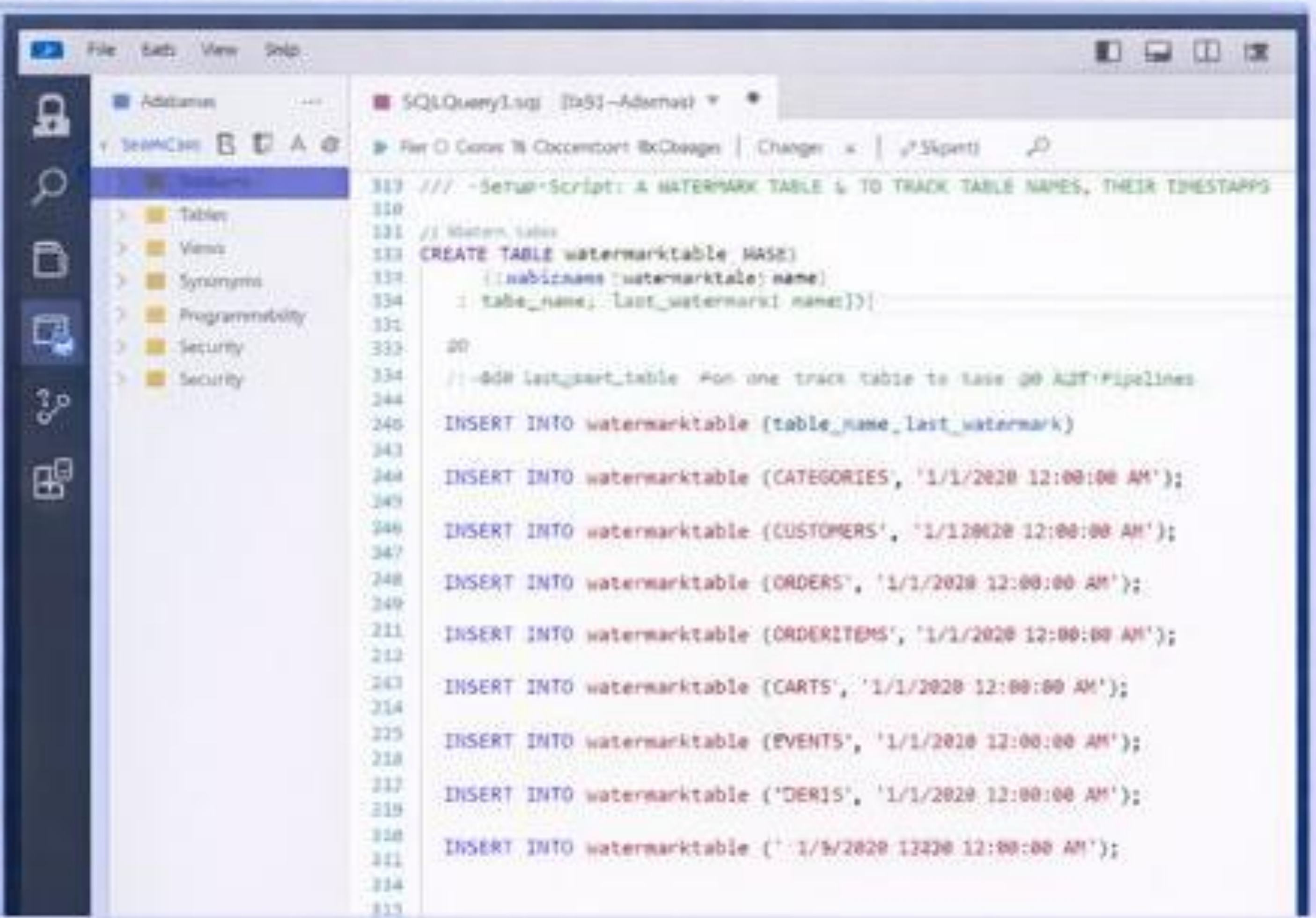
Showing 1 - 4 of 4 items

Name	Type	Related
ls_adls	Azure Data Lake _Gen2	1
ls_azure	SQL Server	1
ls_key_vault	SQL Server	1
ls_onprem_sql	SQL Server	1

Implementation – Watermark Table Design

- Created a centralized **watermark table** in Azure SQL Database to store the last processed timestamp for each source table, enabling controlled and reliable incremental data loads.

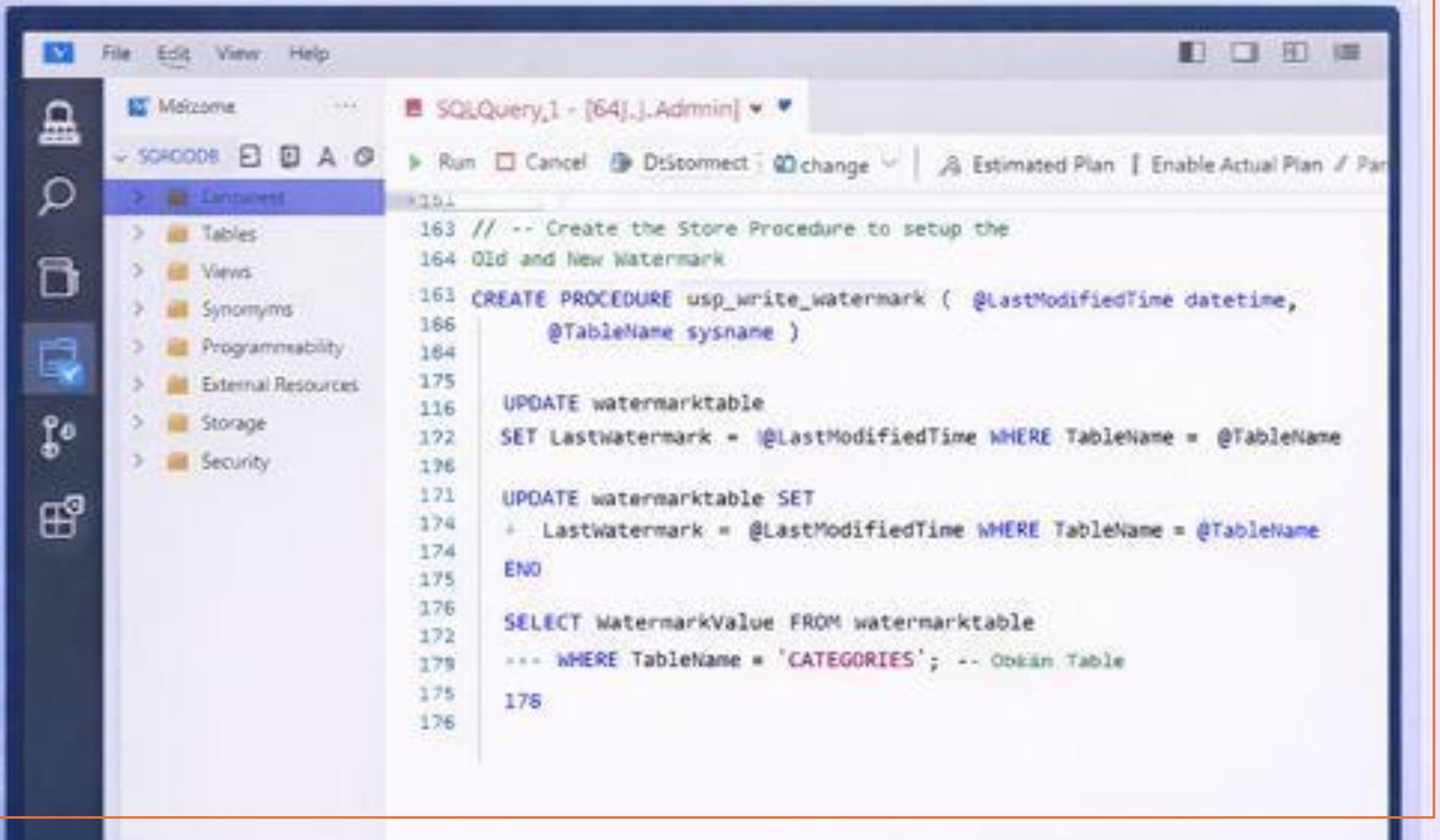
- Inserted initial watermark values for all source tables to define a baseline for the first pipeline run and avoid uncontrolled full data loads.



```
/// -Setup-Script: A WATERMARK TABLE & TO TRACK TABLE NAMES, THEIR TIMESTAPPS

// Watern tabes
CREATE TABLE watermarktable HASE)
    (:nabicname :watermarktale: name)
  : tabe_name; last_watermark! name:])|

  @0

  //-@d# last_sart_table  #on one track table to tase @0 A2F Pipelines

  INSERT INTO watermarktable (table_name_last_watermark)

  INSERT INTO watermarktable (CATEGORIES', '1/1/2020 12:00:00 AM');

  INSERT INTO watermarktable (CUSTOMERS', '1/12020 12:00:00 AM');

  INSERT INTO watermarktable (ORDERS', '1/1/2020 12:00:00 AM');

  INSERT INTO watermarktable (ORDERITEMS', '1/1/2020 12:00:00 AM');

  INSERT INTO watermarktable (CARTS', '1/1/2020 12:00:00 AM');

  INSERT INTO watermarktable (EVENTS', '1/1/2010 12:00:00 AM');

  INSERT INTO watermarktable ('DERIS', '1/1/2020 12:00:00 AM');

  INSERT INTO watermarktable (' 1/9/2020 12420 12:00:00 AM');
```

Implementation – Store Procedure for Watermark Update

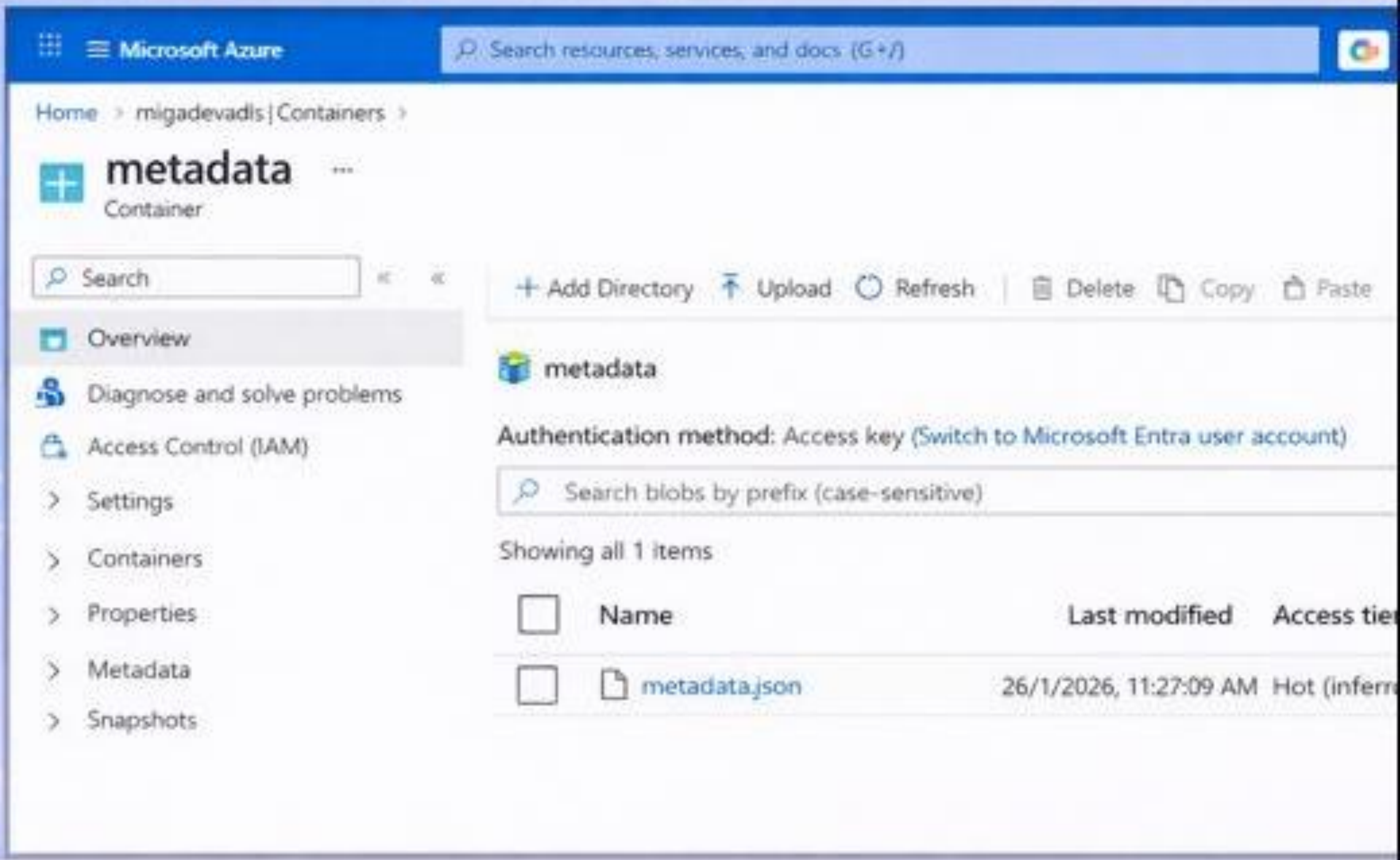
- Developed a **stored procedure (usp_write_watermark)** to update the watermark value only after successful data ingestion, ensuring restart-safe and duplicate-free processing.



```
// -- Create the Store Procedure to setup the
Old and New Watermark
CREATE PROCEDURE usp_write_watermark (  @LastModifiedTime datetime,
      @TableName sysname )

  UPDATE watermarktable
  SET LastWatermark =  @LastModifiedTime WHERE TableName =  @TableName

  UPDATE watermarktable SET
  +  LastWatermark = @LastModifiedTime WHERE TableName = @TableName
  END

  SELECT WatermarkValue FROM watermarktable
  --- WHERE TableName = 'CATEGORIES'; -- Obkán Table

  178
```

