International Journal of Computer Trends and Technology
ISSN: 2231 – 2803 / https://doi.org/10.14445/22312803/IJCTT-V69I2P106

Volume 69 Issue 2, 41-45, February 2021
© 2021 Seventh Sense Research Group®

Original Article

Microsoft Azure Dynamic Pipelines

Sanjeev Kumar

SAP Analytics Solution Architect, GyanSys Inc, 702 Adams Street, Carmel IN, USA

Received Date: 21 December 2020
Revised Date: 04 February 2021
Accepted Date: 07 February 2021

Abstract *- Azure Data Factory is a cloud-based data integration service that allows you to create data-driven workflows in the cloud for orchestrating and automating data movement and data transformation.*

Azure Data Factory does not store any data itself. It allows you to create data-driven workflows to orchestrate the movement of data between supported data stores and the processing of data using compute services in other regions or in an on-premise environment. It also allows you to monitor and manage workflows using both programmatic and UI mechanisms.

Keywords *- Azure ADF, Cloud Computing, Data Factory pipelines*

I. INTRODUCTION

The Azure Data Factory service allows you to create data pipelines that move and transform data and then run the pipelines on a specified schedule (hourly, daily, weekly, etc.). This means the data that is consumed and produced by workflows is time-sliced data, and we can specify the pipeline mode as scheduled (once a day) or one time.

So, what is Azure Data Factory, and how does it work? The pipelines (data-driven workflows) in Azure Data Factory typically perform the following three steps:

- Connect and Collect: Connect to all the required sources of data and processing, such as SaaS services, file shares, FTP, and web services. Then, move the data as needed to a centralized location for subsequent processing by using the Copy Activity in a data pipeline to move data from both on-premise and cloud source data stores to a centralization data store in the cloud for further analysis.
- Transform and Enrich: Once data is present in a centralized data store in the cloud, it is transformed using compute services such as HDInsight Hadoop, Spark, Data Lake Analytics, and Machine Learning.
- Publish: Deliver transformed data from the cloud to on-premise sources like SQL Server or keep it in your cloud storage sources for consumption by BI and analytics tools and other applications.

II. AZURE ADF DYNAMIC PIPELINES

During the recent project I implemented at one of our prestigious clients in the manufacturing space, I encountered a specific requirement to ingest data from 150 different sources. Creating separate pipelines for each data source will end up creating 150 ADF pipelines, and that could be a very complex maintenance effort post-go-live. I did a lot of research and came up with a design to configure Dynamic Pipelines, which means that every data set would run through the same pipeline. To logically design the lifecycle of a dataset from source to the destination inside a Synapse table, I planned to configure 3 pipelines that do the whole data processing to ingest data into Synapse. As a result, all 150 data sets were able to get processed within just 3 pipelines. This is a very sophisticated technical configuration, and I neither find it anywhere over the internet nor found any books. I would like to share this knowledge and experience gained so that anyone who is trying to implement something similar can design the best industry solution, which is robust, maintainable, and re-usable. This did make my client team very impressed. So here we go.

A. *Business Requirement*

Let me first talk about the business requirement. There are around 150 different sources of Point-of-Sales data from different online retailers like Amazon, Home Depot, Way Fair, Lowes, etc. Business users used to go to each point-of-sales platform to download the data for analysis purposes. For example, to find out how many customers browsed their product on Amazon, Home Depot, or Way Fair platforms, for how long they were there on the website, how many products they added to the cart, what other products they added to the cart, did they end up buying all of them or just abandoned the cart. For all such kinds of analysis, business users had to go to each platform separately to download the data. Not only that, after downloading the data, they had to apply multiple business logic to convert that data into some meaningful information.

The intent of this project is to bring all this data from all different sources into one platform and apply the business logic as well before the business team access and create their reports. Microsoft Azure Platform was chosen by the customer due to the latest and greatest cloud technology features like Azure Data Factory, Data Lake, Synapse, Logic Apps and Power BI services, etc.

