VMO2/PRODAPT CONFIDENTIAL

On-perm DataConnector Build Document

Table of Contents

[1. DOCUMENT HISTORY, VERSION, AND AUTHORS 3](#_Toc158193211)

[1.1. DOCUMENT VERSION AND AUTHORS 3](#_Toc158193212)

[1.2. VMO2 REVIEW/APPROVALS 3](#_Toc158193213)

[1.3. ABBREVIATIONS, TERMS AND DEFINITIONS ABBREVIATIONS 3](#_Toc158193214)

[2. DESIGN OBJECTIVES AND SCOPE 4](#_Toc158193215)

[2.1. Design LLD 4](#_Toc158193216)

[2.2. IN SCOPE 5](#_Toc158193217)

[2.3. OUT OF SCOPE AND EXCLUSIONS 5](#_Toc158193218)

[3. Dag DeVELOPMENT 6](#_Toc158193236)

[3.1. Full Load 6](#_Toc158193237)

[3.2. CDC Forward 7](#_Toc158193238)

[3.3. CDC Reverse 1](#_Toc158193239)0

[3.4. Fallout Report 1](#_Toc158193240)1

[3.5. Summary Report 1](#_Toc158193241)2

[4. PLSQL DevelopemEnt 1](#_Toc158193242)3

[4.1. Full Load 1](#_Toc158193243)3

[4.2. CDC Forward 1](#_Toc158193244)3

[4.3. CDC Reverse 1](#_Toc158193245)4

[5. Inventory details 1](#_Toc158193246)6

[5.1. GCOMM INVENTOR DETAIL 1](#_Toc158193247)6

[5.2. GCOMM DB Version details 1](#_Toc158193247)6

[6. MonitoRing 1](#_Toc158193250)7

[6.1. CELERY Flower 1](#_Toc158193251)7

# DOCUMENT HISTORY, VERSION, AND AUTHORS

Heading Text goes here

Paper copies are valid only on the day they are printed. Refer to the author of the document to confirm the currency of this document.

## DOCUMENT VERSION AND AUTHORS

| **Date** | **Version** | **Name** | **Role** | **Comments** |
| --- | --- | --- | --- | --- |
| **12/01/2024** | 1.0.0 | Deivendran Jeyagopi |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## VMO2 REVIEW/APPROVALS

| **Approval Date** | **Version** | **Name** | **Role** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

## ABBREVIATIONS, TERMS AND DEFINITIONS ABBREVIATIONS

Define project related abbreviations.

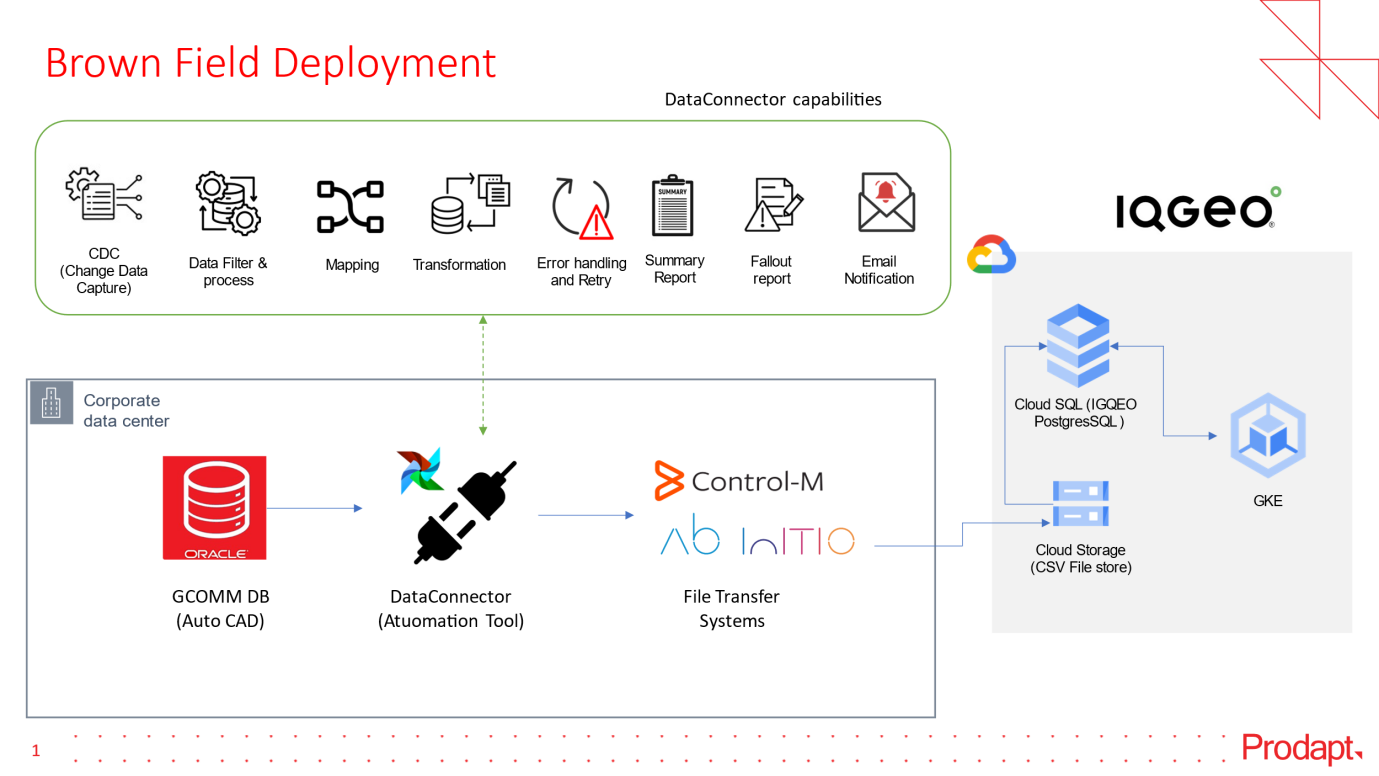
| **Abbreviation** | **Definition** |
| --- | --- |
| **DAG** | Directed Acyclic Graph |
| **FS** | File System |
|  |  |
|  |  |
|  |  |
|  |  |

