**🧠 AI Code Review Service — Simple Architecture Overview**

We have built a **Code Review Service** that automatically analyzes code changes in a **GitLab Merge Request (MR)** using AI. The system is made of **3 independent services**, each doing a specific job:

**🟡 1. GitLab Webhook Listener (Trigger Service)**

* This service is triggered when a Merge Request (MR) is created in GitLab.
* We have set up a **GitLab Webhook** that sends MR details to an **AWS Lambda** function.
* The Lambda function reads the incoming JSON and extracts:
  + MR ID
  + Project ID
  + Action (like open, reopen, update, or merge)
* If the action is **one of these four** (open, reopen, update, merge), it sends a message to **AWS SQS** with this information.

🔹 **Input**: MR event from GitLab  
🔹 **Output**: Message in AWS SQS queue with MR details

**🟢 2. Gen-AI Review Service (Code Analyzer)**

This service runs as an **AWS Lambda with a Docker image**, which includes all AI-related dependencies and the Lambda handler.

* It listens to **SQS messages** (produced by the first service).
* When a new message arrives:
  + If the action is **open, reopen, or update**:
    -  Calls the GitLab API to fetch the MR code diffs (the files changed and the diff content).
    -  Passes each file’s diff through a **Retrieval-Augmented Generation (RAG)** pipeline:
    - This pipeline uses a **FAISS vector index** of the default branch codebase (pre-built by embedding all source code files into vector embeddings via OpenAI embeddings).
    - For each diff, the pipeline retrieves relevant code snippets from the indexed default branch code.
    - It generates a detailed AI-powered code review that includes high-level analysis, security issue detection, code smells, suggestions for improvement, and future risk assessment.

 Aggregates all AI-generated analyses into a comprehensive review.

* + If the action is **merge**:
    -  After the Merge Request is successfully merged into the default branch:
    - Only the **code that changed as part of the merge** is identified.
    - These changed files are **re-embedded** (i.e., reprocessed into vector representations using OpenAI embeddings).
    - The updated embeddings are used to **update the existing FAISS index**:
      * Old embeddings for changed files are removed or replaced.
      * New chunks and their vectors are added to the FAISS index.
    -  This ensures that the **retrieval context** used in future code reviews reflects the latest version of the codebase post-merge.
    -  This process avoids full re-indexing and focuses only on what was modified in the merge, improving efficiency.
* The final result is saved in **DynamoDB**, with:
  + MR ID
  + Project ID
  + Analysis – (AI-generated review text)

🔹 **Input**: Message from SQS  
🔹 **Output**: Code review stored in DynamoDB under MRAnalysisResults table

**Embedding Preprocessing Logic (Default Branch)**

* Prior to running the Gen-AI Review Service, the **default branch** of the repository is processed by a separate ingestion script.
* This script recursively scans all Python source files (.py) in the codebase.
* Each file is split into chunks of manageable size (~1000 characters with overlap).
* Each chunk is embedded using OpenAI embeddings to create vector representations.
* All vectors and metadata (chunk source, text) are saved to persistent storage:
  + FAISS index (index.faiss)
  + Chunk data (chunks.json)
  + Metadata (metadata.pkl)
* This index is loaded by the Gen-AI Review Service at runtime to provide relevant code context for diff analysis.

**🔵 3. GitLab Writer (Comment Poster)**

* **This service listens to DynamoDB Streams for new entries.**
* **When a new record is added (with MR ID, Project ID, and Analysis), it:**
  + **Uses a GitLab Private Token to authenticate**
  + **Finds the correct Merge Request**
  + **Posts the AI-generated Analysis as a comment on the MR**
* **Once the comment is successfully posted, the record is deleted from DynamoDB to avoid duplicate posts.**

**🔹 Input: New entry in DynamoDB  
🔹 Output: Comment added in GitLab MR**