B. *Azure Data Factory*

One of the major components implemented in this project is Azure Data Factory. Azure ADF is used to pick the datasets from multiple sources, and these data sets are in a very unstructured format. This dataset passes through the Azure ADF pipelines for processing and to convert it into a structured format together with business logic applied to it. Finally, via Azure ADF, this data is moved into Synapse and Data Lake. Business users finally access this data via Power BI service by connecting it to Data Lake or Synapse DW.

C. *Dynamic Azure ADF Pipelines*

To save a lot of maintenance effort and also the development time, I came up with a unique solution design to create dynamic ADF pipelines. All 150 datasets pass through one single pipeline dynamically.

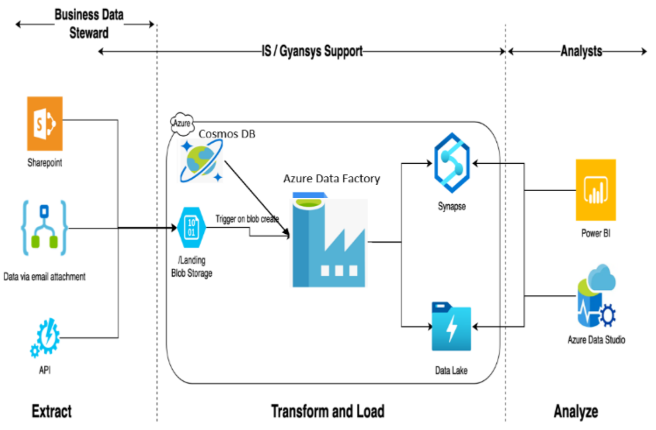


Fig. 1 High-Level Solution Architecture

Azure ADF Dynamic Pipeline components

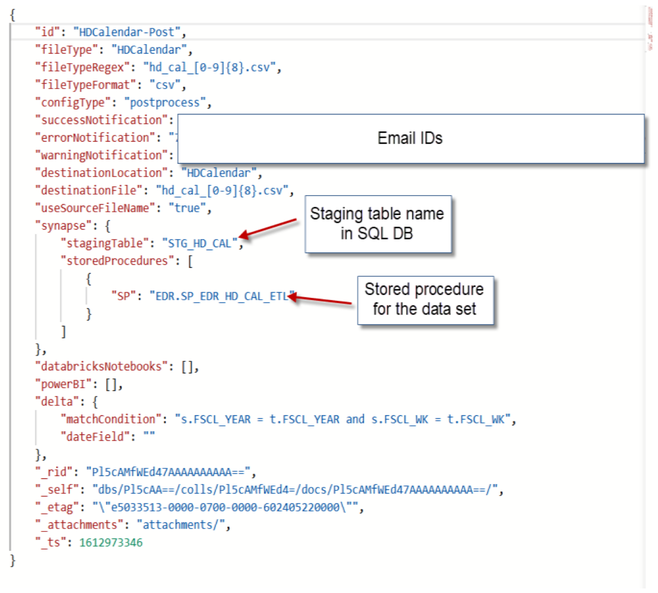
Below are the main components configured to set up a dynamic pipeline:

- Cosmos DB
- Blob Storage Containers
- Synapse
- SQL Stored Procedures
- Data Bricks

D. *Cosmos DB*

Cosmos DB is a unique database that can hold data in JSON format. This database is used to store the configurations related to each data set. When a dataset starts processing within the data factory pipeline, the configuration related to that data set is read from Cosmos DB, and accordingly, that data set is processed, e.g., config related to condition is this dataset is unstructured and has to pass through the data bricks to convert it into a structured format or information related to the fields which will go into the final table in Synapse, information about the stored procedure to be triggered for that particular data set, etc. Example – Cosmos DB dataset configuration.