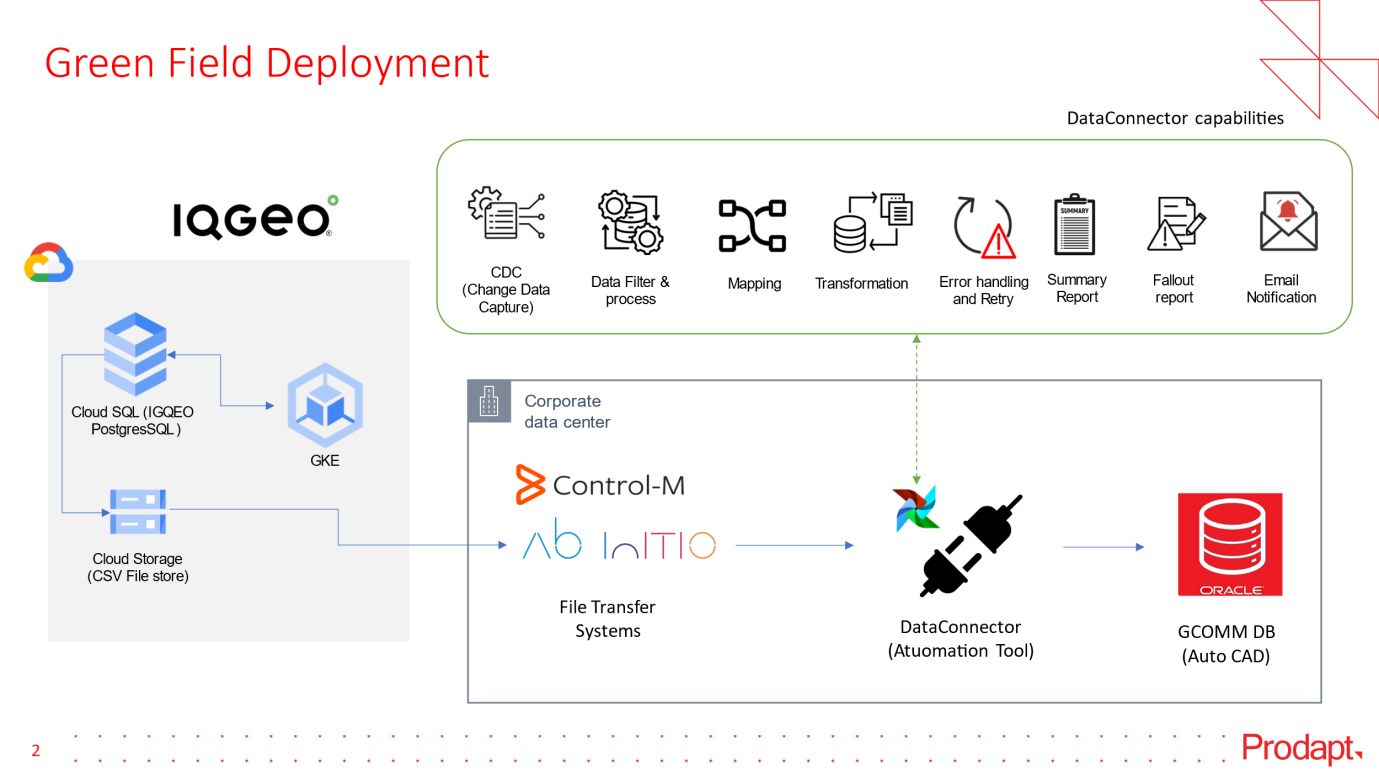
# DESIGN OBJECTIVES AND SCOPE

To retrieve data from the GCOMM database and post the data into IQGEO and vice versa, Airflow must be configured, built, and a DAG developed to process, transform, and transfer the data to the respective NAS storage.

* **Directed Acyclic Graph (DAG):** In Airflow, workflows are defined as DAGs. A DAG is a collection of tasks with directional dependencies between them. Each node in the graph represents a task, and edges represent dependencies between tasks. DAGs are defined using Python scripts.
* **Operators:** Tasks within a DAG are implemented using operators. Operators define the work to be done. Airflow provides a wide range of built-in operators for common tasks like BashOperator, PythonOperator, etc. You can also create custom operators if needed.
* **DAG Definition:** A DAG is defined in a Python script using the DAG class provided by Airflow. Within this definition, you specify the tasks and their dependencies.
* **Scheduling:** Once a DAG is defined, Airflow's scheduler schedules the tasks within the DAG based on their dependencies and the specified schedule.
* **Execution:** Tasks within a DAG can be executed based on various triggers like time-based schedules, external triggers, etc

## Design LLD





## IN SCOPE

* Airflow installation by using python pip
* Airflow configuration as per On-perm DB and GCOMM Data load.
* Development DAG for Full Load, CDC forward & Reverse
* GCOMM DB integration from Airflow server
* Testing connectivity all interface (GCOMM DB, Control, Abinitio and NAS)

## OUT OF SCOPE AND EXCLUSIONS

* Airflow Linux and DB build, configuration and maintenance
* Network configuration.
* FS maintenance

# Dag DEVELOPMENT

A Directed Acyclic Graph (DAG) is a type of graph data structure in which edges have a direction, and no cycles (loops) are allowed. Each edge connects one vertex (node) to another, indicating a directional relationship between them.

