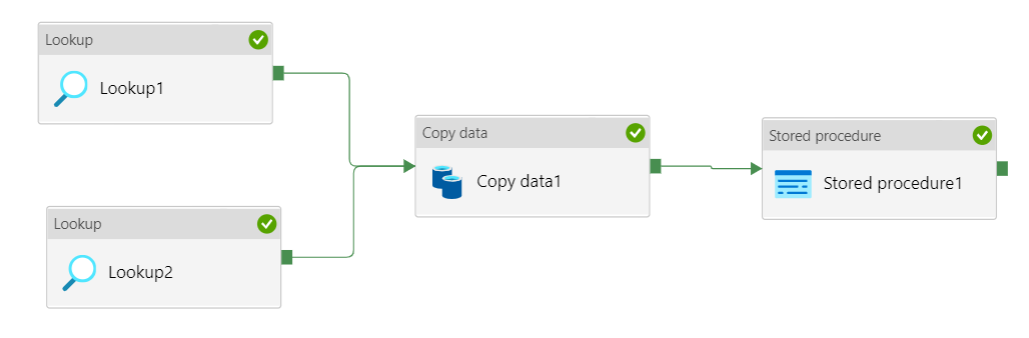
**INCREMENTAL LOAD AND LOGIC APP**

INCREMENTAL LOAD EXECUTION



In this tutorial, you create an Azure Data Factory with a pipeline that loads delta data from a table in Azure SQL Database to Azure Blob storage.

You perform the following steps

* Prepare the data store to store the watermark value.
* Create a data factory.
* Create linked services.
* Create source, sink, and watermark datasets.
* Create a pipeline.
* Run the pipeline.
* Monitor the pipeline run.
* Review results
* Add more data to the source.
* Run the pipeline again.
* Monitor the second pipeline run
* Review results from the second run

**Create a data source table in your SQL database**

1. Open SQL Server Management Studio. In **Server Explorer**, right-click the database, and choose **New Query**.

2. Run the following SQL command against your SQL database to create a table named data\_source\_table as the data source store:

create table data\_source\_table

(

PersonID int,

Name varchar (255),

LastModifytime date time

);

INSERT INTO data\_source\_table

(PersonID, Name, LastModifytime)

VALUES

(1, 'aaaa','9/1/2017 12:56:00 AM'),

(2, 'bbbb','9/2/2017 5:23:00 AM'),

(3, 'cccc','9/3/2017 2:36:00 AM'),

(4, 'dddd','9/4/2017 3:21:00 AM'),

(5, 'eeee','9/5/2017 8:06:00 AM');

In this, you use LastModifytime as the watermark column. The data in the data source store is shown in the following table:

**Create another table in your SQL database to store the high watermark value**

1) Run the following SQL command against your SQL database to create a table named watermarktable to store the watermark value

Create table watermarktable

(

TableName varchar (255),

WatermarkValue datetime,

);

2) Set the default value of the high watermark with the table name of source data store. In this, the table name is data\_source\_table.

INSERT INTO watermarktable

VALUES ('data\_source\_table','1/1/2010 12:00:00 AM')

3) Review the data in the table watermarktable.

Select \* from watermarktable

4) Output:

TableName | WatermarkValue

data\_source\_table | 2010-01-01 00:00:00.000

**Create a stored procedure in your SQL database**

Run the following command to create a stored procedure in your SQL database:

Queries:

Procedure:

CREATE PROCEDURE usp\_write\_watermark @LastModifiedtime datetime, @TableName varchar (50)

