AWS Serverless Task Manager - Setup & Deployment Manual

This manual provides step-by-step instructions for setting up and deploying the AWS Serverless Task Manager application, a serverless TODO application built using AWS Lambda and the Serverless Framework.

Prerequisites

Before beginning the setup process, ensure you have the following:

- Node.js installed on your system
- Active AWS account with appropriate permissions
- Auth0 tenant configured for authentication
- Git installed for version control

JSON Web Key Set (JWKS) Configuration

The application requires JWKS for JWT verification from Auth0. Your JWKS endpoint will follow this format:

```
https://{{domain}}.us.auth0.com/.well-known/jwks.json
```

Replace `[YOUR-DOMAIN]` with your actual Auth0 domain identifier.

Serverless Framework Installation

Install the Serverless Framework globally and verify the installation:

```
npm install -g serverless
serverless --version
serverless login
```

AWS Credentials Configuration

Configure AWS credentials using your IAM access keys:

```
sls config credentials --provider aws --key [ACCESS_KEY_ID] --secret [SECRET_ACCESS_KEY] --profile serverless -o
```

Replace the bracketed values with your actual AWS credentials.

Backend Deployment

Navigate to the backend directory and deploy the serverless infrastructure:

```
cd backend

# Set Node.js options for compatibility
export NODE_OPTIONS=--openssl-legacy-provider

# Install project dependencies
npm install

# Install required serverless plugin
npm install --save-dev serverless-iam-roles-per-function@next

# Address security vulnerabilities
npm audit fix

# Deploy the backend services
serverless deploy --verbose

# Alternative deployment with specific profile
sls deploy -v --aws-profile serverless
```

If you encounter TypeScript version conflicts, install the compatible version:

```
npm i typescript@4.6.4
```

Frontend Setup

Set up the client application:

```
cd client

# Install dependencies
npm install
npm install --save-dev

# Start development server
npm run start
```

Frontend Configuration

Edit the configuration file at `client/src/config.ts` with your specific values:

```
const apiId = 'your-api-gateway-id'
export const apiEndpoint = `https://${apiId}.execute-api.us-east-2.amazonaws.com/dev`

export const authConfig = {
  domain: 'your-auth0-domain.us.auth0.com',
  clientId: 'your-auth0-client-id',
  callbackUrl: 'http://localhost:3000/callback'
}
```

Replace the placeholder values with your actual:

- API Gateway ID (obtained from AWS Console after backend deployment)
- Auth0 domain
- Auth0 client ID

Verification

After successful deployment:

1. Backend API endpoints will be available at the configured API Gateway URL
2. Frontend development server will run on `http://localhost:3000`
3. Authentication will be handled through Auth0 integration

Troubleshooting

Dependency Issues: If you encounter module conflicts, remove and reinstall dependencies:

```
rm -rf node_modules
npm install
```

Deployment Failures: Ensure your AWS credentials have sufficient permissions for Lambda, API Gateway, and DynamoDB operations.

Auth0 Configuration: Verify that your Auth0 application settings match the callback URLs and allowed origins in your configuration.