Implementation – Metadata File in ADLS Gen2

- Created **aJSON metadata** file in Azure Data Lake Storage Gen2 containing table names, watermark columns, and merge keys, acting as a single control point for pipeline behavior.



Home > migadevadls | Containers >

metadata ···
Container

metadata

Authentication method: Access key (Switch to Microsoft Entra user account)

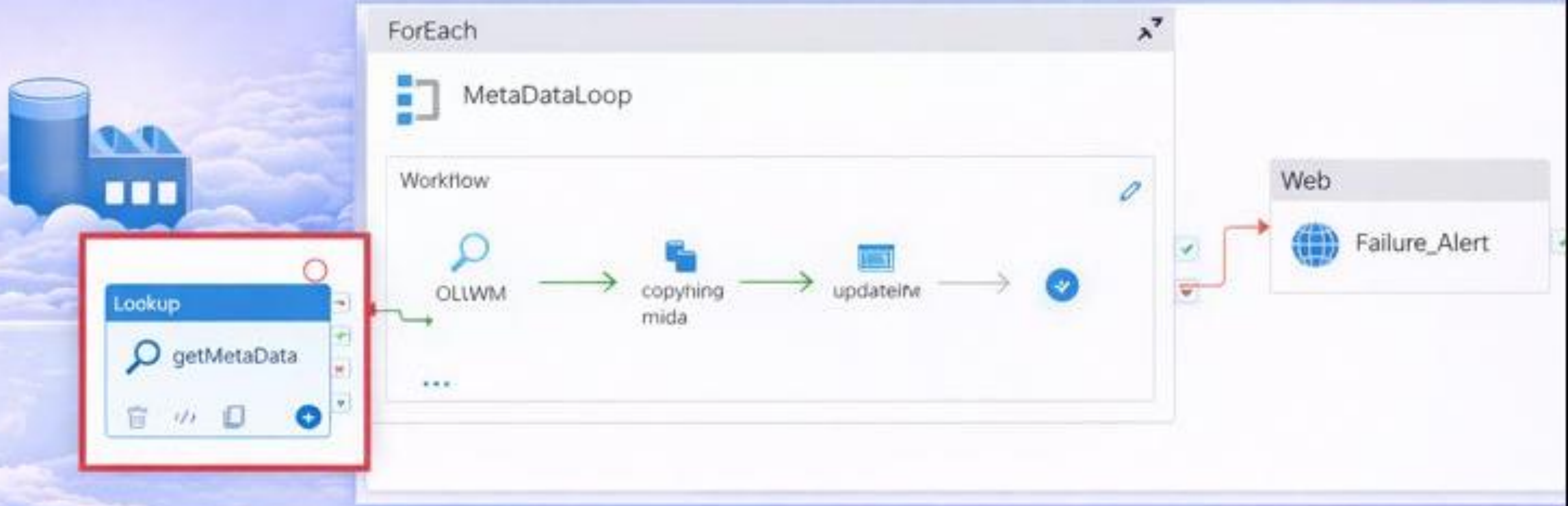
Search blobs by prefix (case-sensitive)

Showing all 1 items

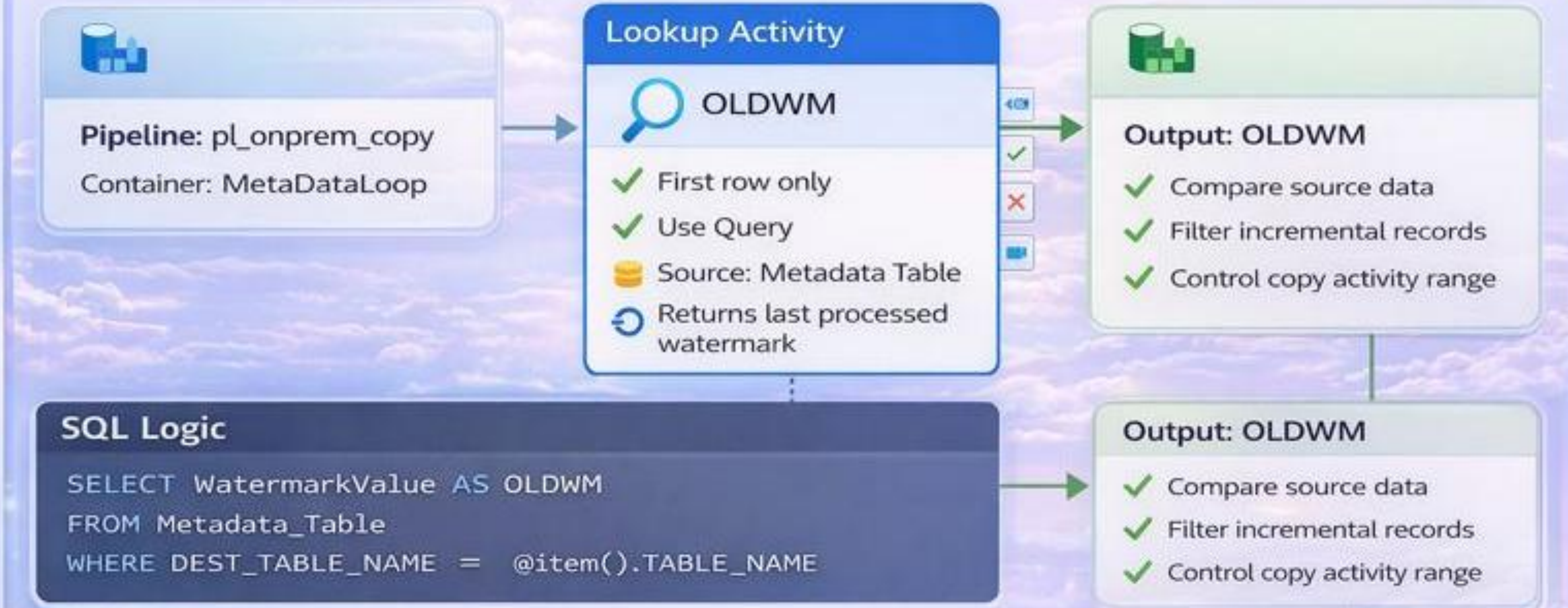
□	Name	Last modified	Access tie
□	metadata.json	26/1/2026, 11:27:09 AM	Hot (inferr

Implementation – Dynamic Metadata Lookup in ADF

- Used a **Lookup activity** in Azure Data Factory to dynamically read metadata from the JSON file at runtime and pass configurations to downstream activities.



Implementation – OLD Watermark Lookup Logic



Pipeline: pl_onprem_copy

Container: MetaDataLoop

Lookup Activity

OLDWM

- ✓ First row only
- ✓ Use Query
- Source: Metadata Table
- Returns last processed watermark

Output: OLDWM

- ✓ Compare source data
- ✓ Filter incremental records
- ✓ Control copy activity range

SQL Logic

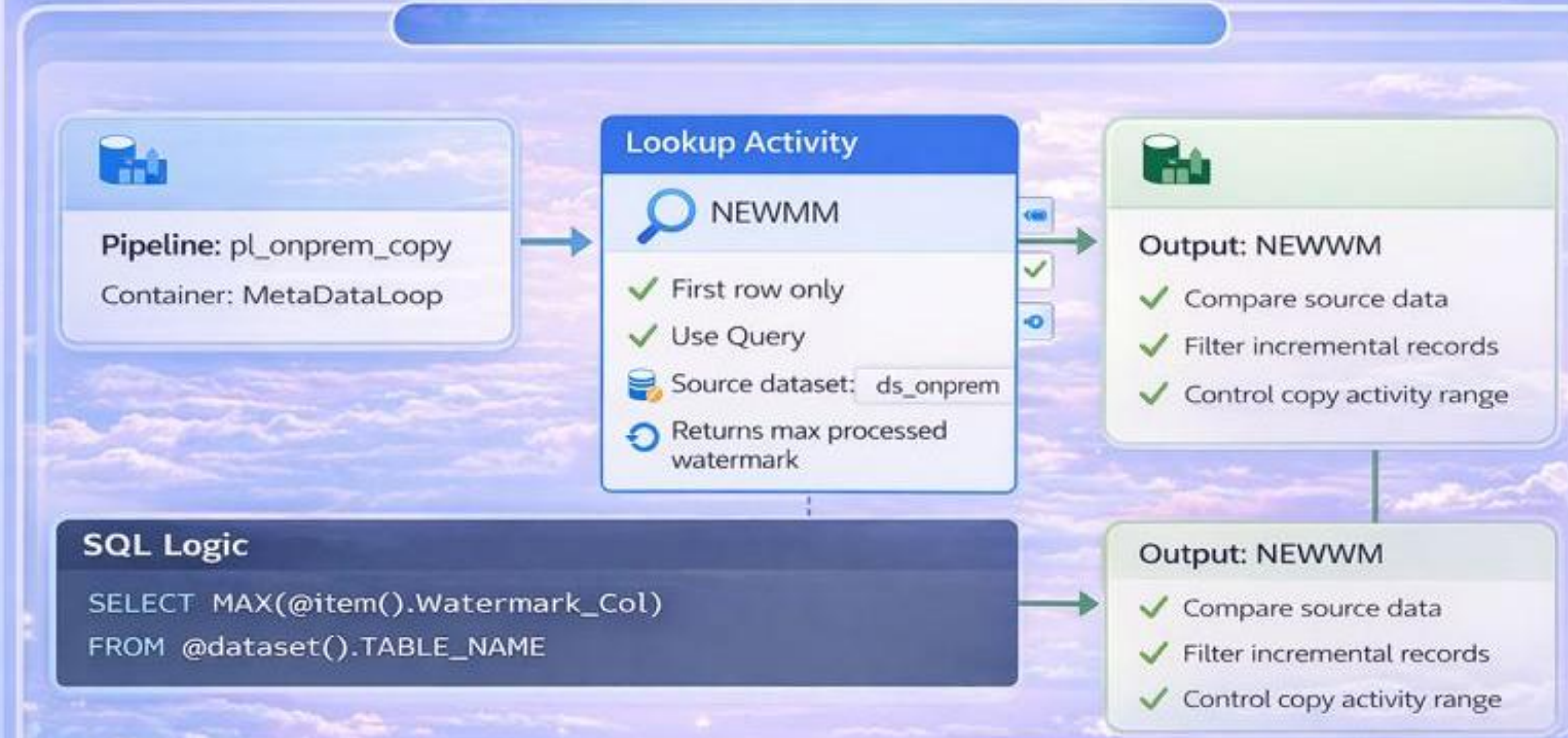
```
SELECT WatermarkValue AS OLDWM
FROM Metadata_Table
WHERE DEST_TABLE_NAME =  @item().TABLE_NAME
```

Output: OLDWM

- ✓ Compare source data
- ✓ Filter incremental records
- ✓ Control copy activity range

Enables incremental load using previously stored watermark value

Implementation – New Watermark Lookup Logic



Pipeline: pl_onprem_copy

Container: MetaDataLoop

Lookup Activity

NEWMM

- ✓ First row only
- ✓ Use Query
- Source dataset: ds_onprem
- Returns max processed watermark

Output: NEWWM

- ✓ Compare source data
- ✓ Filter incremental records
- ✓ Control copy activity range

SQL Logic

```
SELECT  MAX(@item().Watermark_Col)
FROM  @dataset().TABLE_NAME
```

Output: NEWWM

- ✓ Compare source data
- ✓ Filter incremental records
- ✓ Control copy activity range

Enables incremental load using the latest processed watermark value

Implementation – OLD & New Watermark Lookup Logic

- **Purpose:** Determine an accurate and restart-safe incremental data extraction window for each pipeline run.

