# Reference and Troubleshooting

Reference materials, troubleshooting guides, and additional resources for AWS CDK.

## 📖 Reference Materials

### CLI Reference

* [CLI Commands - Acknowledge](ref-cli-cmd-ack.md) - cdk acknowledge command
* [CLI Commands - Bootstrap](ref-cli-cmd-bootstrap.md) - cdk bootstrap command
* [CLI Commands - Context](ref-cli-cmd-context.md) - cdk context command
* [CLI Commands - Deploy](ref-cli-cmd-deploy.md) - cdk deploy command
* [CLI Commands - Destroy](ref-cli-cmd-destroy.md) - cdk destroy command
* [CLI Commands - Diff](ref-cli-cmd-diff.md) - cdk diff command
* [CLI Commands - Doctor](ref-cli-cmd-doctor.md) - cdk doctor command
* [CLI Commands - Import](ref-cli-cmd-import.md) - cdk import command
* [CLI Commands - Init](ref-cli-cmd-init.md) - cdk init command
* [CLI Commands - List](ref-cli-cmd-list.md) - cdk list command
* [CLI Commands - Logs](ref-cli-cmd-logs.md) - cdk logs command
* [CLI Commands - Metadata](ref-cli-cmd-metadata.md) - cdk metadata command
* [CLI Commands - Migrate](ref-cli-cmd-migrate.md) - cdk migrate command
* [CLI Commands - Notices](ref-cli-cmd-notices.md) - cdk notices command
* [CLI Commands - Synthesize](ref-cli-cmd-synth.md) - cdk synth command
* [CLI Commands - Version](ref-cli-cmd-version.md) - cdk version command
* [CLI Commands - Watch](ref-cli-cmd-watch.md) - cdk watch command

### Resource References

* [Resources](resources.md) - Comprehensive resource documentation

### How-To Guides

* [How-Tos](how-tos.md) - Common tasks and solutions

## 🔧 Troubleshooting

### General Troubleshooting

* [Troubleshooting Bootstrap](troubleshoot-bootstrap.md) - Bootstrap issues
* [Troubleshooting Deploy](troubleshoot-deploy.md) - Deployment problems
* [Troubleshooting Guide](troubleshooting.md) - General troubleshooting

## 🔒 Security and Compliance

### Security Reference

* [Security Overview](security.md) - CDK security considerations
* [IAM Security](security-iam.md) - IAM best practices
* [Infrastructure Security](infrastructure-security.md) - Infrastructure protection
* [Compliance Validation](compliance-validation.md) - Compliance frameworks
* [Disaster Recovery](disaster-recovery-resiliency.md) - DR and resiliency
* [Define IAM L2](define-iam-l2.md) - Level 2 IAM constructs

## 📋 Meta Information

### Documentation

* [Documentation History](doc-history.md) - Change history
* [PGP Keys](pgp-keys.md) - Verification keys

### Node.js

* [Node Versions](node-versions.md) - Supported Node.js versions

## 🎯 Quick Reference

### Common CLI Commands

cdk init app --language typescript # Create new app  
cdk bootstrap # Bootstrap environment   
cdk synth # Synthesize CloudFormation  
cdk deploy # Deploy stack  
cdk destroy # Delete stack  
cdk diff # Show differences

### Troubleshooting Checklist

* ☐ CDK CLI version compatibility
* ☐ AWS credentials configured
* ☐ Environment bootstrapped
* ☐ Permissions sufficient
* ☐ Dependencies up to date

## 🔗 Related Sections

* [CLI Usage](../04-deployment/cli.md) - Detailed CLI guide
* [Getting Started](../01-getting-started/) - Initial setup
* [Development](../03-development/) - Development best practices

include::attributes.txt[]

// Attributes

[.topic] [#compliance-validation] = Compliance validation for the {aws} Cloud Development Kit ({aws} CDK) :info\_titleabbrev: Compliance validation

== [abstract]

## Provides information about compliance validation for the {aws} CDK.

// Content start

The {aws} CDK follows the link:https://aws.amazon.com/compliance/shared-responsibility-model/[shared responsibility model] through the specific Amazon Web Services ({aws}) services it supports. For {aws} service security information, see the link:https://docs.aws.amazon.com/security/?id=docs\_gateway#aws-security[{aws} service security documentation page] and link:https://aws.amazon.com/compliance/services-in-scope/[{aws} services that are in scope of {aws} compliance efforts by compliance program].

