Korn Ferry Data Lake

Ver 1.0

June 2021

Contents

[1. Introduction 4](#_Toc75189275)

[1.1. Purpose 4](#_Toc75189276)

[1.2. Scope 4](#_Toc75189277)

[1.3. Out of Scope 4](#_Toc75189278)

[2. Target State Architecture 4](#_Toc75189279)

[2.1. Key Design Considerations 4](#_Toc75189280)

[2.2. Conceptual Architecture 4](#_Toc75189281)

[2.3. Data Flow Diagram 4](#_Toc75189282)

[3. KF Framework 4](#_Toc75189283)

[3.1. Data Ingestion Framework 4](#_Toc75189284)

[3.2. Key components 4](#_Toc75189285)

[3.3. ER Diagram 4](#_Toc75189286)

[3.4. Process Flow 4](#_Toc75189287)

[3.5. Data Validation 4](#_Toc75189289)

[3.6. Modularized Functions 4](#_Toc75189290)

[3.7. Restart ability 4](#_Toc75189291)

[3.8. Retry 4](#_Toc75189292)

[3.9. Logging Mechanism 4](#_Toc75189293)

[3.10. Alert and Notification 4](#_Toc75189294)

[3.11. Exception Handling 4](#_Toc75189295)

[4. Source System 4](#_Toc75189296)

[5. Data Loading Strategy 4](#_Toc75189297)

[5.1. Full Load 4](#_Toc75189298)

[5.1.1. ADF Job name: 4](#_Toc75189299)

[5.1.2. Command to run: 4](#_Toc75189300)

[5.1.3. Steps for Full Load: 4](#_Toc75189301)

[5.2. Delta Load (Incremental load) 4](#_Toc75189302)

[5.2.1. JOB NAME: 4](#_Toc75189303)

[5.2.2. Command to run: 4](#_Toc75189304)

[5.2.3. Steps For Delta Load: 4](#_Toc75189305)

[6. Naming & Coding Standards 5](#_Toc75189306)

[6.1. Coding Standards 5](#_Toc75189307)

[6.2. Table Names 5](#_Toc75189308)

[6.2.1. Data Ingestion Framework Tables 5](#_Toc75189309)

[6.2.2. Data Lake Layer Tables 5](#_Toc75189310)

[6.2.3. Delta Layer Tables 5](#_Toc75189311)

[6.3. Variable Names 5](#_Toc75189312)

[6.4. Function Names 5](#_Toc75189313)

[6.5. Script Names 5](#_Toc75189314)

[7. Retention Strategy 5](#_Toc75189315)

[8. Capacity Plan 5](#_Toc75189316)

[8.1. Hardware 5](#_Toc75189317)

[8.2. Software 5](#_Toc75189318)

[9. Git hub Process 5](#_Toc75189319)

[10. Job Orchestration (ADF) 5](#_Toc75189320)

[11. Spark Process 5](#_Toc75189321)

[11.1. Requirement 5](#_Toc75189322)

[11.2. Process Flow 5](#_Toc75189323)

[11.3. Data Used 5](#_Toc75189324)

[11.4. Step by step Solution 5](#_Toc75189325)

[12. Go Live Process (Dev-Ops) 5](#_Toc75189326)

[13. Run Book 5](#_Toc75189327)

[14. Maintenance 5](#_Toc75189328)

[15. Data Governance 5](#_Toc75189329)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Version No.*** | ***Date*** | ***Description*** | ***Author*** | ***Reviewed By*** |
| 1.0 | 21/06/2021 | Initial document | Prashant Rajguru |  |

# Introduction

## Purpose

The purpose of this project is to extract the SQL based data from DataWarehouse and UAM managed SQL Database Instances and load the data into delta lake tables on top of Azure data lake storage gen 2 so that it can be used for further processing.

## Scope

Scope includes below points -

* Extraction of SQL based data via ADF pipeline to azure data lake gen 2 storage in csv format.
* Loading the processed data into delta lake tables to be stored on top of the azure data lake storage gen 2.
* Monitoring of pipeline
* Logging

## Out of Scope

# Target State Architecture

## Key Design Considerations

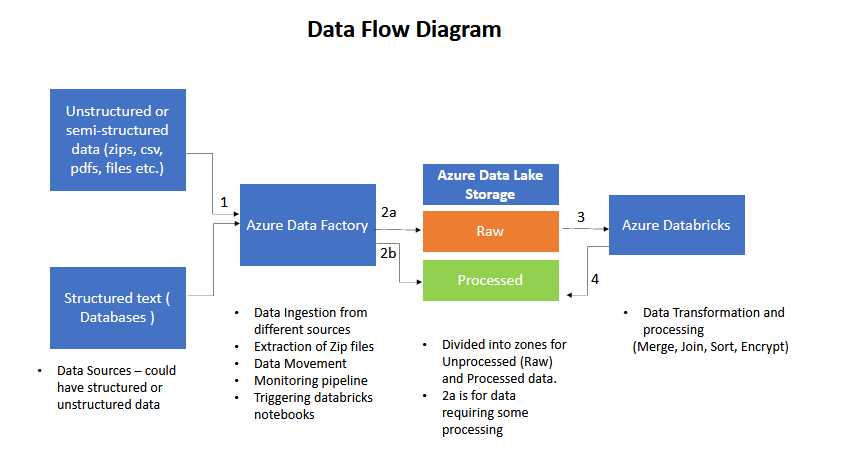
* Full and Incremental zip data extraction – to allow flexibility to extract and copy only newly obtained zip files or to extract all zip files.
* Separate loading of each table presents in Datawarehouse and UAM Databased into separate delta table in the azure data lake storage.