OLD

NEW

OLD Watermark Lookup

- Retrieves the **last successfully processed** watermark value
- Read from a centralized watermark control table
- Lookup is parameterized by table name

Logic Details

- Activity: **Lookup**
- Input Parameter: DEST_TABLE_NAME
- Output: LastWatermarkValue (OLDWM ...

OLD_WATERMARK

✓ Last successfully processed table

Incremental Data Extraction Window

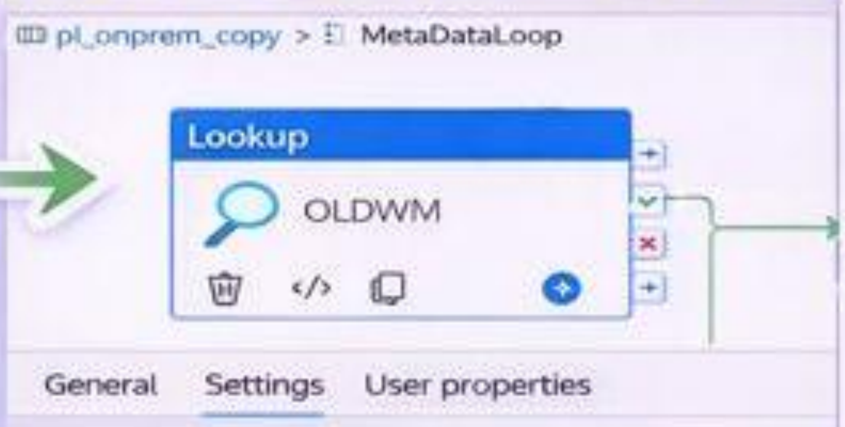
```
updated_at > OLD_WATERMARK
updated_at <= NEW_WATERMARK
```

✓ No duplicate records

✓ No missed data

✓ Fully incremental processing

NEW Watermark Calculation Logic



pl_onprem_copy > MetaDataLoop

DEST_TABLE_NAME

@item().TABLE_NAME

NEW_WATERMARK

✓ Latest available source timestamp

How OLD & NEW Watermarks Work Together

✓ OLD watermark defines where extraction **starts**

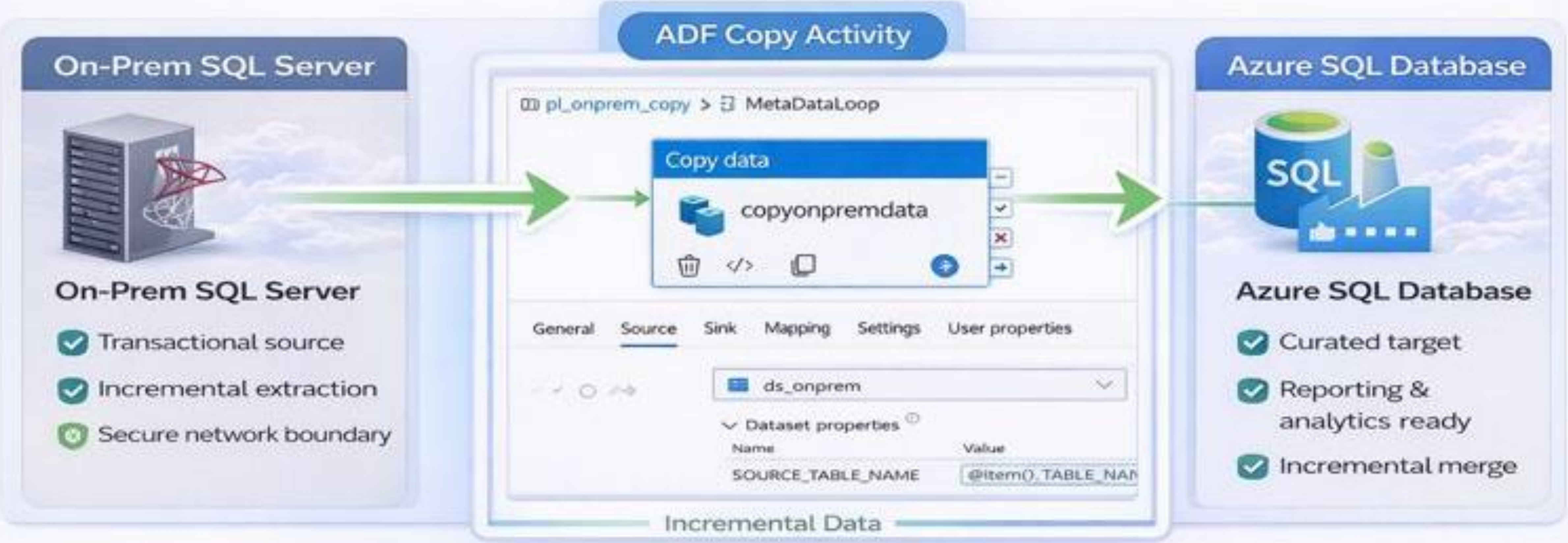
✓ NEW watermark defines where extraction **stops**

✓ After successful ingestion:

- NEW watermark is persisted to the control table

Implementation – Copy Activity (On-Prem to Azure SQL)

Copy Activity – Incremental Data Movement



On-Prem SQL Server

On-Prem SQL Server

- ✓ Transactional source
- ✓ Incremental extraction
- ✓ Secure network boundary

ADF Copy Activity

pl_onprem_copy > MetaDataLoop

Copy data

copyonpremdata

General | **Source** | Sink | Mapping | Settings | User properties

ds_onprem

Dataset properties

Name	Value
SOURCE_TABLE_NAME	@item().TABLE_NAN

Incremental Data

Azure SQL Database

Azure SQL Database

- ✓ Curated target
- ✓ Reporting & analytics ready
- ✓ Incremental merge

Key Points

- ✓ Incremental data loading
- ✓ Metadata-driven table processing
- ✓ Secure hybrid connectivity (SHIR)
- ✓ Scalable & restart-safe

Watermark Update Step

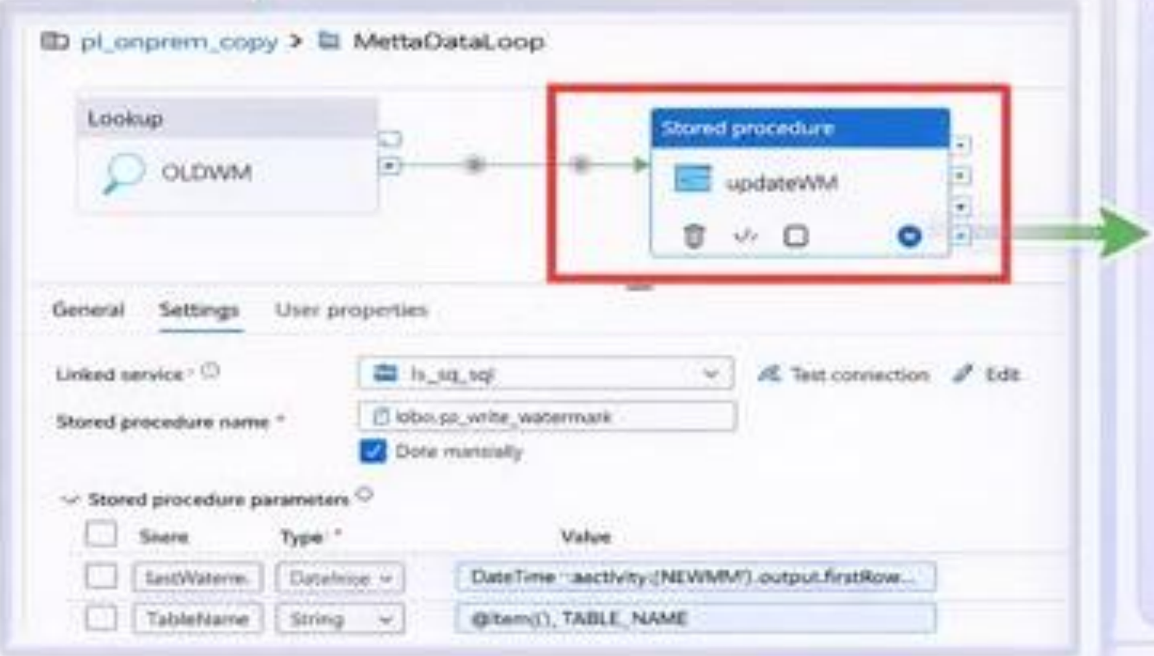
WATERMARK UPDATE STEP

Trigger
After successful Copy Activity

Action
Execute Stored Procedure

Target
Watermark Control Table

Post-Load **Update**



Linked service * ls_sq_sql ⌄ Test connection Edit

Stored procedure name * lobo.pp_write_watermark

☑ Dote manially

Stored procedure parameters

□	Snere	Type *	Value
□	lastWaterm.	Datetiroe ⌄	DateTime ··aectivity:(NEWMM').output.firstRow...
□	TableName	String ⌄	@item(), TABLE_NAME

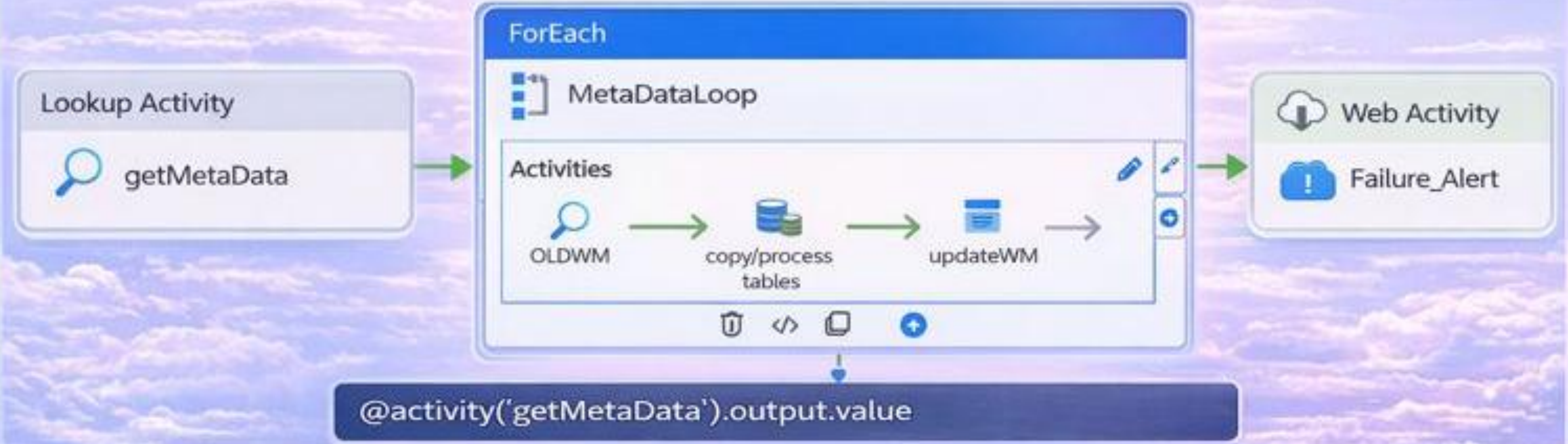
RESULT

- ✓ NEW watermark persisted
- ✓ Becomes OLD watermark
- ✓ Next run starts correctly
- ✓ Restart-safe pipeline

