Amplify Gen 2: Custom Python function with CDK

Mikhail Sliusarev Follow 3 min read · Aug 11, 2024

In this article, I'll walk you through the process of adding a custom Python AWS Lambda function to your AWS Amplify project.

Amplify is a powerful tool for building full-stack applications on AWS. Currently its Gen 2 version does not support deploying Python Lambda functions out of the box (comparing to Gen 1 when it was possible). However, with a few extra steps, you can integrate a Python Lambda into your Amplify workflow using CDK, because Gen 2 hinges on CDK for deploying resources.

While the example project is a Flutter app, the approach shown here is applicable to any Amplify setup, whether you're using React Native, Vue, or another Amplify supported framework. Simply replace `flutter create` with the appropriate initialization command for your framework.

This guide assumes you have an AWS account configured and Node.js installed and have a some experience working with Amplify Gen 2.

Create a project

First, let's create an empty project and initialize Amplify.

```
flutter create amplify_custom_python_function
cd myproject
flutter run
npm create amplify@latest -y
```

At this point, you should have a working app with Amplify initialized. If you check the `amplify` subfolder within your project, you'll find the `Auth` and `Data` modules, as well as the `backend.ts` file, which combines these modules and defines your backend configuration.

```
> tree amplify
amplify
├── auth
│   └── resource.ts
├── backend.ts
├── data
│   └── resource.ts
├── package.json
└── tsconfig.json

2 directories, 5 files
```

Introduce custom functions

Let's create a new folder to store our custom Lambda functions and start by creating a simple "Hello World" Lambda function.

```
cd amplify_custom_python_function/amplify/
mkdir custom-functions
mkdir custom-functions/helloworld
cd custom-functions/helloworld
touch index.py
```

Inside the `index.py` file, add a basic Python Lambda function handler. You can use the following code to get started.

```
def handler(event, context):
    print("Event: ", event)

    return {
        'statusCode': 200,
        'body': 'Hello from Lambda!'
    }
```

Configure CDK stack

Custom functions is a new Amplify module in our project and you'll need to add a `resources.ts` file to it. This file acts as the module's configuration within the Amplify ecosystem.

```
cd amplify_custom_python_function/amplify/custom-functions
touch resources.ts
```

For this guide, we'll keep our CDK stack, which deploys our Lambda function, in the `custom-functions` folder inside `resources.ts` . When defining the stack, you'll need to specify the path from the project root to the folder containing `index.py` so that the stack can locate and package the Lambda content correctly.

```
import { CfnOutput, Stack, StackProps, Duration } from 'aws-cdk-lib';
import { Construct } from 'constructs';
import * as lambda from 'aws-cdk-lib/aws-lambda';
import * as path from 'path';

export class HelloWorldLambdaStack extends Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, props);
    // Define the Lambda function
    const helloWorldFunction = new lambda.Function(this, 'HelloWorldFunction', {
      runtime: lambda.Runtime.PYTHON_3_9, // Specify the runtime
      handler: 'index.handler',           // Specify the handler function
      code: lambda.Code.fromAsset('./amplify/custom-functions/helloworld'),
      functionName: 'HelloWorldFunction',
      description: 'This is my custom Lambda function created using CDK',
      timeout: Duration.seconds(30),
      memorySize: 128,
      environment: {
        TEST: 'test',
      },
    });

    // Output the Lambda function ARN
    new CfnOutput(this, 'HellowWorldFunctionArn', {
      value: helloWorldFunction.functionArn,
      exportName: 'HelloWorldFunctionArn',
    });
  }
}
```

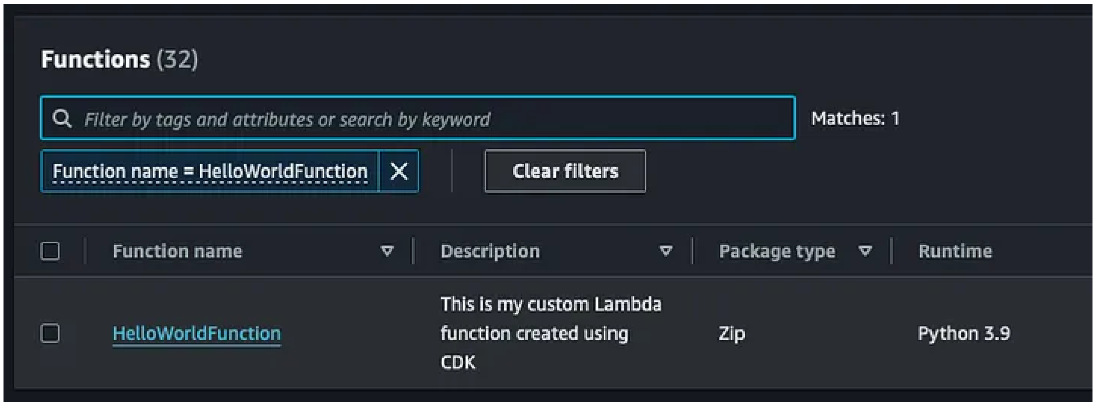
Next, register this CDK configuration with Amplify to ensure the resource is created when you deploy your sandbox.

```
import { defineBackend } from '@aws-amplify/backend';
import { auth } from './auth/resource';
import { data } from './data/resource';
import { HelloWorldLambdaStack } from './custom-functions/resources';

const backend = defineBackend({
  auth,
  data,
});

// Add the HelloWorld custom Lambda stack to the backend
new HelloWorldLambdaStack(
  backend.createStack('HelloWorldLambdaStack'),
  'helloWorldLambdaResource',
  {}
);
```

Once the function is configured, deploy it just as you would with any Amplify Gen 2 infrastructure.

```
npx ampx sandbox --outputs-format dart --outputs-out-dir lib
```

After deployment, you can find your function in AWS, ready to be used.