AS

BEGIN

UPDATE watermarktable

SET [WatermarkValue] = @LastModifiedtime

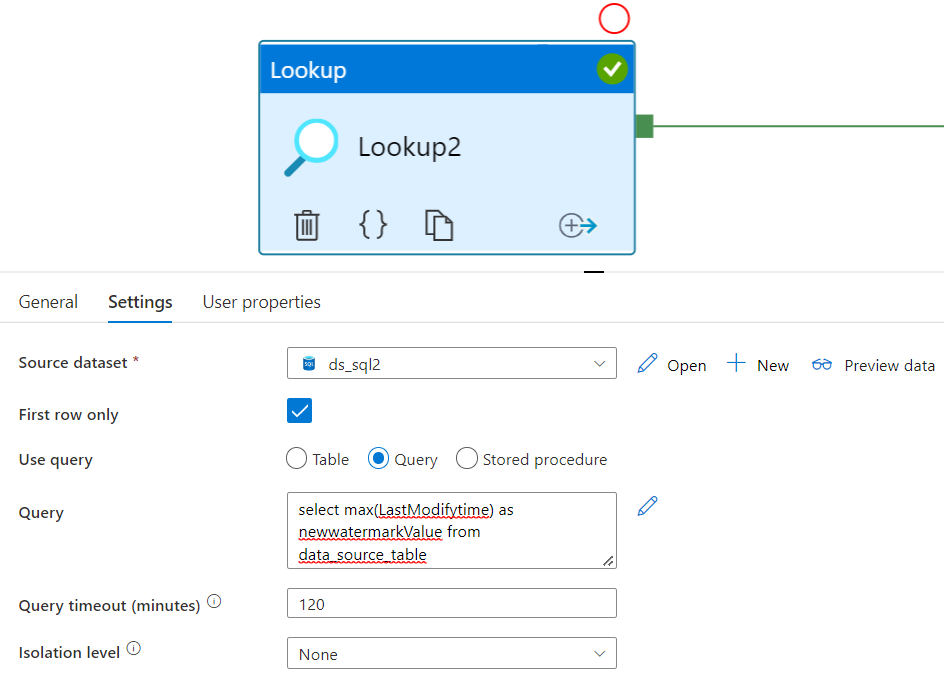
WHERE [TableName] = @TableName

**STEP1**

In **lookup 1** activity we use table **watermarktable** to fetch records based on lastmodifytime (timestamp)

**TABLE:**

SELECT TABLE NAME IS **WATERMARKTABLE**

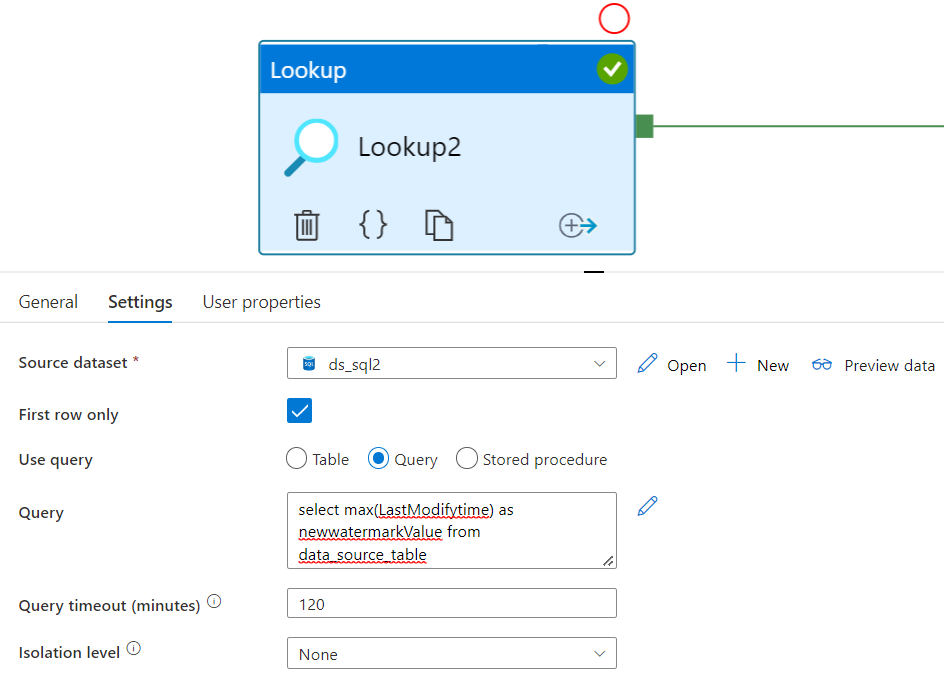


**STEP2**

* In **lookup 2** activity we use query to fetch max of **lastmodifytime**
* Here we select use query given below

QUERY:

**Select MAX (LastModifytime) as NewWatermarkvalue from data\_source\_table**



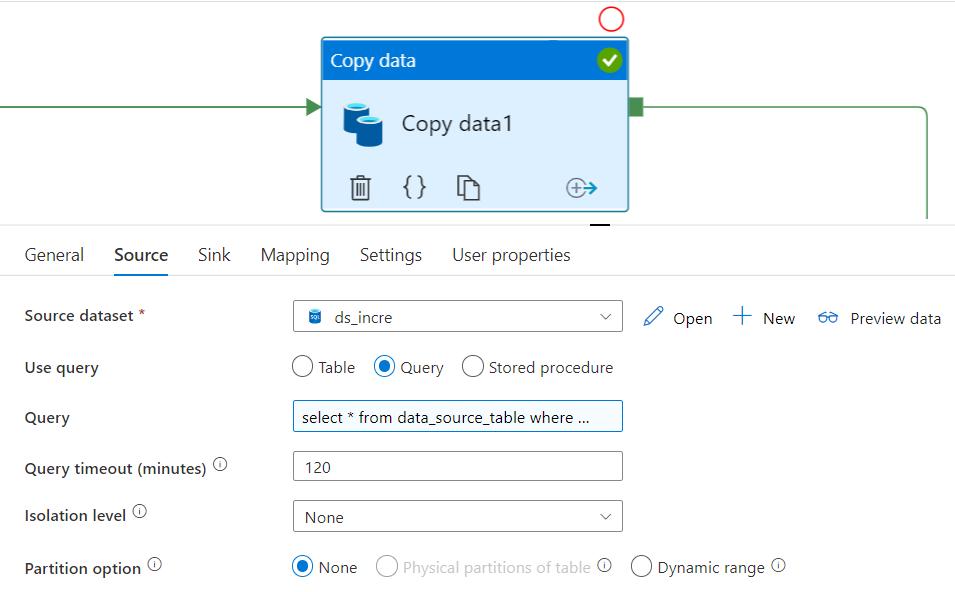
**STEP3**

**SOURCE**

* In **copy data** activity we use query to fetch both loads in copydata
* Select query as given below

**QUERY:**

**Select \* from data\_source\_table where LastModifytime > '@{activity('LookupOldWaterMarkActivity').output.firstRow.WatermarkValue}' and LastModifytime <= '@{activity('LookupNewWaterMarkActivity').output.firstRow.NewWatermarkvalue}'**



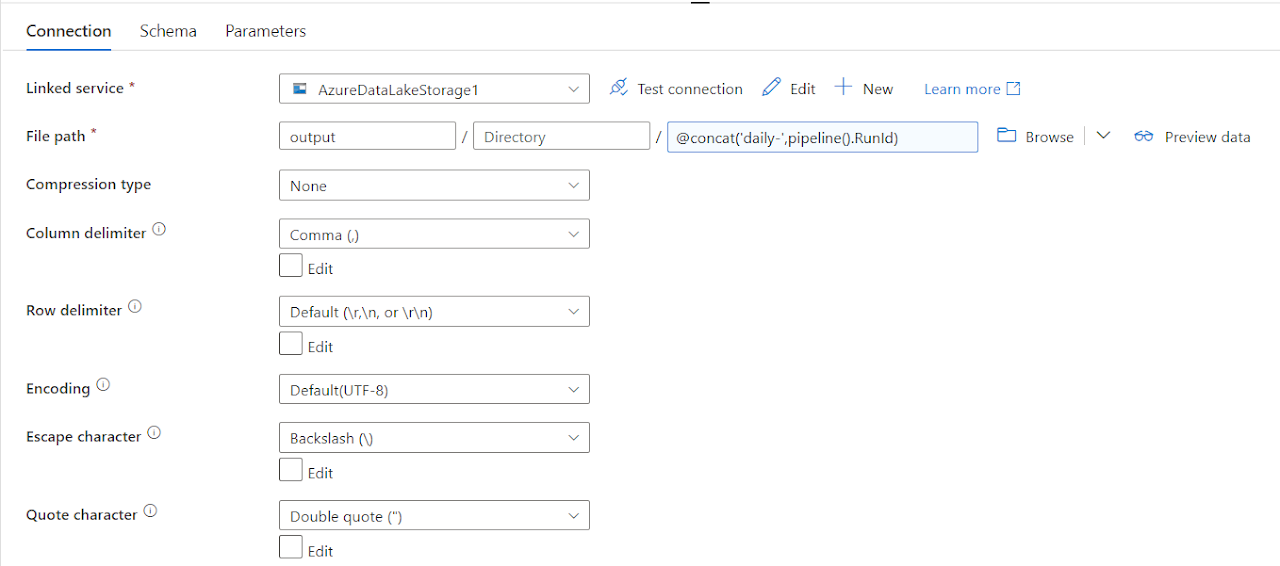
**STEP4**

**SINK:**

* Here we use **data lake** **storage** to store the output with the help **stored proced0ure**
* In sink at **dataset** level in place of container under file path we use a query to get the results

