# Real-Time Serverless Chat Application

## Introduction:

The **Real-Time Serverless Chat Application** is a cloud-based messaging platform designed to enable users to communicate instantly without managing any servers. This project leverages **AWS serverless technologies** to provide scalability, reliability, and cost efficiency. By integrating services such as **AWS Lambda**, **DynamoDB**, **API Gateway (WebSocket)**, **IAM**, and **S3**, the system ensures smooth real-time communication between users with minimal latency.

## Objective:

The main objective of this project is to develop a **fully serverless chat system** that allows real-time message exchange between users while maintaining scalability and security. The focus is on using AWS-managed services to eliminate infrastructure management and provide a highly available and efficient communication platform.

**Key goals:**

- Enable real-time messaging through WebSocket connections.
- Store and retrieve chat data efficiently using DynamoDB.
- Use AWS Lambda for backend logic without maintaining servers.
- Secure access through IAM roles and permissions.
- Host the frontend interface using S3 for easy deployment.

## System Architecture:

Components & Responsibilities:

**Frontend (S3 static site)**

- Hosts the UI (HTML/CSS/JS).
- Opens and maintains a WebSocket connection to the WebSocket API endpoint.
- Sends commands/events (e.g., connect, sendMessage, disconnect).
- Renders incoming messages and presence updates.

**API Gateway — WebSocket API**

- Acts as the WebSocket endpoint for clients.
- Routes messages to Lambda functions based on route keys (e.g., $connect, $disconnect, sendMessage).
- Provides a management API to push messages from backend to connected clients.

**AWS Lambda (backend handlers)**

- $connect handler: authenticate (optional), register connection in DynamoDB.
- $disconnect handler: clean up connection, update presence.
- sendMessage handler: validate message, persist to DynamoDB, broadcast to room participants using API Gateway management API (or enqueue to other Lambdas).

