File Upload and Metadata Management Dashboard Using LocalStack

Objective:

To develop a local dashboard for uploading files and retrieving their metadata, leveraging LocalStack to simulate AWS services like S3 and DynamoDB. This project aims to provide a cost-effective and efficient solution for managing file storage and metadata locally using lambda functions.

Problem Statement:

Managing file uploads and metadata in cloud environments often involves high costs and risks, especially during development and testing. we need a safe, local environment to test file storage workflows and metadata retrieval using lambda functions. without relying on live AWS services.

Proposed Solution:

This project focuses on creating a **dashboard** that allows users to:

1. Upload files to a simulated S3 bucket.
2. Lambda function that will trigger when ever file is uploaded or deleted for
3. Automatically store and retrieve metadata (e.g., file name, size, upload timestamp) from a simulated DynamoDB table.
4. Provide a user-friendly interface for managing files and their metadata.

The solution uses **LocalStack** to simulate AWS services locally, ensuring a cost-effective and risk-free development environment.

Key Features:

1. **File Upload**:
    - Users can upload files via the dashboard.
    - Files are stored in a simulated S3 bucket.
2. **Metadata Management**:
    - File metadata (e.g., name, size, timestamp) is automatically stored in a DynamoDB table using lambda function.
    - Metadata can be retrieved and displayed on the dashboard.
3. **Frontend Dashboard**:
    - Built with React and Vite for a responsive and user-friendly interface.
    - Displays uploaded files and their metadata in a tabular format.
4. **Event-Driven Updates**:
    - A Lambda function (simulated in LocalStack) triggers on file upload to update the DynamoDB table with metadata.
5. **LocalStack Integration**:
    - Simulates AWS services (S3, DynamoDB, Lambda) locally using Docker.

Technical Stack:

- **Frontend**: React, Vite, TailwindCSS
- **Backend**: Node.js, Express.js, aws_sdk, multer
- **AWS Simulation**: LocalStack (S3, DynamoDB, Lambda)
- **Infrastructure**: Docker, Docker Compose

Implementation Plan:

1. **Setup LocalStack:**
   - Configure Docker Compose to run LocalStack with S3, DynamoDB, and Lambda.

2. **Setup Commands:**
   - Figuring out terminal commands to create aws services (S3 bucket, lambda function, DynamoDB, and IAM roles), and connecting them. Uploading lambda function code, changing there configurations, and adding event notifications etc.
3. **Lambda Function:**
   - Write a Python-based Lambda function to handle S3 events and update DynamoDB with metadata.
4. **Develop Backend:**
   - Build APIs for file upload and metadata retrieval.
   - Integrate with LocalStack services (S3 and DynamoDB).
5. **Develop Frontend:**
   - Create a React-based dashboard for file upload and metadata display.
   - Implement features to list uploaded files and their metadata.
6. **Testing and Documentation:**
   - Test the system locally using LocalStack.
   - Provide clear documentation for setup and usage.

Expected Outcomes:

- A functional dashboard for uploading files and retrieving metadata.
- A local simulation of AWS services for cost-effective development.
- A streamlined workflow for managing file storage and metadata.

s

Budget:

- **Development Tools**: Free (Node.js, React, LocalStack Community Edition).
- **Infrastructure**: Local machine with Docker installed.

Diagram