**SPOTIFY END-END DATA PIPELINE PROJECT (AWS, PYTHON)**

**Technologies Used:** Python, AWS S3, AWS Lambda, AWS Glue, AWS Athena, AWS IAM

**Step 1: Creating the Spotify Developer Account**

<https://developer.spotify.com/>

We need to have the Client ID and Client Secret key to extract the data and perform the analysis.

**Step 2: In the Jupyter Notebook (Doing Locally)**

1. We need to install the required libraries and must create Spotify object to access the TOP 50 India Playlist.
2. Here is the URL for the top 50 India playlist: <https://open.spotify.com/playlist/37i9dQZEVXbLZ52XmnySJg>
3. Next, we need to make use of the spotify.playlist\_tracks(playlist\_URI) to access the all the songs as a JSON Object.
4. This playlist contains the data about albums, artists, and songs
5. We will extract the required fields from the JSON object and store in the respective lists like albums, artists, and songs.
6. Now will convert the above lists to Dataframe so that it will much more understandable to the user.
7. We have changed the datetimes columns in the Dataframe.

**Step 3: AWS S3 Bucket Creations**

1. We will create two buckets Raw\_Data and Transformed\_Data.
2. In the Raw\_Data bucket we will create two folders named Before\_Processed and Processed.
3. In the Transformed\_Data bucket we will create three folders named Albums Data, Artists Data and Songs Data.

**Step 4: AWS Lambda Code**

1. Now we will write the AWS lambda code.
2. We have to create an **Extract function** with Runtime as python 3.8 because we use python to extract the data.
3. I have created this function with a new role.
4. We will use the same code python code which we used in the jupyter notebook.
5. In the function, we need to set the environment variables because it is not a good practise to hard code the sensitive values in the lambda code itself.
   1. For setting the environment variables, Go to Configuration -> Environment Variables.
   2. In our case we need to set two environment variables i.e., Client ID and Client Secret.
6. We must manually upload the spotipy module into the AWS Lambda Layer. (We did this in jupyter notebook by running pip install spotipy)
   1. For this We need to go Layers and create a layer as spotipy layer and upload the spotipy zip file.
   2. Next, we need to add this spotipy layer to our code.
7. Now we will dump this data to S3 bucket using boto3 module in AWS.
8. Usually, we will get errors like Timeout Error or Put\_Object Access denied.
   1. Timeout Error, it will occur because the default execution of the lambda function is set as 3 sec. We need to increase this one by going to Configuration -> General Configuration -> Timeout. The maximum runtime for the Lambda function is 15 minutes.
   2. Put\_Object Access Denied because when we are creating the function, we just created a new role with a basic permission in the IAM. Now we must give the S3FullAccess to the same role by going to Configurations -> Permissions -> Click on Role Name -> Add Permissions -> Attach Policies -> S3 Full Access -> Save.
9. Now we will create one more **function in lambda for Transformation job.**
10. We have created the above function with an existing role which we created in the Extract lambda function. In this function we will do two tasks.
    1. The first task is we will transform the data in the bucket Raw\_Data/before\_processed to Transformed\_Data/Albums or Transformed\_Data/Artists or Transformed\_Data/Songs.
    2. The second task is once we are done with the transformation, we will move the data from Raw\_Data/Before\_Processed to Raw\_Data/Processed.
11. In the function code, first we will list the files in the Raw\_Data/before\_processed and get all the filenames in it.
    1. Now we will iterate over each file and create the Lists for Albums, Artists and Songs by transforming the data which means will pick the respective fields in the JSON object.
    2. We will write three different functions for three categories.
    3. Now we will convert the list to pandas dataframe.
    4. Finally, we will store this dataframes in CSV format at their respective locations in the S3.
12. In this function code, second when once we are done with the transformation, will move files from Raw\_Data/Before\_Processed to Raw\_Data/Processed.

**Creating triggers to automate the process of Extraction and Transformation lambdas:**

1. We will create trigger for the Extract job which will run automatically for every minute, day, hour based on our needs.
   1. To do this, we must click on the Extract function in lambda, Add Triger -> Select a Source -> EventBridge (CloudWatch Events)
   2. Then we must create a new role, Give me some rule name.
   3. Finally, we must give the Schedule expression, whether we can use rate or cron expressions.
   4. I am using rate (1 minute) which will run for every 1 minute. (You can give like rate (1 day) like that as well.)
2. We will create a trigger for the Transformation which will run whenever there is an Object or file get created in the S3 bucket (Raw\_Data/Before\_Processed)
   1. To do this, we must click on the Transformation function in lambda, Add Trigger -> Select a Source -> S3.
   2. Provide the bucket name.
   3. Select the event type as All Object Create Events which means whenever there is an object create run this function.
   4. Provide the Prefix (Optional) - Before\_Processed/
   5. Provide the Suffix (Optional) - .Json
   6. We must change the permissions of the role which is there for Transform job.
   7. Attach Policy -> AWS LambdaRole (We are giving because for communication of two lambda functions.)

**Step 5: AWS Glue**

1. We will create a crawlers which crawls the files in the Transformed\_Data/Songs, Transformed\_Data/Artists, Transformed\_Data/Albums which will infer the schema of the CSV files in it and create tables on top it.
2. We must create 3 crawlers namely for Songs, Albums and Artists.
3. Creating Crawlers
   1. Go to Crawlers
   2. Create Crawler
   3. Provide the name
   4. Add a Data Source (Because we are creating on new S3 bucket and inferring it)
   5. Provide the S3 path (In our case BucketName/Transformed\_Data/Name) The name may be songs, artists and albums based on the crawler that we are creating.
   6. In that path give / at the end to dig into it.
   7. Create new IAM Role
   8. Give the name whatever you want.
   9. Add Database (Assuming that we don’t have before.)
   10. Name of the Database
   11. We can schedule the crawler.
   12. Next
   13. Finally create the crawler
4. Now we will create the crawlers for all like Artists, Albums and Songs.
5. Then we can run each crawler.
6. Sometimes, the crawlers will not be able to give the correct name for the column names in the table, that time we must edit manually AWS Glue, Tables.
7. For this we have to add one more addition for the Glue table property as skip.header.count=1.

**Step 6: AWS Athena**

1. Already tables get created in the AWS Glue for Songs, Artists, and Albums.
2. Now we can query in the AWS Athena by selecting the respective Database and Tables.