Key Points

- ✓ Idempotent processing
- ✓ Reliable incremental runs
- ✓ No duplicate loads
- ✓ Scalable & restart-safe

Implementation – For Each Loop for Table Processing



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

- ✓ Iterates over each table returned
- ✓ Executes copy and process activities in sequence
- ✓ Alerts on failure via web activity

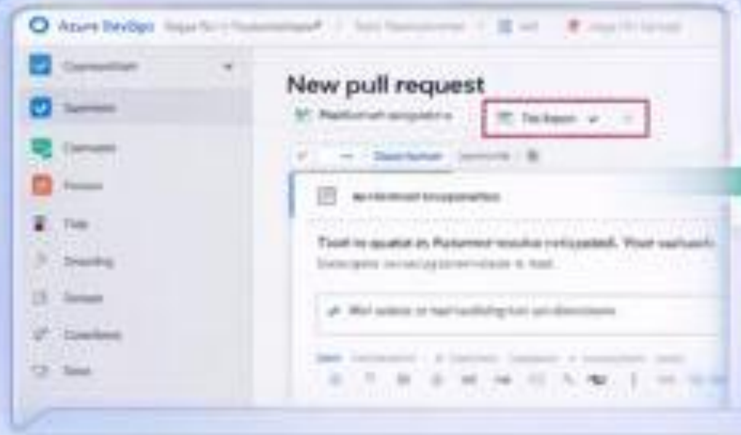
Sequentially executes tasks for each table returned by getMetaData activity

Implementation – Development Workflow

End-to-End Deployment Lifecycle

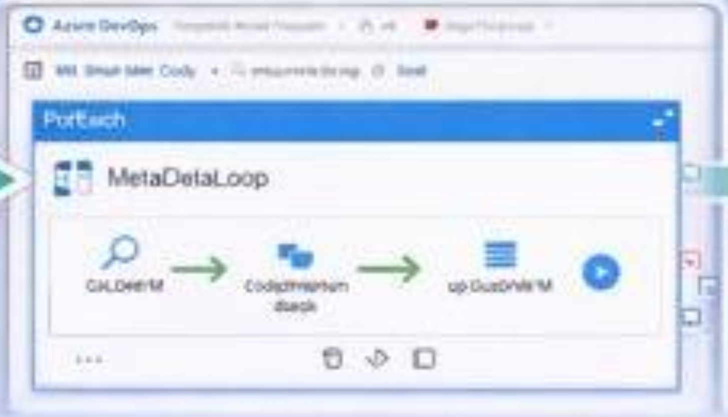
DEVELOP IN FEATURE BRANCH

- ✓ Azure DevOps feature branch
- ✓ Build & validate pipelines
- ✓ Isolated development environment



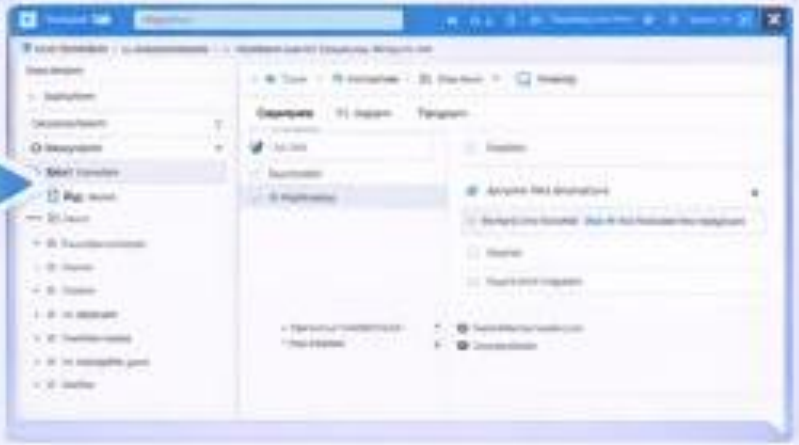
MERGE & DEPLOY WITH AZURE DEVOPS

- ✓ Pull request with code review
- ✓ Artifacts published & merged
- ✓ Deployment to main branch



AUTOMATED SCHEDULING

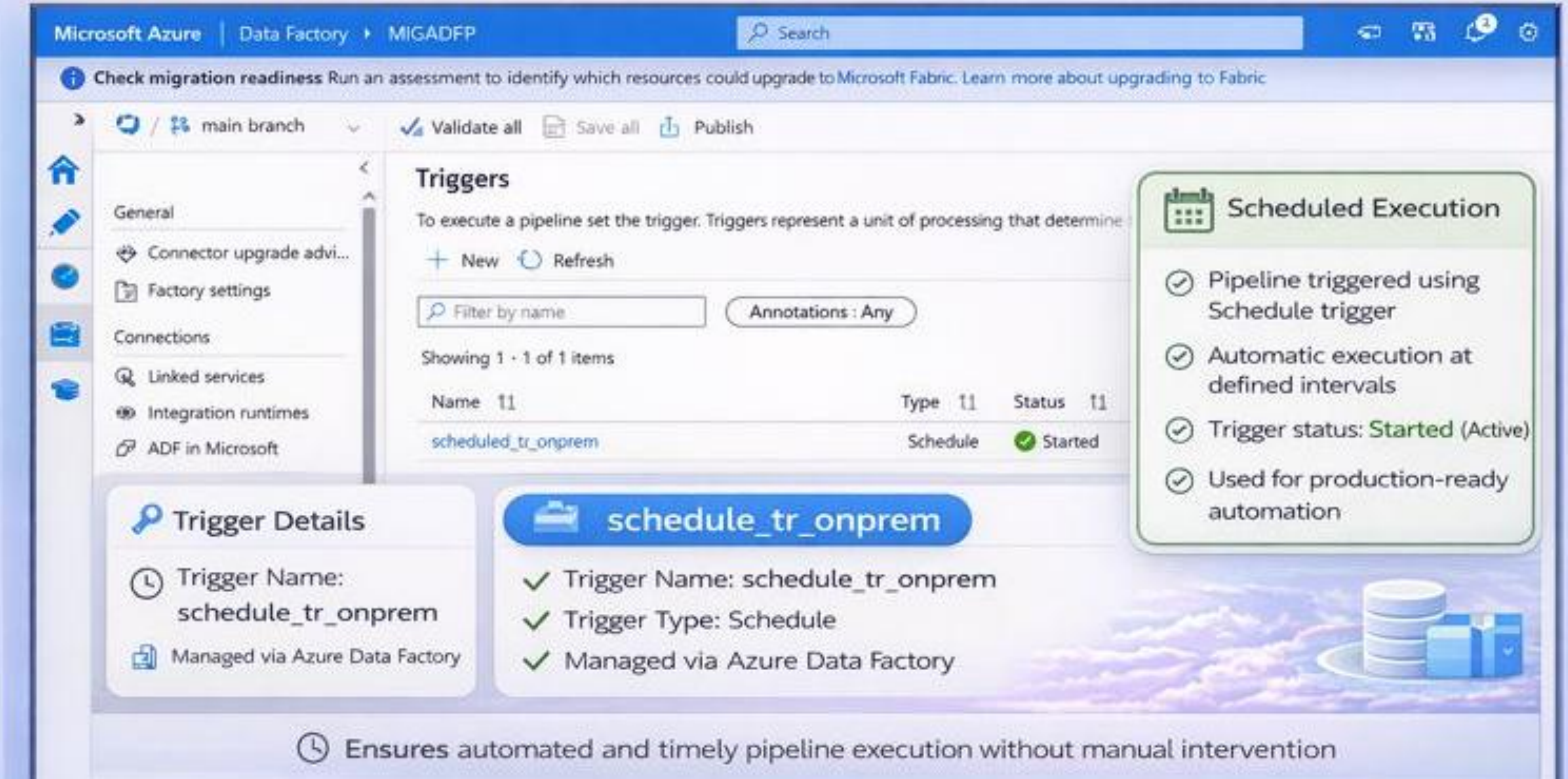
- ✓ Scheduled triggers enabled
- ✓ Processed in production
- ✓ Continuously monitored



Key Points

- ✓ Code merged after approval
- ✓ Artifacts published to main line
- ✓ Artifacts published
- ✓ Scalable & consistent
- ✓ Scaleband ettefuve main line
- ✓ Monitored & stable production

Implementation – Scheduled Execution



Check migration readiness Run an assessment to identify which resources could upgrade to Microsoft Fabric. Learn more about upgrading to Fabric

Triggers

To execute a pipeline set the trigger. Triggers represent a unit of processing that determine

Showing 1 - 1 of 1 items

Name	Type	Status
scheduled_tr_onprem	Schedule	✓ Started

Scheduled Execution

- ✓ Pipeline triggered using Schedule trigger
- ✓ Automatic execution at defined intervals
- ✓ Trigger status: Started (Active)
- ✓ Used for production-ready automation

Trigger Details

Trigger Name: schedule_tr_onprem

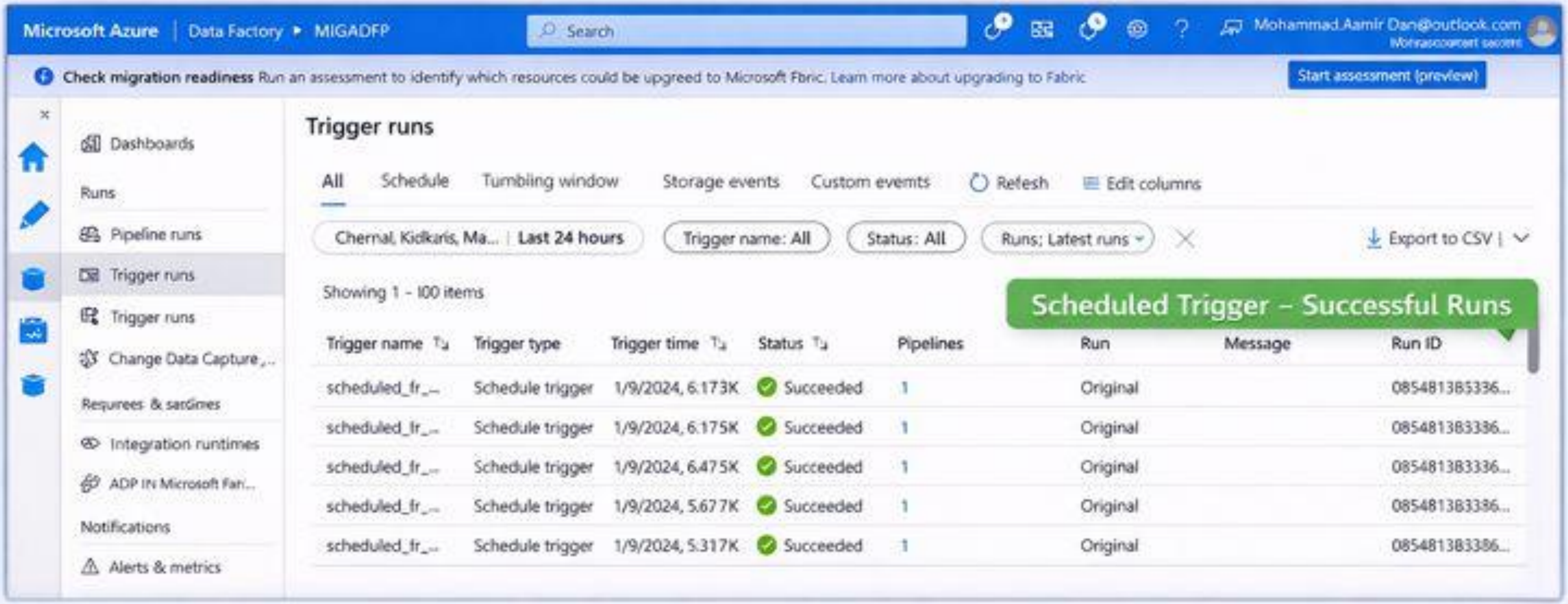
Managed via Azure Data Factory

schedule_tr_onprem

- ✓ Trigger Name: schedule_tr_onprem
- ✓ Trigger Type: Schedule
- ✓ Managed via Azure Data Factory