You can find more information during the ADF discussion ahead on how to use this configuration within pipelines.



```
{
    "id": "HDCalendar-Post",
    "fileType": "HDCalendar",
    "fileTypeRegex": "hd_cal_[0-9]{8}.csv",
    "fileTypeFormat": "csv",
    "configType": "postprocess",
    "successNotification":
    "errorNotification": "
    "warningNotification":
    "destinationLocation": "HDCalendar",
    "destinationFile": "hd_cal_[0-9]{8}.csv",
    "useSourceFileName": "true",
    "synapse": {
        "stagingTable": "STG_HD_CAL",
        "storedProcedures": [
            {
                "SP": "EDR.SP_EDR_HD_CAL_ETL"
            }
        ]
    },
    "databricksNotebooks": [],
    "powerBI": [],
    "delta": {
        "matchCondition": "s.FSCL_YEAR = t.FSCL_YEAR and s.FSCL_WK = t.FSCL_WK",
        "dateField": ""
    },
    "_rid": "Pl5cAMfWEd47AAAAAAAAAA==",
    "_self": "dbs/Pl5cAA==/colls/Pl5cAMfWEd4=/docs/Pl5cAMfWEd47AAAAAAAAAA==/",
    "_etag": "\"e5033513-0000-0700-0000-602405220000\"",
    "_attachments": "attachments/",
    "_ts": 1612973346
}
```

Email IDs
Staging table name in SQL DB
Stored procedure for the data set

E. *Blob Storage container*

Files or data sets from the shared drive or other online platforms are captured and landed into the blob storage in Excel, CSC, JSON formats. These files are either dropped manually or via email or pulled via APIs. As soon as a Blob or a data set hits the Blob storage container, the data factory pipeline gets triggered to process it.

Synapse: This is the final data warehouse service where the processed data is stored in tables,

F. *SQL Stored Procedures*

After converting the file from an unstructured to a structured format, stored procedure/s are run to apply the business logic and turn that data into meaningful information and finally to store that data in the synapse table.

G. *Databricks*

If a dataset is in an unstructured format, that needs to be converted into a structured format first based on the fields required by the business to analyze on. This processing happens via data bricks. Databricks component consists of a python programming paradigm which does this conversion from unstructured to the structured format.

H. *Dynamic Pipeline configuration:*

To logically divide the whole data set processing until Synapse table, there are 3 pipelines configured – Router, Pre-process and Post-process. The router pipeline picks the data set file and passes it to either pre-process pipeline or post-process pipeline. If any dataset needs to be converted from an unstructured to a structured format, they are passed to the pre-processing pipeline and processed here. After conversion, the original data set file is converted into

multiple structure CSV files, and these files are further passed to the post-processing pipeline to move data to the Synapse table. Any files which are already structured and need no pre-processing are directly passed to the post-processing pipeline for Synapse storage. To achieve the dynamic behavior of the pipelines, there is extensive use of variables and parameters within the pipelines. Any information related to datasets is not at all hard-coded anywhere in the pipelines, and everything is achieved via Cosmos config and variables/parameters within the pipeline.
For the dynamic pipeline config settings, I will talk about only one pipeline, which the first pipeline that picks the data set, i.e., Router data factory pipeline. I divided the dynamic process into 2 parts:

I. Main Pipeline or Parent Pipeline

The purpose of this pipeline is to pick the dataset and pass it to another processor pipeline that does all the processing. If for any reason any activity in the processor pipeline fails, the main pipeline will send out appropriate error information returned by the processor pipeline to the authorized recipients via email. Also, the error is logged in the COSMOS DB, which is further connected to Power BI, where a full-fledged dashboard is set up to analyze the errors, and preventive action is taken based on the analysis performed.

Processor pipeline: This pipeline reads the cosmos config, picks up the files from the blob storage container, and processes it accordingly.

