■ Deploy OITH Stripe Payments to AWS Lambda

This guide will help you deploy the Stripe payment function to AWS Lambda.

Prerequisites

1. **AWS Account** with Lambda access
2. **Stripe Account** with API keys (stripe.com)
3. **DynamoDB Table** already created (from userSync deployment)

Step 1: Get Your Stripe Keys

1. Go to Stripe Dashboard
2. Copy your keys:
- **Secret key**: `sk_test_xxx` (for testing) or `sk_live_xxx` (for production)
- You already have your publishable key in the app

Step 2: Create Lambda Function

1. Go to AWS Lambda Console
2. Click **"Create function"**
3. Choose **"Author from scratch"**
4. Fill in:
- **Function name:** `oith-payments`
- **Runtime:** `Node.js 20.x`
- **Architecture:** `x86_64`
5. Click **"Create function"**

Step 3: Add Stripe Layer

Since Lambda needs the `stripe` npm package, you have two options:

Option A: Use a Pre-built Layer (Easiest)

1. In your Lambda function, click **"Layers"**
2. Click **"Add a layer"**
3. Choose **"Specify an ARN"**
4. Use this public Stripe layer ARN (us-east-1):
`arn:aws:lambda:us-east-1:770693421928:layer:Klayers-p312-stripe:1`
(If in different region, search for "Klayers stripe" for your region)

Option B: Create Your Own Layer

1. On your computer, create a folder:
`bash mkdir nodejs && cd nodejs npm init -y npm install stripe cd .. zip -r stripe-layer.zip nodejs`
2. In AWS Lambda Console, go to **Layers** → **Create layer**
3. Upload `stripe-layer.zip`
4. Add this layer to your function

Step 4: Add Lambda Code

1. In the Lambda function page, scroll to **"Code source"**
2. Delete the default code
3. Copy the **entire contents** of `paymentHandler.mjs` and paste it
4. **Important:** Rename the file from `index.mjs` to `paymentHandler.mjs` or update the handler
5. Click **"Deploy"**

Set Handler

1. Go to **Runtime settings** → **Edit**
2. Set **Handler** to: `paymentHandler.handler`
3. Click **"Save"**

Step 5: Configure Environment Variables

1. Go to **Configuration** → **Environment variables**
2. Click **"Edit"**
3. Add these variables:

Key	Value
`STRIPE_SECRET_KEY`	`sk_test_51Sct6c...` (your secret key)
`DYNAMODB_TABLE`	`oith-users`
`DOMAIN`	`https://main.d3cpep2ztx08x2.amplifyapp.com`

4. Click **"Save"**

Step 6: Add Permissions

1. Go to **Configuration** → **Permissions**
2. Click on the **Role name** link
3. Click **"Add permissions"** → **"Attach policies"**
4. Add these policies:
- `AmazonDynamoDBFullAccess`
5. Click **"Add permissions"**

Step 7: Increase Timeout

1. Go to **Configuration** → **General configuration**
2. Click **"Edit"**
3. Set **Timeout** to `30 seconds` (Stripe API calls can take time)
4. Click **"Save"**

Step 8: Create API Gateway

1. Go to AWS API Gateway Console

2. Click **"Create API"**
3. Choose **"HTTP API"** → **"Build"**
4. Click **"Add integration"**:
- **Integration type:** Lambda
- **Lambda function:** `oith-payments`
5. **API name:** `oith-payments-api`
6. Click **"Next"**

Configure Routes

Add these routes (all pointing to your Lambda):

Method	Path
GET	`/api/health`
GET	`/api/plans`
POST	`/api/create-payment-intent`
POST	`/api/create-checkout-session`
GET	`/api/verify-payment/{sessionId}`
GET	`/api/subscription/{customerId}`
POST	`/api/cancel-subscription`
POST	`/webhook`

7. Click **"Next"** → **"Next"** → **"Create"**

Step 9: Enable CORS

1. In API Gateway, go to **CORS**
2. Click **"Configure"**
3. Set:
- **Access-Control-Allow-Origin:** `*`
- **Access-Control-Allow-Headers:** `*`

- **Access-Control-Allow-Methods:** `GET, POST, PUT, DELETE, OPTIONS`

4. Click **"Save"**

Step 10: Get Your API URL

After creating, you'll see an **Invoke URL** like:

```
https://xxxxxxxxxx.execute-api.us-east-1.amazonaws.com
```

Copy this URL!

Step 11: Update Your App

Update `app.js` with your API URL. Find this section:

```
const PAYMENT_CONFIG = { ... apiUrl: 'http://localhost:3000/api',
```

Change it to:

```
const PAYMENT_CONFIG = { ... apiUrl:
'https://xxxxxxxxxx.execute-api.us-east-1.amazonaws.com/api',
```

Then commit and push:

```
git add prototype/app.js git commit -m "Update payment API URL for production" git push
origin main
```

Step 12: Test It!

1. Open your app: https://main.d3cpep2ztx08x2.amplifyapp.com/prototype/index.html
2. Go through registration to the payment screen
3. Use test card: `4242 4242 4242 4242`
4. Any future expiry date and any CVC
5. Payment should process successfully!

■ Done!

Your payment system is now live on AWS!

Optional: Set Up Webhooks

For real-time subscription updates:

1. Go to Stripe Webhooks
2. Click **"Add endpoint"**
3. Set **Endpoint URL**: `https://xxxxxxxxxx.execute-api.us-east-1.amazonaws.com/webhook`
4. Select events:
- `checkout.session.completed`
- `customer.subscription.updated`
- `customer.subscription.deleted`
- `invoice.payment_failed`
- `invoice.paid`
5. Copy the **Signing secret** (`whsec_xxx`)
6. Add to Lambda environment variables:
- `STRIPE_WEBHOOK_SECRET`: `whsec_xxx`

Troubleshooting

"Module not found: stripe"

- Make sure you added the Stripe layer to your Lambda function

CORS Errors

- Double-check API Gateway CORS settings
- Ensure Lambda returns proper CORS headers

"Invalid API Key"

- Verify `STRIPE_SECRET_KEY` environment variable is set correctly

- Make sure you're using the secret key (starts with `sk_`), not publishable key

Payment Not Saving to Database

- Check CloudWatch Logs for errors
- Verify DynamoDB permissions are attached

Timeout Errors

- Increase Lambda timeout to 30 seconds
- Check CloudWatch for slow operations

Test Cards

Card Number	Result
4242 4242 4242 4242	Success
4000 0000 0000 3220	3D Secure required
4000 0000 0000 0002	Decline

Use any future expiry date and any 3-digit CVC.