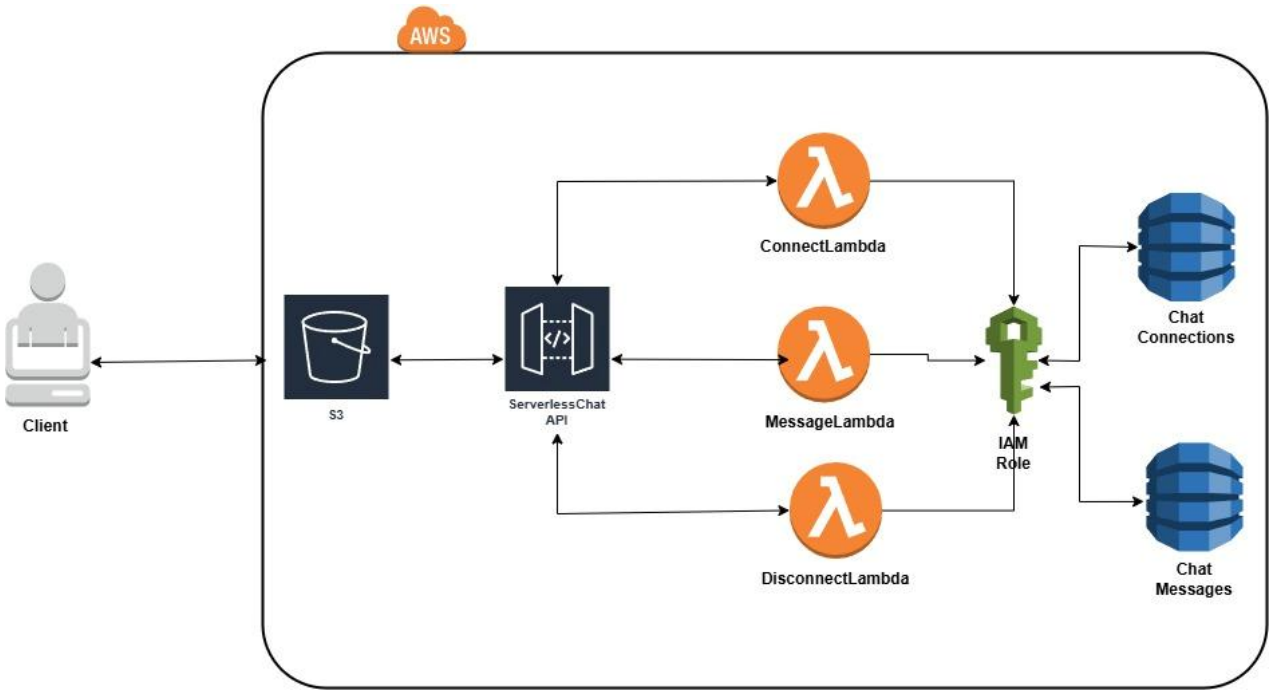
**DynamoDB**

- Stores chat messages (partitioned by roomId or conversationId) with timestamps.
- Stores connection records and presence metadata (connectionId -> userId, lastSeen, currentRoom).
- Designed with appropriate keys and indexes for efficient queries (e.g., GSI for recent messages).

**IAM**

- Fine-grained IAM roles for Lambda (permissions to read/write DynamoDB and call API Gateway management API).
- Least privilege for S3 read (public/private hosting) and any admin roles.

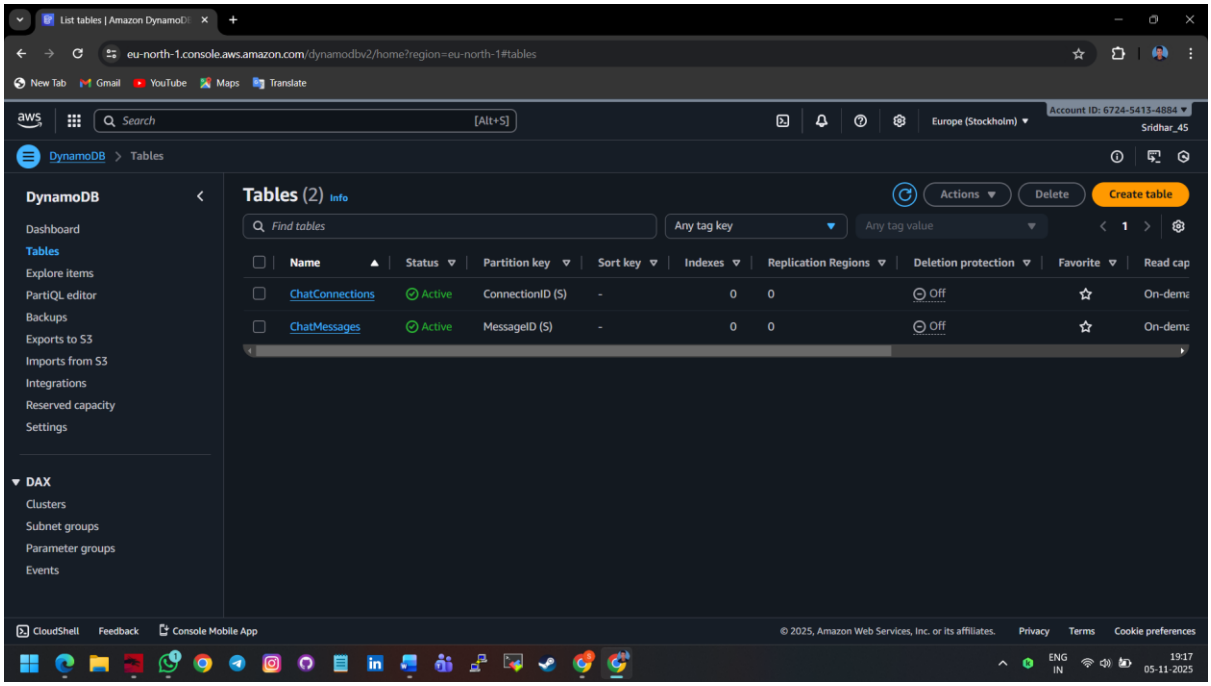
## Diagram:



## Steps Involved in this Project:

### Step1:

- ✓ Set Up the DynamoDB Table:
- ✓ Created a Two DynamoDB table



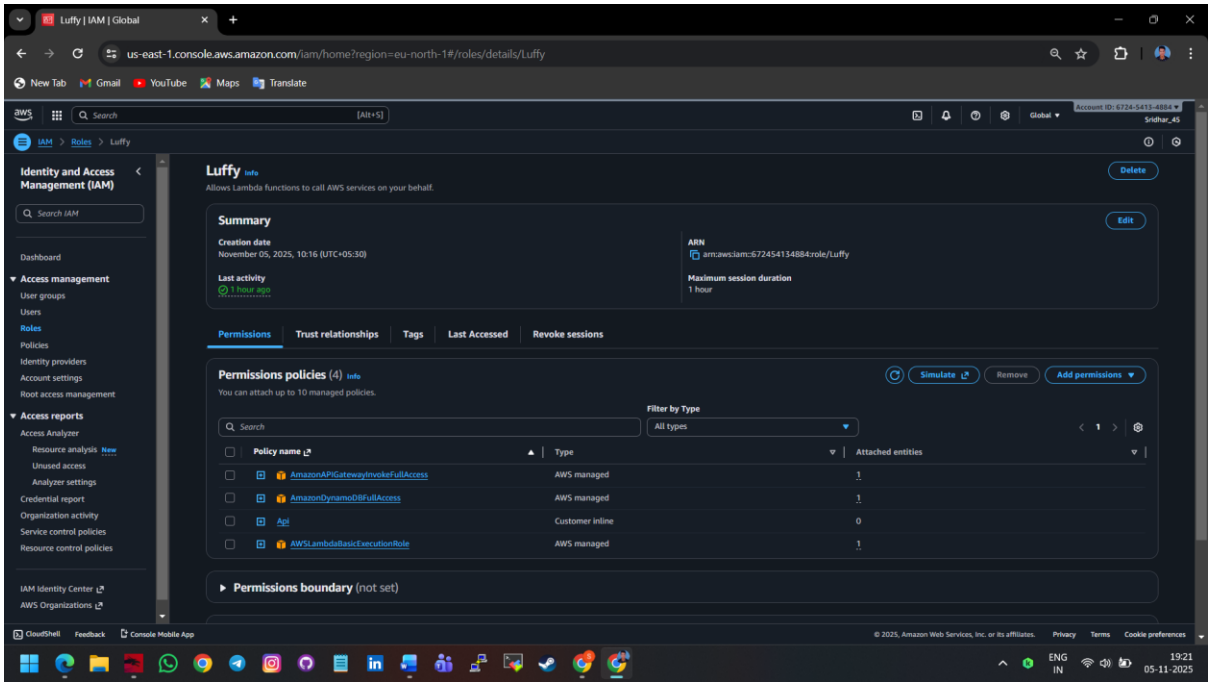

## Step 2: Create IAM role for Lambda Function:

Created an IAM role for Lambda with permissions to:

# Access DynamoDB tables.

Policies:

- ✓ AmazonDynamoDBFullAccess
- ✓ AWSlambdaBasicExecutionRole
- ✓ AmazonApiInvokeFullAccess