III. MAIN/PARENT PIPELINE

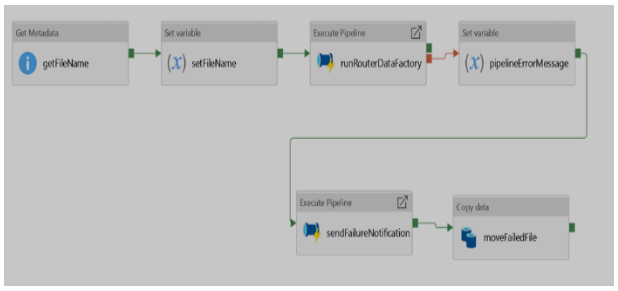
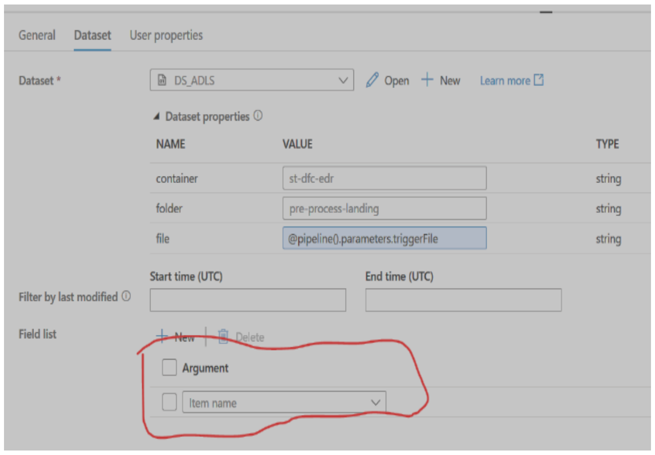


Fig. 2 Architecture

get filename: It's a 'Get Metadata' activity. Notice the Field list configuration selected as 'Item name.' As soon as the blob hits the folder in the blob container, this pipeline is triggered, and the 'Get Metadata' activity picks up the file name from the folder.



General | Dataset | User properties

Dataset *: DS_ADLS — Open — New — Learn more

Dataset properties

NAME	VALUE	TYPE
container	st-dfc-edr	string
folder	pre-process-landing	string
file	@pipeline().parameters.triggerFile	string

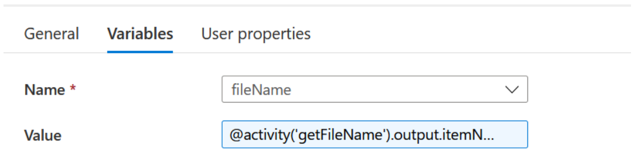
Filter by last modified: Start time (UTC) — End time (UTC)

Field list: New | Delete

Argument

Item name

set filename: This is a 'Set Variable' activity. The name of the file is passed to this variable.

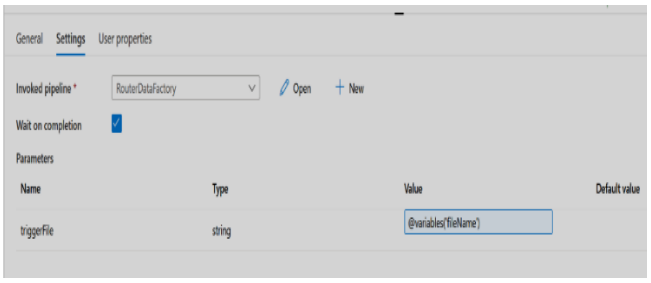


General | Variables | User properties

Name *: fileName

Value: @activity('getFileName').output.itemN...

Run Router Data Factory: This is the 'Execute Pipeline' activity and triggers the processor pipeline.



General | Settings | User properties

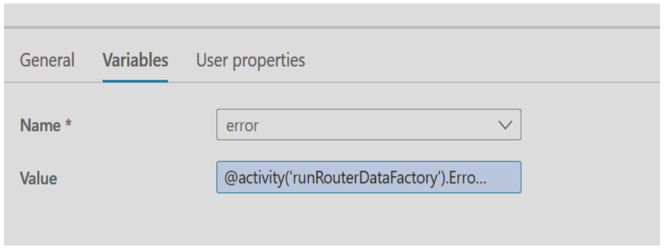
Invoked pipeline *: RouterDataFactory — Open — New

