Facial Recognition System - Project Explanation

Step 1: Laying the Foundation - Infrastructure Setup

I started by launching an EC2 instance, which acted as my main computing environment. This is where I configured the AWS CLI to interact with services like S3, Rekognition, DynamoDB, and Lambda. With the CLI set up, I created a Rekognition collection-a container for storing facial data-and cloned a GitHub repository that held the initial codebase and resources for the project.

Step 2: Building the Core - AWS Services Integration

Next, I configured the key AWS components. I created an S3 bucket to store the uploaded images, then a DynamoDB table to store the face metadata-like face IDs and image references. The interesting part was setting up a Lambda function that would automatically trigger every time a new image was uploaded to the S3 bucket. This function would call AWS Rekognition, detect and index any faces in the image, and then save those face prints and metadata into DynamoDB.

Step 3: Facial Recognition in Action

Once the pipeline was ready, I uploaded several images to the S3 bucket. This triggered the Lambda function, which processed the images in real-time-detecting faces and storing the corresponding data. I verified that the DynamoDB table was correctly storing the FaceId, image path, and related metadata.

Step 4: Bringing It All Together with Docker

Finally, I built and ran a Dockerized web application that interacted with these AWS components. The app allowed users to upload images and matched them against the stored face data. This helped demonstrate a full facial recognition workflow-from image input, through face detection and storage, to visual output via a web interface.

Key Takeaways

This project really taught me how to architect serverless workflows using AWS. I got hands-on experience with event-driven development using Lambda and S3 triggers, and I understood how to coordinate services like Rekognition, DynamoDB, and Docker. It gave me a strong foundation in designing scalable and decoupled systems.