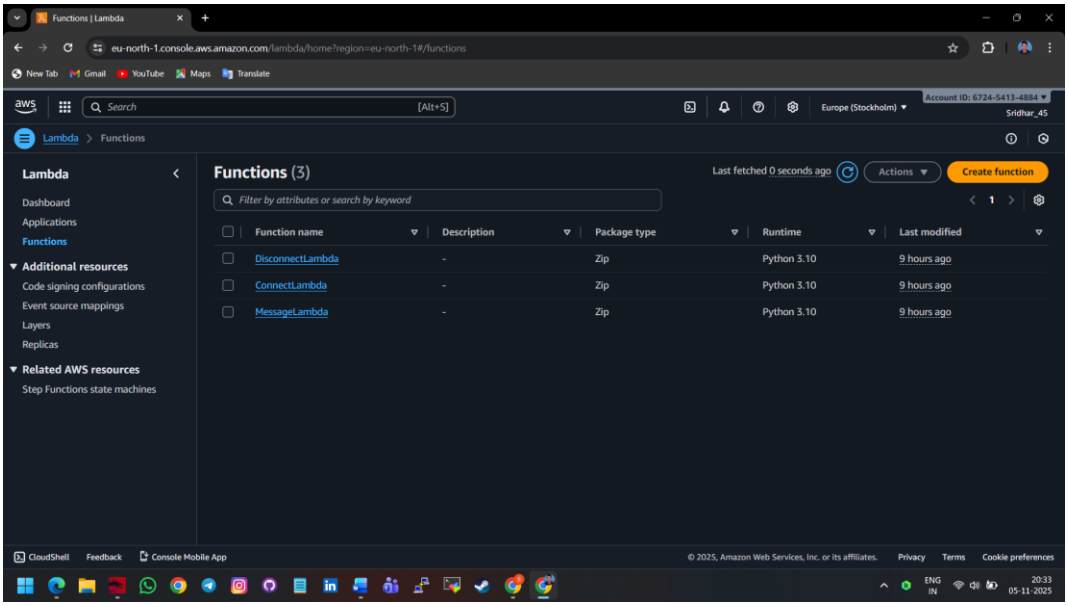



**IAM Role For Lambda**

## Step 3: Create a Lambda Function:

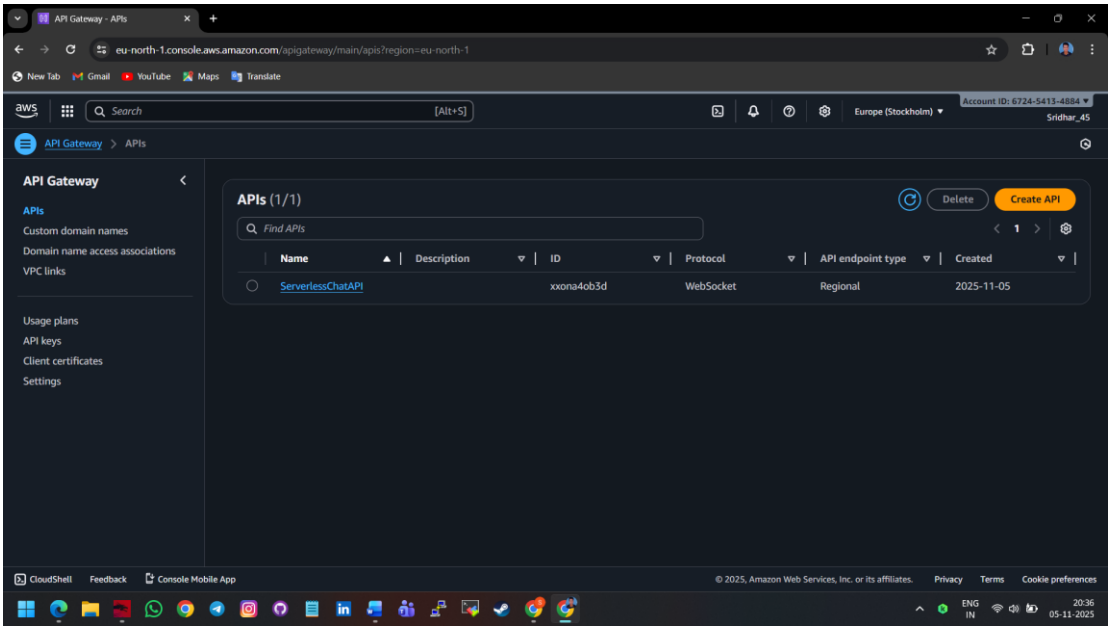
Created Lambda functions in Python to handle:

- ConnectLambda
- DisconnectLambda
- MessageLambda



## Step 4: API Creation With AWS API Gateway :

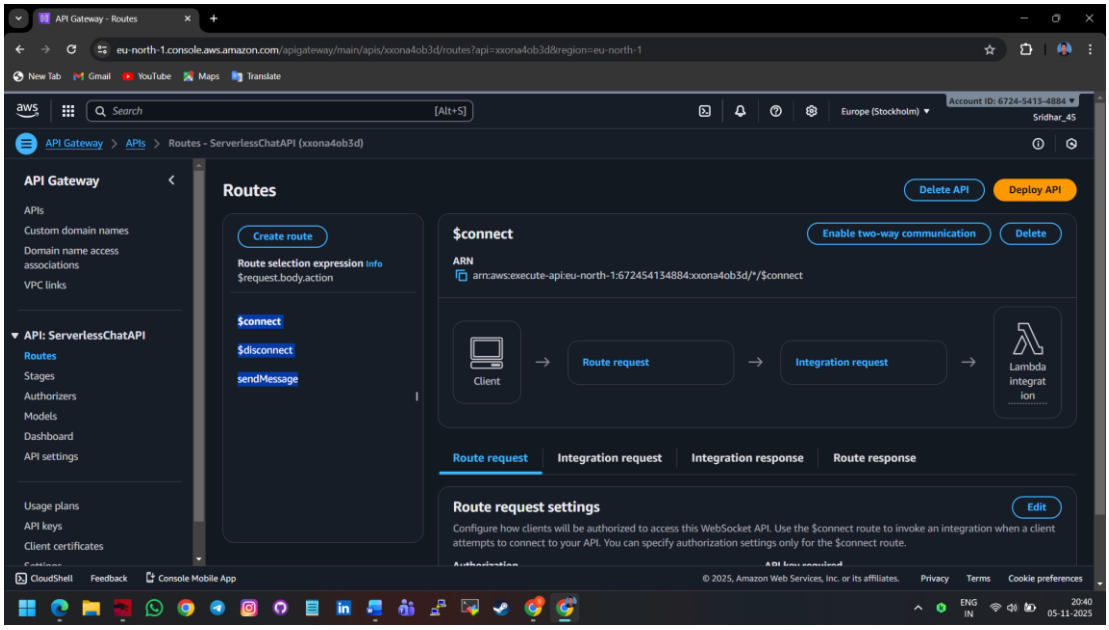
Created a WEBSOCKET API in API Gateway.




Serverless Named API Websocket

## Add Routes:

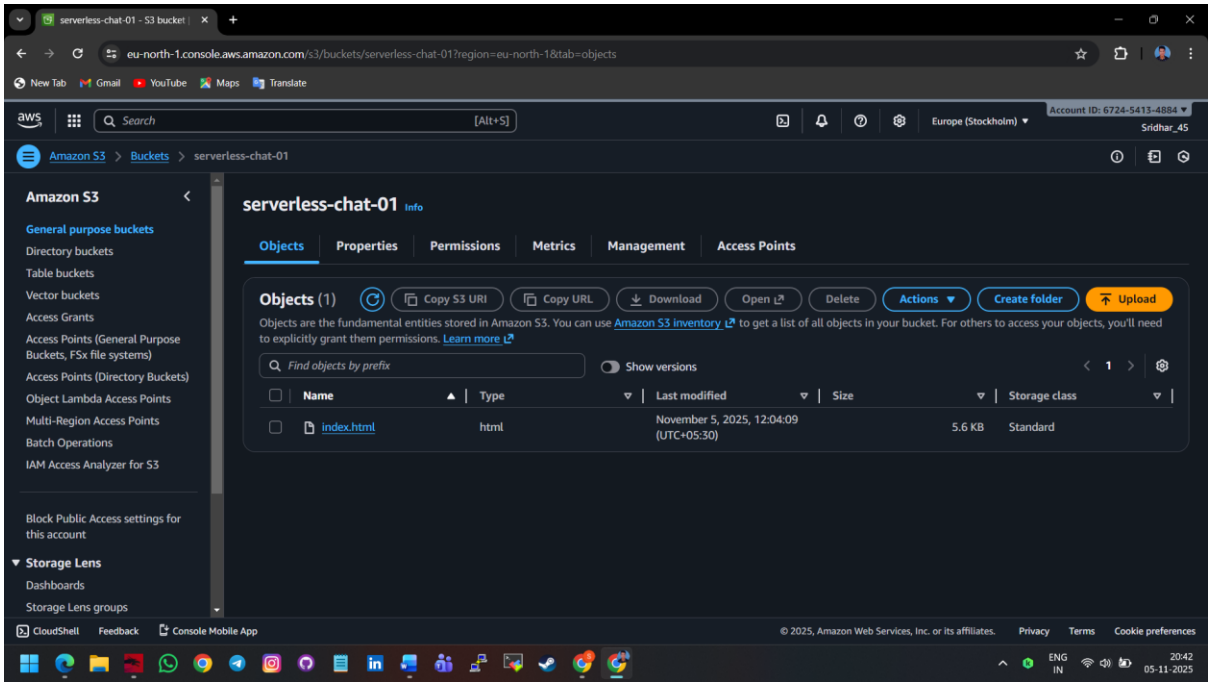
- $connect
- $disconnect
- SendMessage (Custom)