Wait on completion: ✓

Parameters

Name	Type	Value	Default value
triggerFile	string	@variables('fileName')	

Pipeline Error Message: This is a 'Set Variable' activity. Notice that this activity is connected as a failure process of the previous activity. The previous activity runs the processor pipeline, so this means if there is any activity that fails in the processor pipeline, the error returned gets stored in a variable.
@activity('runRouterDataFactory').Error.Message

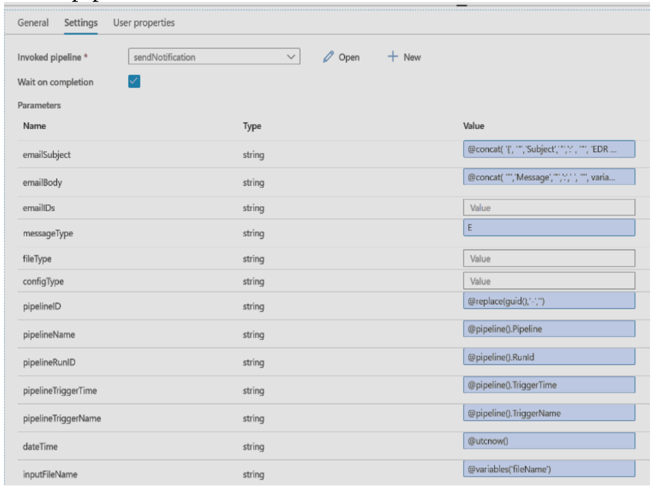


General | Variables | User properties

Name *: error

Value: @activity('runRouterDataFactory').Erro...

Send Failure Notification: This is another 'Execute Pipeline' activity and triggers a separate pipeline to send out an error notification. All the necessary information is passed to the pipeline to send the email notification.


General | Settings | User properties

Invoked pipeline *: sendNotification
Wait on completion: ✓

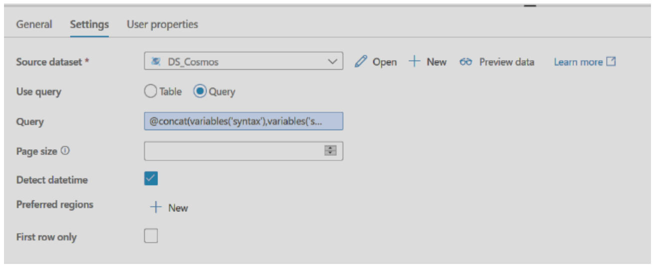
Parameters

Name	Type	Value
emailSubject	string	@concat('{', '"', 'Subject', '"', ':' , '"', 'EDR ...
emailBody	string	@concat('"', 'Message', '"', ':', ' ', '"', varia...
emailIDs	string	Value
messageType	string	E
fileType	string	Value
configType	string	Value
pipelineID	string	@replace(guid(),'-','')
pipelineName	string	@pipeline().Pipeline
pipelineRunID	string	@pipeline().RunId
pipelineTriggerTime	string	@pipeline().TriggerTime
pipelineTriggerName	string	@pipeline().TriggerName
dateTime	string	@utcnow()
inputFileName	string	@variables('fileName')

Move Failed File: This is a 'Copy data' activity. Since there was an error in the processor pipeline, the dataset needs to be re-processed after fixing the error. This activity moves the failed data file from the landing folder to a different 'error' folder, which holds the error data files. They can be fixed and moved back to the landing folder for re-processing.

IV. PROCESSOR PIPELINE

A. Architecture


General | Settings | User properties

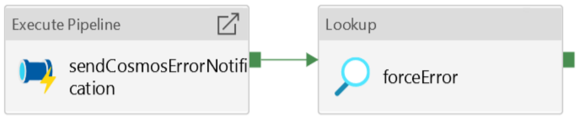
Source dataset *: DS_Cosmos
Use query: Table [✓ Query]
Query: @concat(variables('syntax'),variables('s...
Page size:
Detect datetime: ✓
Preferred regions: + New
First row only: ☐

B. Check Cosmos Configs Returned

This is an 'IF condition' activity. This condition checks if a configuration related to the data set in the process is available in Cosmos DB or not and is read by the Lookup activity.

