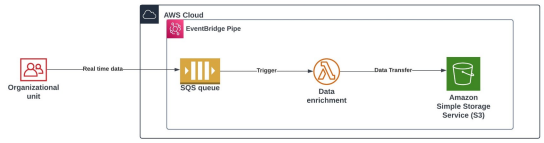
Airbnb -PDF

Objective:

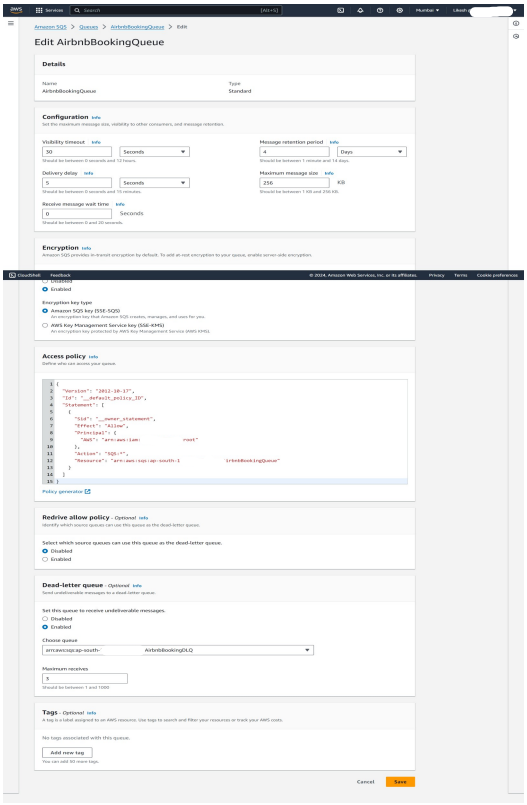
Build a simulated data pipeline for Airbnb booking data that integrates various AWS services, demonstrating real-time data processing, filtering, and storage.

Architecture diagram:



Step1 :- Create an Amazon SQS Queue with DLQ

1. Create an SQS Standard Queue named AirbnbBookingQueue.
2. Setup a Dead Letter Queue (DLQ):
   a. Create another SQS queue named AirbnbBookingDLQ.
   b. Configure the AirbnbBookingQueue to send messages to AirbnbBookingDLQ after 3 unsuccessful delivery attempts.



Visibility timeout - It is the duration during which a message is hidden from other consumers after being retrieved by one consumer.

Message retention period - It is the duration for which a message is stored in the queue before it expires and is automatically deleted.

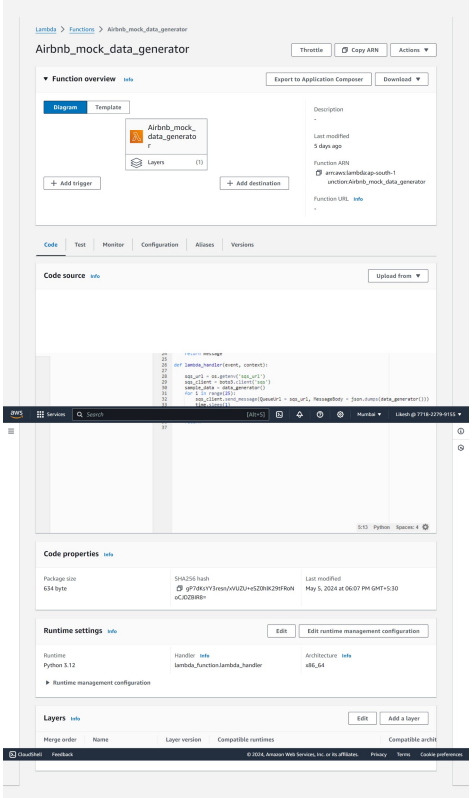
Delivery delay - It is the period between when a message is sent to the queue and when it becomes available for consumers to retrieve, allowing for a delay in message delivery.

Receive message wait time - It is the maximum time a consumer will wait for a message to become available in the queue before receiving an empty response, allowing for efficient message retrieval.

Step2:- Create Producer Lambda Function

1. Create a Lambda function named ProduceAirbnbBookingData. This function will generate mock Airbnb booking data and publish it to AirbnbBookingQueue.
2. Add enough permissions to publish data onto SQS.
3. Suggested Mock Data Structure:

```
{
"bookingId": "UUID",
 "userId": "UserID",
"propertyId": "PropertyID",
 "location": "City, Country",
"startDate": "YYYY-MM-DD",
"endDate": "YYYY-MM-DD",
"price": "Price in USD"
}
```

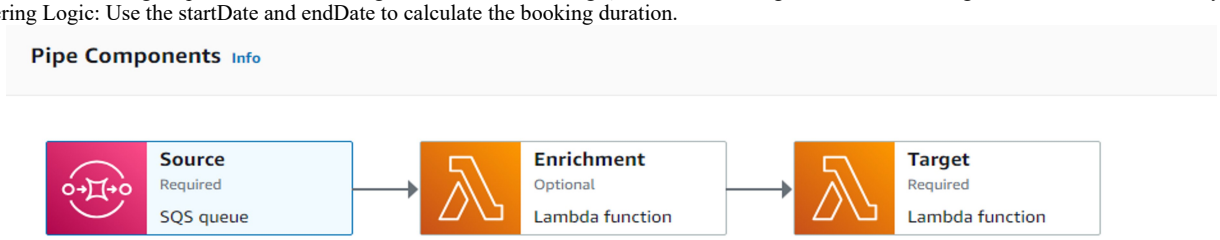


Step3:- Create an standard s3 bucket

1. Create an standard s3 bucket "airbnb-booking-records" to store the booking records.

Step4:- Setup EventBridge Pipe

1. Create an EventBridge Pipe to consume messages from AirbnbBookingQueue. Filter messages where the booking duration is more than 1 day.
2. Filtering Logic: Use the startDate and endDate to calculate the booking duration.



3. **Source:**
   a. The source of our data is the AirbnbBookingQueue SQS.
4. **Filtering at Enrichment Level:**
   a. Filtering of the incoming data occurs at the enrichment level.
   b. We've implemented a Lambda function for this task.
   c. The Lambda function processes each message from the SQS queue.
   d. If the booking's start date is equal to its end date, the message is filtered out.
5. **Data Storage:**
   a. Filtered/enriched data is stored in an S3 bucket.
   b. For clarity, we've designed the process in two steps:
      i. A Lambda function receives the filtered data.
      ii. Inside this Lambda function, the data is written into an S3 bucket.
   c. The S3 bucket's name and other configurations are handled using environment variables.

Lambda Functions:

1. Data Generation and Queuing:
This Lambda function generates simulated booking data and sends it to the SQS queue.

2. Filtering at Enrichment Level:
This Lambda function filters the incoming data based on the booking's start and end dates.

3. Data Storage in S3:
Upon receiving filtered data, this Lambda function writes it into the designated S3 bucket.

Find all these lambda_function codes, here.