## Conceptual Architecture

## Data Flow Diagram

## A high-level architecture with data flow diagram is provided below. In the case of DW and UAM data, the source data is present in Managed SQL database instances which is fed into the Azure Data Factory after configuring linked services to connect to and Managed SQl instances (via Self-hosted Integration runtime).

## It is loaded into ADLS gen2 via ADF into a staging directory. Databricks notebook is later run on the extracted Jsons and it creates delta tables batchwise and saves them to ADLS Gen 2 Storage account.



# KF Framework

## Data Ingestion Framework

ADF pipeline is used for data ingestion and extraction of SQL table data from Managed SQL instances to azure data lake storage gen 2. Databricks is later used for loading the data into delta tables.

## Key components

ADF Pipeline-

## ER Diagram

## Process Flow

## Data Validation

## Modularized Functions

## Restart ability

## Retry

## Logging Mechanism

## Alert and Notification

## Exception Handling

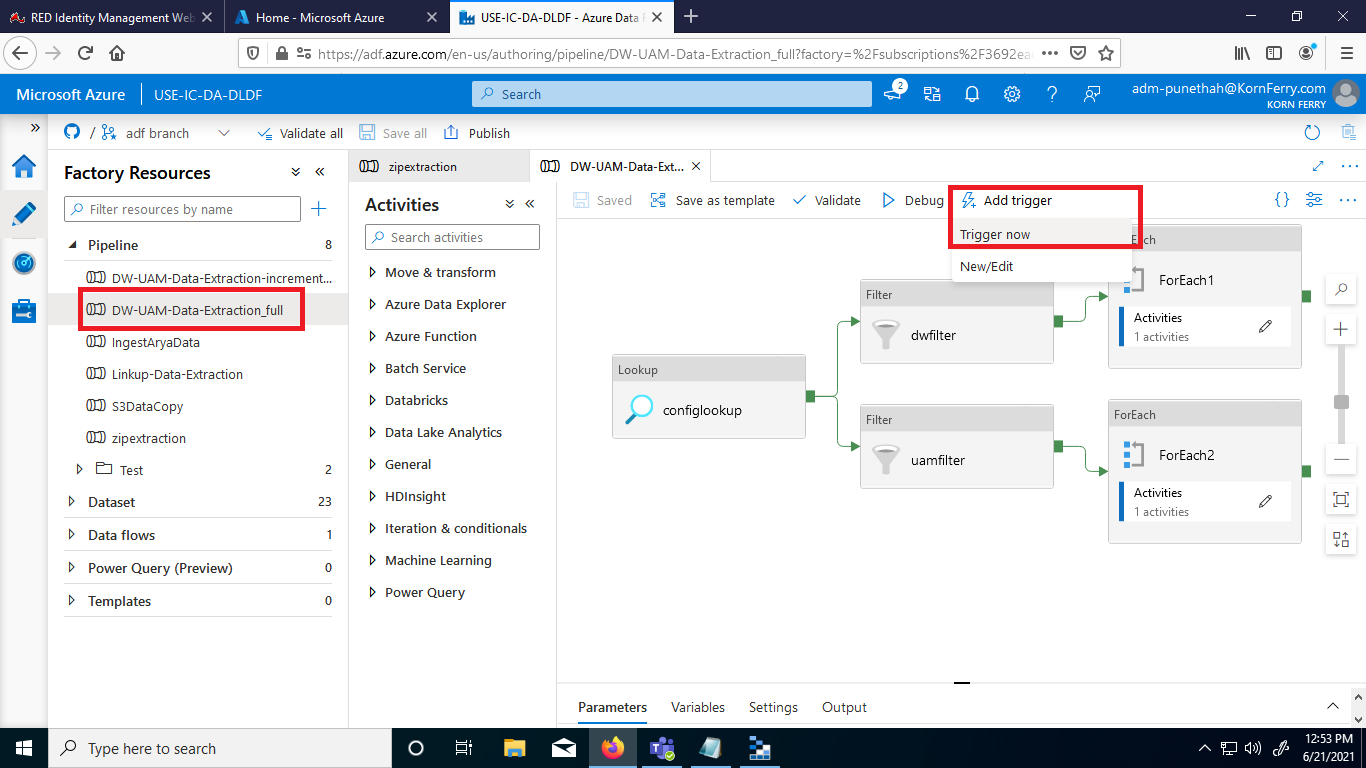
# Source System

# Data Loading Strategy

## Full Load

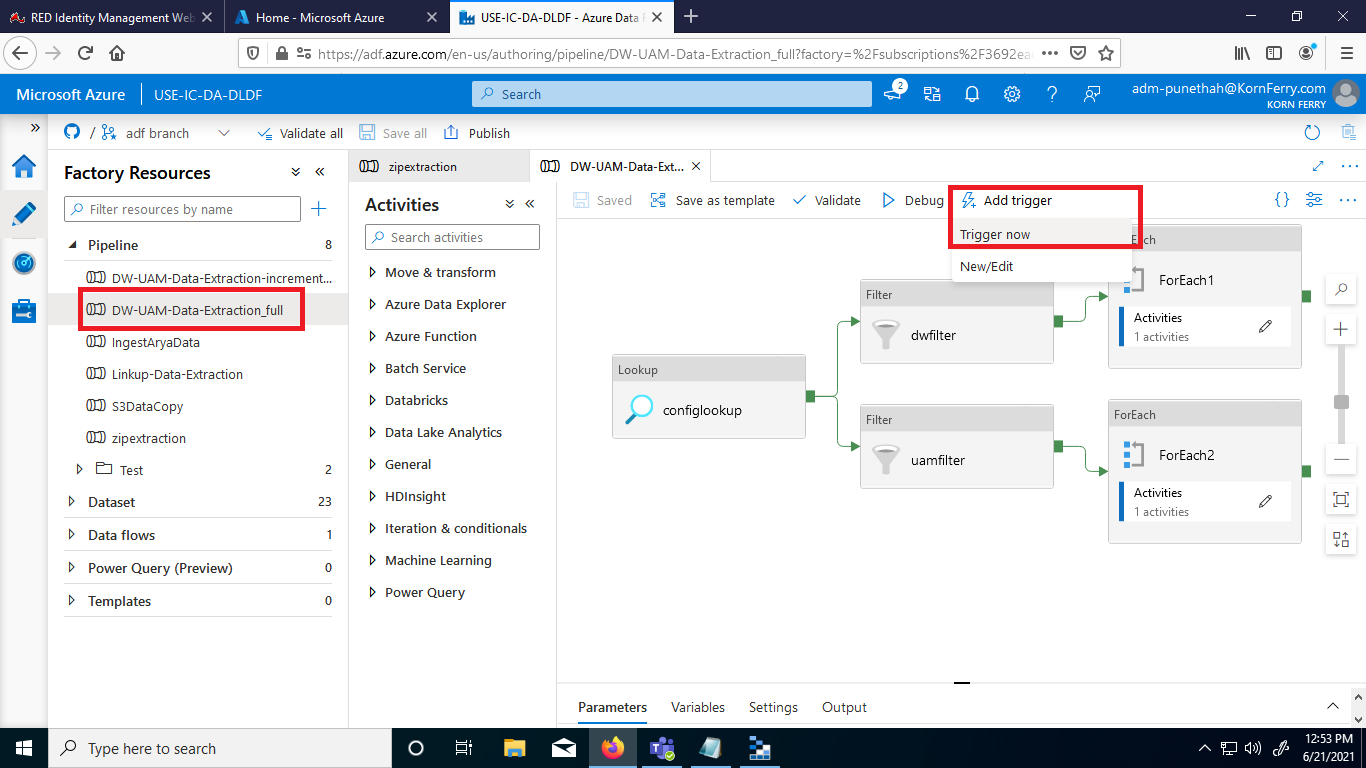
## ADF Job name: DW-UAM-Data-Extraction\_full