<u>If 'True'</u>: Do Nothing.

<u>If False:</u> This means config is not available, and it makes no sense to move further in the pipeline. Raise an error message and force pipeline to error.

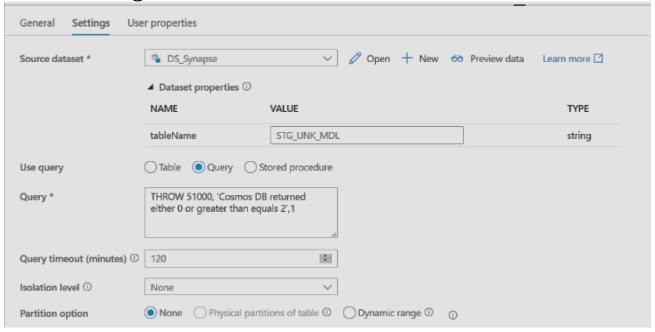


C. Send Cosmos Error Notification

This is an 'Execute Pipeline' activity and triggers Send Notification pipeline with all necessary information required, i.e., email body, subject, and email IDs.

D. Force erro

This is a Lookup activity. But it is a trick to use this activity to throw an error instead of reading any data. Check out the below config.


General | Settings | User properties

Source dataset *: DS_Synapse

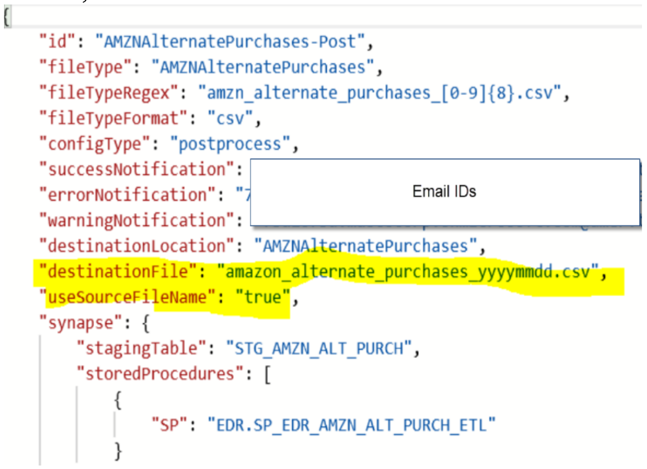
Dataset properties

NAME	VALUE	TYPE
tableName	STG_UNK_MDL	string

Use query: Table [✓ Query] Stored procedure
Query *:
```
THROW 51000, 'Cosmos DB returned
either 0 or greater than equals 2',1
```
Query timeout (minutes): 120
Isolation level: None
Partition option: [✓ None] Physical partitions of table Dynamic range

E. Check Destination File Name

This is an 'If Condition' activity. This is one example where I can show how Cosmos DB config is used within pipelines. There is a placeholder in Cosmos DB that holds the name of the 'Destination' file, which means when the file is processed, which means that after the file is converted from unstructured to structured data, the processed file is dropped into another folder in the storage container and the name of that file will not be similar to the name with the file was dropped in the source folder, but it will be a new name as maintained in the Cosmos config as **"destination file"** below;


```
{
    "id": "AMZNAlternatePurchases-Post",
    "fileType": "AMZNAlternatePurchases",
    "fileTypeRegex": "amzn_alternate_purchases_[0-9]{8}.csv",
    "fileTypeFormat": "csv",
    "configType": "postprocess",
    "successNotification":    Email IDs
    "errorNotification": "
    "warningNotification":
    "destinationLocation": "AMZNAlternatePurchases",
    "destinationFile": "amazon_alternate_purchases_yyyymmdd.csv",
    "useSourceFileName": "true",
    "synapse": {
        "stagingTable": "STG_AMZN_ALT_PURCH",
        "storedProcedures": [
            {
                "SP": "EDR.SP_EDR_AMZN_ALT_PURCH_ETL"
            }
```

Property 'useSourceFileName' has value "true" or "false," which will be looked at in the Data Factory in this 'IF condition.'