The security and compliance of {aws} services is assessed by third-party auditors as part of multiple {aws} compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others. {aws} provides a frequently updated list of {aws} services in scope of specific compliance programs at link:https://aws.amazon.com/compliance/services-in-scope/[{aws} Services in Scope by Compliance Program].

Third-party audit reports are available for you to download using {aws} Artifact. For more information, see link:https://docs.aws.amazon.com/artifact/latest/ug/downloading-documents.html[Downloading Reports in {aws} Artifact].

For more information about {aws} compliance programs, see https://aws.amazon.com/compliance/programs/[{aws} Compliance Programs].

Your compliance responsibility when using the {aws} CDK to access an {aws} service is determined by the sensitivity of your data, your organization’s compliance objectives, and applicable laws and regulations. If your use of an {aws} service is subject to compliance with standards such as HIPAA, PCI, or FedRAMP, {aws} provides resources to help:

* link:https://aws.amazon.com/quickstart/?quickstart-all.sort-by=item.additionalFields.updateDate&quickstart-all.sort-order=desc&awsf.quickstart-homepage-filter=categories%23security-identity-compliance[Security and Compliance Quick Start Guides] – Deployment guides that discuss architectural considerations and provide steps for deploying security-focused and compliance-focused baseline environments on {aws}.
* link:https://aws.amazon.com/compliance/resources/[{aws} Compliance Resources] – A collection of workbooks and guides that might apply to your industry and location.
* link:https://aws.amazon.com/config/[{aws} Config] – A service that assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
* link:https://aws.amazon.com/security-hub/[{aws} Security Hub] – A comprehensive view of your security state within {aws} that helps you check your compliance with security industry standards and best practices.

include::attributes.txt[]

// Attributes

[.topic] [#define-iam-l2] = Define permissions for L2 constructs with the {aws} CDK :info\_titleabbrev: Define permissions for L2 constructs :keywords: {aws} CDK, {aws} Construct Library, {aws} CloudFormation, IAM, Permissions, Roles, Policies,

== [abstract]

## Define {aws} Identity and Access Management (IAM) roles and policies for L2 constructs when using the {aws} Cloud Development Kit ({aws} CDK).

// Content start

Define {aws} Identity and Access Management (IAM) roles and policies for L2 constructs when using the {aws} Cloud Development Kit ({aws} CDK).

[#define-iam-l2-grant] == Use grant methods to define permissions

When you define your infrastructure using L2 constructs from the {aws} Construct Library, you can use the provided *grant methods* to specify the permissions your resources will require. The {aws} CDK will automatically create the IAM roles needed for all {aws} resources that require them.

The following is an example that defines permissions between an {aws} Lambda function and Amazon Simple Storage Service (Amazon S3) bucket. Here, the grantRead method of the Bucket L2 construct is used to define these permissions:

==== [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’; import \* as kms from ‘aws-cdk-lib/aws-kms’;

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

…. const key = new kms.Key(this, ‘BucketKey’); const bucket = new s3.Bucket(this, ‘Bucket’, { encryptionKey: key, }); const handler = new lambda.Function(this, ‘Handler’, { runtime: lambda.Runtime.NODEJS\_20\_X, handler: ‘index.handler’, code: lambda.Code.fromAsset(‘lambda’), });

// Define permissions between function and S3 bucket using grantRead method bucket.grantRead(handler); ….