## Command to run:

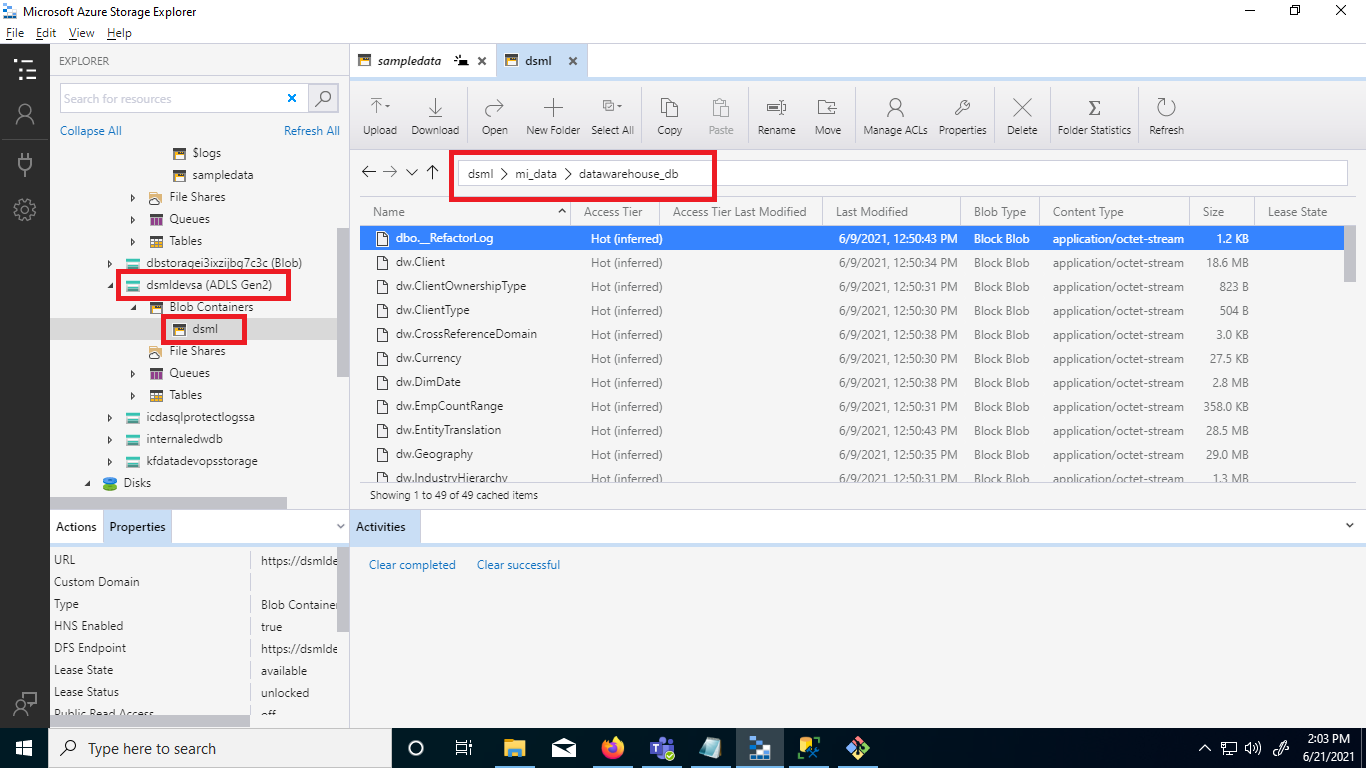


## Steps for Full Load:

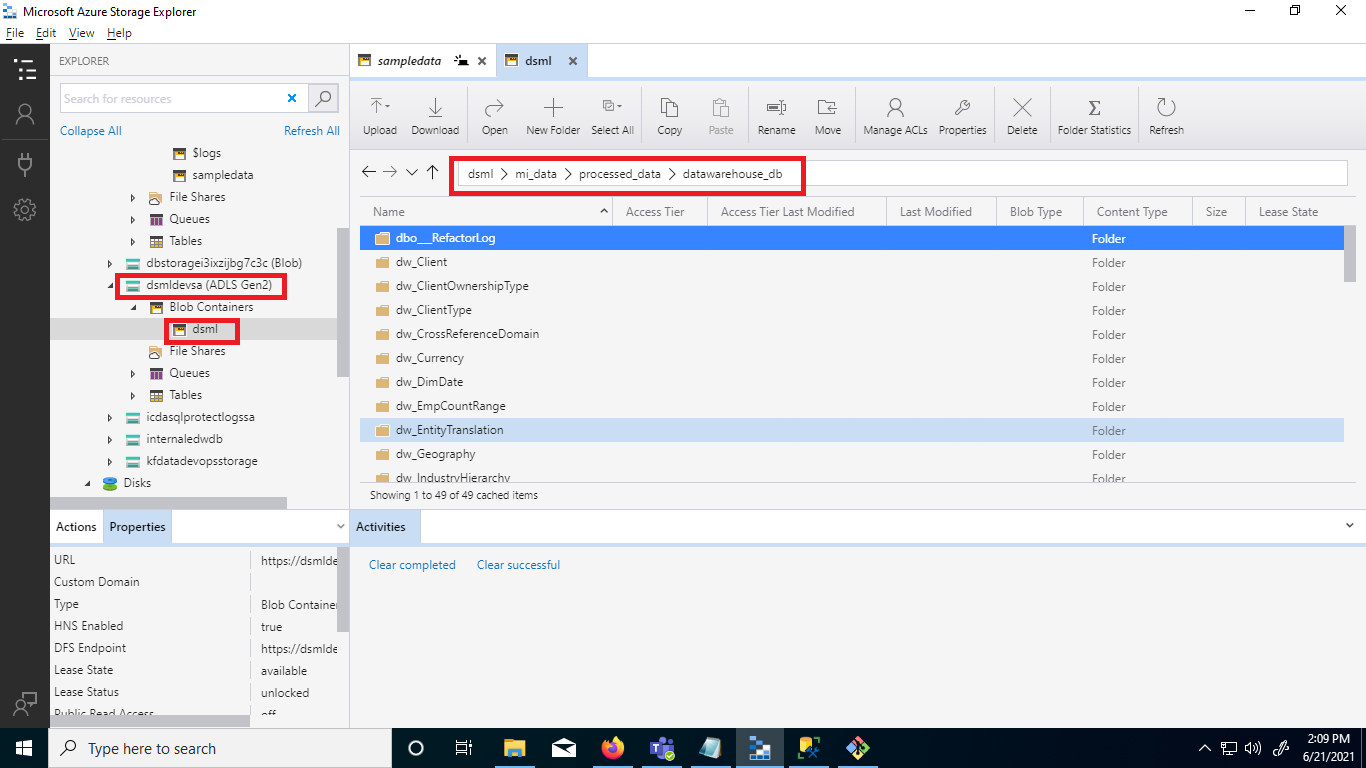
* Run the DW-UAM-Data\_extraction full data extraction pipeline from Azure data factory (adf.azure.com )