Functions (32)

Matches: 1

Function name = HelloWorldFunction

□	Function name	Description	Package type	Runtime
□	HelloWorldFunction	This is my custom Lambda function created using CDK	Zip	Python 3.9

Conclusion

I hope this guide helped you understand the basics of configuring a custom Python Lambda function with AWS Amplify Gen 2.

While these steps provide a simple way to get started, real-world projects will involve additional complexities, such as managing dependencies, writing unit tests, and mocking Lambda function events.

Amplify AWS Lambda Function Python

Written by Mikhail Sliusarev

5 followers · 3 following

Follow

Responses (5)

Tehfiestygoat

What are your thoughts?

chris.write
Oct 2, 2024

Thanks a lot!! But how do I invoke that lambda from UI?

15 Reply

Adam Cooke
Dec 29, 2024

this would be much more useful if it showed how to use lambda layers or a without and also how to reference within the data/resource.ts file

Reply

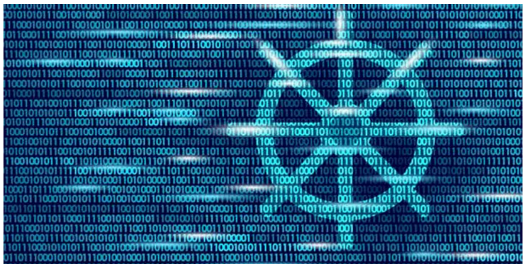
Jim McGowen
Nov 20, 2024

What file does the code that starts with "import { defineBackend }" go in?

Reply

See all responses

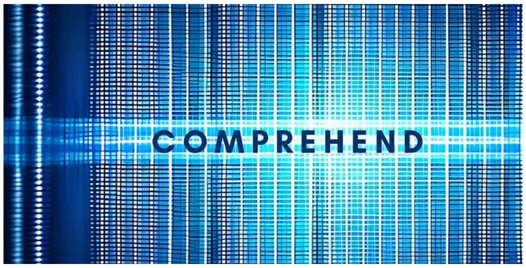
More from Mikhail Sliusarev



Mikhail Sliusarev

Set up a k8s cluster on DigitalOcean

Setting up k8s could be daunting for a newcomer. In this article, I tried reducing the...



Mikhail Sliusarev

Get ICD codes in Flutter with Comprehend Medical and Amplify

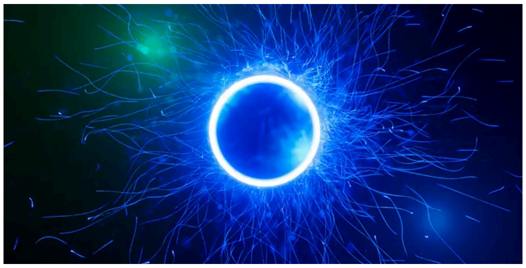
ICD 10 — is a structured way to cluster diagnoses. Along with LOINC and SNOMED...

Jun 30, 2023 1

Jul 22, 2023 1

See all from Mikhail Sliusarev

Recommended from Medium



Will Lockett

The AI Bubble Is About To Burst, But The Next Bubble Is Already...

Techbros are preparing their latest bandwagon.

✦ Sep 14 11.9K 387

Yashwanth Sai

I have built around 300 agents, worked at 5 startups. Here's what ...

Lessons learnt after working with agents for over an year.

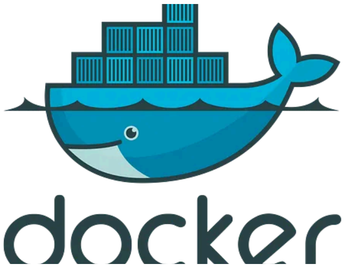
✦ Aug 23 4.2K 160

Nikita Palnov

Amazon Bedrock — Using RAG & Knowledge Bases (pt.1 — Basics)

Welcome !

May 3 · 21

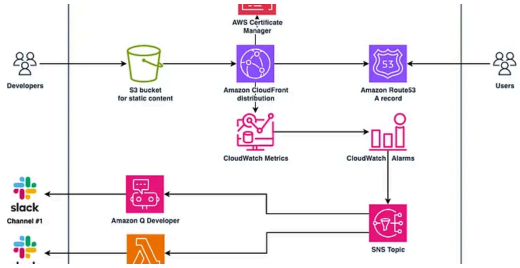


Abhinav

Docker Is Dead — And It's About Time

Docker changed the game when it launched in 2013, making containers accessible and...

Jun 8 · 6.6K · 186



Andrii Shykhov

Send CloudWatch alarm notifications to Slack via Amazon ...

Monitor a static website hosted on Amazon CloudFront and deliver CloudWatch alarm...

Sep 14 · 1

Yash Batra

The 47-Line Code That Made One Developer $2 Million from AI

In late 2024, a solo indie developer pushed 47 lines of Python code to GitHub.

Jul 8 · 4.1K · 122

See more recommendations