A diagram of a diagram of a data flow

Description automatically generated

A white background with black text

Description automatically generated

A screenshot of a data

Description automatically generated A screenshot of a document

Description automatically generated

A white background with black text

Description automatically generated A screenshot of a white page

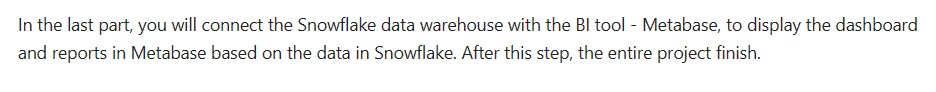
Description automatically generated

A diagram of data in a computer

Description automatically generated A white background with black text

Description automatically generated A diagram of data transformation

Description automatically generated A diagram of data analysis

Description automatically generated 

**8. Steps**

**8.1/ Downloading data from the S3 bucket (inventory table) to snowflake using AWS lambda**

The url of the S3 bucket is as follows. It’s fixed, but the inventory file is updated every day.

'<https://de-materials-tpcds.s3.ca-central-1.amazonaws.com/inventory.csv>'

We will use the library requests, and snowflake-connector-python, compile these library as layer adding to AWS lambda function.

**8.1.1/ coding for the EC2 instance**

*# Create Ubuntu EC2 instance t2.micro amd 8gb volume. Don't forget to attach key pair otherwise all settings should be default.*

*# Update EC2 instance*

sudo yum update

sudo yum upgrade

#create layer

mkdir **-**p lambda\_layers**/**python**/**lib**/**python3.7**/**site**-**packages

python3 **-**m venv venv

source venv**/**bin**/**activate

*# 1) install the dependencies in the desired folder*

pip3 install **-**r requirements**.**txt **-**t lambda\_layers**/**python**/**lib**/**python3.7**/**site**-**packages**/.**

*# 2) Zip the lambda\_layers folder*

cd lambda\_layers

zip **-**r snowflake\_lambda\_layer**.**zip **\***

*# 3) publish layer in s3*

*# Create s3 bucket in the same region as lambda function. You can name it anything. For this example, we will use lambda-layer-bucket as the name.*

aws s3 cp snowflake\_lambda\_layer**.**zip s3:**//lambda-**layer**-**bucket

*# 4) Add layer into lambda and choose s3 as the upload mechanism.*

*# You can follow instructions in this URL(https://docs.aws.amazon.com/lambda/latest/dg/creating-deleting-layers.html)*

*# under "To create a layer (console)" heading if you are not sure how to do this.*

*# 5) After uploading the layer, you will find the layer under custom layer option when you try to add layer to your lambda function.*

We need to read the URL, and download the file to tmp/inventory.csv ('tmp' folder is the only folder you can use save files temporarily on Lambda)

**8.1.2/ Uploading file to Snowflake using lambda**

**A screenshot of a computer

Description automatically generated**

Prior to running the Lambda function, create schema and table on the snowflake.

In the Lambda function, we need:

* Tell Snowflake which schema we are going to use;
* Create File Format for the file;
* Create a stage;
* Upload file from the lambda 'tmp' folder to the Snowflake stage;
* Copy the file from the stage to the table in the Snowflake.

#coding for lambda function

import json

import toml

import requests

import snowflake.connector as sf

def lambda\_handler(event, context):

#loading config data

config = toml.load('config.toml')

url = config['source']['url']

account = config['snowflake']['account']

user = config['snowflake']['user']

password = config['snowflake']['password']

warehouse = config['snowflake']['warehouse']

database = config['snowflake']['database']

schema = config['snowflake']['schema']

table = config['snowflake']['table']

role = config['snowflake']['role']

stage = config['snowflake']['stage']

#download file, and copy to temp folder on lambda

response = requests.get(url)

file\_name = 'temp.csv'

temp\_file\_path = '/tmp/temp.csv'

with open(temp\_file\_path,'wb') as temp\_file:

temp\_file.write(response.content)