* Pipeline processes all the tables specified in config table with Enabled flag set to true
* The files are copied to the folder mi\_data/datawarehouse\_db for DataWarehouse database data and to mi\_data/uam\_db for UAM database data. One file is created for each table in the sql. Files exported are in csv format. (Copying full data takes approximately 8 hours if run for all tables)



* Once run, the databricks script to load the csv files into delta tables can be run which will import these tables into delta table format. In ADLS the delta tables are saved to the below directory for datawarehouse\_db (similar path conventions followed for UAM)



## Delta Load (Incremental load)

## JOB NAME:

## The job we using **Workspace/uam\_datawarehouse/read\_csv.**

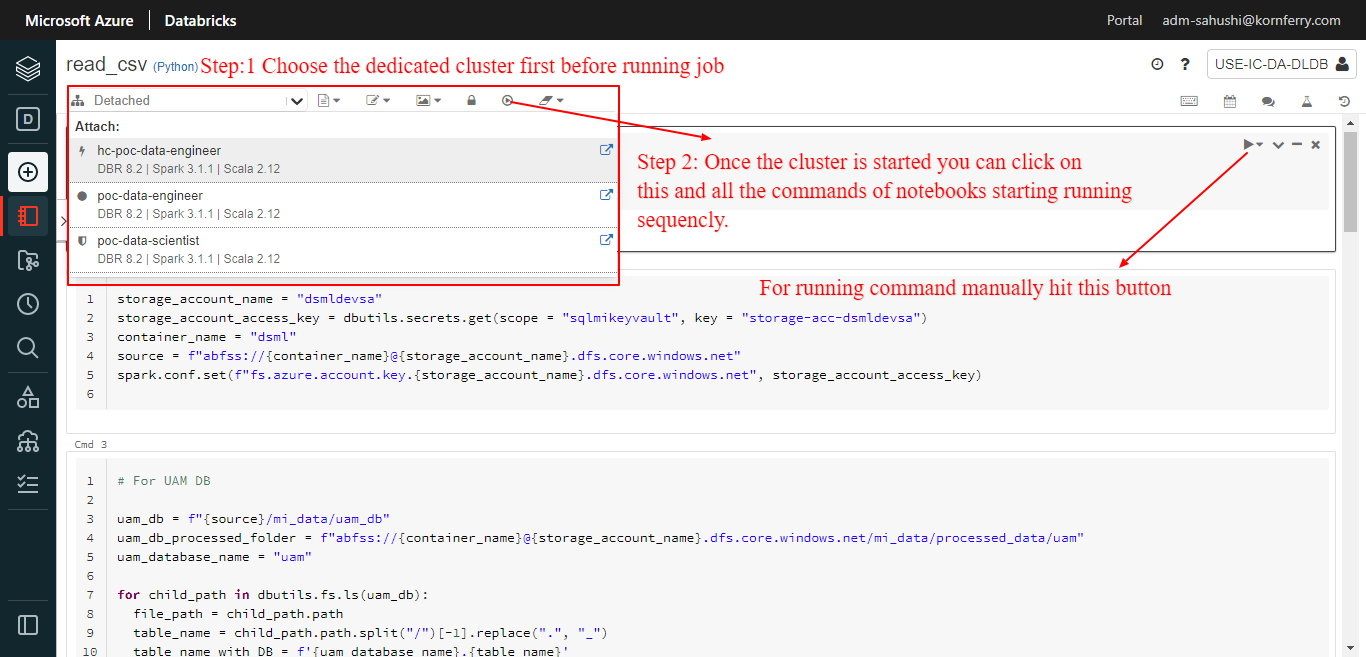
Github Link -[**File Link**](https://github.com/HayGroup/mldatalake/blob/databricks/source/notebooks/uam_datawarehouse/read_csv.py)

## Command to run: Step 1:

## 1 - Check cluster is running or not if not run the cluster

2 - Hit the run button as I highlighted in the below picture.

## Steps For Delta Load:



# Naming & Coding Standards

## Coding Standards

1 – Auto Pep8

2 – Boiler Plate code.

