Absolutely! I'll break everything down into very simple, beginner-friendly steps, with explanations so you can follow easily. We'll build a system that:

* Watches your DynamoDB table for new or updated entries
* When a new entry appears or updates, Lambda is triggered automatically
* Lambda reads the data (mr\_iid, project\_id, analysis)
* Posts or updates a comment on the related Merge Request (MR) in GitLab using the analysis text

**Step-by-step Guide for Beginners**

**Step 1: Prepare your DynamoDB Table for Streaming**

* Go to your **AWS Management Console**.
* Find **DynamoDB** service.
* Open your table called MRAnalysisReport.
* Click on the **Exports and streams** tab.
* Turn on **DynamoDB Streams** with option **NEW\_IMAGE**.
  + This means the stream will capture the new version of an item whenever it is added or updated.

Why? Because DynamoDB Streams will tell your Lambda whenever a new entry is added or updated in your table.

**Step 2: Create a Lambda Function in AWS**

* Go to **AWS Lambda** service in the Console.
* Click **Create function**.
* Choose **Author from scratch**.
* Give your function a name, e.g. ProcessMRAnalysis.
* Choose **Python 3.9** (or the latest Python runtime).
* For **Execution role**, create or select a role that has:
  + Permission to read from DynamoDB Streams.
  + Permission to write logs to CloudWatch.
  + Access to the internet (so it can call GitLab's API).

**Step 3: Connect DynamoDB Stream to Lambda as a Trigger**

* Open your Lambda function.
* Go to **Configuration > Triggers**.
* Click **Add trigger**.
* Choose **DynamoDB** as the trigger type.
* Select your table’s stream ARN (it appears once you enable Streams).
* Set **Batch size** to 1 (process one record at a time).
* Enable the trigger and save.

Now your Lambda will run **automatically** whenever a new item is inserted or updated in DynamoDB.

**Step 4: Write Your Lambda Code**

Your Lambda function will:

* Read the incoming record from DynamoDB Stream.
* Extract the fields mr\_iid, project\_id, and analysis.
* Call GitLab API to add or update the MR comment with the analysis text.

Here’s a simple example code to put inside your Lambda function:

import json

import requests

import urllib.parse

import boto3

# Your GitLab API URL and token (replace with your token)

GITLAB\_API\_URL = '<https://gitlab.com/api/v4>'

GITLAB\_TOKEN = 'your\_gitlab\_personal\_access\_token'

# DynamoDB setup

dynamodb = boto3.client('dynamodb')

DYNAMODB\_TABLE = 'YourDynamoDBTableName' # Replace with your actual DynamoDB table name

def post\_gitlab\_comment(project\_id, mr\_iid, analysis):

# URL encode project\_id if it contains slashes or special chars

project\_id\_encoded = urllib.parse.quote\_plus(str(project\_id))

url\_notes = f"{GITLAB\_API\_URL}/projects/{project\_id\_encoded}/merge\_requests/{mr\_iid}/notes"

headers = {'PRIVATE-TOKEN': GITLAB\_TOKEN}

data = {'body': analysis}

post\_resp = requests.post(url\_notes, headers=headers, data=data)

if post\_resp.status\_code == 201:

print(f"Posted new comment on MR {mr\_iid}")

return True

else:

print(f"Failed to post comment: {post\_resp.status\_code} {post\_resp.text}")

return False

def delete\_dynamo\_record(mr\_iid, project\_id):

try:

response = dynamodb.delete\_item(

TableName=DYNAMODB\_TABLE,

Key={

'mr\_iid': {'S': mr\_iid},

'project\_id': {'S': project\_id}

}

)

print(f"Deleted DynamoDB record for MR {mr\_iid}, Project {project\_id}")

return True

except Exception as e:

print(f"Failed to delete DynamoDB record: {e}")

return False

def lambda\_handler(event, context):

for record in event['Records']:

if record['eventName'] in ['INSERT', 'MODIFY']:

new\_image = record['dynamodb']['NewImage']

mr\_iid = new\_image['mr\_iid']['S']

project\_id = new\_image['project\_id']['S']

analysis = new\_image['analysis']['S']

print(f"Processing MR {mr\_iid} for project {project\_id}")

# Post comment to GitLab MR

success = post\_gitlab\_comment(project\_id, mr\_iid, analysis)

# If post successful, delete the record from DynamoDB

if success:

delete\_dynamo\_record(mr\_iid, project\_id)

return {

'statusCode': 200,

'body': json.dumps('Processing done.')

}

**Step 5: Deploy and Test**

* Save your Lambda function code.
* Make sure your Lambda role has the right permissions.
* Go to DynamoDB and insert or update an item in your MRAnalysisReport table:

Example item:

|  |  |  |
| --- | --- | --- |
| **mr\_iid** | **project\_id** | **analysis** |
| "123" | "mygroup/myproject" | "This is the latest analysis" |

* Check Lambda logs in **CloudWatch** to see if it ran without errors.
* Check your GitLab MR to see if the comment appears or updates.

**Summary for You:**

|  |  |
| --- | --- |
| **What you do** | **Why** |
| Enable DynamoDB Streams on your table | To notify Lambda of changes |
| Create a Lambda function with Python | To process changes and call GitLab API |
| Attach DynamoDB Stream as trigger | So Lambda runs automatically when your table changes |
| Write Lambda code to post/update GitLab comments | To send your analysis back to GitLab MR |
| Test by adding/updating DynamoDB entries | To verify the whole flow works |