:doctype: book

include::attributes.txt[]

// Attributes [.topic] :info\_titleabbrev: Tokens :info\_abstract: In the {aws} Cloud Development Kit ({aws} CDK), tokens are placeholders for values that aren’t known when defining constructs or synthesizing stacks. These values will be fully resolved at deployment, when your actual infrastructure is created. When developing {aws} CDK applications, you will work with tokens to manage these values across your application. :keywords: {aws} CDK, Tokens, {aws} CloudFormation, concepts

[#tokens] = Tokens and the {aws} CDK

== [abstract]

## In the {aws} Cloud Development Kit ({aws} CDK), *tokens* are placeholders for values that aren’t known when defining constructs or synthesizing stacks. These values will be fully resolved at deployment, when your actual infrastructure is created. When developing {aws} CDK applications, you will work with tokens to manage these values across your application.

// Content start

In the {aws} Cloud Development Kit ({aws} CDK), *tokens* are placeholders for values that aren’t known when defining constructs or synthesizing stacks. These values will be fully resolved at deployment, when your actual infrastructure is created. When developing {aws} CDK applications, you will work with tokens to manage these values across your application.

[#tokens-example] == Token example

The following is an example of a CDK stack that defines a construct for an Amazon Simple Storage Service (Amazon S3) bucket. Since the name of our bucket is not yet known, the value for bucketName is stored as a token:

==== [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’;

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’);

// Store value of the S3 bucket name const myBucketName = myBucket.bucketName;

// Print the current value for the S3 bucket name at synthesis console.log(“myBucketName:” + bucketName); } } —- ….

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

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

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

// Store value of the S3 bucket name const myBucketName = myBucket.bucketName;

// Print the current value for the S3 bucket name at synthesis console.log(“myBucketName:” + myBucketName); } } ….

== 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

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")  
  
# Store the value of the S3 bucket name  
my\_bucket\_name = my\_bucket.bucket\_name  
  
# Print the current value for the S3 bucket name at synthesis  
print(f"myBucketName: {my\_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 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();  
  
// Store the token for the bucket name  
String myBucketName = myBucket.getBucketName();  
  
// Print the token at synthesis  
System.out.println("myBucketName: " + myBucketName);

} } —- ….

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

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”);

…. // Store the token for the bucket name var myBucketName = myBucket.BucketName;

// Print the token at synthesis  
 System.Console.WriteLine($"myBucketName: {myBucketName}");  
}

} } —- ….

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/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{})

// Store the token for the bucket name myBucketName := myBucket.BucketName()

// Print the token at synthesis fmt.Println(“myBucketName:”, \*myBucketName)

return stack } ….

== // …

When we run cdk synth to synthesize our stack, the value for myBucketName will be displayed in the token format of ${Token[TOKEN.<1234>]}. This token format is a result of how the {aws} CDK encodes tokens. In this example, the token is encoded as a string:

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

$ cdk synth –quiet myBucketName: ${Token[TOKEN.21]} —

Since the value for our bucket name is not known at synthesis, the token is rendered as myBucket<unique-hash>. Our {aws} CloudFormation template uses the Ref intrinsic function to reference its value, which will be known at deployment:

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

Resources: myBucket<5AF9C99B>: # … Outputs: bucketNameOutput: Description: The name of the S3 bucket Value: Ref: myBucket<5AF9C99B> — ====

For more information on how the unique hash is generated, see xref:how-synth-default-logical-ids[Generated logical IDs in your {aws} CloudFormation template].