## Table Names

## Data Ingestion Framework Tables

## Data Lake Layer Tables: Mantling in DBFS storage.

## Delta Layer Database Jobs\_Full & Jobs\_Diff

## Variable Names

## Function Names

## Script Names

# Retention Strategy

# Capacity Plan

## Hardware

## Software - Azure Data Factory, Azure Databricks

# Git hub Process

Git can be connected to ADF using the below configurations as shown below-

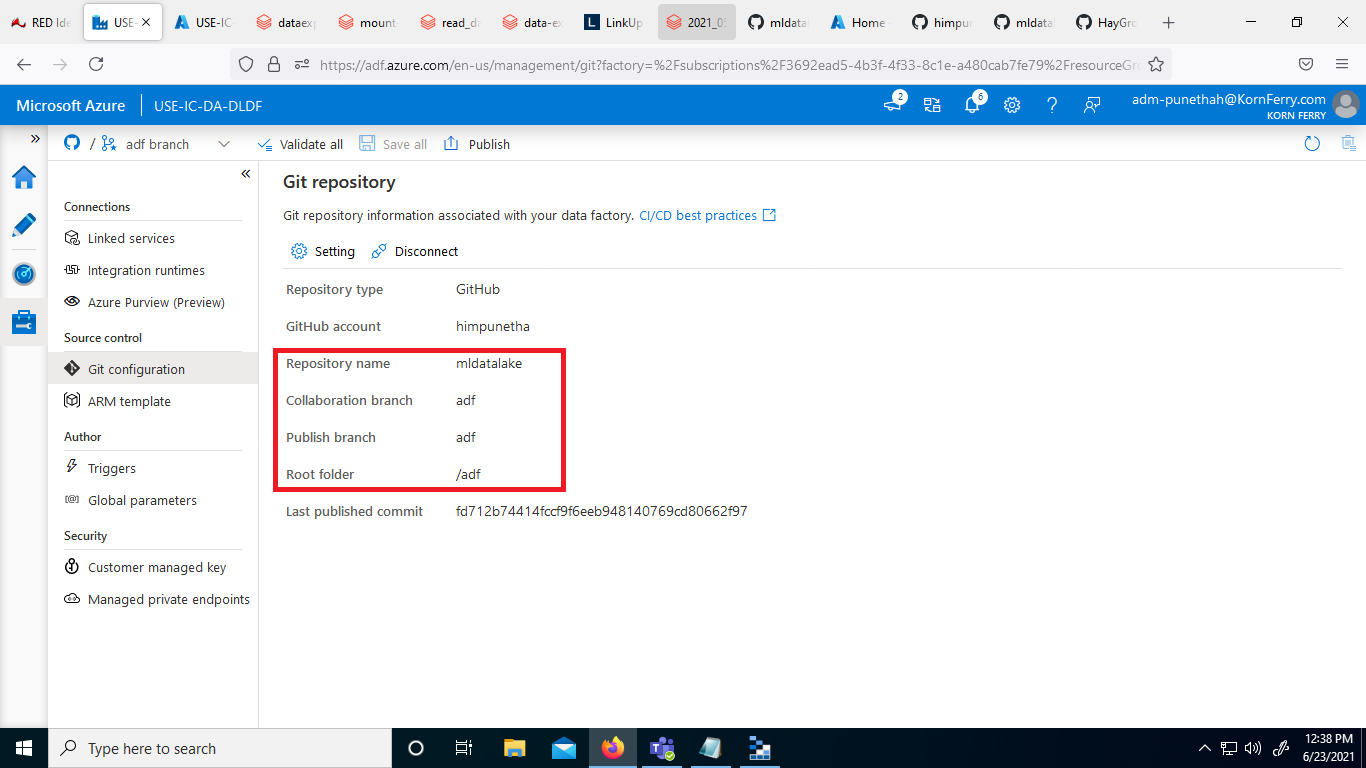
Repository name – mldatalake (<https://github.com/HayGroup/mldatalake/tree/main>)

Collaboration branch – adf

Publish branch – adf

Root Folder - /adf

The new ADF where the code is to be deployed must be created with same name as USE-IC-DA-DLDF . The repository would be needed to be forked in the beginning and then adf can be connected to github with the same configuration as provided above. Select Import existing resources to repository as yes. After connecting, ADF will read the publish\_config.json file present in the adf branch of the repo and then it will import resources. Refresh the adf page once and check if the resources are imported. For more details , check - <https://docs.microsoft.com/en-us/azure/data-factory/source-control>



# Job Orchestration (ADF)

ADF Config table for pipeline – A config table is used for specifying the tables whose data needs to be extracted. This table is stored in Managed SQL instance inside database DataLake. The details on setting up the config table are defined below.

One time set-up of config table in Database

* Create config table – Config table for adf can be created by running the below command for the database where it needs to be created.

Github link - <https://github.com/HayGroup/mldatalake/blob/main/Documentation/createadfconfig.sql>

Create Schema datalake

CREATE TABLE datalake.adfconf(

[Source] [nvarchar](100) NULL,

[Source\_table] [nvarchar](100) NULL,

[Dest\_table] [nvarchar](100) NOT NULL,

[Columns] [nvarchar](100) NOT NULL,

[Watermark\_Column] [varchar](100) NULL,

[Watermark\_Value] [datetime] NULL,

[Enabled] [int] NOT NULL,

[Load\_Flag] [nvarchar](100) NULL,

[Status] [nvarchar](50) NULL,

[Comment] [varchar](255) NULL

)

* Insert a unique record for each table which needs to be copied and exported. Example insertion query is provided below.