#establish connection to snowflake

conn = sf.connect(user=user,password=password,account=account,warehouse=warehouse,database=database,schema=schema, role=role)

cursor = conn.cursor()

use\_schema = f"use schema {schema}"

cursor.execute(use\_schema)

file\_format = f"CREATE or REPLACE FILE FORMAT CSV\_COMMA TYPE ='CSV' FIELD\_DELIMITER = ',' SKIP\_HEADER = 1;"

cursor.execute(file\_format)

create\_stage = f"create or replace stage {stage} file\_format = CSV\_COMMA;"

cursor.execute(create\_stage)

file\_put = f"PUT 'file://{temp\_file\_path}' @{stage};"

cursor.execute(file\_put)

list\_stage = f"list @{stage}"

cursor.execute(list\_stage)

copy\_to = f"copy into {schema}.{table} from @{stage}/{file\_name} FILE\_FORMAT = CSV\_COMMA ;"

cursor.execute(copy\_to)

#close connection

cursor.close()

conn.close()

return {

'statusCode': 200,

'body': json.dumps('Hello from Lambda!')

}

#coding for the file config.toml

[source]

url = 'https://de-materials-tpcds.s3.ca-central-1.amazonaws.com/inventory.csv'

[snowflake]

account = 'LPGDBSA-KQB04012'

user = '-----------'

password = ‘----------'

warehouse = 'COMPUTE\_WH'

database = 'TPCDS'

schema = 'RAW'

table = 'inventory'

role = 'accountadmin'

stage = 'khapham'

**8.1.3/ Schedule the lambda function to run on every night at 2am (EST)**

Use the EventBridge in Lambda function to make it run on very night at 2am (EST)

**8.2/ Downloading data from the AWS RDS database using Airbyte**

**8.2.1/ Launch an Ubuntu EC2 instance in AWS with configuration as follows:**

. Type: t2.large

. AMI: Ubuntu Server 22.04 LTS (free tier)

. Firewall: allow all traffic to

. Storage: 20Gb

**8.2.2/ Install docker, and docker compose to this EC2**

Next is to install Docker to this EC2 instance

sudo apt update

sudo apt install -y docker.io

# add ubuntu user to the docker group using the below command

sudo usermod -aG docker ubuntu

exit

ssh -i <path to your pem file> ubuntu@<public IP of the EC2 instance>

*# install docker compose v2 manually*

*# reference https://docs.docker.com/compose/install/linux/*

DOCKER\_CONFIG**=**${DOCKER\_CONFIG**:-**$HOME/.docker}

mkdir -p $DOCKER\_CONFIG/cli-plugins

curl -SL https://github.com/docker/compose/releases/download/v2.18.1/docker-compose-linux-x86\_64 -o $DOCKER\_CONFIG/cli-plugins/docker-compose

chmod +x $DOCKER\_CONFIG/cli-plugins/docker-compose

docker compose version

**8.2.3/ Install Airbyte to this EC2 using docker, and docker compose**

*# Install Airbyte*

wget https://s3.amazonaws.com/weclouddata/data/data/run-ab-platform.sh

#start airbyte

chmod +x run-ab-platform.sh

./run-ab-platform.sh -b

once the airbyte UI pop up, key in username as airbyte, and password as password.

**8.2.4/ Setting connection between Airbyte and AWS RDS**

Setting up the Sources:

Select Postgres

Source name -> DE-RDS

Host -> de-rds.czm23kqmbd6o.ca-central-1.rds.amazonaws.com

port -> 5432

Database Name -> tpcds

schema -> raw\_st

username -> postgres

password -> weclouddatade

Setting up destination:

* Name: Snowflake
* Host -> https://mqb43056.snowflakecomputing.com
* Role: ACCOUNTADMIN
* Warehouse: tpcds
* Database: TPCDS
* Default Schema: RAW
* Username: ---------------
* Password: ---------------------

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**8.3/ Transform data downloaded to snowflake using DBT**

**schedule cron job to run dbt transform every day at 6am**

**crontab -e**

**0 6 \* \* \* <path to the bash script>**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**