**GitLab Webhook Listener Service (gitlab-webhook-listener)**

* + **Job:** This is the "front door" of your system. It *only* listens for incoming webhooks from GitLab.
  + **Why a separate service?** It needs to be super fast and highly available. If it's slow, GitLab will think your service is down and keep retrying. It shouldn't do heavy processing.
  + **Input:** HTTP POST requests from GitLab (webhook payloads).
  + **Output:** Sends a message to a "Message Queue" (like a shared inbox) telling another service that a new GitLab event happened.
  + **Technology:** Very lightweight web framework (e.g., Flask in Python).
  + **Example Action:** Receives an MR webhook, quickly checks its secret token, then puts a message like {"event\_type": "merge\_request", "project\_id": 123, "mr\_iid": 456, "action": "open"} onto a queue.

* + When you create a Merge Request in GitLab, GitLab sends a **webhook** (like a message) to your AWS Lambda function.
  + The Lambda function quickly checks the webhook and then **puts a message into the AWS SQS queue**.
  + The queue acts like a **mailbox**, holding all messages safely until the next service is ready.
  + The **Code Fetcher & Preprocessor** service reads messages from this queue one by one, fetches code from GitLab, and processes it.
  + This setup keeps your system fast and reliable because the webhook listener doesn’t wait for processing — it just delivers the message and responds quickly.

**🪜 STEP-BY-STEP: Modify Lambda to Push to SQS**

**✅ Prerequisites**

Before modifying the Lambda:

* + **AWS SQS Queue** — You’ll create one.
  + **IAM Role Permissions** — Lambda must have permission to write to SQS.

**✅ Step 1: Create SQS Queue**

* + Go to **AWS Console > SQS**: <https://console.aws.amazon.com/sqs>
  + Click **Create queue**
  + Choose **Standard Queue**
  + Name it something like gitlab-webhook-events
  + Click **Create Queue**
  + Copy the **Queue URL**

**✅ Step 2: Add IAM Permissions to Lambda**

* + Go to your **Lambda function**
  + Click the **Configuration** tab → **Permissions**
  + Click the **Execution role name** (it opens in IAM)
  + In the IAM role page, click **Add permissions > Attach policies**
  + Search for and add: **AmazonSQSFullAccess** (for now; we’ll restrict it later)

**✅ Step 3: Add Environment Variable for Queue URL**

* + Go back to your **Lambda function**
  + Configuration > **Environment variables** > Edit
  + Add:
    - SQS\_QUEUE\_URL → paste your SQS queue URL here

**✅ Step 4: Update Lambda Code to Send to SQS**

Replace your existing Lambda code with this:

python

CopyEdit

import json  
import os  
import boto3  
import urllib3

http = urllib3.PoolManager()  
sqs = boto3.client('sqs')

GITLAB\_TOKEN = os.environ['GITLAB\_TOKEN']  
SQS\_QUEUE\_URL = os.environ['SQS\_QUEUE\_URL']  
GITLAB\_API\_URL = "<https://gitlab.com/api/v4>"

def lambda\_handler(event, context):  
 try:  
 body = json.loads(event['body'])  
 print("Webhook received:", body)

# Extract necessary info from webhook  
 event\_type = event['headers'].get('X-Gitlab-Event', 'unknown')  
 project\_id = body['project']['id']  
 mr\_iid = body['object\_attributes']['iid']  
 action = body['object\_attributes']['action']

# Build message for SQS  
 message = {  
 "event\_type": "merge\_request",  
 "project\_id": project\_id,  
 "mr\_iid": mr\_iid,  
 "action": action  
 }

# Send to SQS  
 response = sqs.send\_message(  
 QueueUrl=SQS\_QUEUE\_URL,  
 MessageBody=json.dumps(message)  
 )

print("Message sent to SQS:", response)

return {  
 'statusCode': 200,  
 'body': json.dumps({'message': 'ok'})  
 }

except Exception as e:  
 print("Error:", str(e))  
 return {  
 'statusCode': 500,  
 'body': json.dumps({'message': 'error', 'error': str(e)})  
 }

Click **Deploy** after pasting the code.

**✅ Step 5: Test the New Setup**

* + Create a new Merge Request in your GitLab test project
  + Check:
    - **Lambda logs** in CloudWatch: it should show the webhook and successful SQS send
    - **SQS**: Go to the queue and check for messages (under "Monitoring" → “Number of messages available”)

**✅ Summary**

You now have a production-ready **GitLab Webhook Listener Service** that:

🎯 Accepts GitLab Merge Request webhooks

📨 Extracts key info

📬 Sends the message to SQS

⚡ Responds fast (GitLab is happy!)