} } —

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’); const kms = require(‘aws-cdk-lib/aws-kms’);

class CdkDemoStack extends Stack {

constructor(scope, id, props) { super(scope, id, props);

…. const key = new kms.Key(this, ‘BucketKey’); const bucket = new s3.Bucket(this, ‘Bucket’, { encryptionKey: key, }); const handler = new lambda.Function(this, ‘Handler’, { runtime: lambda.Runtime.NODEJS\_20\_X, handler: ‘index.handler’, code: lambda.Code.fromAsset(‘lambda’), });

// Define permissions between function and S3 bucket using grantRead method bucket.grantRead(handler); ….

} }

== // …

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

class CdkDemoStack(Stack):

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

key = kms.Key(self, 'BucketKey')  
bucket = s3.Bucket(self, 'Bucket')  
handler = \_lambda.Function(  
 self,  
 'Handler',  
 runtime = \_lambda.Runtime.NODEJS\_20\_X,  
 handler = 'index.handler',  
 code = \_lambda.Code.from\_asset('lambda'),  
)  
  
# Define permissions between function and S3 bucket using grantRead method  
bucket.grantRead(handler) ----

….

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

import software.amazon.awscdk.core.App; import software.amazon.awscdk.core.Stack; import software.amazon.awscdk.core.StackProps; import software.amazon.awscdk.services.kms.Key; import software.amazon.awscdk.services.kms.KeyProps; import software.amazon.awscdk.services.s3.Bucket; import software.amazon.awscdk.services.s3.BucketProps; import software.amazon.awscdk.services.lambda.Function; import software.amazon.awscdk.services.lambda.FunctionProps; import software.amazon.awscdk.services.lambda.Runtime; import software.amazon.awscdk.services.lambda.Code; import software.constructs.Construct;

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

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

Key key = new Key(this, "BucketKey", KeyProps.builder().build());  
  
Bucket bucket = new Bucket(this, "Bucket", BucketProps.builder()  
 .encryptionKey(key)  
 .build());  
  
Function handler = new Function(this, "Handler", FunctionProps.builder()  
 .runtime(Runtime.NODEJS\_20\_X)  
 .handler("index.handler")  
 .code(Code.fromAsset("lambda"))  
 .build());  
  
// Define permissions between function and S3 bucket using grantRead method  
bucket.grantRead(handler);

}

public static void main(final String[] args) { App app = new App();

new CdkDemoStack(app, "CdkDemoStack");  
  
app.synth();

} } —- ….

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

namespace CdkDemo { public class CdkDemoStack : Stack { internal CdkDemoStack(Construct scope, string id, IStackProps props = null) : base(scope, id, props) { var key = new Key(this, “BucketKey”);

…. var bucket = new Bucket(this, “Bucket”, new BucketProps { EncryptionKey = key });

var handler = new Function(this, "Handler", new FunctionProps  
 {  
 Runtime = Runtime.NODEJS\_20\_X,  
 Handler = "index.handler",  
 Code = Code.FromAsset("lambda")  
 });  
  
 // Define permissions between function and S3 bucket using grantRead method  
 bucket.GrantRead(handler);  
}

} } —- ….

Go:: + [source,go,subs=“verbatim,attributes”] — package main import ( “github.com/aws/aws-cdk-go/awscdk/v2” “github.com/aws/aws-cdk-go/awscdk/v2/awskms” “github.com/aws/aws-cdk-go/awscdk/v2/awss3” “github.com/aws/aws-cdk-go/awscdk/v2/awslambda” “github.com/aws/constructs-go/constructs/v10” “github.com/aws/jsii-runtime-go” ) // … func NewCdkDemoStack(scope constructs.Construct, id string, props \*CdkDemoStackProps) awscdk.Stack { stack := awscdk.NewStack(scope, &id, &props.StackProps) key := awskms.NewKey(stack, jsii.String(“BucketKey”), nil) bucket := awss3.NewBucket(stack, jsii.String(“Bucket”), &awss3.BucketProps{ EncryptionKey: key, }) handler := awslambda.NewFunction(stack, jsii.String(“Handler”), &awslambda.FunctionProps{ Runtime: awslambda.Runtime\_NODEJS\_20\_X(), Handler: jsii.String(“index.handler”), Code: awslambda.Code\_FromAsset(jsii.String(“lambda”), &awss3assets.AssetOptions{}), }) bucket.GrantRead(handler) return stack } // … — ====

When you use grant methods of L2 constructs to define permissions between resources, the {aws} CDK will create roles with least privilege policies based on the method you specify. As a security best practice, we recommend that you use the method that applies only the permissions that you require. For example, if you only need to grant permissions for a Lambda function to read from an Amazon S3 bucket, use the grantRead method instead of grantReadWrite.

