Customer Engagement Analytics

April 19, 2024

# 

# Table of Contents

[Table of Contents](#_q2aosf8sa6eu)

[Summary](#_ktkgta1vf50p)

[Description](#_shlv8yc3mvg2)

[System](#_wlh24v3ka964)

[Architecture](#_dnwnhywmiqfk)

[Amazon Web Services](#_y0eg3toy3sc4)

[Snowflake](#_dxmoh85wfqgj)

[Github](#_ipk2cd4dv1yc)

[Environments](#_okeu8jq0udlz)

[Development](#_rpzzxaub5kz7)

[User Acceptance Testing (UAT)](#_so98xa43gnag)

[Production](#_yeg50jp7w1ec)

[Replication Rules](#_srbayw3ntbfw)

[Costs](#_lm0lihr2j1l9)

[Programming Languages/File Formats](#_75qfvu1j3s05)

[SAM/Cloudformation (yaml)](#_qs8zyqx4z62z)

[Schemachange (SQL)](#_oabv8it4aeko)

[Github Workflow Files (yaml/bash)](#_suf7wnbvbx7s)

[Glue Jobs/Data Checks (python)](#_s180fwq5occy)

[Data](#_wrk0ogp8g31g)

[Datapull High Level Summary](#_kyukuff1bhcy)

[Data Processing Layers](#_jxglmxqoj8m2)

[Data Sources](#_sfqxssr5rcpm)

[Manual Upload Procedures](#_94w983c9oxv8)

[Upload Rules:](#_yr9p2enxeem1)

[Supported Uploads:](#_sild5r1887op)

[Specialty Mapping](#_ph4alg2mjvma)

[ICS Mapping](#_uovfthv46dd7)

[Hallo Healthcare](#_g4iey99jc22f)

[Data Ingest Frequency](#_41atckjqm4vh)

[Data Ingest Notifications](#_2jdlxwar1k4r)

[Reminder Notifications](#_bsflvt52vys7)

[Status Notifications](#_rgiw0mrmlqdu)

[Error Notifications](#_1msx82c0lgg)

[Data Auditing](#_nzgn8wke966y)

[Data Ingest System (Detailed)](#_a0252afk95f8)

[Data Aggregation](#_gt735ni3wuky)

[Data Ingest Checks](#_n24rgbcv48ot)

[Data Mapping](#_jzekmudzlgbs)

[Data Access](#_ivdrxdrma40d)

[Provision\_Role](#_f7r47iu1011f)

[Service\_Role](#_a13545tmdtj7)

[Read\_Only\_Role](#_uu8njdvv62do)

[Tableau\_Read\_Only\_Role](#_8kt9frhkmcow)

[Database Schemas](#_4j72iialtdry)

[STG](#_5yrp23mvhnav)

[INT](#_hopnbkio3o9q)

[PUB](#_o53ukj4bn17g)

[Data Access API](#_wavrs7ddsfej)

[Architecture](#_t97zm2iisher)

[Summary](#_aaxjz9gqv2kb)

[Authentication](#_g3z67yswnlzs)

[Authorization](#_hl3xs79sw1ch)

[Query](#_uftrembncxkk)

[Export](#_frb17opagd66)

[More Details](#_yrx6ulxllt7v)

# 

# 

# Summary

This document is a description of the Amarin Customer Engagement Analytics (CEA) application. It will describe the functionality of the system, data mapping, data pull frequency, and other details related to its deployment and ongoing processes.

## Description

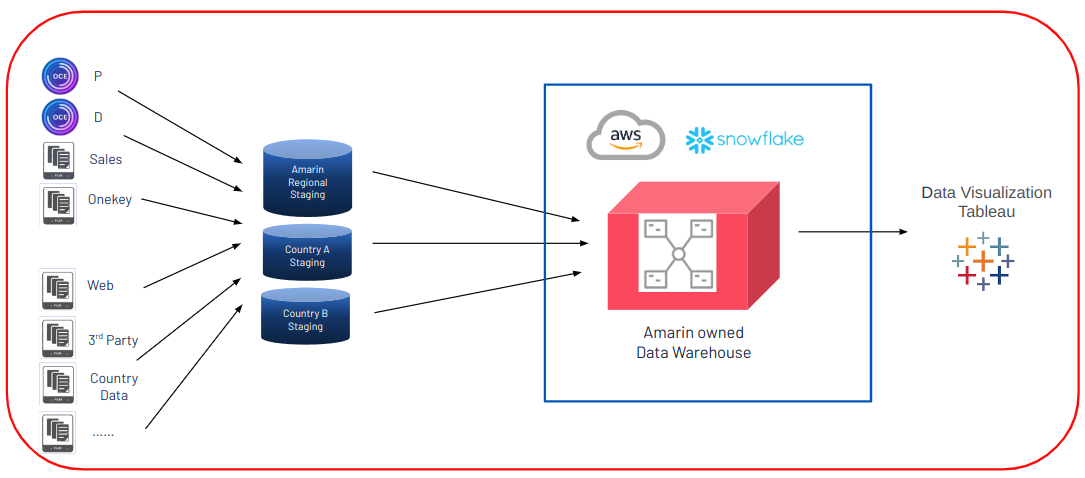
Customer Engagement Analytics accelerates the digital transformation of an organization by aggregating, organizing, and displaying data to allow for:

1. More Effective Campaigns
2. Increased Reach
3. Better Investment Transparency
4. Higher Frequency of Communication
5. Increased Personalization

# 

# System

The CEA system runs in Amazon Web Services (AWS) and stores all final data outputs in Snowflake. These platforms make it simple for the application to easily expand in scope over time while maintaining data compliance, security, and redundancy.



## 

## Architecture

Customer Engagement Analytics leverages many built-in services provided by AWS. These services are advantageous because AWS assumes responsibility in maintaining, updating, and optimizing them over time. This reduces the costs associated with a team required to monitor, migrate, and update infrastructure. Due to this, long term maintenance responsibilities can be primarily focused on improvements to the existing system, data quality monitoring, and evolving customer needs with limited attention paid toward infrastructure upkeep. Data ingestion and quality checks are orchestrated in AWS, data is stored in Snowflake, and the infrastructure is deployed through Github Actions.