## Full Load

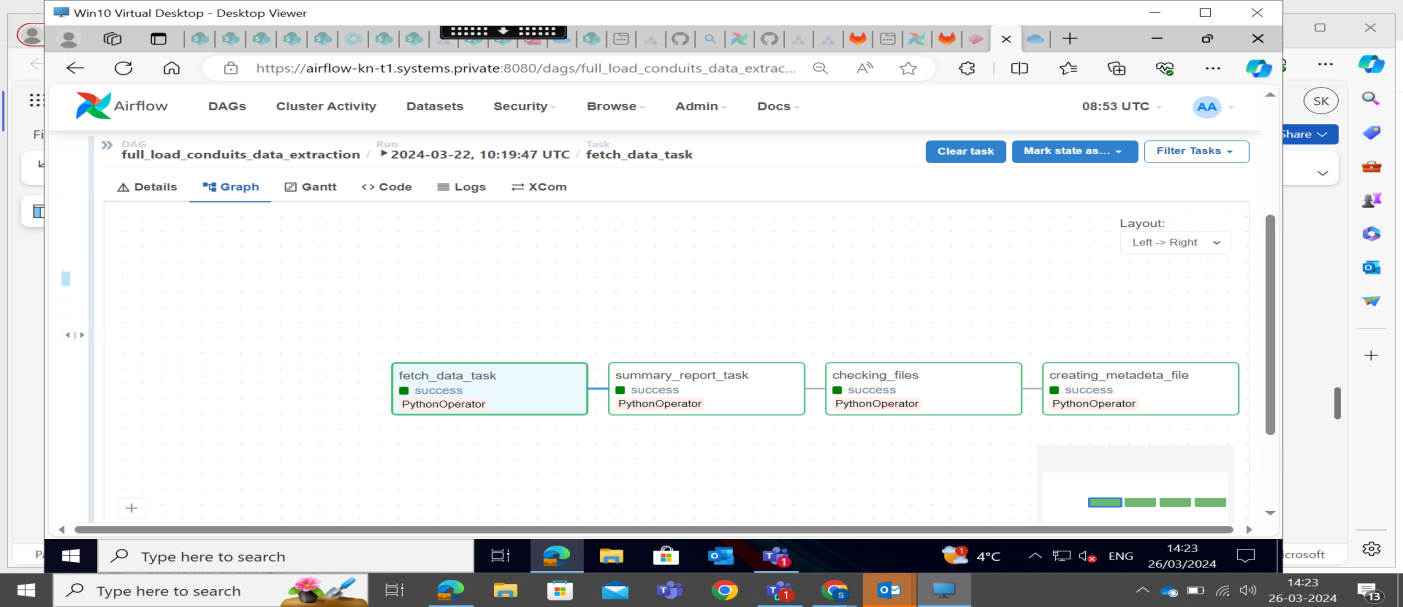
In Full load Forward Connector, we are getting the data from GCOMM databases. Airflow Dag extract the data from the GCOMM tables and load it to the CSV file. CSV files along with the metadata file which contains the vmx\_version,got zipped into the output folder, which is taken up by the Control M team, first it loads the data to NAS Storage and then it runs the script to transfer the file to GCP bucket, from where it is loading to IQGEO tables. Following are the tasks present in the full load dags

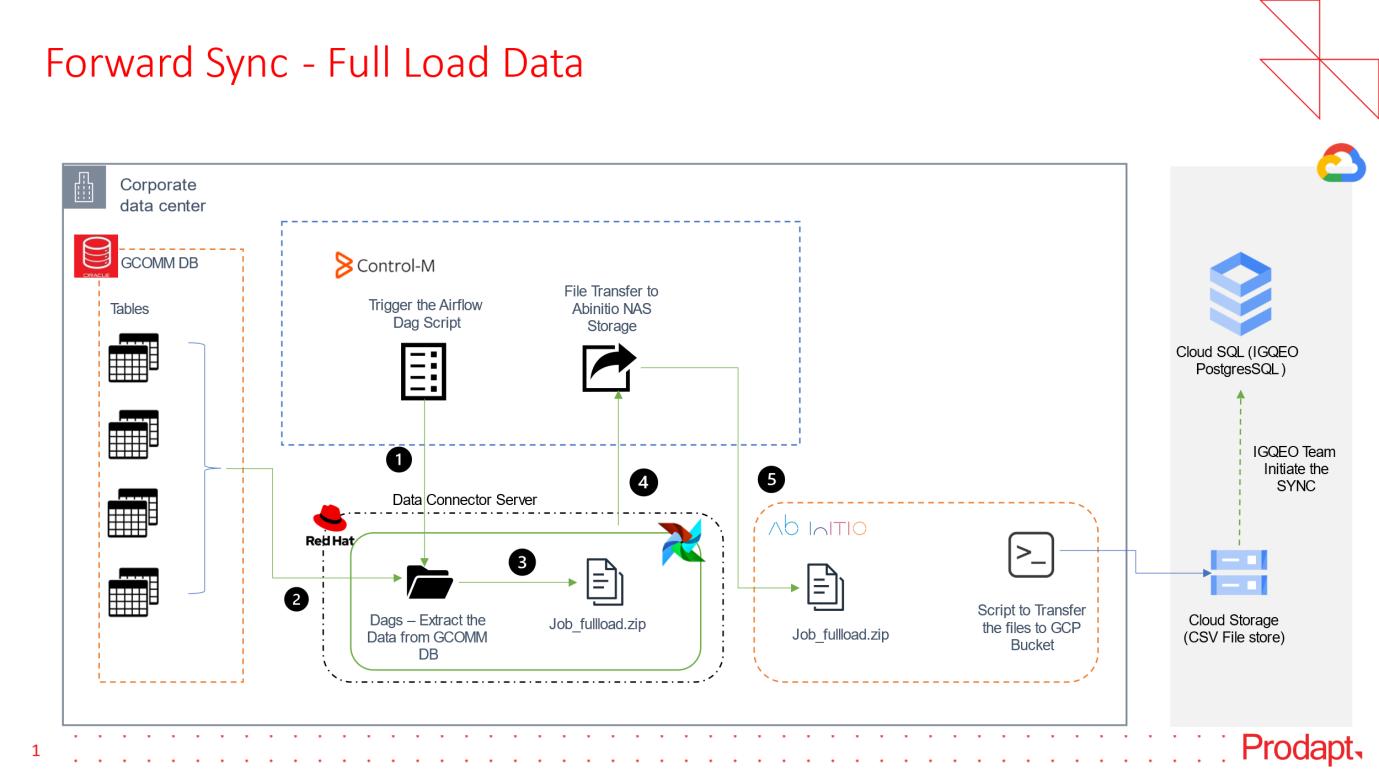
Fetch\_data\_task -> In this task we first connect to the GCOMM database and extract the data from the tables and load it to csv with a chunk size of 1000000 records.

Summary\_report\_task -> In this task we have created a summary file for the object that includes the details like feature\_name, no\_of\_csv\_files\_extracted, total\_data\_extraction, total\_time

Checking\_empty\_files -> This task will create the empty csv file with the header only if there is no record present at the source table for the object.

Creating\_metadata\_file -> This task creates the metadata file that contains details like vmx\_version, design, owner, owner\_email.