For each method that you use, the CDK creates a unique IAM role for the specified resources. If necessary, you can also directly modify the policy that will be attached to the role. The following is an example:

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — import { aws\_iam as iam } from ‘aws-cdk-lib’;

handler.addToRolePolicy(new iam.PolicyStatement({ actions: [‘s3:GetObject’, ’s3:List\_\_’], resources: [ bucket.bucketArn, bucket.arnForObjects(’\_\_’), ] })); —

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

handler.addToRolePolicy(new iam.PolicyStatement({ actions: [‘s3:GetObject’, ’s3:List\_\_’], resources: [ bucket.bucketArn, bucket.arnForObjects(’\_\_’), ] })); —

Python:: + [source,python,subs=“verbatim,attributes”] — from aws\_cdk import aws\_iam as iam

handler.add\_to\_role\_policy(iam.PolicyStatement( actions=[‘s3:GetObject’, ’s3:List\_\_’], resources=[ bucket.bucket\_arn, bucket.arn\_for\_objects(’\_\_’), ] )) —

Java:: + [source,java,subs=“verbatim,attributes”] — import software.amazon.awscdk.services.iam.PolicyStatement; import software.amazon.awscdk.services.iam.PolicyStatementProps;

handler.addToRolePolicy(PolicyStatement.Builder.create() .actions(Arrays.asList(“s3:GetObject”, “s3:List\_\_“)) .resources(Arrays.asList( bucket.getBucketArn(), bucket.arnForObjects(”\_\_“) )) .build()); —

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

handler.AddToRolePolicy(new PolicyStatement(new PolicyStatementProps { Actions = new[] { “s3:GetObject”, “s3:List\_\_” }, Resources = new[] { bucket.BucketArn, bucket.ArnForObjects(“\_\_“) } })); —

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

import ( // … “github.com/aws/aws-cdk-go/awscdk/v2/awsiam” // … )

// …

func NewCdkDemoStack(scope constructs.Construct, id string, props \*CdkDemoStackProps) awscdk.Stack { // …

…. handler.AddToRolePolicy(awsiam.NewPolicyStatement(&awsiam.PolicyStatementProps{ Actions: jsii.Strings(“s3:GetObject”, “s3:List*”), Resources: jsii.Strings(bucket.BucketArn(), bucket.ArnForObjects(”*”)), }))

// … } —- ==== ….

However, we recommend that you use the grant methods when available.

[#define-iam-l2-manual] == Manually create and use IAM roles

If you prefer not to use the CDK grant methods to create and manage permissions, you must manually create and configure them. You can create IAM roles using the {aws} Management Console, {aws} CLI, or {aws} SDKs. Then, you can pass them into your CDK application manually or use the *role customization* feature.

[#define-iam-l2-manual-role] === Reference and manage all roles manually

Constructs that require a role have an optional role property that you can use to pass in a role object.

*To reference a role manually*:: + . Use Role.fromRoleName() to reference your pre-existing role. The following is an example: + [source,javascript,subs=“verbatim,attributes”] — const existingRole = Role.fromRoleName(stack, ‘Role’, ‘my-pre-existing-role’, { mutable: false // Prevent CDK from attempting to add policies to this role }) — + . Pass the pre-existing role when defining your resource. The following is an example: + [source,javascript,subs=“verbatim,attributes”] — const handler = new lambda.Function(stack, ‘Handler’, { runtime: lambda.Runtime.NODEJS\_20\_XZ, handler: ‘index.handler’, code: lambda.Code.fromAsset(path.join(\_\_dirname, ‘lambda-handler’)), // Pass in pre-existing role role: existingRole, }); —

[#define-iam-l2-manual-customization] === Use the role customization feature

The {aws} CDK *role customization* feature generates a report of roles and policies in your CDK app. You can use this feature to generate a report. Then you can substitute pre-created roles for them.