The following services are implemented in CEA:

### Amazon Web Services

1. Data Ingest
   1. Glue Job - serverless compute for data ingest
   2. AppFlow - no code compute for data ingest
2. Data Storage
   1. S3 - long term, lost cost, high availability data storage for raw data
3. Security
   1. KMS - encryption keys for data security
   2. IAM - a user/permission management system used to restrict access to various parts of the system
   3. Secrets Manager - secure storage of usernames and passwords for access to data sources
   4. Cloudtrail - logging (audit trail) of user/service activity within the system
   5. VPC - forces all data ingest resources to make requests from a specific ip address and limits inbound/outbound requests
4. Data Auditing
   1. Glue Crawler - serverless compute for data auditing/schema determination
   2. Glue Data Catalog - storage to monitor changes in ingested data structures
   3. Cloudwatch - logging of data ingest statuses and errors
   4. SNS - a notification system used for automated data ingest to Snowflake and error/warning notification emails to support
   5. Lambda - small serverless compute used for data checks
5. Orchestration
   1. Step Functions - determines when data ingest jobs run, when to check data quality, when to provide error notifications, and when to run data consolidation processes
   2. Eventbridge Scheduler - schedules the date and time that a service (in the case, a step function) should run to perform data ingests

### Snowflake

1. Data Ingest
   1. Integrations - connect snowflake to data source systems (AWS), send error notifications
   2. Stages - determine what locations in data source systems should be accessible
   3. Pipes - transfer/transform data from stages to database tables
2. Data Storage
   1. Databases - used to contain the entire CEA data set within each environment (dev, uat, prod)
   2. Schemas - used to separate raw data (STG), transformed data (INT), and user-facing data (PUB)
   3. Tables - used to store individual data sets
   4. Views - custom consolidated data from tables used to display specific data
3. Data Transformation
   1. Stored Procedures - migrate/transform data across schemas
   2. Warehouses - determine the compute resources allocated to running transformations, queries, data loads, etc.
4. Data Security
   1. Users - determine who can log into the snowflake account
   2. Roles - determine what resources a user can access and the operations they can perform within a snowflake account

### Github

1. Repository - stores all code and infrastructure definitions for system deployment
2. Secrets - securely stored credentials used during deployment through github -cannot be seen once stored
3. Environments - logical partitions that allow varying secret values with the same name to be stored; this allows for (secure) multiple deployments of infrastructure within a repository and rule-based deployments
4. Actions - A serverless compute system that leverages the secrets within a given environment to perform actions (in this case, the deployment of the system into AWS and Snowflake)

## 

## Environments

CEA leverages 3 different environments to allow for the safe development of features:

### Development

Allows developers to work with the CEA processes and systems while accessing (non-critical) data.

### User Acceptance Testing (UAT)

Once development is complete, a developer will make a “pull request” in Github to the production branch. This will trigger a deployment into the UAT environment for verification prior to a production release

### Production

After UAT verification is completed, a pull request will be merged into the production branch by the reviewers. The production branch will contain all live data used by customers.

#### Replication Rules

Since some data is manually uploaded into CEA, replication rules have been placed on the S3 bucket for each environment. This allows manual uploads to automatically be replicated from PROD to UAT, and additionally from UAT to DEV.

## Costs

The data ingest process runs in a serverless format -meaning that most of its resources are utilized when the data ingest process is running, and not continuously. There are 4 sources of continuous costs within the system:

1. KMS - Encryption keys (based on number of keys used)
2. VPC - to maintain a static IP address, the vpc must run 24/7
3. S3 - data stored
4. Glue Data Catalog - based on size of schemas
5. Snowflake Database - based on size of data

## 

## Programming Languages/File Formats

The coding languages used within the CEA system have been intentionally consolidated to minimize the technical expertise needed to update and maintain it.

### SAM/Cloudformation (yaml)

AWS Serverless Application Management (SAM) is used to compile and deploy infrastructure; this is a superset of AWS Cloudformation. Yaml files were chosen to capture these instructions.

### Schemachange (SQL)

All snowflake database management is performed using the python-based package called schemachange. Schemachange follows the instructions from SQL files stored in various directories to create resources in snowflake. It records its operations in a snowflake change history table.

### Github Workflow Files (yaml/bash)

Github Actions uses yaml-based workflow files for orchestration of deployment operations. In this case, github actions determines when SAM should run (AWS deployment), and when Schemachange should run (Snowflake Deployment)

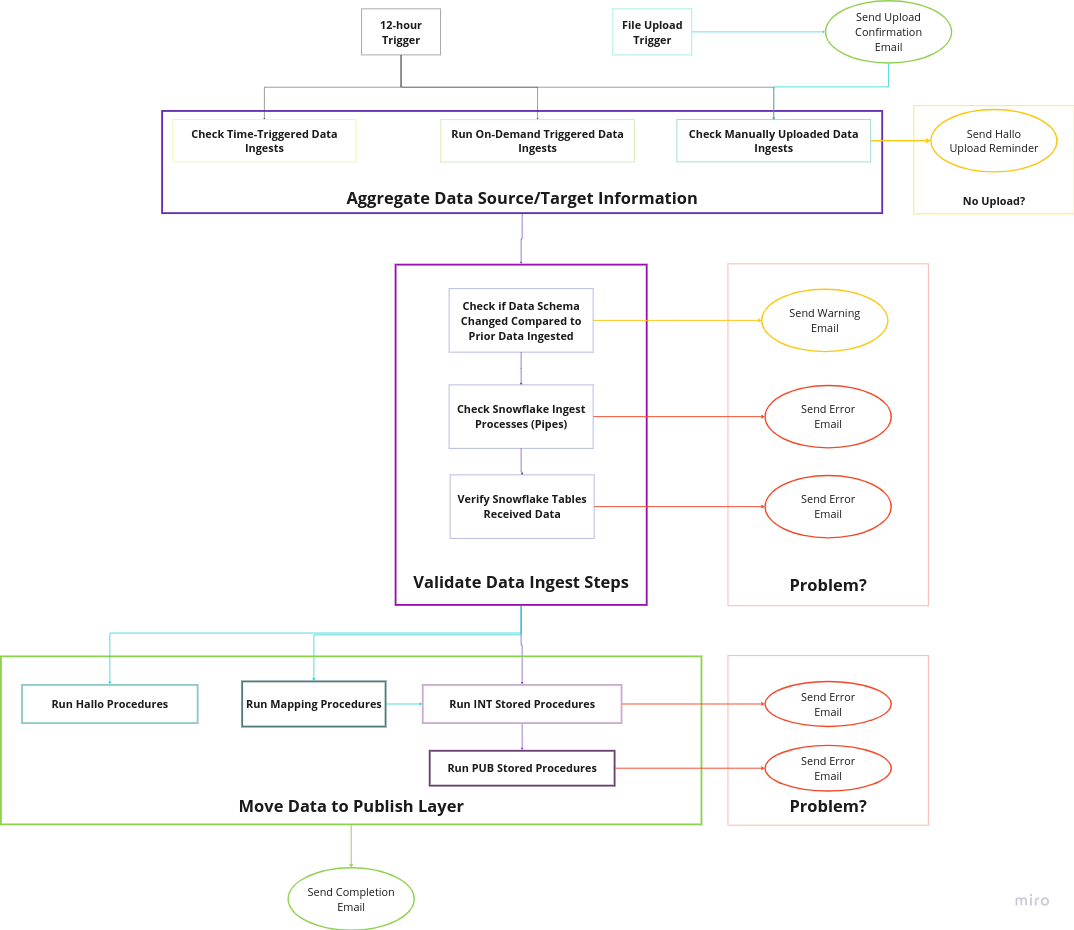
### Glue Jobs/Data Checks (python)

All code written to perform various data ingest operations and checks is written in python.