**FILE:**

**@concat('daily-', pipeline().RunId)**



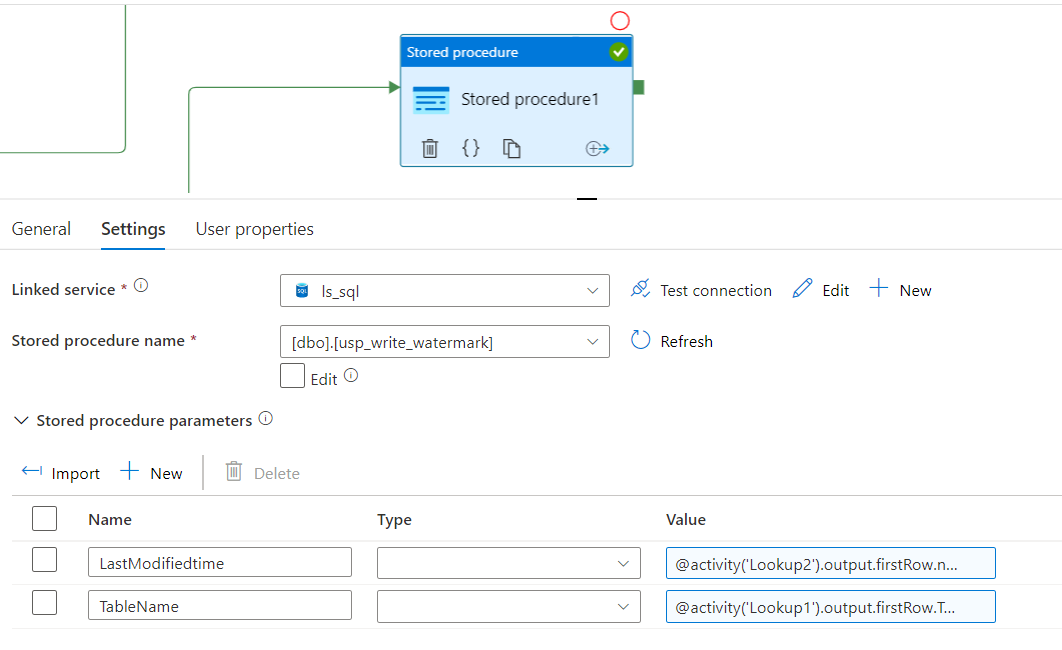
**STEP 5**

In **stored procedure** activity we used a procedure table and also we used the **parameters** to fetch the records **incrementally**

**PARAMETERS:**

**LastModifiedtime:@activity('Lookup2').output.firstRow.newwatermarkValue**

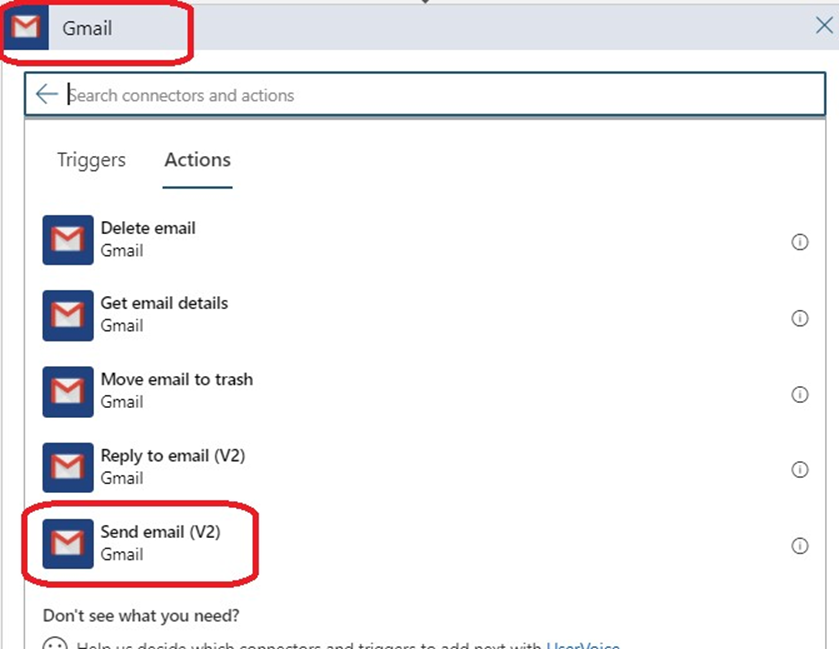
**TableName : @activity('Lookup1').output.firstRow.TableName**