Ensures automated and timely pipeline execution without manual intervention

Implementation – Trigger Runs Monitoring



Microsoft Azure | Data Factory ▸ MIGADFP
Search
Mohammad.Aamir Dan@outlook.com
Monrascoatent sacotti

Check migration readiness Run an assessment to identify which resources could be upgreed to Microsoft Fbric. Learn more about upgrading to Fabric

Trigger runs

Chernal, Kidkaris, Ma... | **Last 24 hours** · Trigger name: All · Status: All · Runs; Latest runs

Showing 1 - 100 items

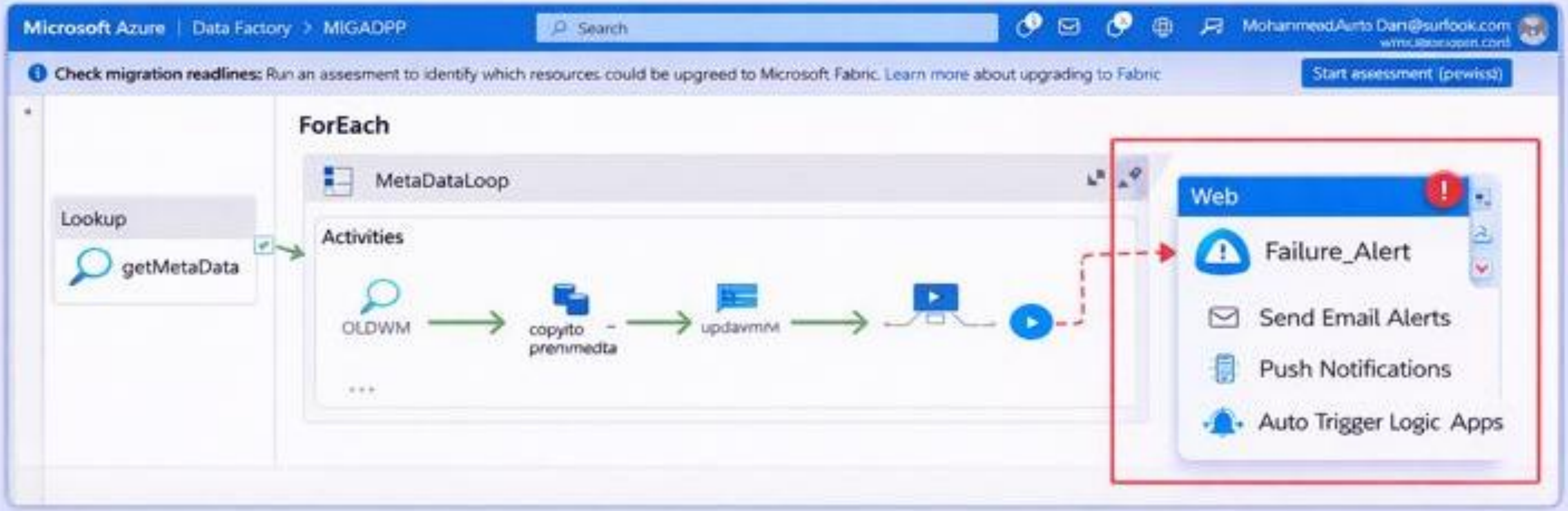
Scheduled Trigger – Successful Runs

Trigger name	Trigger type	Trigger time	Status	Pipelines	Run	Message	Run ID
scheduled_fr_...	Schedule trigger	1/9/2024, 6:173K	✓ Succeeded	1	Original		0854813B5336...
scheduled_fr_...	Schedule trigger	1/9/2024, 6:175K	✓ Succeeded	1	Original		0854813B3336...
scheduled_fr_...	Schedule trigger	1/9/2024, 6.475K	✓ Succeeded	1	Original		0854813B3336...
scheduled_fr_...	Schedule trigger	1/9/2024, 5.677K	✓ Succeeded	1	Original		0854813B3386...
scheduled_fr_...	Schedule trigger	1/9/2024, 5.317K	✓ Succeeded	1	Original		0854813B3386...

- ✓ Automated trigger execution
  → и hours
- ✓ Multiple successful runs
  > latest
- ✓ Continuous monitoring & reliability

Implementation – Failure Monitoring & Alerts

Integrated **Azure Logic Apps** to send automated alerts on pipeline failure, improving operational visibility and response time.



Check migration readlines: Run an assesment to identify which resources could be upgreed to Microsoft Fabric. Learn more about upgrading to Fabric

Real-Time Failure Detection

1 hour

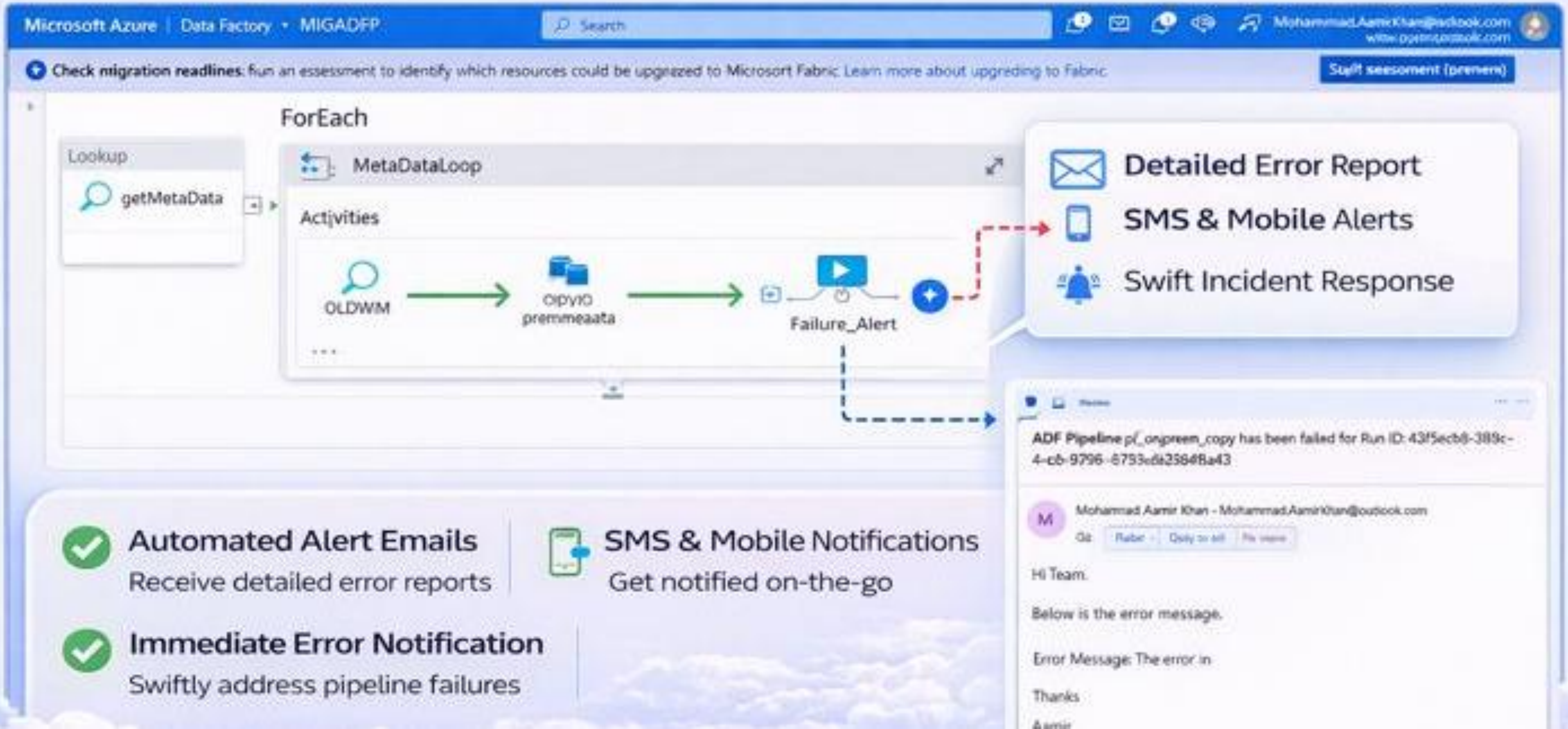
Instant Alerts via Email & SMS

Receive notifications immediately

Improved Operational & reliability

Implementation – Failure Pipeline Alerts

Integrated **Azure Logic Apps** to send automated alerts on pipeline failure, improving operational visibility and response time.


Microsoft Azure | Data Factory • MIGADFP
Search
MohammadAamirKhan@adkook.com
Check migration readlines: Run an essessment to identify which resources could be upgnazed to Microsort Fabric Learn more about upgreding to Fabric
Stuft seesoment (prenem)
ForEach
Lookup
getMetaData
MetaDataLoop
Actjvities
OLDWM
olpylo premmeaata
Failure_Alert
...
Detailed Error Report
SMS & Mobile Alerts
Swift Incident Response
M
Rabel
Ouly to all
Fe wape

ADF Pipeline pl_onpreem_copy has been failed for Run ID: 43f5ecb8-389c-4-cb-9796-6753cdt2384f8a43

Mohammad Aamir Khan - Mohammad.AamirKhan@outlook.com

Hi Team.

Below is the error message.