## CDC Forward

In CDC Forward Connector, whenever a DML(Insert,update,Delete) happened in the GCOMMS appplication that will be captured in the MODIFICATION\_LOG table in the Database. Airflow Dag will trigger the procedure PRC\_CDC\_PROCESS to extract the data and load it to the CSV file. CSV files along with metadata file which contains the vmx\_version, got zipped into the output folder with name of job\_cdc along with the current\_datetime(job\_cdc\_{current\_datetime}) and from there it is taken up by Control M team to load it to NAS storage, and then it runs the script to transfer the data to GCP bucket from where it is loaded to IQGEO tables. Below are the details of the task present in CDC forward dag

**store\_reverse\_dag\_state\_task**-In this task we are putting the checkpoint to find the present dag run state of reverse dag, if the dag state is running then we are skipping all the tasks further, if not then all the other tasks are triggered according to the task dependencies order.

**fetch\_query\_for\_all\_remaining\_features**- In this task we will trigger the procedure "PKG\_DATA\_CONNECTOR.PRC\_CDC\_PROCESS" and store the output query in v\_query\_value variable and will return the boolean of v\_query\_value, bool(v\_query\_value)

**fetch\_query\_for\_fiber\_task** - In this task we will trigger the procedure "PKG\_DATA\_CONNECTOR.PRC\_CDC\_PROCESS\_FIBER" and store the output query in v\_query\_value2 variable and will return the boolean of v\_query\_value bool(v\_query\_value2)

**fetch\_query\_for\_connections\_task**- In this task we will trigger the procedure "PKG\_DATA\_CONNECTOR.PRC\_CDC\_PROCESS\_CONNECTIONS" and store the output query in v\_query\_value3 variable and will return the boolean of v\_query\_value bool(v\_query\_value3)

**data\_validation\_task**- In this task we will vaildate whether data is present or not, If the all three above task returns none then this task also returns none which means there is no CDC data and all downstream tasks are skipped, If any one of the above taks returns true which means CDC data is present all the downstream tasks will be runned in the sequential order according to the task dependency order.

**data\_extract\_for\_all\_remaining\_features\_task**- In this task we will execute the query which is returned by fetch\_query\_for\_all\_remaining\_features task and write data into csv file, agin we will trigger the "PKG\_DATA\_CONNECTOR.PRC\_CDC\_PROCESS" procedure and write the data into csv file file, we will trigger the procedure again again untill the query is none.

**data\_extract\_for\_fiber\_task**-In this task we will execute the query which is returned by fetch\_query\_for\_fiber\_task task and write data into csv file, agin we will trigger the "PKG\_DATA\_CONNECTOR.PRC\_CDC\_PROCESS\_FIBER" and write the data into csv file, we will trigger the procedure again again untill the query is none.

**data\_extract\_for\_connections\_task**-In this task we will execute the query which is returned by fetch\_query\_for\_connections\_task task and writedata into csv file, agin we will trigger the "PKG\_DATA\_CONNECTOR.PRC\_CDC\_PROCESS\_CONNECTIONS" and write the data into csv file, we will trigger the procedure again again untill the query is none.

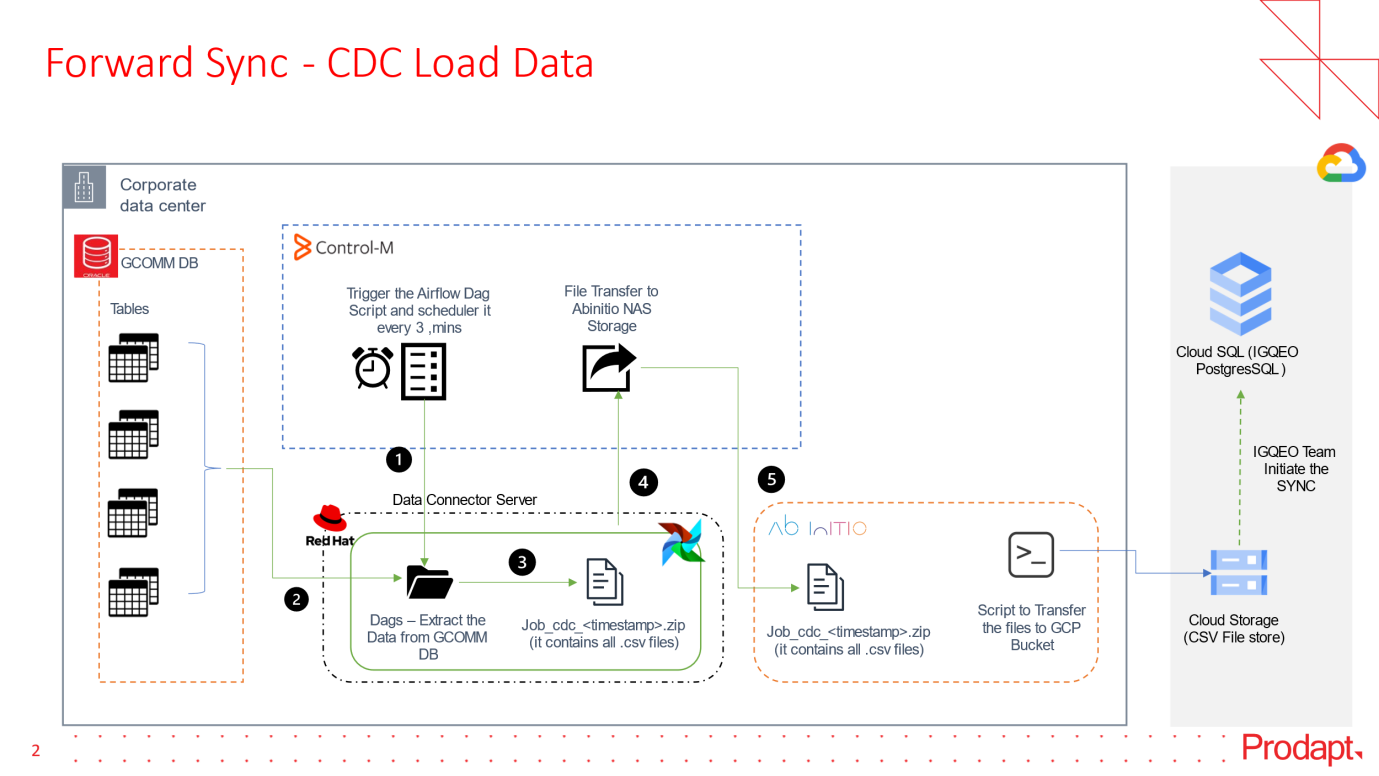
**summary\_task**-In this task we will create a summary file which consists of ['feature\_name','execution\_time','data\_export\_time','db\_count','total\_Data\_extraction\_count\_in\_files','No\_of\_csv\_files\_extracted','total\_time'] columns and store it in "/usr/data/GCOMM\_IQGEO/extraction\_summary/extraction\_summary\_cdc\_{current\_datetime}.csv**creating\_metadata\_file**-In this task we will create a metadata file which will consists 'vmx\_version', 'design', 'owner', 'owner\_email' details.

