[#tokens-passing] == Passing tokens

Tokens can be passed around as if they were the actual value they represent. The following is an example that passes the token for our bucket name to a construct for an {aws} Lambda function:

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — import \* as cdk from ‘aws-cdk-lib’; import { Construct } from ‘constructs’; import \* as s3 from ‘aws-cdk-lib/aws-s3’; import \* as lambda from ‘aws-cdk-lib/aws-lambda’;

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

…. // Define an S3 bucket const myBucket = new s3.Bucket(this, ‘myBucket’);

// …

// Define a Lambda function const myFunction = new lambda.Function(this, “myFunction”, { runtime: lambda.Runtime.NODEJS\_20\_X, handler: “index.handler”, code: lambda.Code.fromInline(exports.handler = async function(event) { return { statusCode: 200, body: JSON.stringify('Hello World!'), }; };), functionName: myBucketName + “Function”, // Pass token for the S3 bucket name environment: { BUCKET\_NAME: myBucketName, // Pass token for the S3 bucket name } }); } } —- ….

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const { Stack, Duration } = require(‘aws-cdk-lib’); const s3 = require(‘aws-cdk-lib/aws-s3’); const lambda = require(‘aws-cdk-lib/aws-lambda’);

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

…. // Define an S3 bucket const myBucket = new s3.Bucket(this, ‘myBucket’);

// …

// Define a Lambda function const myFunction = new lambda.Function(this, ‘myFunction’, { runtime: lambda.Runtime.NODEJS\_20\_X, handler: ‘index.handler’, code: lambda.Code.fromInline(exports.handler = async function(event) { return { statusCode: 200, body: JSON.stringify('Hello World!'), }; };), functionName: myBucketName + ‘Function’, // Pass token for the S3 bucket name environment: { BUCKET\_NAME: myBucketName, // Pass token for the S3 bucket name } }); } } ….

== module.exports = { CdkDemoAppStack }

Python:: + [source,python,subs=“verbatim,attributes”] — from aws\_cdk import ( Stack ) from constructs import Construct from aws\_cdk import aws\_s3 as s3 from aws\_cdk import aws\_lambda as \_lambda

class CdkDemoAppStack(Stack):

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

# Define an S3 bucket  
my\_bucket = s3.Bucket(self, "myBucket")  
  
# ...  
  
# Define a Lambda function  
my\_function = \_lambda.Function(self, "myFunction",  
 runtime=\_lambda.Runtime.NODEJS\_20\_X,  
 handler="index.handler",  
 code=\_lambda.Code.from\_inline("""  
 exports.handler = async function(event) {  
 return {  
 statusCode: 200,  
 body: JSON.stringify('Hello World!'),  
 };  
 };  
 """),  
 function\_name=f"{my\_bucket\_name}Function", # Pass token for the S3 bucket name  
 environment={  
 "BUCKET\_NAME": my\_bucket\_name # Pass token for the S3 bucket name  
 }  
) ----

….

Java:: + [source,java,subs=“verbatim,attributes”] — package com.myorg;

import software.constructs.Construct; import software.amazon.awscdk.Stack; import software.amazon.awscdk.StackProps; import software.amazon.awscdk.services.s3.Bucket; import software.amazon.awscdk.services.lambda.Code; import software.amazon.awscdk.services.lambda.Function; import software.amazon.awscdk.services.lambda.Runtime;

import java.util.Map;

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

…. public CdkDemoAppStack(final Construct scope, final String id, final StackProps props) { super(scope, id, props);

// Define an S3 bucket  
Bucket myBucket = Bucket.Builder.create(this, "myBucket")  
 .build();  
  
// ...  
  
// Define a Lambda function  
Function myFunction = Function.Builder.create(this, "myFunction")  
 .runtime(Runtime.NODEJS\_20\_X)  
 .handler("index.handler")  
 .code(Code.fromInline(  
 "exports.handler = async function(event) {" +  
 "return {" +  
 "statusCode: 200," +  
 "body: JSON.stringify('Hello World!')," +  
 "};" +  
 "};"  
 ))  
 .functionName(myBucketName + "Function") // Pass the token for the s3 bucket to the function construct  
 .environment(Map.of("BUCKET\_NAME", myBucketName)) // Pass the bucket name as environment variable  
 .build();

} } —- ….

C#:: + [source,csharp,subs=“verbatim,attributes”] — using Amazon.CDK; using Constructs; using Amazon.CDK.{aws}.S3; using Amazon.CDK.{aws}.Lambda; using System; using System.Collections.Generic;

namespace CdkDemoApp { public class CdkDemoAppStack : Stack { internal CdkDemoAppStack(Construct scope, string id, IStackProps props = null) : base(scope, id, props) { // Define an S3 bucket var myBucket = new Bucket(this, “myBucket”);

…. // …

// Define a Lambda function  
 var myFunction = new Function(this, "myFunction", new FunctionProps  
 {  
 Runtime = Runtime.NODEJS\_20\_X,  
 Handler = "index.handler",  
 Code = Code.FromInline(@"  
 exports.handler = async function(event) {  
 return {  
 statusCode: 200,  
 body: JSON.stringify('Hello World!'),  
 };  
 };  
 "),  
 // Pass the token for the S3 bucket name  
 Environment = new Dictionary<string, string>  
 {  
 { "BUCKET\_NAME", myBucketName }  
 },  
 FunctionName = $"{myBucketName}Function" // Pass the token for the s3 bucket to the function construct  
 });  
}

} } —- ….

Go:: + [source,go,subs=“verbatim,attributes”] — package main

import ( “fmt”

“github.com/aws/aws-cdk-go/awscdk/v2” “github.com/aws/aws-cdk-go/awscdk/v2/awslambda” “github.com/aws/aws-cdk-go/awscdk/v2/awss3” “github.com/aws/constructs-go/constructs/v10” “github.com/aws/jsii-runtime-go” )

type CdkDemoAppStackProps struct { awscdk.StackProps }

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

…. // Define an S3 bucket myBucket := awss3.NewBucket(stack, jsii.String(“myBucket”), &awss3.BucketProps{})

// …

// Define a Lambda function myFunction := awslambda.NewFunction(stack, jsii.String(“myFunction”), &awslambda.FunctionProps{ Runtime: awslambda.Runtime\_NODEJS\_20\_X(), Handler: jsii.String(“index.handler”), Code: awslambda.Code\_FromInline(jsii.String(exports.handler = async function(event) { return { statusCode: 200, body: JSON.stringify('Hello World!'), }; };)), FunctionName: jsii.String(fmt.Sprintf(“%sFunction”, *myBucketName)), // Pass the token for the S3 bucket to the function name Environment: &map[string]*string{ “BUCKET\_NAME”: myBucketName, }, })

return stack } // … —- ….

When we synthesize our template, the Ref and Fn::Join intrinsic functions are used to specify the values, which will be known at deployment:

== [source,yaml,subs=“verbatim,attributes”]

Resources: myBucket<5AF9C99B>: Type: {aws}::S3::Bucket # … myFunction<884E1557>: Type: {aws}::Lambda::Function Properties: # … Environment: Variables: BUCKET\_NAME: Ref: myBucket<5AF9C99B> FunctionName: Fn::Join: - “” - - Ref: myBucket<5AF9C99B> - Function # … — ====