A diagram of a cloud computing process

Description automatically generated

Step 1:

Using python to connect to the Climate Watch API. We are going to extract the data on emission in JSON format. Here is the simplified JSON schema: {

{**"data":** [

{"id”,"iso\_code3", "country", "data\_source", "sector", "gas", "unit", "emissions": [{"year","value”]}],

**"meta"**: {

"years": [ ],

"header\_years": [],

"sorting": {

"sort\_col": "iso\_code3",

"sort\_dir": "ASC"

},

"columns": ["id", "iso\_code3","country","data\_source", "sector","gas"]

}}

Step 2:

Load the emission data into an Amazon S3 bucket that we are going to be calling a landing zone. Immediately we get that data inside the Amazon S3 bucket that is going to trigger a lambda function (written by python language), which is going to help to copy that information from the landing zone into the next S3 bucket (intermediate zone). Since we don’t want to anyone to tamper with the data in the first S3 bucket (the landing zone), we want every other connection to be with the intermediate zone. So in the first and second S3 bucket, the emission data is still stored in the JSON format.

Step 3:

After the data is in the intermediate zone, it is also going to trigger another lambda function (written by python language) that then transform the data, then it’s going to load the data inside the third S3 bucket.

By looking at the target data schema, there are 6 tables: Data, Country, Series, Country-Series, Series-Time, and FootNote.

In this transformation step, the lambda function will first check if the value in column "iso\_code3" and “Country” in the source data are matched with the column “Country Code” and “Short Name” in the Country table in the target data. If not, make some changes to make sure they are matched. And also change the column name to “Country Code” and “Short Name”.

By looking at the Series table in target data schema, we need to create a column named “Series Code” in based on “sector”, “gas”, and “unit” column in the source data. For example, if in the source data, the sector is Bunker Fuels, gas is C02, and unit is MtCO2e, then the Series Code is CC.CO2.EMSE.BF.

By looking at the Country-Series table, there is a column named “DESCRIPTION”, we create a new column named source data description, value is “Source :Climate Watch”.

By looking at the Series-Time table, also have a column named “DESCRIPTION”, we create a new column named Series Time description, value is “The sample was drawn from the XXX sector only” where XXX is coming from the sector column in the source data.

Now we only keep the columns that is relevant to the target schemas. The source data is transformed to a table: the schema is following:

A screenshot of a computer

Description automatically generated

Based on the transformed table, we can add/merge additional data to the existing 6 tables. For example, the Data table in the target data can be made by selecting Country Code, Short Name, Series Code, Year and Data from the transformed table.

Step 4: And then we can load the transform data inside an Amazon Redshift cluster that we are going to be provisioning

All above steps are going to be orchestrated by the Apache airflow that is going to be running in the Amazon EC2 instance.