Error Message: The error in

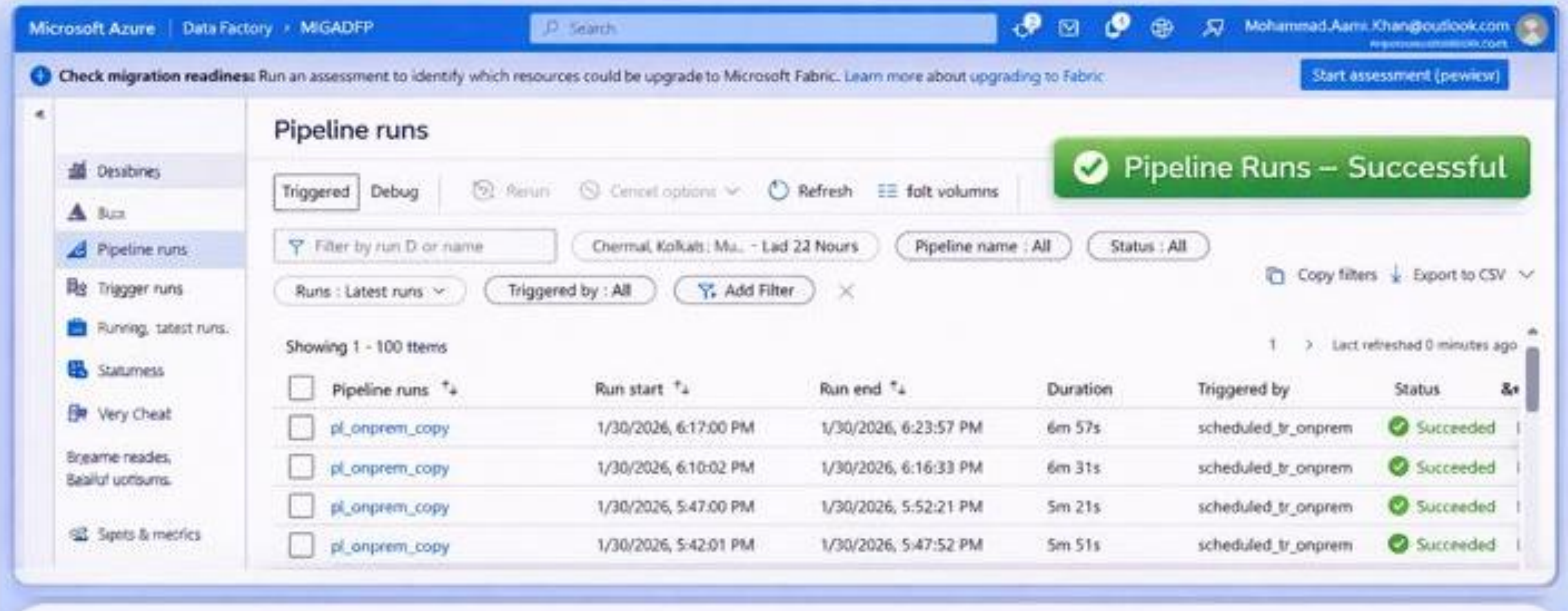
Thanks

Aamir

Automated Alert Emails
Receive detailed error reports

SMS & Mobile Notifications
Get notified on-the-go

Immediate Error Notification
Swiftly address pipeline failures



Microsoft Azure | Data Factory › MIGADFP

Mohammad.Aami.Khan@outlook.com

Check migration readiness: Run an assessment to identify which resources could be upgrade to Microsoft Fabric. Learn more about upgrading to Fabric

Start assessment (pewiesr)

Pipeline runs

✓ Pipeline Runs – Successful

Filter by run D or name

Chermal, Kolkat : Mu.. - Lad 22 Nours | Pipeline name : All | Status : All

Runs : Latest runs | Triggered by : All | Add Filter

Showing 1 - 100 ttems

1 › Lact refreshed 0 minutes ago

☐	Pipeline runs	Run start	Run end	Duration	Triggered by	Status
☐	pl_onprem_copy	1/30/2026, 6:17:00 PM	1/30/2026, 6:23:57 PM	6m 57s	scheduled_tr_onprem	✓ Succeeded
☐	pl_onprem_copy	1/30/2026, 6:10:02 PM	1/30/2026, 6:16:33 PM	6m 31s	scheduled_tr_onprem	✓ Succeeded
☐	pl_onprem_copy	1/30/2026, 5:47:00 PM	1/30/2026, 5:52:21 PM	5m 21s	scheduled_tr_onprem	✓ Succeeded
☐	pl_onprem_copy	1/30/2026, 5:42:01 PM	1/30/2026, 5:47:52 PM	5m 51s	scheduled_tr_onprem	✓ Succeeded

Scheduled Pipeline Executions

Triggered automatically on schedule

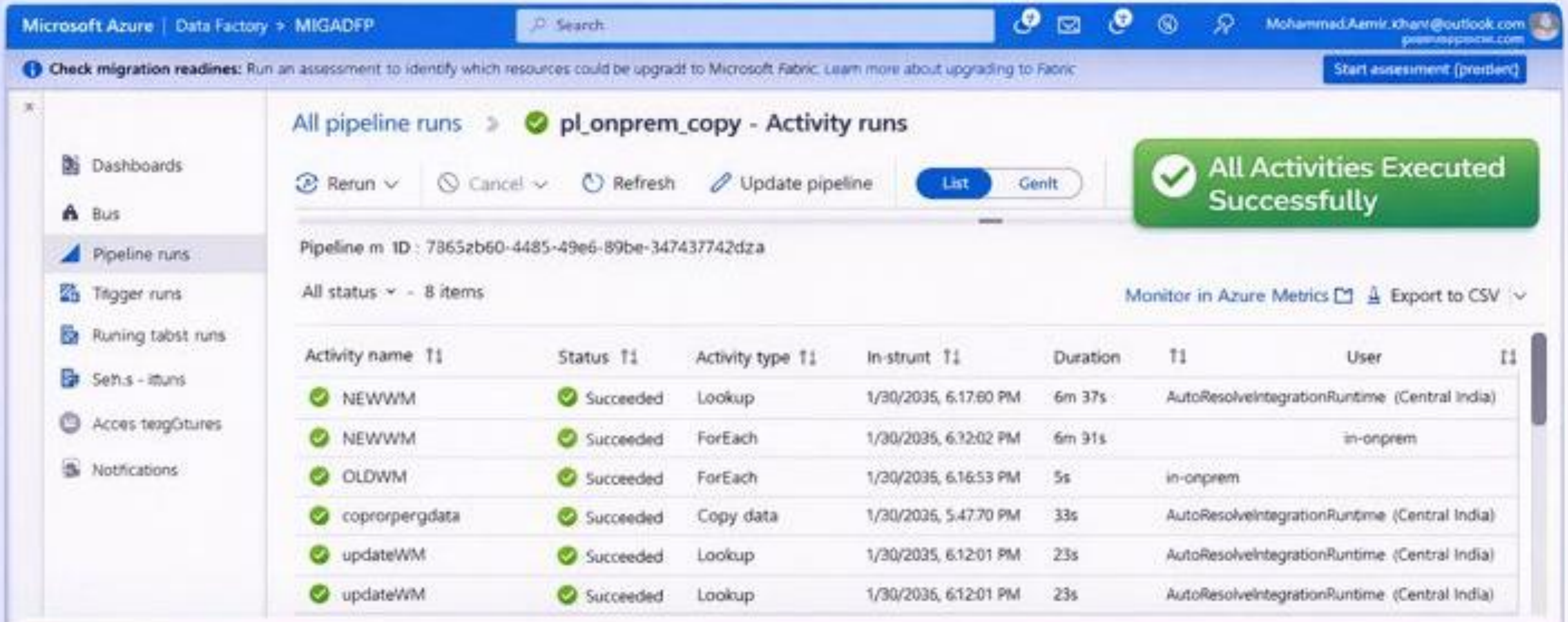
Multiple Successful Runs

Consistent & reliable pipeline executions

Execution Duration & Status Monitored

Track the run time and outcomes

ADF Pipeline Execution: Step-by-Step Breakdown



Check migration readines: Run an assessment to identify which resources could be upgradt to Microsoft Fabric. Learn more about upgrading to Fabric

Start assessment (prediert)

All pipeline runs > pl_onprem_copy - Activity runs

All Activities Executed Successfully

Pipeline m ID : 7865zb60-4485-49e6-89be-347437742dza

All status - 8 items

Activity name ↑↓	Status ↑↓	Activity type ↑↓	In-strunt ↑↓	Duration	↑↓	User ↑↓
✓ NEWWM	✓ Succeeded	Lookup	1/30/2035, 6.17.60 PM	6m 37s	AutoResolveIntegrationRuntime (Central India)	
✓ NEWWM	✓ Succeeded	ForEach	1/30/2035, 6.12.02 PM	6m 31s		in-onprem
✓ OLDWM	✓ Succeeded	ForEach	1/30/2035, 6.16.53 PM	5s	in-onprem	
✓ coprorpergdata	✓ Succeeded	Copy data	1/30/2026, 5.47.70 PM	33s	AutoResolveIntegrationRuntime (Central India)	
✓ updateWM	✓ Succeeded	Lookup	1/30/2035, 6.12.01 PM	23s	AutoResolveIntegrationRuntime (Central India)	
✓ updateWM	✓ Succeeded	Lookup	1/30/2036, 6.12.01 PM	23s	AutoResolveIntegrationRuntime (Central India)	

Lookup Activities

Metadata & watermark resolution

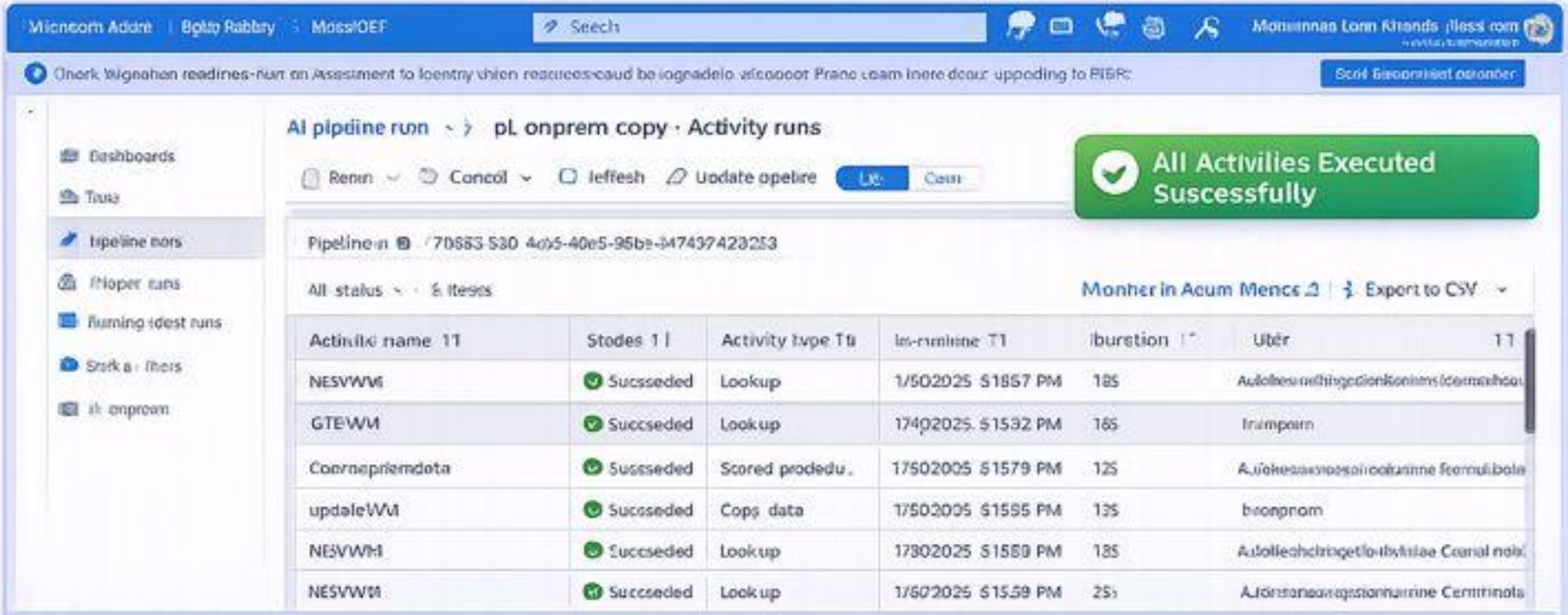
Copy Activity

Incremental data ingestion

Stored Procedure

Watermark updated after load

Activity-Level Execution Details – pl_onprem_copy Pipeline



Micnsorn Addre | Bgbp Rabbry | MossIOEF
Seech
Monunnas Lonn Khands ,llessi rom
Onork Wignahen readines-hurt on Assesiment to loentry vhien reaoueesaud be iognadelo afcooonr Prane ıoam inere dcaur uppoding to RIBR:
Scol Sseonnlet aranoer
Dashboards
Tuua
lıpeline nors
Wloper runs
Runing idest runs
Sorlk a · lhers
ıl onprom
Renn
Concol
Ieffesh
Update opelire
Lıe
Caun