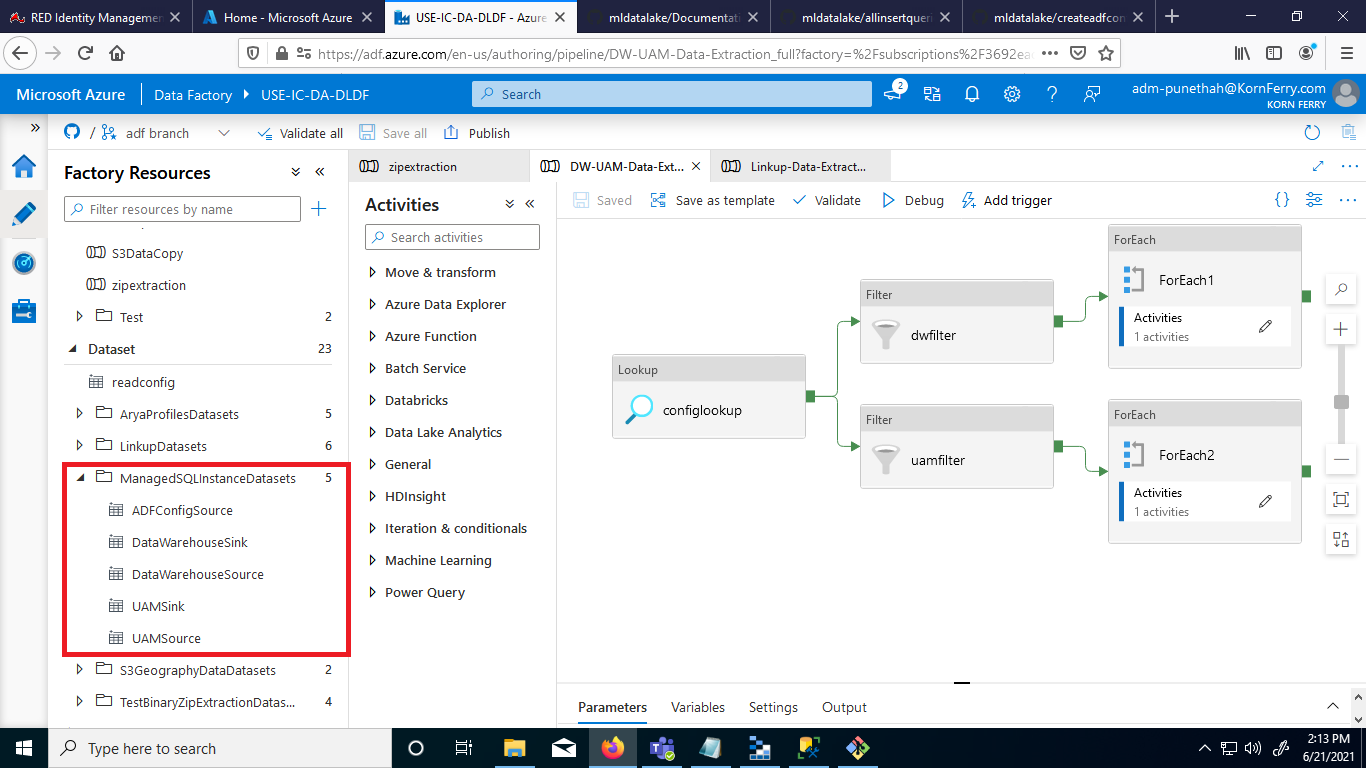
Insert into datalake.adfconf (Source, Source\_table, Dest\_table, Columns, Watermark\_Column, Watermark\_Value, Enabled, Load\_Flag) values ('sqldw','dw.Client','dw.Client','\*','UpdateDate','1900-01-01 00:00:00',1,'Full')

Github link of sql file containing insert queries (only for copying full data)- <https://github.com/HayGroup/mldatalake/blob/main/Documentation/allinsertqueries-fulldata.sql>

* In the config table if Enabled flag is set to false then that table will not be copied by adf pipeline.

**ADF - Pipeline Data Sources**

* ADFConfigSource – Dataset referring to Managed SQL Database where ADF config table is present.
* DataWarehouseSource – Source dataset is Managed SQL Database DataWarehouse
* DataWarehouseSink - Sink dataset refers to location in ADLS gen 2 storage where tables are to be exported.
* UAMSource - Source dataset is Managed SQL Database UAM
* UAMSink - Sink dataset refers to location in ADLS gen 2 storage where tables are to be exported.



**ADF - Pipeline Activities**

There are 7 activities in the pipeline out of which 4 activities run for UAM tables and 4 activities run for Datawarehouse tables.

* + Configlookup – This activity performs a lookup on the config table which is created for orchestration purposes. It gets the entries added in the config table and passes only those table entries further for which Enabled flag is set to true.
  + DWFilter – Used for filtering the table entries which are present in DataWarehouse database and only allowing those table entries to pass further to loop
  + ForEach (following DWFilter) – This activity is run for all the entries which were obtained after DWFilter activity
  + If Activity inside ForEach (DWFilter) - If activity checks if incremental flag is set to true or false. For full data copy it runs the false portion of if activity.