# 

# Data

## Datapull High Level Summary

The CEA Data Pull System was designed with a “layered” approach. While it contains many checks/verifications on an individual data source level, each layer acts in a similar fashion for all data sources. The following is a high level diagram which illustrates how the CEA system checks for data, processes it, and sends emails to alert those who monitor the system.  
  


## 

## Data Processing Layers

As displayed, the CEA can be pictured as a set of layers. An individual layer runs operations for each individual data source in parallel. However, a layer must wait for the previous layer to complete for **all data sources** before the next layer is activated. Below is a description of how each layer works:

1. **Trigger Event** - the event that triggers the CEA data pull process is critical to the system’s functionality.
   1. If the recurring time trigger starts the system, every data source will be inspected and processed.
   2. If a manual upload triggers the system, data processing/inspection will be skipped for every data source except for the upload (and tables dependent on the upload)
2. **Aggregate Data Source/Target Information** - The initial steps of CEA for each data source involve the system checking the status of the process that pulled data itself, or triggering the process to pull the data on-demand. Additionally, lists of tables, pipes (the connections between S3 and Snowflake), and a description name of the source are aggregated for further use.
   1. Time-Triggered Sources - Some data sources are required to have a time-dependent recurring pull applied to them. In this layer they must be synchronized with the start of CEA and are verified for completion before CEA continues to inspect data quality.
   2. CEA-Triggered Source - Data sources that are able to be triggered by CEA itself are triggered at this level and verified for completion prior.
   3. Manual Upload Sources
3. **Validate Data Ingest Steps** - The middle layer takes the aggregated information and performs a series of checks to ensure that data properly traveled from the data sources to Snowflake.
4. **Move Data to Publish Layer** - The final layer involves snowflake operations which move and transform the inserted data from the STG layer to the INT and PUB layers.

## Data Sources

The following data sources are currently integrated into the CEA system:

1. Iqvia OCE Personal
2. Iqvia OCE Digital
3. Iqvia Sales
4. Google Analytics
5. LinkedIn Ads
6. Hallo Healthcare (manual upload)
7. Mapping Definition Files (manual upload)

## 

## Manual Upload Procedures

There are multiple different manual uploads that are currently supported in CEA. All of them follow the same rules.

The manual uploads are inspected through the following process:

1. If an excel sheet: check for the indicated sheet name to ensure the proper data is transferred
2. Map all columns with the rules shown below. Any columns not included in the schema are ignored
3. Attempt to move the data from a raw

### Upload Rules:

1. Columns
   1. Columns can be changed in order but **not** in name.The snowflake tables have predefined columns that are not dynamically updated.
   2. Currently the following changes are made to column names automatically:
      1. All spaces, colons, and forward slashes are replaced by underscores
         1. “/ “ → “\_”
         2. “/” → “\_”
         3. “: “ → “\_”
         4. “-” → “\_”
         5. “ “ → “\_”
         6. “,” → ”\_”
         7. “€” → “euro”
         8. “(“ → “”
         9. “)” → “”
      2. All case sensitivity is removed. So **case sensitivity is not an issue**

**Ex:**

Account: Top Parent Onekey id **→**account\_top\_parent\_onekey\_id

ICS/ Country **→**ics\_country

* 1. Only the columns defined in the table will be ingested. All others will be ignored.

1. File Format
   1. Files are assumed to be in either Excel or CSV format. After upload, these are converted automatically to JSON for improved flexibility.
      1. Zip files can be uploaded, and will be extracted. However, only CSV extraction from zip files has been tested at the present time.
   2. This must be defined and kept static.
   3. Sheet name for Excel files is explicitly defined and must be kept constant

### 

### Supported Uploads:

ES Monthly Sales

**Format:**This data is assumed to be in Excel format and assumes that all data is in the sheet called “Monthly Sales”.

**Columns:**The following columns are currently expected:

1. DOT\_POTENTIAL
2. GEO\_DESCRIPTION
3. HOSPITAL\_AREA\_ID
4. HOSPITAL\_AREA\_NAME
5. PERIOD
6. PRIORITY
7. PRODUCT
8. PROVINCE
9. RANKING
10. RBM
11. REGION\_CCAA
12. SALES\_REP
13. TERRITORY
14. TOTAL\_UNITS
15. TOTAL\_VALUES\_MNF\_EUR
16. TYPE

ES Weekly Sales

**Format:**This data is assumed to be in Excel format and assumes that all data is in the sheet called “ES Weekly Sales”.

**Columns:**The following columns are currently expected:

1. CLIENTE
2. DATE
3. MANAGER
4. PRODUCTO
5. PROVINCIA
6. REGION
7. UNITS
8. VALUE

EU Geography

**Format:**This data is assumed to be in Excel format and assumes that all data is in the sheet called “EU Geo”.

**Columns:**The following columns are currently expected:

1. COUNTRY
2. MANAGER
3. NAME
4. REGION
5. TERRITORY

FI Sales

**Format:**This data is assumed to be in Excel format and assumes that all data is in the sheet called “Sheet1”.

**Columns:**The following columns are currently expected:

1. CITY
2. CUSTOMER
3. DATE
4. LOT\_NUMBER
5. MAT\_\_NET\_EURO
6. MAT\_\_QTY
7. PRODUCT
8. PRODUCT\_NUMBER

#### Hallo Healthcare

**Format:** This data is assumed to be in Excel format and assumes that all data is in the sheet called “Sales Details”.

**Columns:** The following columns are currently expected:

1. WEEK
2. W
3. PRIORITY
4. ICS\_COUNTRY
5. KAM
6. RBM
7. DATE
8. MONTH
9. YEAR
10. PRODUCT\_DESCRIPTION
11. CUSTOMER\_TYPE
12. CUSTOMER\_NAME
13. CLASS\_OF\_TRADE
14. HEALTH\_BODY
15. POST\_CODE
16. REGION
17. TERRITORY
18. UNITS
19. SALES\_VALUE

#### ICS Mapping

**Format:** This data is assumed to be in Excel format and assumes that all data is in the sheet called “EU ICS Mapping”.