*To use the role customization feature*:: + . Add Role.customizeRoles() somewhere towards the top of your CDK application. The following is an example: + [source,javascript,subs=“verbatim,attributes”] — const stack = new Stack(app, ‘LambdaStack’);

// Add this to use the role customization feature iam.Role.customizeRoles(stack);

// Define your resources using L2 constructs const key = new kms.Key(stack, ‘BucketKey’); const bucket = new s3.Bucket(stack, ‘Bucket’, { encryptionKey: key, }); const handler = new lambda.Function(stack, ‘Handler’, { runtime: lambda.Runtime.NODEJS\_16\_X, handler: ‘index.handler’, code: lambda.Code.fromAsset(path.join(\_\_dirname, ‘lambda-handler’)), });

// The grantRead() is still important. Even though it actually doesn’t mutate // any policies, it indicates the need for them. bucket.grantRead(handler); — + . When you synthesize your application, the CDK will throw an error, indicating that you need to provide the pre-created role name to Role.customizeRoles(). The following is an example of the generated report: + —++++++(LambdaStack/Handler/ServiceRole) AssumeRole Policy: [ { “Action”: “sts:AssumeRole”, “Effect”: “Allow”, “Principal”: { “Service”: “lambda.amazonaws.com” } } ] Managed Policy ARNs: [ “arn:(PARTITION):iam::aws:policy/service-role/AWSLambdaBasicExecutionRole” ] Managed Policies Statements: NONE Identity Policy Statements: [ { “Action”: [ “s3:GetObject*”, ”s3:GetBucket*”, “s3:List\*” ], “Effect”: “Allow”, “Resource”: [ “(LambdaStack/Bucket/Resource.Arn)”, “(LambdaStack/Bucket/Resource.Arn)/\*” ] } ] —- + . Once the role is created, you can pass it into your application for the resource that it applies to. For example, if the name of the role created for LambdaStack/Handler/ServiceRole is lambda-service-role, you would update your CDK app as follows: + [source,javascript,subs=“verbatim,attributes”] —- const stack = new Stack(app, ‘LambdaStack’); // Add this to pass in the role iam.Role.customizeRoles(stack, { usePrecreatedRoles: { ‘LambdaStack/Handler/ServiceRole’: ‘lambda-service-role’, }, }); —- + The CDK will now use the pre-created role name anywhere that the role is referenced in the CDK application. It will also continue to generate the report so that any future policy changes can be referenced. + You will notice that the reference to the Amazon S3 bucket ARN in the report is rendered as (LambdaStack/Bucket/Resource.Arn) instead of the actual ARN of the bucket. This is because the bucket ARN is a deploy time value that is not known at synthesis (the bucket hasn't been created yet). This is another example of why we recommend allowing CDK to manage IAM roles and permissions by using the providedgrantmethods. In order to create the role with the initial policy, the admin will have to create the policy with broader permissions (for example,arn:aws:s3:::\*`).++++++

include::attributes.txt[]

// Attributes

[.topic] [#disaster-recovery-resiliency] = Resilience for the {aws} Cloud Development Kit ({aws} CDK) :info\_titleabbrev: Resilience

== [abstract]

## Provides information about resilience for the {aws} CDK.

// Content start

The Amazon Web Services ({aws}) global infrastructure is built around {aws} Regions and Availability Zones.

{aws} Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking.

With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about {aws} Regions and Availability Zones, see link:https://aws.amazon.com/about-aws/global-infrastructure/[{aws} Global Infrastructure].