# Spark Process

## Requirement

1 - Microsoft Azure Subscriptions

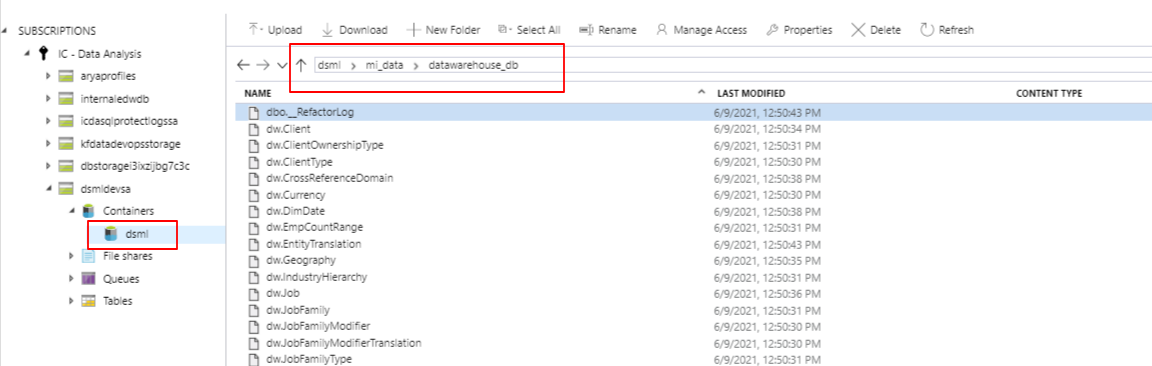
2 - Azure Databricks Instance/Subscription

3 - Databricks Cluster with Scala/SQL/Python language configured.

4 - Notebook that contain multiple commands in sequence for data pipeline to perform ETL

## Process Flow

ADLS Gen2 Storage Location



GitHub Link: <https://github.com/HayGroup/mldatalake/tree/databricks/source/notebooks/uam_datawarehouse>

Step 1: Initialize all the required variables in the top of the notebooks so you can use them in the entire notebook

Step 2: Now navigate using dbutils within ADLs GEN2 instance container to the desired files.



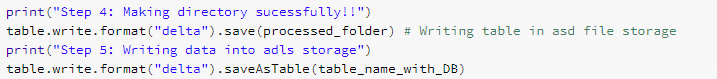
Step 3: Now read that file according to their format in spark data frame.



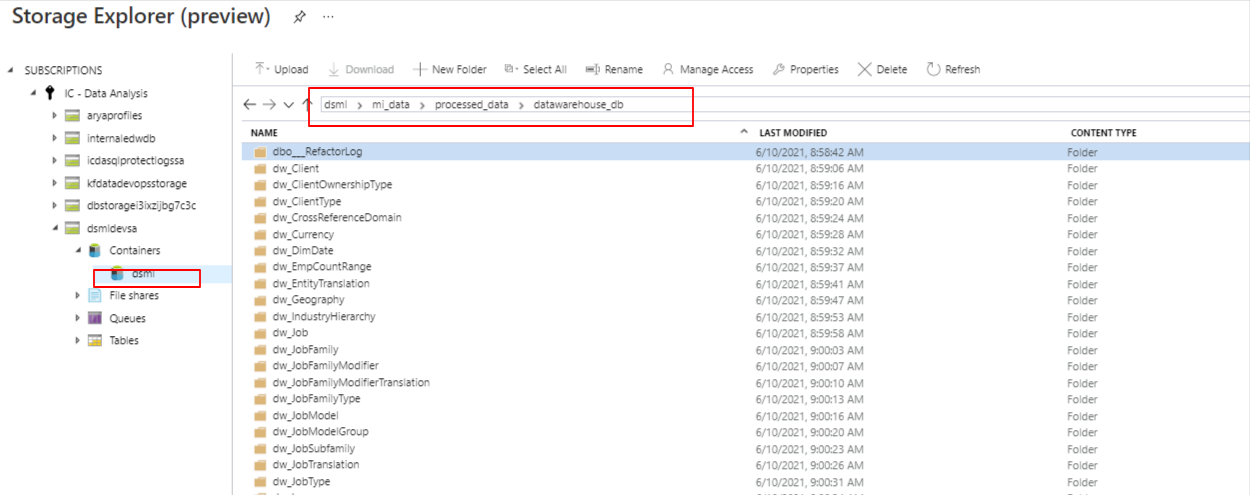
Step 4: Now make temp table in memory form that data frame to check that is compatible to write in delta table.



Step 5 : Once the step 4 is verified now write that temporary table in DBFS in delta format.



All the Delta tables are dumped in the ADLS Gen2



## Data Used

1 – We can read that data in pyspark in data frame format

2 – We can use that in SQL format also

3 – That Data strictly maintains the ACID properties

4 – We can use in various place for analysis or for model traning, The whole fetch on the tip of the finger

## Step by step Solution

# Go Live Process (Dev-Ops)

# Run Book

# Maintenance

# Data Governance