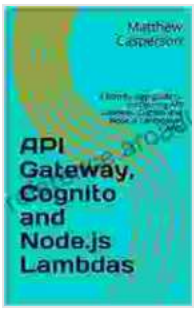
Step-by-Step Guide to Configuring API Gateway, Cognito, and Node.js Lambdas



API Gateway, Cognito and Node.js Lambdas: A step by step guide to configuring API Gateway, Cognito and Node.js Lambdas in AWS (AWS Cloud Guides)

by Brennien Coker

★★★★★ 5 out of 5

Language	: English
File size	: 16089 KB
Text-to-Speech	: Enabled
Screen Reader	: Supported
Enhanced typesetting	: Enabled
Print length	: 62 pages
Lending	: Enabled

In this guide, we will walk through the steps of setting up and configuring Our Book Library API Gateway, Our Book Library Cognito, and Node.js Lambdas to create a secure and scalable serverless application. This guide assumes that you have a basic understanding of AWS and serverless architecture.

Prerequisites

- An AWS account
- Node.js installed on your local machine
- The AWS CLI installed and configured

Step 1: Create an API Gateway REST API

1. Open the AWS Management Console and navigate to the API Gateway service.

2. Click on the "Create API" button.

3. Select the "REST API" option and click on the "Create API" button.

4. Enter a name for your API and click on the "Create" button.

Step 2: Create a Cognito User Pool

1. Open the AWS Management Console and navigate to the Cognito service.

2. Click on the "Create a user pool" button.

3. Enter a name for your user pool and click on the "Create" button.

4. On the "General settings" page, configure the following settings:

- Pool name: Enter a name for your user pool.
- Domain name: Enter a domain name for your user pool (e.g., example.auth.us-east-1.Our Book Librarycognito.com).
- App client name: Enter a name for your app client (e.g., myapp).

5. Click on the "Create pool" button.

Step 3: Create a Cognito Identity Pool

1. Open the AWS Management Console and navigate to the Cognito service.

2. Click on the "Create an identity pool" button.

3. Enter a name for your identity pool and click on the "Create" button.

4. On the "Identity pool settings" page, configure the following settings:

- Identity pool name: Enter a name for your identity pool.
- Authentication providers: Select the "Our Book Library Cognito user pools" option and select your user pool from the dropdown list.

5. Click on the "Create pool" button.

Step 4: Create a Node.js Lambda function

1. Open your text editor and create a new file named `index.js`.

2. Paste the following code into the file:

javascript const AWS = require('aws-sdk');

exports.handler = async (event) => { const identity = await new AWS.CognitoIdentity({apiVersion: '2014-06-30'}).getIdentityPoolForUser({IdentityPoolId: process.env.AWS_COGNITO_IDENTITY_POOL_ID, Logins: {[`cognito-idp.${process.env.AWS_COGNITO_REGION}.Our Book Libraryaws.com/${process.env.AWS_COGNITO_USER_POOL_ID}`]: event.requestContext.authorizer.claims.sub}}).promise();

// Get the user's attributes from the Cognito user pool const cognito = new AWS.CognitoIdentityServiceProvider({apiVersion: '2016-04-18'}); const user = await cognito.getUser({AccessToken: event.requestContext.authorizer.claims.access_token}).promise();

return { statusCode: 200, body: JSON.stringify({ message: `Hello, ${user.Username}!`, identityId: identity.IdentityId, attributes: user.Attributes }) }; };

3. Save the file.

Step 5: Deploy the Lambda function

1. Open the AWS Management Console and navigate to the Lambda service.

2. Click on the "Create function" button.

3. Select the "Author from scratch" option and click on the "Next" button.

4. Enter a name for your function (e.g., myfunction).

5. Select the "Runtime" option (e.g., Node.js 16.x).

6. Select the "Execution role" option (e.g., Create a new role from AWS policy templates).

7. Select the "Policy template" option (e.g., Our Book Library Cognito Authorizer).

8. Click on the "Create function" button.

Step 6: Connect API Gateway to the Lambda function

1. Open the AWS Management Console and navigate to the API Gateway service.

2. Click on the "APIs" link in the sidebar.

3. Click on the name of the API that you created in Step 1.

4. Click on the "Resources" link in the sidebar.

5. Click on the "Create Resource" button.

6. Enter a name for your resource (e.g., myresource).

7. Click on the "Create" button.

8. Click on the "Methods" link in the sidebar.

9. Click on the "Create Method" button.

10. Select the "GET" method.

11. Click on the "Integration" tab.

12. Select the "Lambda Function" option.

13. Enter the ARN of your Lambda function (e.g., arn:aws:lambda:us-east-1:123456789012:function:myfunction).

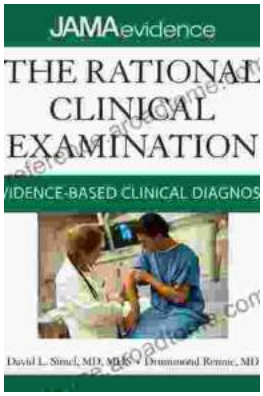
14. Click on the "Save" button.

Step 7: Test the API

1. Open a terminal window and curl the API endpoint (e.g., curl https://example.execute-api.us-east-1.Our Book Libraryaws.com/myresource).

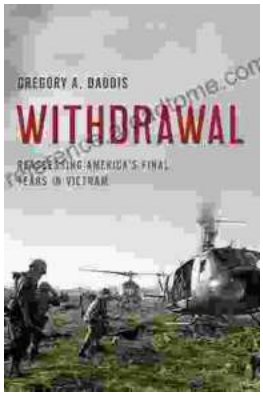
2. You should see a response from the API.

In this guide, we walked through the steps of setting up and configuring API Gateway, Cognito, and Node.js Lambdas to create a secure and scalable serverless application. This guide provides a solid foundation for building more complex serverless applications.



Unlock the Secrets of Accurate Clinical Diagnosis: Discover Evidence-Based Insights from JAMA Archives Journals

Harnessing the Power of Scientific Evidence In the ever-evolving landscape of healthcare, accurate clinical diagnosis stands as the cornerstone of...



Withdrawal: Reassessing America's Final Years in Vietnam

The Controversial Withdrawal The withdrawal of American forces from Vietnam was one of the most controversial events in American history. The war...