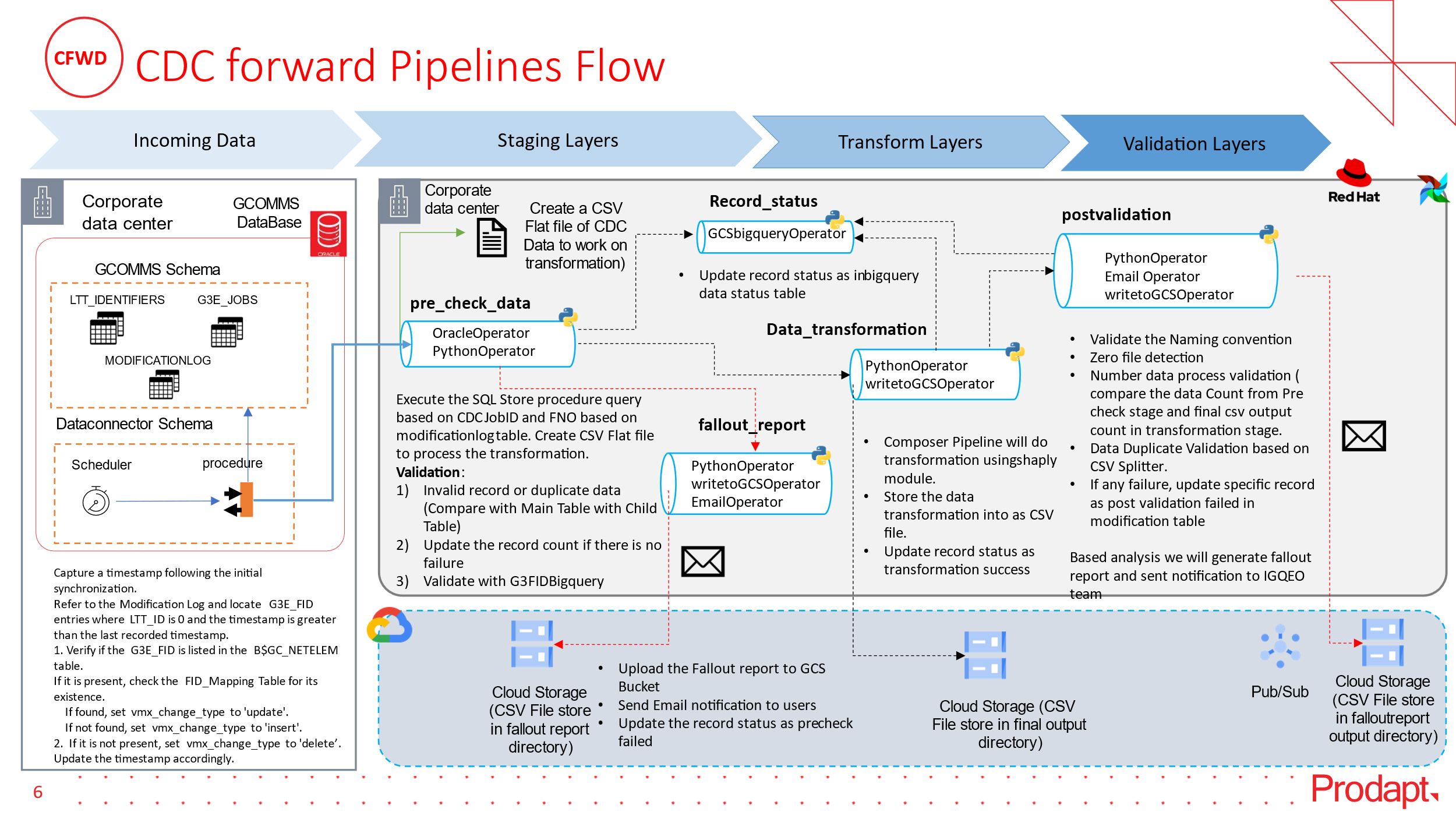
**zip\_and\_move\_task**-In this task we will zip the job\_CDC\_{current\_datetime} present in "/usr/data/GCOMM\_IQGEO/process/job\_cdc\_{current\_datetime}" and move to the "/usr/data/GCOMM\_IQGEO/process/job\_cdc\_{current\_datetime}.zip"

**update\_query\_task-**In this task we are updating the table TB\_CDC\_PROCESS from processed “P” to “Y” and truncating the table TB\_splice\_connect after every run.

**error\_handling\_task**-In this task we write all the error records into csv file and store it in "/usr/data/GCOMM\_IQGEO/error\_reports/cdc\_error\_fid\_{current\_datetime}.csv"

**email\_task**-In this task we will sent the eamil if all the above tasks are successful with an attachment of extraction\_summary





## 3.3.CDC Reverse

In Reverse Data Connector, first the Control team from GCP bucket places the file to NAS storage and from there it loads the CSV files in the input folder as zip file from IQGEO databases. Airflow Dag will load the data from the zip file in PLSQL table and call the PLSQL procedure. The procedure will perform insert,update and delete functionalities which will be updated in GCOMMS UI

* Inserting one extra column (vmx\_region) through python script as NONE for fiber\_cable\_lable.
* Inserting two extra columns (vmx\_region, vmx\_id) as NONE through python script for connections.

Below are the task details present in the reverse Dag

list\_and\_sort\_task: This task is a PythonOperator that executes the list\_and\_sort\_zip\_files function. This function lists all the ZIP files in a specified folder (source\_path) and sorts them based on their timestamps. It provides the sorted list of ZIP files as output.

move\_and\_unzip\_task: This task is a BashOperator that executes a bash command (move\_command). The command moves ZIP files from the source folder (source\_path) to the destination folder (destination\_path). It's assumed that the ZIP files contain data that needs to be processed.

unzip\_files\_one\_by\_one\_task: This task is a PythonOperator that executes the unzip\_files\_one\_by\_one function. This function unzips the files one by one based on their timestamps. It takes the list of sorted ZIP files as input, unzips them, and returns the name of the processed ZIP file.