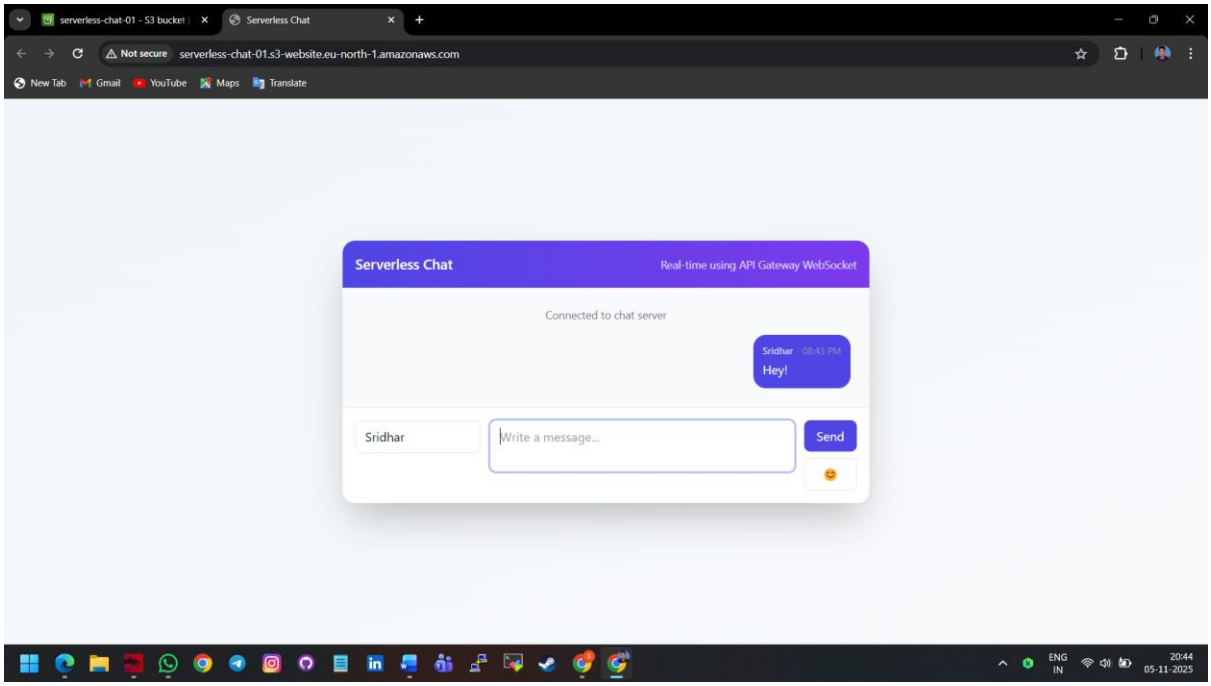

Three Routes

## Step 5: Hosted The Frontend on Amazon S3 :

- Created an S3 bucket to host the static web application files (HTML, CSS, JS).
- Enabled Static Website Hosting and uploaded all necessary frontend assets.




# Final Output:



## Conclusion:

- This project successfully demonstrates the power of AWS serverless architecture in building a scalable and real-time communication platform.
- By using DynamoDB, Lambda, API Gateway (WebSocket), IAM, and S3, the chat application delivers instant messaging without the need to manage servers.
- The solution is highly cost-effective, reliable, and easy to maintain, making it an ideal model for modern real-time applications.