**Columns:** The following columns are currently expected:

1. ACCOUNT\_TOP\_PARENT\_ONEKEY\_ID
2. ACCOUNT\_TOP\_PARENT
3. PRIORITY\_ACCOUNTS
4. ICS\_NAME

MIDAS

**Format:**This data is assumed to be in Excel format and assumes that all data is in the sheet called “MIDAS Tableau”.

**Columns:**The following columns are currently expected:

1. ATC4
2. CALENDAR\_QUARTER
3. CORPORATION
4. COUNTRY
5. INT\_DDD\_UNITS
6. INT\_WHO\_ATC4
7. MANUFACTURER
8. MOLECULE
9. PACK
10. PACK\_SIZE
11. PACK\_STRENGTH
12. PRODUCT
13. STANDARD\_UNITS
14. UNITS
15. USD\_MNF

NL Data

**Format:**This data is assumed to be in a .Zip file containing three Excel files. The three files are assumed to be called “area”, “article” and “sales”

**(area)Columns:**

1. AREA
2. AREA\_LEVEL
3. F2
4. F2\_DESCRIPTION
5. INHABITANT\_QUANTITY
6. PROVINCE\_NAME

**(article)Columns:**

1. COMBINED\_STRENGTH
2. COMPANY\_NAME
3. COMPANY\_STATUS
4. COUNTING\_UNITS\_AMOUNT
5. COUNTING\_UNITS\_UNIT
6. COUNTRY
7. DDD
8. EAN
9. EHIBC
10. EPHMRA\_CODE
11. F2\_DESCRIPTION
12. FARMINFORMCODE
13. FI\_DESCRIPTION
14. FI\_FORM
15. FI\_NAME
16. FI\_ROUTE
17. FI\_SUBNAME
18. FOTC\_CODE
19. GENERIC\_NAME
20. GRIP\_UNITS\_AMOUNT
21. GRIP\_UNITS\_UNIT
22. INTRODUCTION\_DATE
23. MANUFACTURER
24. MKKK
25. PACK\_FORM
26. PACK\_SIZE\_AMOUNT
27. PACK\_SIZE\_UNIT
28. PRODUCT\_STATUS
29. PRODUCT\_TYPE
30. REGISTRATION
31. STRENGTH\_AMOUNT
32. STRENGTH\_UNIT
33. SUBPACK\_SIZE\_AMOUNT
34. SUBPACK\_SIZE\_UNIT
35. WHO\_CODE

**(sales)Columns:**

1. AREA
2. AREA\_LEVEL
3. FARMINFORMCODE
4. LINE\_CODE
5. PACKS
6. SALES\_AIE
7. SALES\_MONTH
8. VERSION

NL Supporting Files

**Format:**This data is assumed to be in Excel format with two sheets and assumes that the sheets are called “Brick” and “DDD”.

**(Brick)Columns:**

1. AREA
2. BRICK
3. TERRITORIES

**(DDD)Columns:**

1. COEFF
2. FARMINFORMCODE

SE Sales

**Format:**This data is assumed to be in Excel format and assumes that all data is in the sheet called “Sheet1”.

**Columns:**The following columns are currently expected:

1. DATE
2. ITEM
3. CUSTOMER CHAIN
4. CUSTOMER
5. REGION
6. QTY
7. WEEK
8. YEAR
9. TERRITORY

#### Specialty Mapping

**Format:** This data is assumed to be in Excel format and assumes that all data is in the sheet called “Sheet1”.

**Columns:** The following columns are currently expected:

1. COUNTRY
2. HCP\_SPECIALTY\_LABEL
3. AMARIN\_SPECIALTY\_TRANSLATION

UK Source of Business

**Format:**This data is assumed to be in Excel format and assumes that all data is in the sheet called “Sheet1”.

**Columns:**The following columns are currently expected:

1. AMARIN\_SOURCES\_OF\_BUSINESS
2. AMARIN\_SOURCES\_OF\_BUSINESS\_ROLE
3. COUNTRY\_IDENTIFIER\_I
4. COUNTRY\_IDENTIFIER\_II
5. ONEKEYID

## Data Ingest Frequency

Data is currently ingested **twice per day** during the times of **1am** and **1pm UTC**. The current data ingest process takes approximately **1 hour** to complete in totality. For manual data uploads, the system attempts to look for new files on a daily basis, but will not perform updates to these datasets if new files are not available.

## Data Ingest Notifications

Email is the method currently used for the CEA notification system. There are 3 levels of notifications: reminder notifications, error notifications, and status notifications.

#### Reminder Notifications

These notifications are enabled to remind manual upload stakeholders to upload files:

1. A **reminder email** is sent for any file that hasn't been uploaded within the specified time period.

Reminders Enabled For:

1. Hallo Healthcare - 1 week

#### Status Notifications

These notifications are enabled to allow to manual upload stakeholders to monitor the process:

1. A **confirmation email** is sent upon upload.
2. A **confirmation email** is sent upon completion.
3. An **error email** is sent if there's an issue during the process.

#### Error Notifications

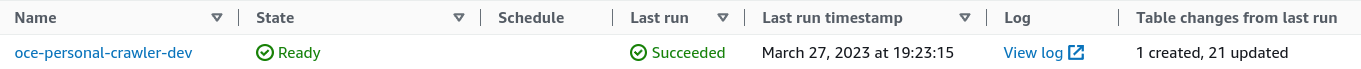
These notifications are enabled to allow to Tech support team to monitor the process:

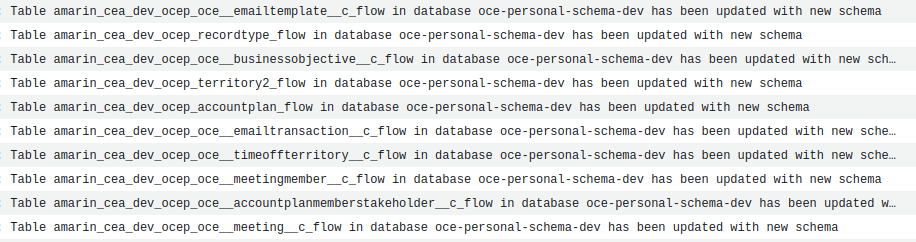
1. The support team receives an **error email** for each data error received.
   1. Support team will notify the stakeholders within 12 hours of receiving one of these emails
   2. Support team will create a ticket to track the error
2. The support team receives a **warning email** for each schema that changes.
   1. Support team will investigate the extent of the data change
   2. Support team will create a ticket to capture this change
   3. Support team will notify stakeholders in the event that this change will affect the data structure

## Data Auditing

AWS Glue Crawlers run with each data pull to check the data consistency of all data sources. Their output is a database containing tables of the fields, their data types, and a changelog indicating what fields were updated, added, or deleted.