Al pipdine run › pL onprem copy · Activity runs

All Activilies Executed Susscessfully

Pipeline ın /70663 530 4ab5-40e5-95be-b47437423253

All status · 6 iteses

Monher in Acum Mencs | Export to CSV

Actinlıki name	Stodes	Activity lype	In-rınlıne	Iburstion	Ubr
NESVWM	Sucseded	Lookup	1/502025 51857 PM	18S	Aulolkesnothingetionkonimsidomahoo
GTEWM	Succseded	Lookup	17402025 51532 PM	16S	Inımporn
Coornapriemdata	Susseded	Scored prodedu..	17502005 51579 PM	12S	A.uiokesanoosaiooduınne Romulbola
updaleWM	Sucseded	Cops data	1/502005 51595 PM	13S	beonprom
NESVWM	Succeseded	Lookup	17302025 51559 PM	18S	Aulolleohdringetlodbylalae Conal noid
NESVWM	Succeded	Look up	1/502025 51559 PM	25S	A.donsonaoqestionnurrine Cemrinola

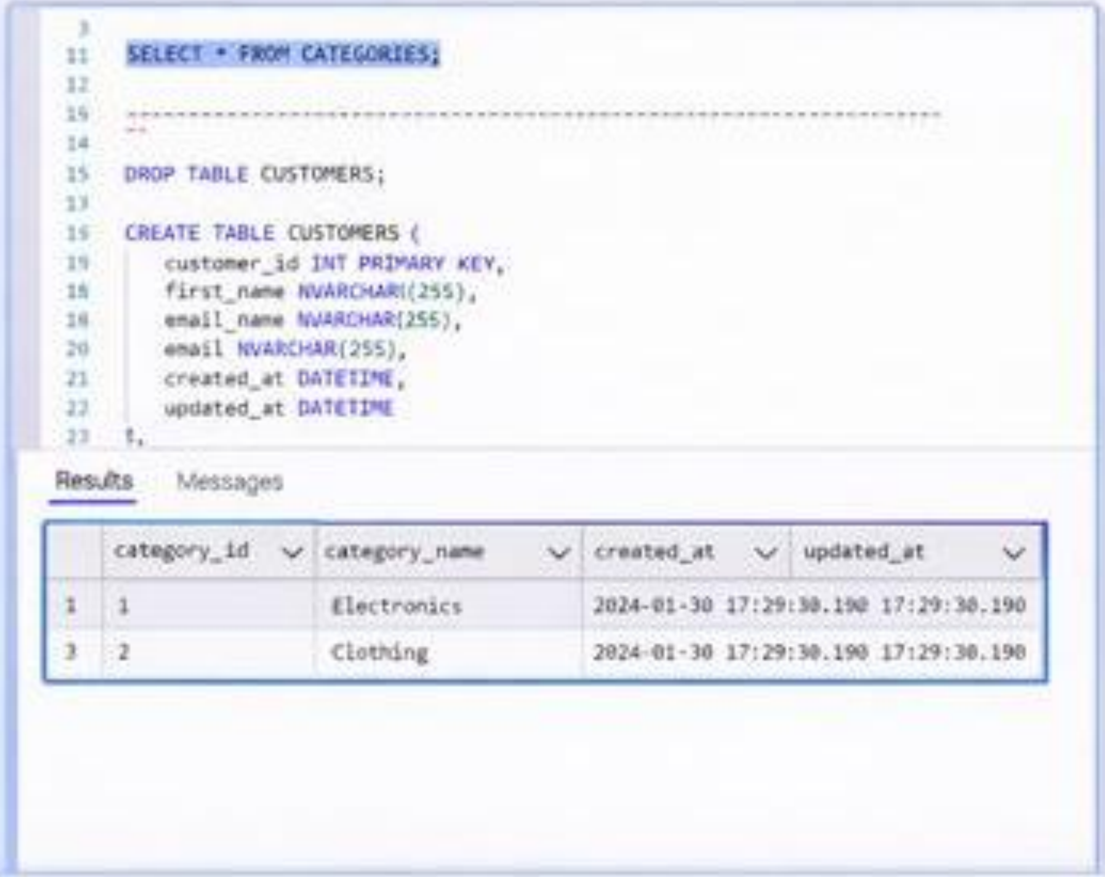
Lookup Activities
Mefodata & waterrnark resolution

Copy Activities
Incrernental data hgeslion

Stored Procedures
Watermark updsted post load

Incremental Data Ingestion Validation in Azure Data Studio

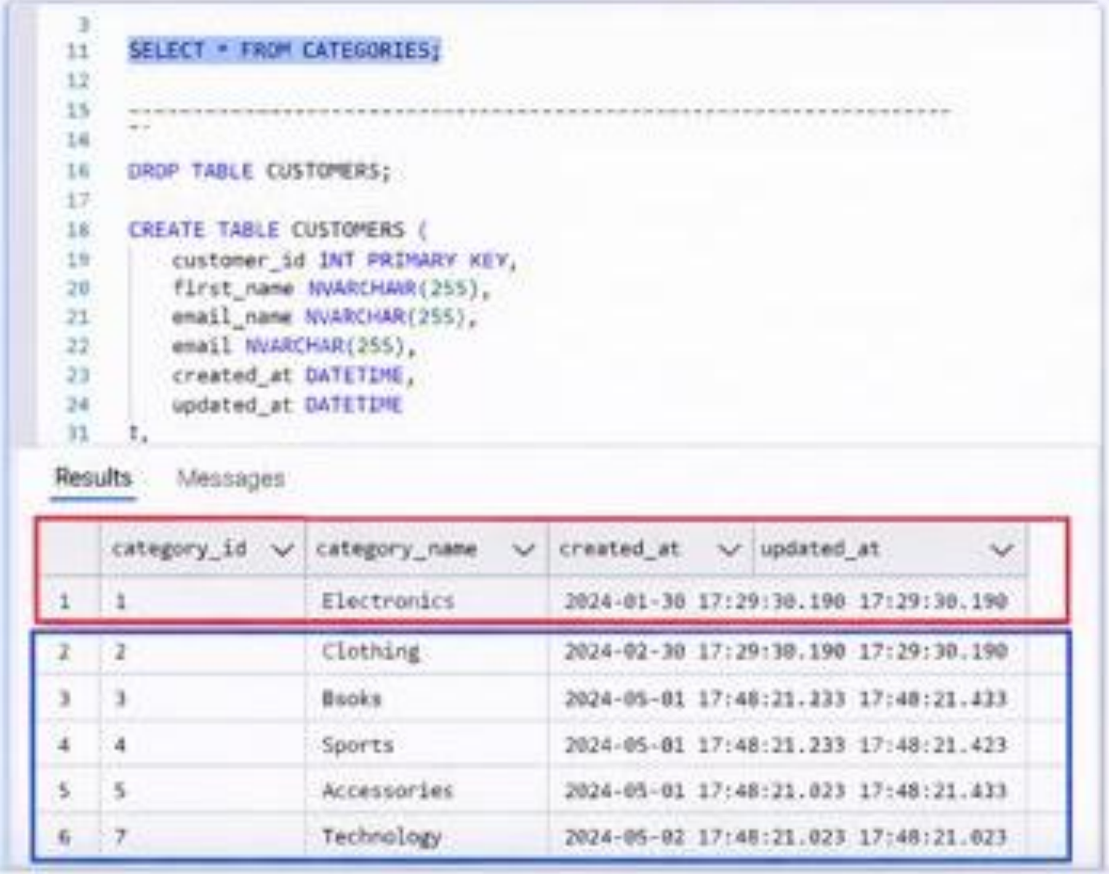
Before Incremental Load



```sql
SELECT * FROM CATEGORIES;

----------------------------------------------------------------------
--

DROP TABLE CUSTOMERS;

CREATE TABLE CUSTOMERS (
    customer_id INT PRIMARY KEY,
    first_name NVARCHARI(255),
    email_name NVARCHAR(255),
    email NVARCHAR(255),
    created_at DATETIME,
    updated_at DATETIME
t,
```

	category_id	category_name	created_at	updated_at
1	1	Electronics	2024-01-30 17:29:30.190	17:29:30.190
3	2	Clothing	2024-01-30 17:29:30.190	17:29:30.190

After Incremental Load



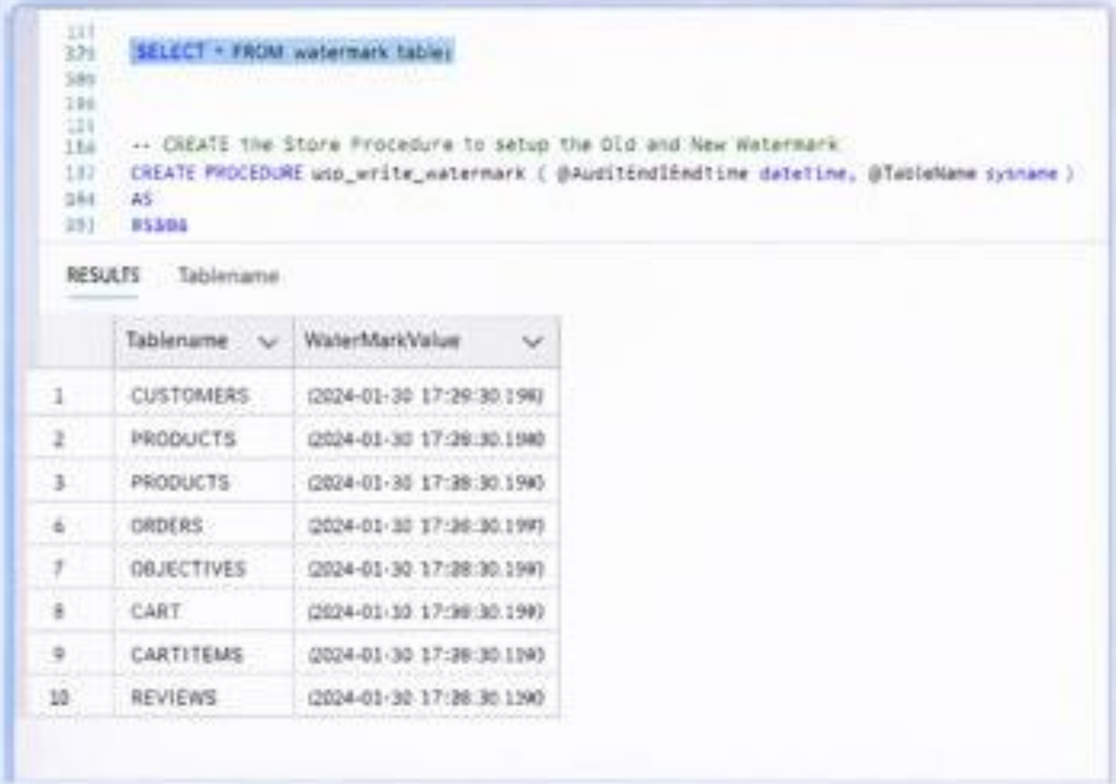
```sql
SELECT * FROM CATEGORIES;

----------------------------------------------------------------------
--

DROP TABLE CUSTOMERS;

CREATE TABLE CUSTOMERS (
    customer_id INT PRIMARY KEY,
    first_name NVARCHAR(255),
    email_name NVARCHAR(255),
    email NVARCHAR(255),
    created_at DATETIME,
    updated_at DATETIME
t,
```

	category_id	category_name	created_at	updated_at
1	1	Electronics	2024-01-30 17:29:30.190	17:29:30.190
2	2	Clothing	2024-02-30 17:29:30.190	17:29:30.190
3	3	Books	2024-05-01 17:48:21.233	17:48:21.433
4	4	Sports	2024-05-01 17:48:21.233	17:48:21.423
5	5	Accessories	2024-05-01 17:48:21.023	17:48:21.433
6	7	Technology	2024-05-02 17:48:21.023	17:48:21.023

- ✓ Incremental records ingested successfully
- ✓ Existing records unchanged
- ✓ Watermark logic validated
- ✓ Data consistency preserved