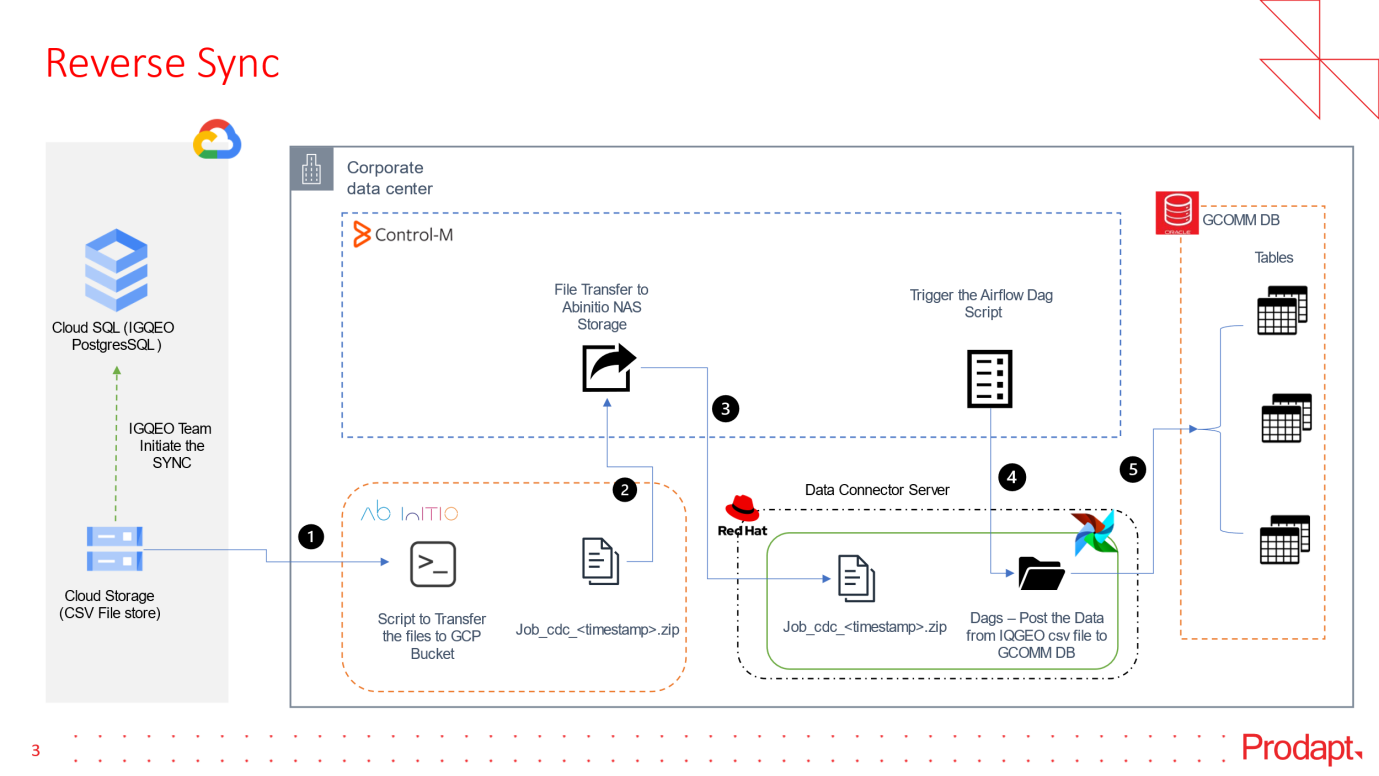
load\_data\_task: This task is a PythonOperator that executes the load\_data\_into\_tables function. It loads data from CSV files into corresponding tables in the database. It takes the name of the processed ZIP file as input.

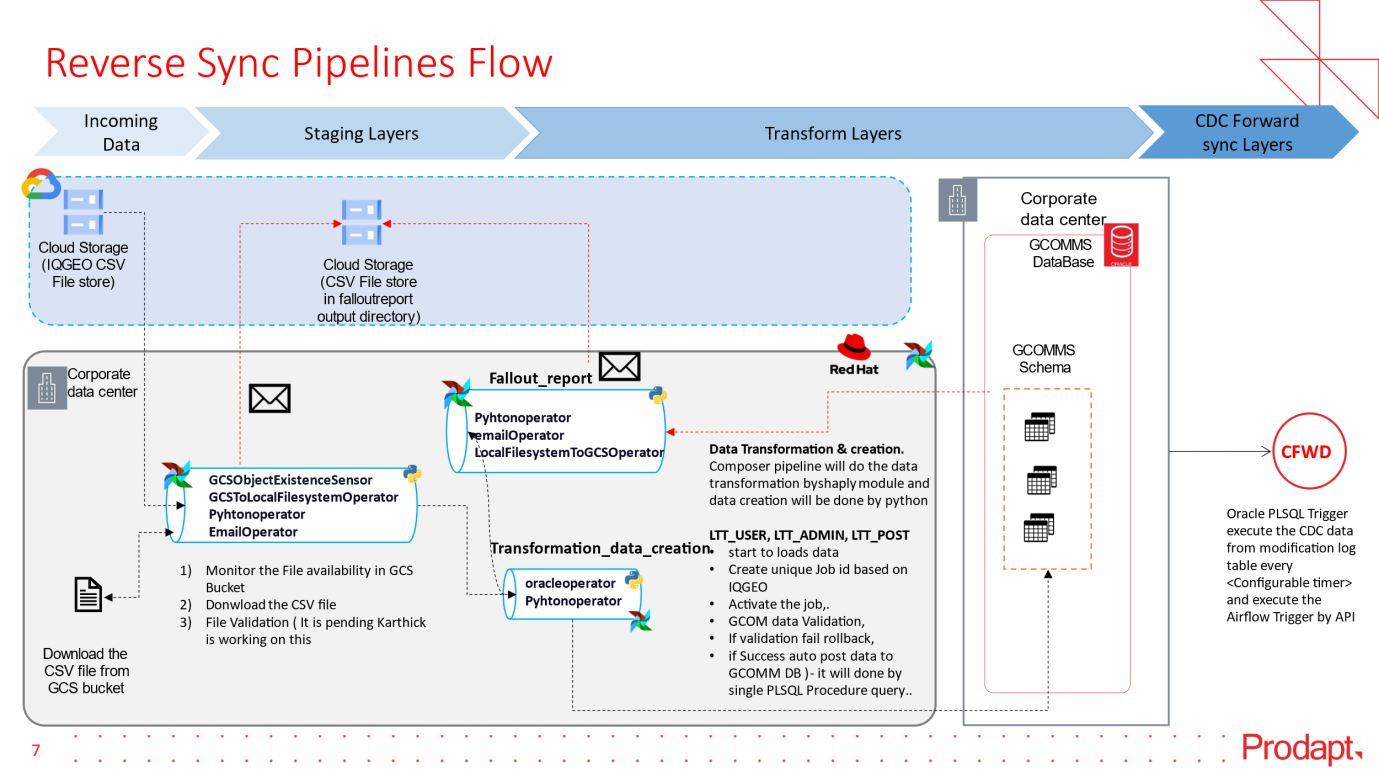
call\_procedure\_task: This task is a PythonOperator that executes the call\_stored\_procedure function. It calls a stored procedure in the database to perform specific actions based on the loaded data. It takes the name of the processed ZIP file as input.