The images below show an example of this process:

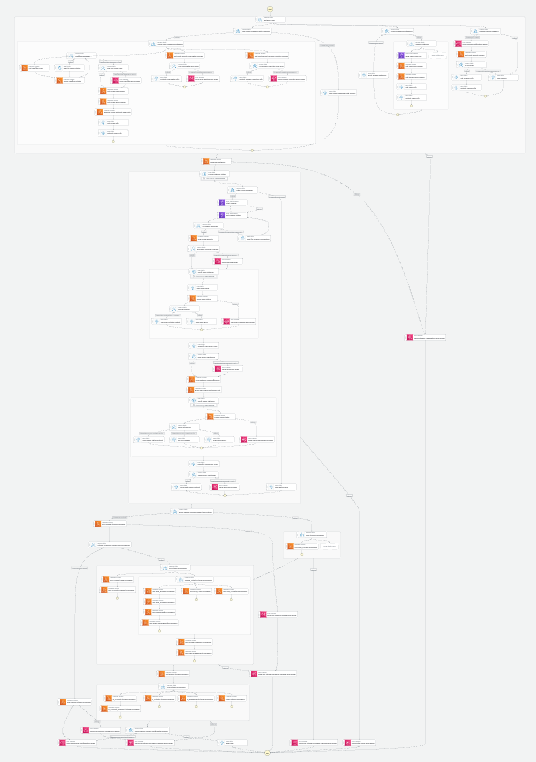




## 

## Data Ingest System (Detailed)

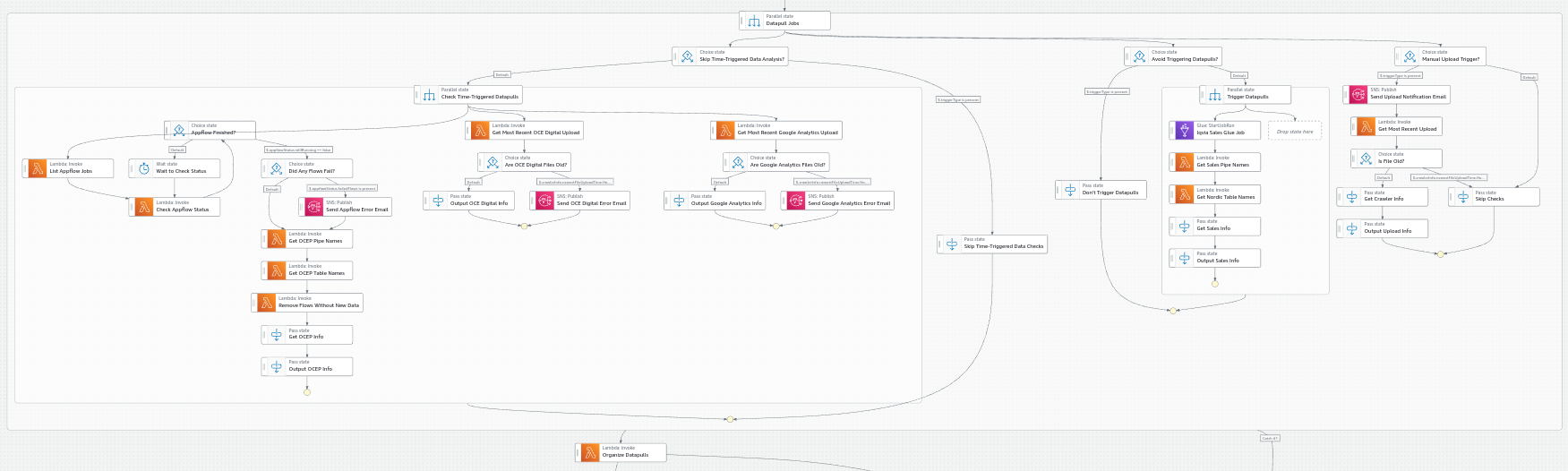
The entire CEA system is orchestrated by the following step function in AWS:

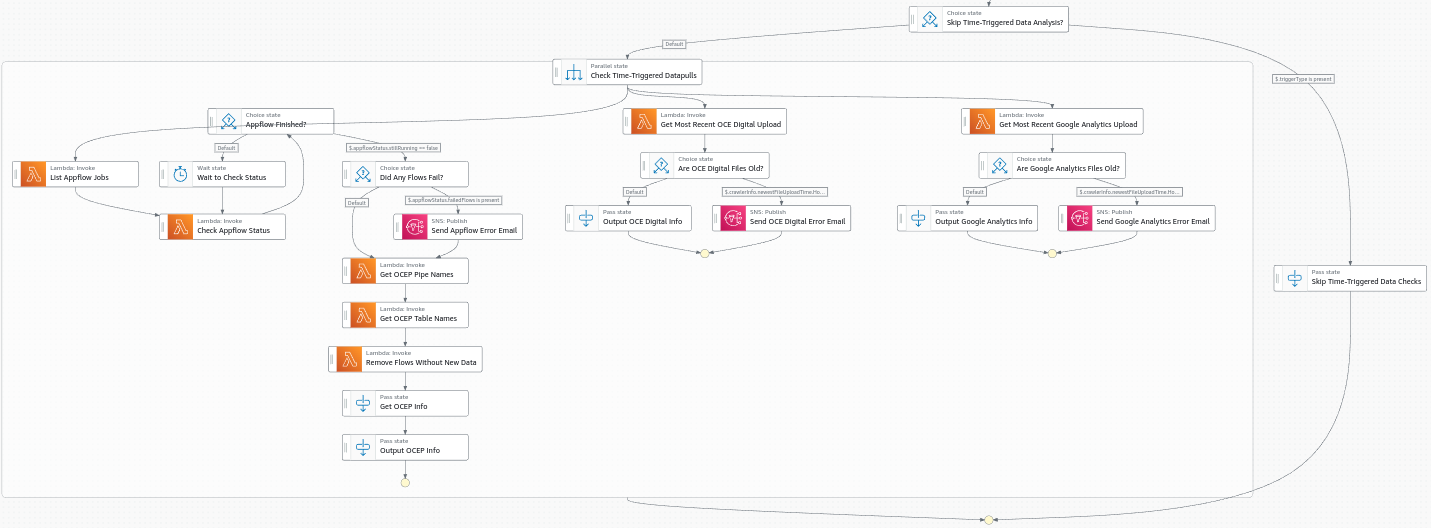


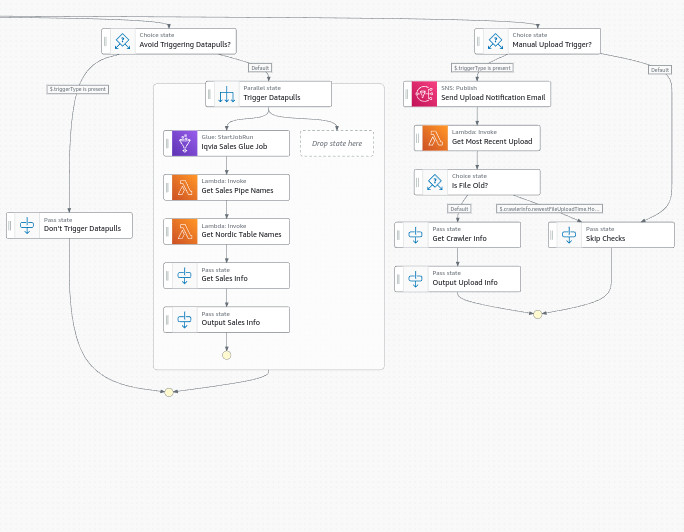
### Data Aggregation

Data Aggregation is run in parallel:

1. OCE Personal - This is set on a recurring time interval so AWS can manage the incremental pulls. Due to this, OCE Personal data pulls begin an hour before the step function. The step function then checks to ensure all OCE Personal flows are finished prior to continuing. It also checks to see if any flows failed and sends notifications accordingly.
2. OCE Digital - This is pushed to S3 by Salesforce. Due to this, the system checks that a file was sent within the past 24 hours.
3. Google Analytics - This is pushed to S3 by Google Cloud. Due to this, the system checks that a file was sent within the past 24 hours.
4. Infohub - This is triggered by the step function to pull the data and completion is verified before it continues. **[As of 10/2023, Infohub has been temporarily removed until needed]**
5. Iqvia Sales Data - This is triggered by the step function to pull the data and completion is verified before it continues.
6. Manual Uploads - Manual files are checked only when a file is uploaded. All details needed for the manual upload checks are sent into the step function. Otherwise, these checks are skipped.





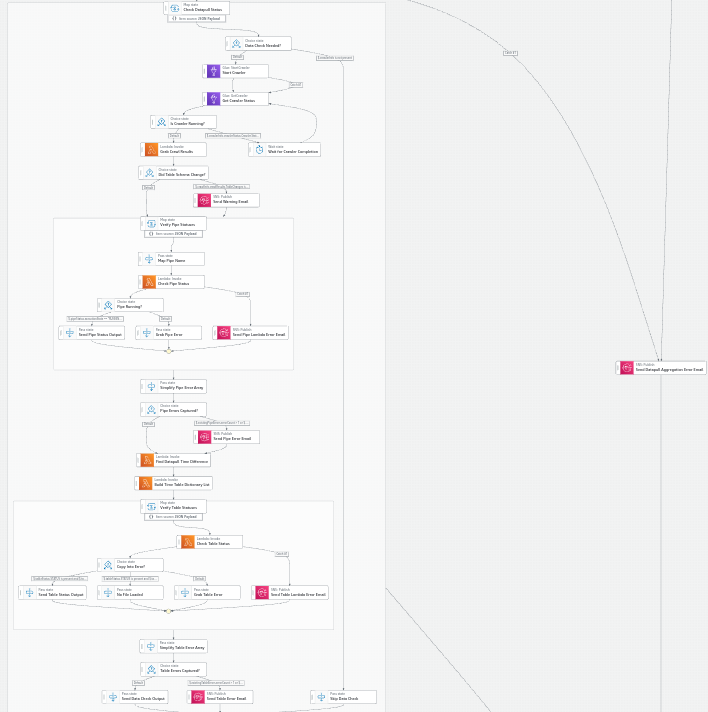


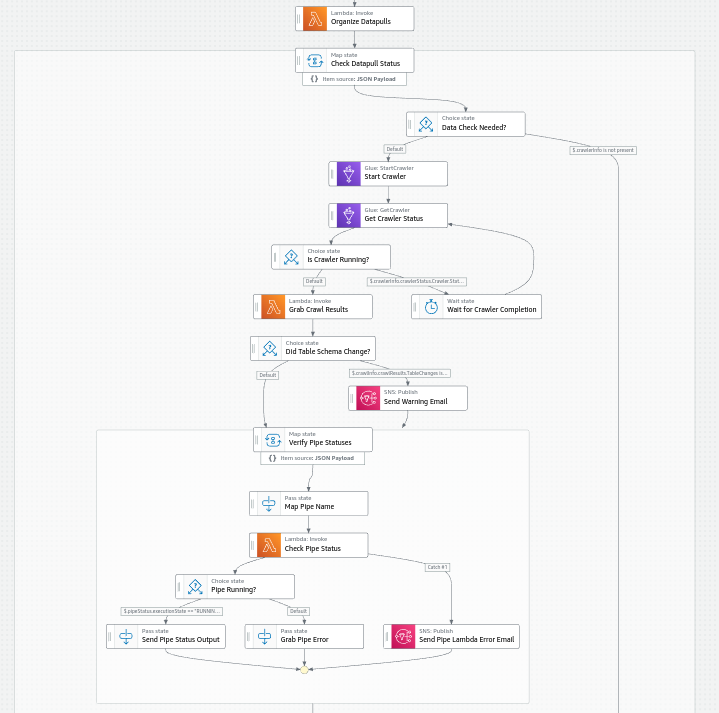
### 

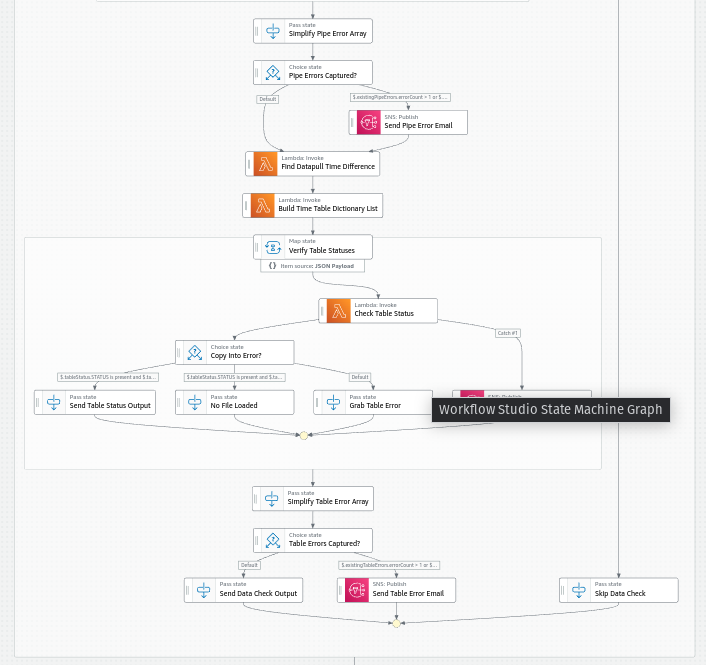
### Data Ingest Checks

Data Ingest Checks are run in parallel:

1. Run Glue Crawler to examine schema consistency
2. Check if the data schema has changed
3. If the data schema has changed, notify the CEA support team
4. Check the status of the snowflake ingest pipe to ensure it was possible for data to transfer to snowflake
5. If the pipes are facing errors, notify the CEA support team
6. Check the table's copy history to ensure the data successfully copied in
7. If the table has failures, notify the CEA support team



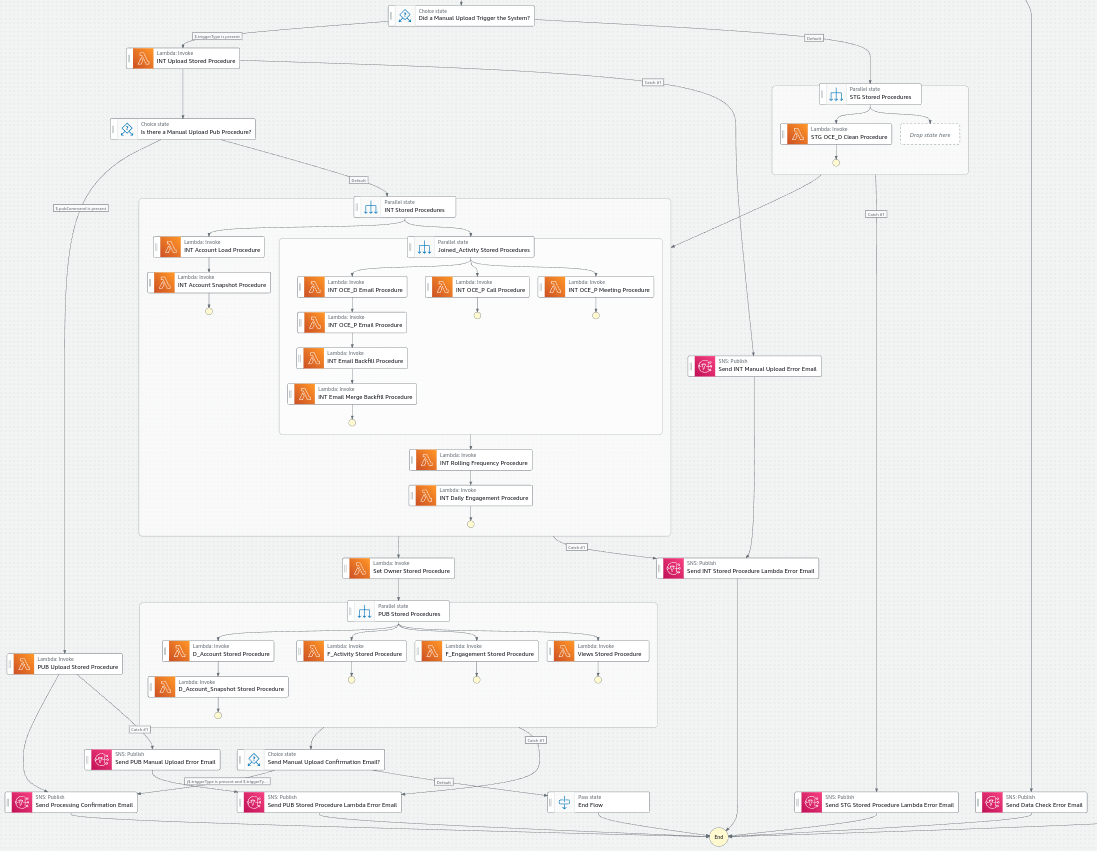


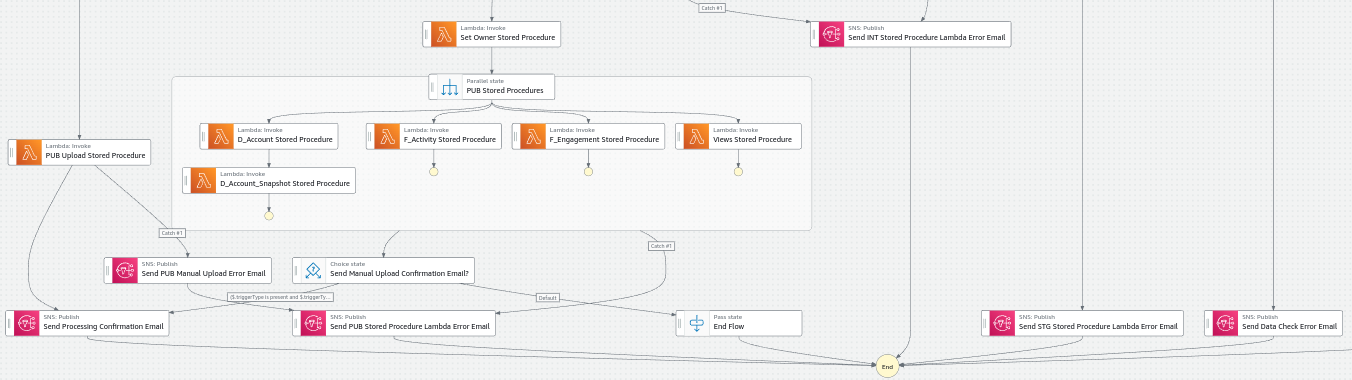
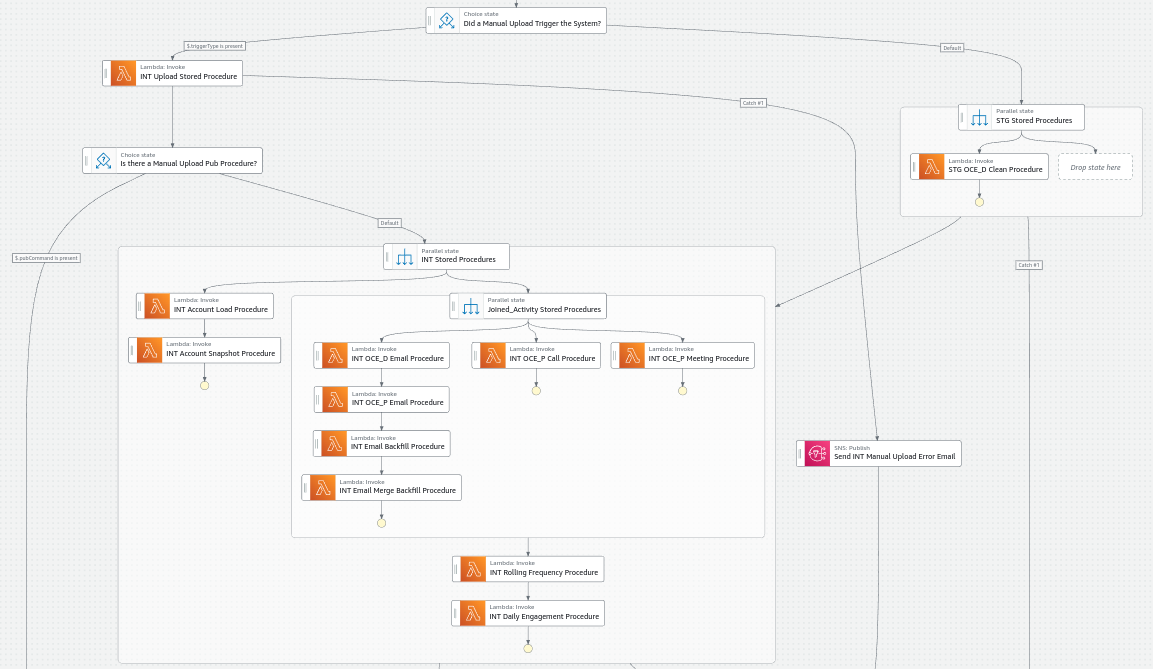