Tracking Watermark Changes After Incremental Load

Before Incremental Load

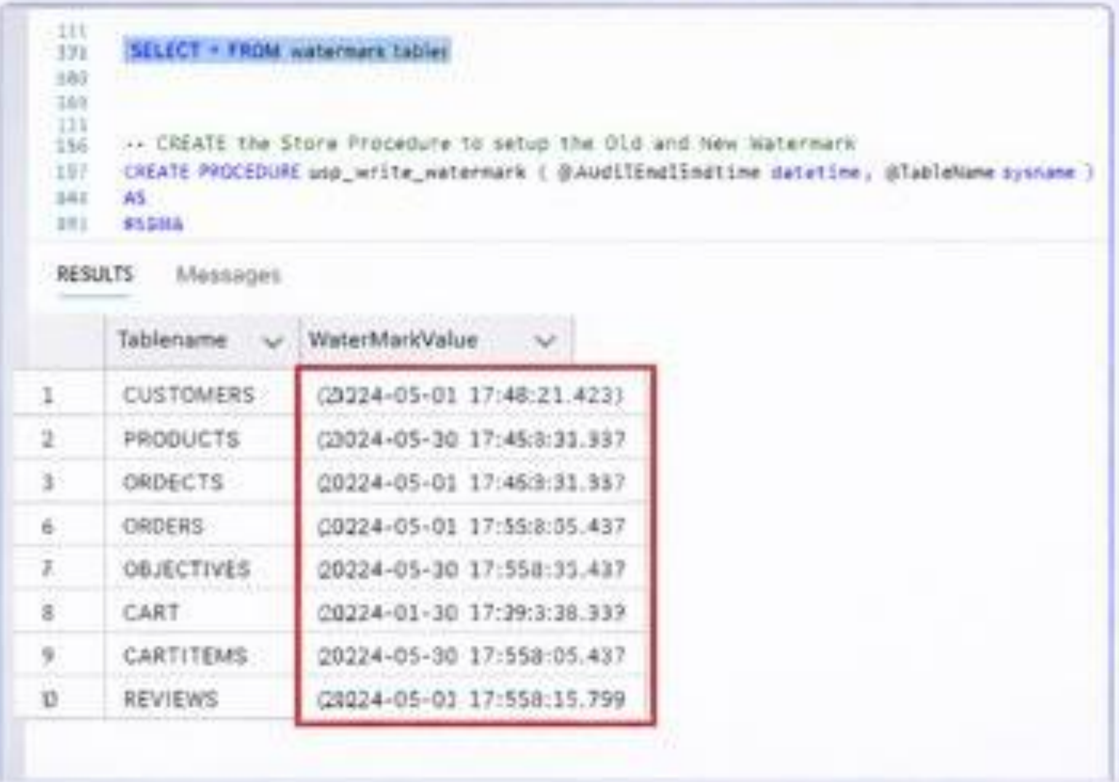


```
SELECT * FROM watermark table;



-- CREATE the Store Procedure to setup the Old and New Watermark
CREATE PROCEDURE usp_write_watermark ( @AuditEndIEndtime dateTime, @TableName sysname )
AS
RS$BA
```

RESULTS   Tablename

	Tablename	WaterMarkValue
1	CUSTOMERS	(2024-01-30 17:29:30.198)
2	PRODUCTS	(2024-01-30 17:28:30.1980
3	PRODUCTS	(2024-01-30 17:38:30.1980
6	ORDERS	(2024-01-30 17:30:30.199)
7	OBJECTIVES	(2024-01-30 17:28:30.199)
8	CART	(2024-01-10 17:30:30.199)
9	CARTITEMS	(2024-01-30 17:38:30.190)
10	REVIEWS	(2024-01-30 17:28:30.1390

After Incremental Load



```
SELECT * FROM watermark tables



-- CREATE the Store Procedure to setup the Old and New Watermark
CREATE PROCEDURE usp_write_watermark ( @AudlTEndIEndtime datetime, @TableName sysname )
AS
#SGBA
```

RESULTS   Messages

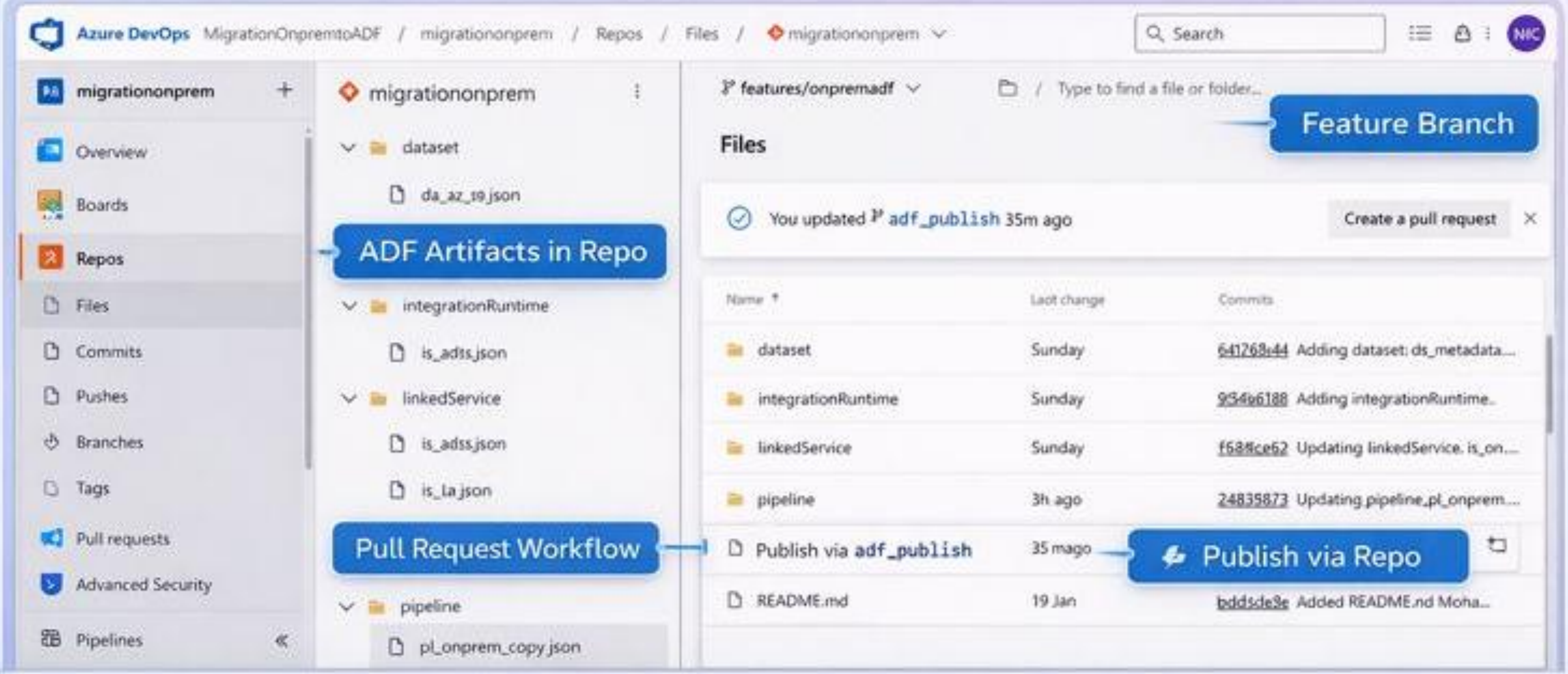
	Tablename	WaterMarkValue
1	CUSTOMERS	(20224-05-01 17:48:21.423)
2	PRODUCTS	(22024-05-30 17:45:8:31.337
3	ORDECTS	(20224-05-01 17:46:3:31.337
6	ORDERS	(20224-05-01 17:55:8:05.437
7	OBJECTIVES	20224-05-30 17:558:35.437
8	CART	(20224-01-30 17:39:3:38.337
9	CARTITEMS	20224-05-30 17:558:05.437
10	REVIEWS	(28024-05-01 17:558:15.799

✓ Watermark updated for all tables

✓ Pipeliine processed new data seamlessly

✓ Consistent incremental update tracking

ADF Git Integration in Azure DevOps



Feature Branch

ADF Artifacts in Repo

Pull Request Workflow

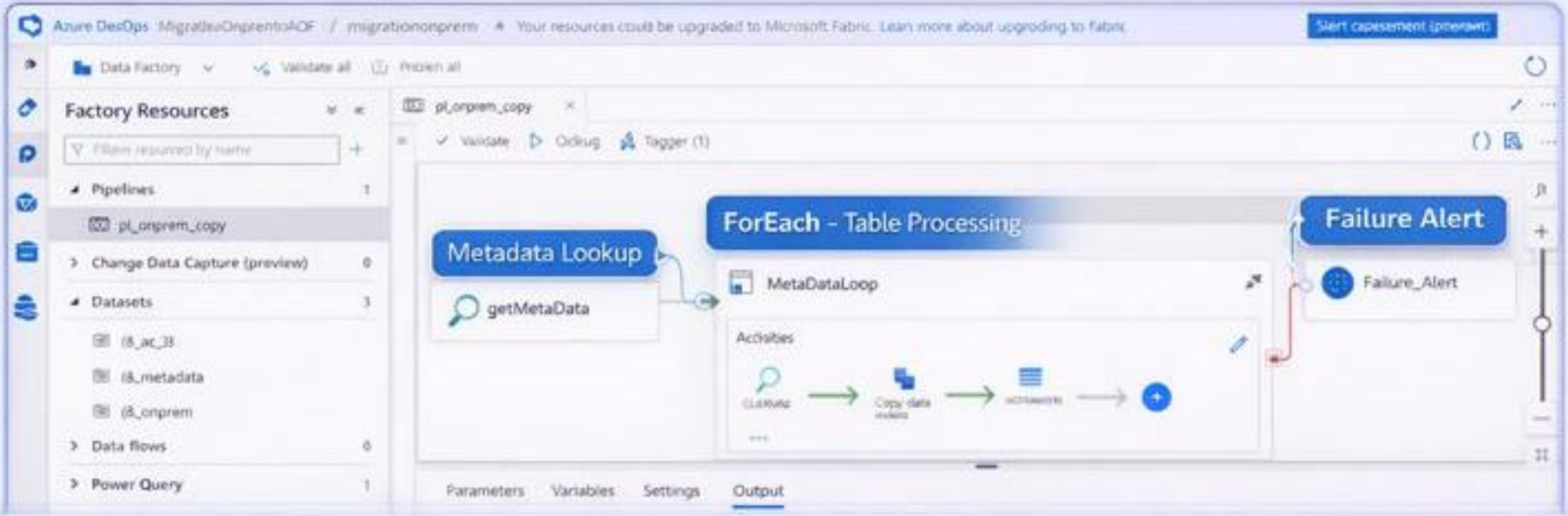
Publish via Repo

You updated adf_publish 35m ago

Name	Last change	Commits
dataset	Sunday	641268a44 Adding dataset: ds_metadata...
integrationRuntime	Sunday	9S4a6188 Adding integrationRuntime..
linkedService	Sunday	f68fce62 Updating linkedService. is_on...
pipeline	3h ago	24835873 Updating pipeline.pl_onprem...
Publish via adf_publish	35 mago	
README.md	19 Jan	bddsde9e Added README.nd Moha...

- ✓ Version-controlled ADF pipelines
- ✓ Feature-branch-based development
- ✓ Pull request & code review process

Read metadata → process incrementally → update watermark → monitor & alert



Read Metadata → ForEach Loop → Copy Data → Update Watermark→ Monitor & Alert

Read Metadata	ForEach Loop	Copy Data	Update Watermark	Monitor & Alert
Identify tables & watermark	Dynamic table processing	Incremental ingestion	Persist new state	Failure notifications

Thank You

Mohammad Aamir Khan

Aspiring Data Engineer