If "True":

Copy the file to the destination folder with the file name maintained in the Cosmos DB.

If "False":

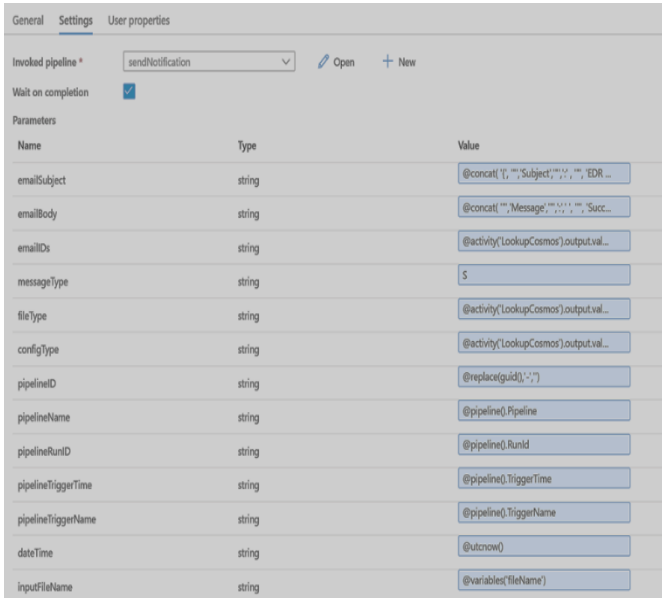
Copy the file to the destination folder with the file name, same as the source file.

F. *Move Completed File*

This is a “Copy” activity. Once the file is processed, it’s a success, and the file is moved to a different folder titled ‘completed.’ You can also attach a timestamp with the file name to keep track of the files processed with time.

G. *Send Success Notification*

This is the ‘Execute pipeline’ activity. Once the file is processed successfully, a success message is sent out. This activity triggers another process chain, ‘send Notification.’



Invoked pipeline * sendNotification

Wait on completion ✓

Parameters

Name	Type	Value
emailSubject	string	@concat('{', '"','Subject','"',':' , '"', 'EDR ...
emailBody	string	@concat('"','Message','"',':' , '"', 'Succ...
emailIDs	string	@activity('LookupCosmos').output.val...
messageType	string	S
fileType	string	@activity('LookupCosmos').output.val...
configType	string	@activity('LookupCosmos').output.val...
pipelineID	string	@replace(guid(),'-','')
pipelineName	string	@pipeline().Pipeline
pipelineRunID	string	@pipeline().RunId
pipelineTriggerTime	string	@pipeline().TriggerTime
pipelineTriggerName	string	@pipeline().TriggerName
dateTime	string	@utcnow()
inputFileName	string	@variables('fileName')

Can find trends in the data provided and able to come up with state-of-the-art dashboards and reports with just a click of a button. Data security provided by SAP Analytics Cloud is incomparable to any other tool available as of today in this space. The cloud connector and cloud agent config keep the data always secure via HTTPS protocol and do not let any hacking take place. The planning feature of SAP Analytics Cloud makes it possible to plan for the near future or predictive planning for long-term business decisions.

V. CONCLUSION

Azure Data Factory is a very robust data integration and ingestion platform that can process terabytes to petabytes of data stored in the storage containers or data in transit via outside sources. With the right design, data ingestion to the Azure platform can be made dynamic, easy to maintain, and easy to fix. Hope this article is helpful in designing efficient data factory pipelines that will save a lot of time and effort.

VI. REFERENCES

[1] https://docs.microsoft.com/en-us/azure/data-factory/#:~:text=Azure%20Data%20Factory%20is%20Azure's,with%20full%20compatibility%20in%20ADF.

[2] https://docs.microsoft.com/en-us/azure/data-factory/introduction.