The {aws} CDK follows the link:https://aws.amazon.com/compliance/shared-responsibility-model/[shared responsibility model] through the specific Amazon Web Services ({aws}) services it supports. For {aws} service security information, see the link:https://docs.aws.amazon.com/security/?id=docs\_gateway#aws-security[{aws} service security documentation page] and https://aws.amazon.com/compliance/services-in-scope/[{aws} services that are in scope of {aws} compliance efforts by compliance program].

:doctype: book

include::attributes.txt[]

[.topic] [#doc-history] = {aws} CDK Developer Guide history :info\_titleabbrev: Document history :info\_abstract: {aws} CDK Developer Guide history

== [abstract]

## {aws} CDK Developer Guide history

// Content start

See link:https://github.com/awslabs/aws-cdk/releases[Releases] for information about {aws} CDK releases. The {aws} CDK is updated approximately once a week. Maintenance versions may be released between weekly releases to address critical issues. Each release includes a matched {aws} CDK Toolkit (CDK CLI), {aws} Construct Library, and API Reference. Updates to this Guide generally do not synchronize with {aws} CDK releases.

= [NOTE]

# The table below represents significant documentation milestones. We fix errors and improve content on an ongoing basis.

[.updates] == Updates

[.update,date=“2025-03-27”] === Document bootstrap template versions v26 and v27

For more information, see link:https://docs.aws.amazon.com/cdk/v2/guide/bootstrapping-env.html#bootstrap-template-history[Bootstrap template version history].

[.update,date=“2025-03-21”] === Add documentation for building and deploying container image assets

When you build container image assets with the {aws} CDK, Docker is utilized by default to perform these actions. If you want to use a different container management tool, you can replace Docker. For more information, see https://docs.aws.amazon.com/cdk/v2/guide/build-containers.html[Build and deploy container image assets in CDK apps].

[.update,date=“2024-02-02”] === Add documentation for CDK Migrate feature

Use the {aws} CDK CLI cdk migrate command to migrate deployed {aws} resources, deployed {aws} CloudFormation stacks, and local {aws} CloudFormation templates to {aws} CDK. For more information, see https://docs.aws.amazon.com/cdk/v2/guide/migrate.html[Migrate to {aws} CDK].

[.update,date=“2023-03-23”] === IAM best practices updates

Updated guide to align with the IAM best practices. For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html[Security best practices in IAM].

[.update,date=“2022-04-20”] === Document cdk.json

Add documentation of cdk.json configuration values.

[.update,date=“2022-04-07”] === Dependency management

Add topic on managing dependencies with the {aws} CDK.

[.update,date=“2022-03-09”] === Remove double-braces from Java examples

Replace this anti-pattern with Java 9 Map.of throughout.

[.update,date=“2021-12-04”] === {aws} CDK v2 release

Version 2 of the {aws} CDK Developer Guide is released.

See https://github.com/awslabs/aws-cdk/releases[Releases] for information about {aws} CDK releases. The {aws} CDK is updated approximately once a week. Maintenance versions may be released between weekly releases to address critical issues. Each release includes a matched {aws} CDK Toolkit (CDK CLI), {aws} Construct Library, and API Reference. Updates to this Guide generally do not synchronize with {aws} CDK releases.

[.level] == {blank}

[.update-history] |=== |===

include::attributes.txt[]

// Attributes

[.topic] [#how-tos] = {aws} CDK tutorials and examples :info\_titleabbrev: Tutorials and examples :keywords: {aws} CDK, {aws} Cloud Development Kit ({aws} CDK), IaC, Infrastructure as Code, serverless, serverless application, Lambda, API Gateway

== [abstract]

## This section contains tutorials and examples for the {aws} Cloud Development Kit ({aws} CDK).

// Content start

This section contains tutorials and examples for the {aws} Cloud Development Kit ({aws} CDK).

[#tutorials] *Tutorials*:: Tutorials provide you with step-by-step instructions that you can follow to implement a task using the {aws} CDK.