move\_to\_archive\_task: This task is a PythonOperator that executes the move\_to\_archive function. It moves the processed ZIP file along with any associated files or directories to an archive folder (archive\_folder). It takes the name of the processed ZIP file as input.

move\_to\_error\_task: This task is a PythonOperator that executes the move\_to\_error function. It moves the ZIP file to an error folder (error\_folder) if any errors occur during processing. It takes the name of the processed ZIP file as input.

dag\_status\_check\_task: This task is a PythonOperator that checks the status of the DAG after processing. If the move\_to\_error\_task succeeds, it raises an error to indicate that the DAG status check failed.





## 3.4.Fallout Report

In the Fallout Report for Full load, we have created 3 reports

1. Missing Fid
2. Duplicate Fid
3. Missing Fid Summary

For CDC we have created one report that provides the details for processed fid , duplicate fid, error fid and completed Fid

## 3.5.Summary Report

## Full Load

As part of the summary report, we are capturing feature\_name for which we are extracting the data , No\_of\_csv\_files\_extracted contains the details of total csv files created for that feature since 1 csv file contains 1million records , total\_Data\_extraction\_count contains the total number of records present in the table for that feature, total\_time in sec gives time requires to extract the records and load it to the csv file for full load.

## FORWARD CDC

As part of the summary report, we are capturing feature\_name for which we are extracting the data ,execution time in sec gives the time taken for the execution of query and storing the result in a variable, Data\_export time in sec gives time taken to write the records in the csv file, No\_of\_csv\_files\_extracted contains the details of total csv files created for that feature since 1 csv file contains 1million records , total\_Data\_extraction\_count contains the total number of records present in the table for that feature, total\_time in sec gives time requires to extract the records and load it to the csv file for CDC.

# PLSQL Development

Overview of PLSQL Development

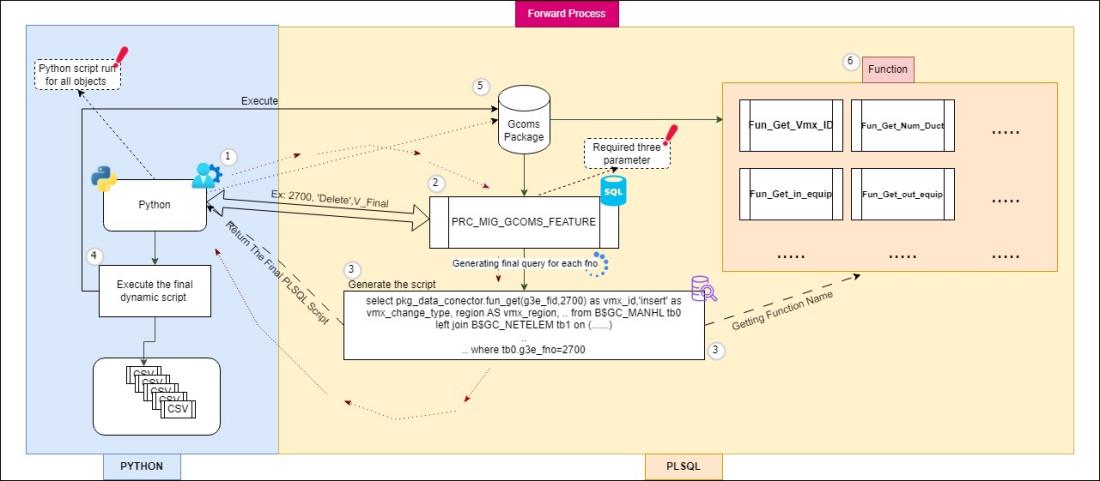
## Full Load

Prerequisites

Ensure that you have met all of the following prerequisites:

* CPU-16
* Memory-32GB
* Oracle sqlite
* Oracle DB
* gcoms's data schema need to be present

Workflow diagram



Automation explanation:

The data migration process, termed the Forward Connector, facilitates the seamless transfer of data from GCOMMS to IQGEO . PL/SQL queries play a pivotal role in orchestrating and executing the migration, effectively facilitating the relocation of data between servers. There are different types of features likes manhole,conduit,swept\_tee,etc... Every feature has millions of records, and those data will be migrated.