Stored Procedures

Stored Procedures are run in layers for INT and PUB.

* If the INT layer fails, the PUB layer is skipped. This is to prevent there from being mismatched partially updated data.
* Error emails are sent for any errors experienced.
* The Activity Stored procedures are dependent on each other and run as follows:
  + OCE Digital
  + OCE Personal
  + Email Backfill (for non-onekey id entries)
  + Email Merge (for non-onekey id entries)
* The manual upload procedures have been generalized to be evaluated on whether or not they affect INT or PUB layer tables other than themselves. All stored procedures tied to the uploads are sent into the step function when the upload is made.





## 

## Data Mapping

### Data Access

There are 4 roles set up to access/manage the data within Snowflake. In reality, the roles follow the naming schema of **[project name]\_[role name]\_[environment]** but only the role name has been shown below for simplicity:

#### Provision\_Role

This role “owns” all infrastructure within an environment. It has the broadest permissions out of any role and is only used for infrastructure deployment.

#### Service\_Role

This role is used on a regular basis by the AWS system to check and update data within all tables. It has the second largest permission set in the database because it monitors and controls the entire end-to-end CEA data pull process.

#### Read\_Only\_Role

This role has full “select” permissions on all tables in the environment. It can see all 3 data schemas (below) but cannot manipulate the data itself.

#### Tableau\_Read\_Only\_Role

This role has minimum permissions in CEA. It is only allowed to read from the data in the PUB schema.

### 

### Database Schemas

The data processing is handled in 3 distinct schemas. The schemas are described qualitatively below due to the large number of rules, data sources, and complex manipulations involved in the CEA system. For more information on how the data travels from STG to INT to PUB, please reference the CEA\_Data\_Mapping.xlsx file:

#### STG

The tables in this schema are connected to AWS -making STG the direct ingestion point for all data into Snowflake:

* The tables are updated automatically through Snowflake Pipes that are connected to Amazon S3.
* While some data may be type-formatted in this schema, this is considered a holding place for raw data overall.
* Manual uploads are aggregated with a timestamp attached to determine which should be moved into INT.

#### INT

This is the schema where most data formatting and manipulation occurs:

* Many data sources are aggregated together into single tables
* Fields are customized based on desired logic
* The data from the most recent manual upload is only stored
* All data moves into this schema through stored procedures run either on file upload or through the time-driven CEA data pull step function.

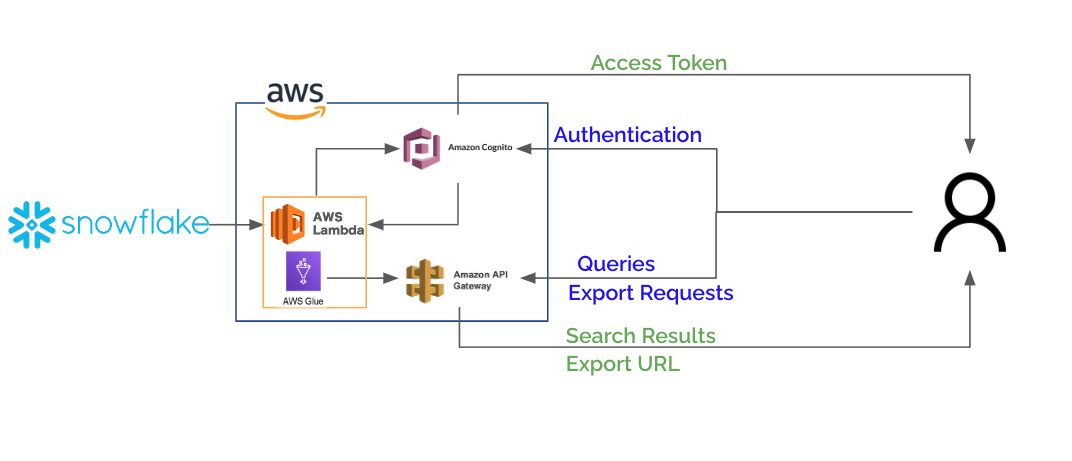
#### PUB

The tables in this schema are the ones that are designed to be user-facing and in the fully prepared format.

* Some data aggregation/manipulation may occur, but it is generally for simplifying information/displaying the data
* There are both tables and views in this schema
* All data moves into this schema through stored procedures run either on file upload or through the time-driven CEA data pull step function.

## Data Access API

### Architecture



#### Summary

This API was established to provide other applications with easy access to data within the CEA warehouse. The system provides the ability to query based on specific parameters, return only desired columns, paginate on results, and export bulk data in file format.

#### Authentication

This system uses OAuth 2.0 to authenticate access to the API. A consumer application must use a supplied secret and id to request an auth token -which expires after a short period of time.

#### Authorization

Consumer applications are only provided with the access needed to api endpoints through specific scopes. Scope permissions must be requested during authentication and are granted on an as-needed basis by the administrator.

#### 

#### Query

The query syntax is abstracted from SQL in order to simplify expression building. It uses human-readable operators like “and”, “or”, “in”, etc. to help consuming applications easily leverage the system and to additionally mitigate the risks of SQL injection.

#### Export

Export capabilities are also simplified and the files are provided through a short-lived presigned url. This allows for easy download. The current supported export formats are “JSON” and “Parquet”.

#### More Details

For specific details on the API and it’s usage, refer to the API Usage Instructions document.

## 