[#hello-world-function] == Step 6: Define your Lambda function

In this step, you import the link:https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.aws\_lambda-readme.html[aws\_lambda] module from the {aws} Construct Library and use the link:https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.aws\_lambda.Function.html[Function] L2 construct.

Modify your CDK stack file as follows:

==== [role=“tablist”] TypeScript:: Located in lib/hello-cdk-stack.ts: + [source,javascript,subs=“verbatim,attributes”] — import \* as cdk from ‘aws-cdk-lib’; import { Construct } from ‘constructs’; // Import the Lambda module import \* as lambda from ‘aws-cdk-lib/aws-lambda’;

export class HelloCdkStack extends cdk.Stack { constructor(scope: Construct, id: string, props?: cdk.StackProps) { super(scope, id, props);

// Define the Lambda function resource const myFunction = new lambda.Function(this, “HelloWorldFunction”, { runtime: lambda.Runtime.NODEJS\_20\_X, // Provide any supported Node.js runtime handler: “index.handler”, code: lambda.Code.fromInline(exports.handler = async function(event) { return { statusCode: 200, body: JSON.stringify('Hello World!'), }; };), }); } } —-

JavaScript:: Located in lib/hello-cdk-stack.js: + [source,javascript,subs=“verbatim,attributes”] — const { Stack } = require(‘aws-cdk-lib’); // Import the Lambda module const lambda = require(‘aws-cdk-lib/aws-lambda’);

class HelloCdkStack extends Stack { constructor(scope, id, props) { super(scope, id, props);

// Define the Lambda function resource const myFunction = new lambda.Function(this, “HelloWorldFunction”, { runtime: lambda.Runtime.NODEJS\_20\_X, // Provide any supported Node.js runtime handler: “index.handler”, code: lambda.Code.fromInline(exports.handler = async function(event) { return { statusCode: 200, body: JSON.stringify('Hello World!'), }; };), });

} }

== module.exports = { HelloCdkStack }

Python:: Located in hello\_cdk/hello\_cdk\_stack.py: + [source,python,subs=“verbatim,attributes”] — from aws\_cdk import ( Stack, aws\_lambda as \_lambda, # Import the Lambda module ) from constructs import Construct

class HelloCdkStack(Stack):

def *init*(self, scope: Construct, construct\_id: str, **kwargs) -> None: super().*init*(scope, construct\_id,** kwargs)

# Define the Lambda function resource my\_function = \_lambda.Function( self, “HelloWorldFunction”, runtime = \_lambda.Runtime.NODEJS\_20\_X, # Provide any supported Node.js runtime handler = “index.handler”, code = \_lambda.Code.from\_inline( ““” exports.handler = async function(event) { return { statusCode: 200, body: JSON.stringify(‘Hello World!’), }; }; ““” ), ) —-

Java:: Located in +src/main/java/.../HelloCdkStack.java+: + [source,java,subs=“verbatim,attributes”] — package com.myorg;

import software.constructs.Construct; import software.amazon.awscdk.Stack; import software.amazon.awscdk.StackProps; // Import Lambda function import software.amazon.awscdk.services.lambda.Code; import software.amazon.awscdk.services.lambda.Function; import software.amazon.awscdk.services.lambda.Runtime;

public class HelloCdkStack extends Stack { public HelloCdkStack(final Construct scope, final String id) { this(scope, id, null); }

public HelloCdkStack(final Construct scope, final String id, final StackProps props) { super(scope, id, props);

// Define the Lambda function resource Function myFunction = Function.Builder.create(this, “HelloWorldFunction”) .runtime(Runtime.NODEJS\_20\_X) // Provide any supported Node.js runtime .handler(“index.handler”) .code(Code.fromInline( “exports.handler = async function(event) {” + ” return {” + ” statusCode: 200,” + ” body: JSON.stringify(‘Hello World!’)” + ” };” + “};”)) .build();

} } —

C#:: Located in +src/main/java/.../HelloCdkStack.java+: + [source,csharp,subs=“verbatim,attributes”] — using Amazon.CDK; using Constructs; // Import the Lambda module using Amazon.CDK.{aws}.Lambda;

namespace HelloCdk { public class HelloCdkStack : Stack { internal HelloCdkStack(Construct scope, string id, IStackProps props = null) : base(scope, id, props) { // Define the Lambda function resource var myFunction = new Function(this, “HelloWorldFunction”, new FunctionProps { Runtime = Runtime.NODEJS\_20\_X, // Provide any supported Node.js runtime Handler = “index.handler”, Code = Code.FromInline(@” exports.handler = async function(event) { return { statusCode: 200, body: JSON.stringify(‘Hello World!’), }; }; “), }); } } } —

Go:: Located in hello-cdk.go: + [source,go,subs=“verbatim,attributes”] — package main

import ( “github.com/aws/aws-cdk-go/awscdk/v2” “github.com/aws/constructs-go/constructs/v10” “github.com/aws/jsii-runtime-go” // Import the Lambda module “github.com/aws/aws-cdk-go/awscdk/v2/awslambda” )

type HelloCdkStackProps struct { awscdk.StackProps }

func NewHelloCdkStack(scope constructs.Construct, id string, props \*HelloCdkStackProps) awscdk.Stack { var sprops awscdk.StackProps if props != nil { sprops = props.StackProps } stack := awscdk.NewStack(scope, &id, &sprops)

// Define the Lambda function resource myFunction := awslambda.NewFunction(stack, jsii.String(“HelloWorldFunction”), &awslambda.FunctionProps{ Runtime: awslambda.Runtime\_NODEJS\_20\_X(), // Provide any supported Node.js runtime Handler: jsii.String(“index.handler”), Code: awslambda.Code\_FromInline(jsii.String(exports.handler = async function(event) { return { statusCode: 200, body: JSON.stringify('Hello World!'), }; };)), })

return stack }

== // …

====

Let’s take a closer look at the Function construct. Like all constructs, the Function class takes three parameters:

* *scope* – Defines your Stack instance as the parent of the Function construct. All constructs that define {aws} resources are created within the scope of a stack. You can define constructs inside of constructs, creating a hierarchy (tree). Here, and in most cases, the scope is this (self in Python).
* *Id* – The construct ID of the Function within your {aws} CDK app. This ID, plus a hash based on the function’s location within the stack, uniquely identifies the function during deployment. The {aws} CDK also references this ID when you update the construct in your app and re-deploy to update the deployed resource. Here, your construct ID is HelloWorldFunction. Functions can also have a name, specified with the functionName property. This is different from the construct ID.
* *props* – A bundle of values that define properties of the function. Here you define the runtime, handler, and code properties.
* Props are represented differently in the languages supported by the {aws} CDK.
* \*\* In TypeScript and JavaScript, props is a single argument and you pass in an object containing the desired properties. \*\* In Python, props are passed as keyword arguments. \*\* In Java, a Builder is provided to pass the props. There are two: one for FunctionProps, and a second for Function to let you build the construct and its props object in one step. This code uses the latter. \*\* In C#, you instantiate a FunctionProps object using an object initializer and pass it as the third parameter.

If a construct’s props are optional, you can omit the props parameter entirely.

All constructs take these same three arguments, so it’s easy to stay oriented as you learn about new ones. And as you might expect, you can subclass any construct to extend it to suit your needs, or if you want to change its defaults.