**LOGIC APPS**

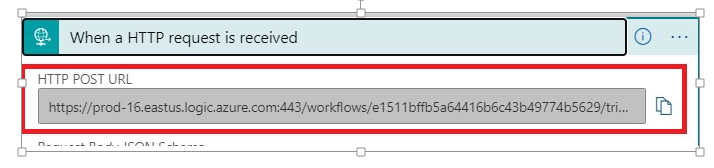
**STEP1**

We can use this for multiple scenarios. Logic app (email) and adf (error handling). The communication between these two azure parts is done with a json message via an http request (post). The json message contains the name of the data factory and the pipeline that failed, an error message and an email address. You could of course hardcode the email address in logic apps, but now you can reuse the logic app for various pipelines or data factories and notify different people.



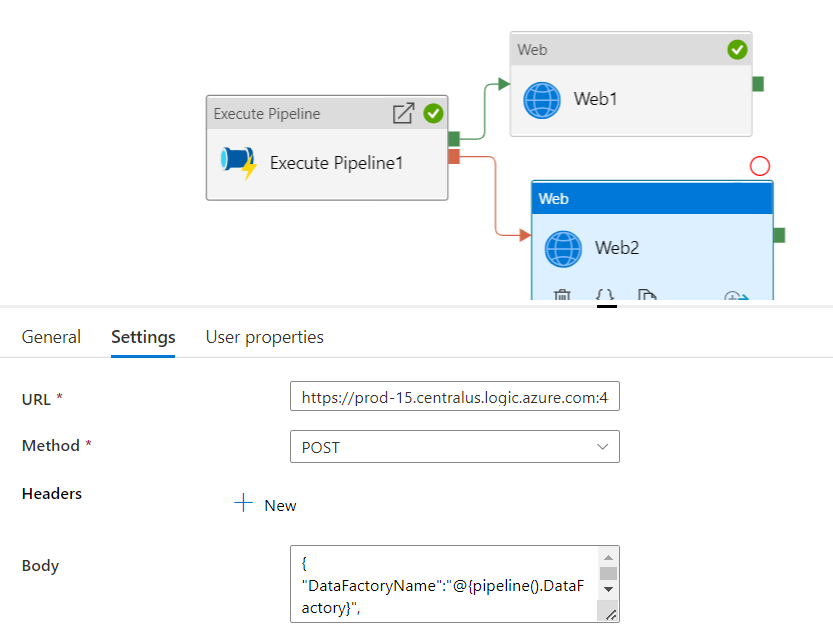
**STEP2**

* Next, we will add a new step to our logic app, called “send an email”. I will use Gmail but if you want to use another email provider pick that one.
* It’s the first time you connect Gmail account on azure? Then you need to connect your Gmail account to azure by signing in. (note: allow pop-ups in your browser.)



**STEP3**

After creation of azure logic app and saving the logic app, azure created an endpoint URL for our logic apps, you’ll find in the first step. Copy this URL to a notepad, we’ll need this later.



**STEP4**

* HERE WE USE EXCUTE PIPELINE ACTIVITY TO EXECUTE CHILD PIPELINE USING TWO WEB ACTIVITIES ONE FOR SECCEEDED AND OTHER FOR FAILURE.
* AFTER COMPLETION OF EXECUTION WE WILL GET THE EMAIL NOTIFICATION WETHER IT SUCCEEDED OR FAILED