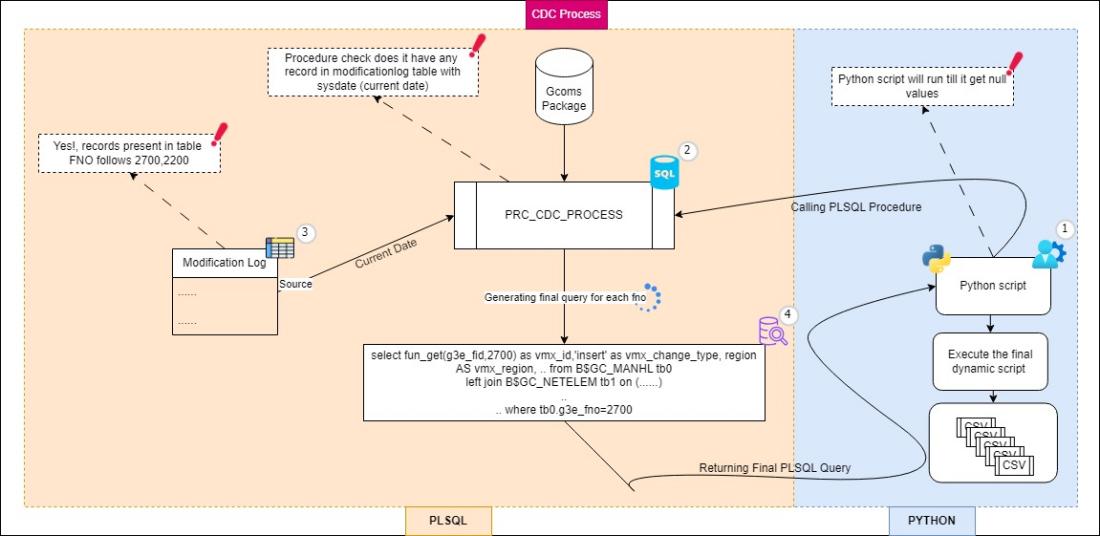
## CDC Forward

Prerequisites

Ensure that you have met all of the following prerequisites:

* CPU-16
* Memory-32GB
* Oracle sqlite
* Oracle DB
* modificationlog and fibertablemodlog table need to present

Worflow diagram



Automation explanation:

The purpose of the CDC(Change Data Capture) process is to identify and track changes like Insert, Update and Delete data in a database in real-time and generate PLSQL script accordingly.

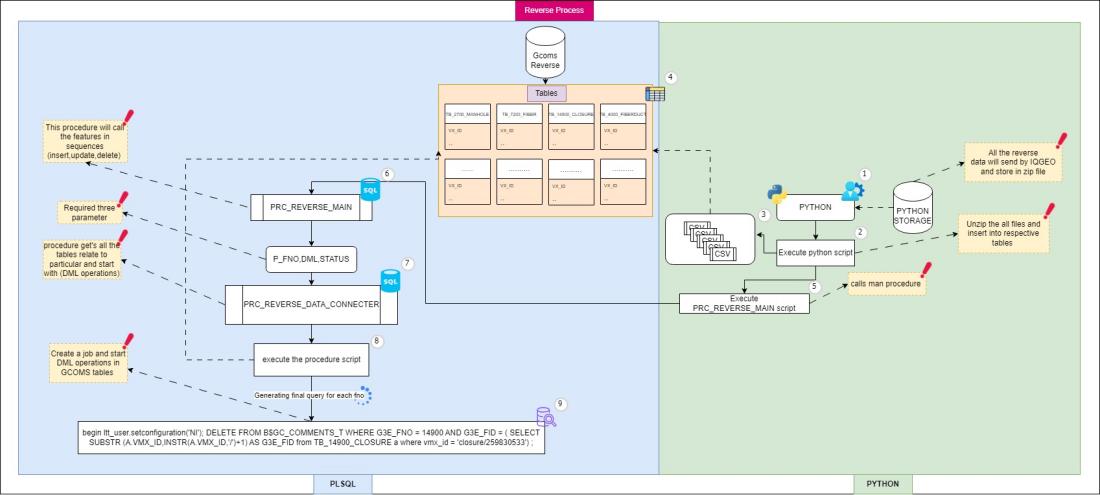
## CDC Reverse

Prerequisites

Ensure that you have met all of the following prerequisites:

* CPU-16
* Memory-32GB
* Oracle sqlite
* Oracle DB
* gcoms's data schema need to be present

Worflow diagram



Automation explanation:

The Reverse Connector, streamlines data transfer from IQGEO to GCOMMS. PL/SQL queries play a pivotal role in orchestrating and executing the migration, effectively facilitating the relocation of data between servers. there are different types of features likes manhole,conduit,swept\_tee,etc. base on the IQGEO CSV files

# 5. Inventory details

## 5.1.GCOMM INVENTOR DETAIL

[VMO2 share path link](https://prodaptcloud.sharepoint.com/:x:/r/sites/VM-O2GAPAssessment/Shared%20Documents/General/Data_Connector/Access%20Matrix/VMO2_Servers.xlsx?d=wa5ddfc47eee543cdb83fcd3cc51cdbb2&csf=1&web=1&e=3fcJiZ)

[VMO2 On-boarding Form.xlsx](https://prodaptcloud.sharepoint.com/:x:/r/sites/VM-O2GAPAssessment/Shared%20Documents/General/Data_Connector/Access%20Matrix/VMO2%20On-boarding%20Form.xlsx?d=wfb5d8e52ee124a14a3f0e1572883599f&csf=1&web=1&e=PJc7tB)

## 5.2.GCOMM DB Version details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Server name** | Environment | OS version | Database name | DB version |
| **gcomsdb-kn-t1** | Test | Linux x86\_64 | gcom1t\_PDB1\_GCOM1T | **19.15.0.0.220419** |
| **gcomsdb-kn-p11** | **Prod** | Linux x86\_64 | GIS1P\_PDB1\_GIS1P | **19.17.0.0.221018** |
| **gcomsdb-kn-p12** | **Prod** | Linux x86\_64 | gisap\_PDB1\_GISAP | **19.15.0.0.220419** |
| **gcomsdb-kn-u11** | PreProd | Linux x86\_64 | GIS1U\_PDB1\_GIS1U | **19.17.0.0.221018** |
| **gcomsdb-kn-d11** | Dev | Linux x86\_64 | GIS1D\_PDB1\_GIS3D  GIS1T\_PDB1\_GIS2T  GIS1D\_PDB1\_GIS1D  GIS1T\_PDB1\_GIS1T | **19.17.0.0.221018** |
| **gcomsdb-kn-d12** | Dev | Linux x86\_64 | GIS2D\_PDB1\_GIS4D  GIS2D\_PDB1\_GIS2D | **19.17.0.0.221018** |

# 6. MONITORING

## 6.1.CELERY Flower

Celery Flower is a web-based user interface (UI) for monitoring and managing Apache Airflow tasks, also known as Directed Acyclic Graphs (DAGs). It provides an intuitive visual representation of the DAGs, allowing users to easily track their progress, view logs, and troubleshoot issues.