[#examples] *Examples*:: Examples show and explain how specific tasks can be implemented using the {aws} CDK. They do not provide direct step-by-step instructions for you to follow. + For more examples of {aws} CDK stacks and apps in your favorite supported programming language, see the https://github.com/aws-samples/aws-cdk-examples[{aws} CDK Examples] repository on GitHub.

include::serverless\_example.adoc[leveloffset=+1]

include::create-multiple-stacks.adoc[leveloffset=+1]

include::ecs\_example.adoc[leveloffset=+1]

include::attributes.txt[]

// Attributes

[.topic] [#infrastructure-security] = Infrastructure security for the {aws} Cloud Development Kit ({aws} CDK) :info\_titleabbrev: Infrastructure security

== [abstract]

## Provides information about infrastructure security for the {aws} CDK.

The {aws} CDK follows the link:https://aws.amazon.com/compliance/shared-responsibility-model/[shared responsibility model] through the specific Amazon Web Services ({aws}) services it supports. For {aws} service security information, see the link:https://docs.aws.amazon.com/security/?id=docs\_gateway#aws-security[{aws} service security documentation page] and https://aws.amazon.com/compliance/services-in-scope/[{aws} services that are in scope of {aws} compliance efforts by compliance program].

include::attributes.txt[]

// Attributes [.topic] [#node-versions] = Supported Node.js versions for the {aws} CDK :info\_titleabbrev: Supported Node versions :keywords: {aws} CDK, {aws} Cloud Development Kit ({aws} CDK), node versions, supported versions, aws-cdk-cli, aws-cdk-lib

== [abstract]

## The {aws} Cloud Development Kit ({aws} CDK) depends on Node.js for its functionality. This includes core CDK components such as the {aws} CDK CLI (aws-cdk-cli), {aws} Construct Library (aws-cdk-lib), as well as broader tools such as [.noloc]JSII, [.noloc]Projen, and [.noloc]CDK8s. This page documents the Node.js runtime versions compatible with each release of the {aws} CDK, helping you maintain a supported development environment.

// Content start

The {aws} Cloud Development Kit ({aws} CDK) depends on Node.js for its functionality. This includes core CDK components such as the {aws} CDK CLI (aws-cdk-cli), {aws} Construct Library (aws-cdk-lib), as well as broader tools such as [.noloc]JSII, [.noloc]Projen, and [.noloc]CDK8s. This page documents the Node.js runtime versions compatible with the {aws} CDK, helping you maintain a supported development environment.

The {aws} CDK aims to maintain compatibility with actively supported Node.js Long Term Support (LTS) versions. As Node.js versions reach their end-of-life, the {aws} CDK also phases out support for these versions to ensure security, performance, and access to the latest features.

Keeping your CDK components and tools on supported Node.js versions ensures that you receive security updates, bug fixes, and access to the latest CDK features and improvements.

[#node-version-timeline] == Node.js version support timeline

The {aws} CDK supports Node.js versions beyond their official End-of-Life (EOL) dates to give you time to upgrade your environment. This support extends for 6 months (for example, when Node.js versions reach EOL in April, support continues until October). The version support timeline table below documents specific CDK support end dates as they are determined. This approach gives you a defined window to plan and implement version upgrades while maintaining access to CDK functionality and updates.

[cols=“1,1,1”, options=“header”] |=== |Node.js version |Node EOL date |CDK support status

|=== | 22.x | 2027-04-30 | Support expected until October 2027. |===

|=== | 20.x | 2026-04-30 | Support expected until October 2026. |===

|=== | 18.x | 2025-05-30 | Support ends 2025-11-30. |===

|=== | 16.x | 2023-09-11 | Support ends 2025-05-30. |===

|=== | 14.x | 2023-04-30 | Support ends 2025-05-30. |===

:doctype: book :pp: {plus}{plus}

include::attributes.txt[]

// Attributes

[#pgp-keys] = OpenPGP keys for the {aws} CDK and jsii :doctype: book :info\_titleabbrev: OpenPGP keys

== [abstract]

## Current and historical OpenPGP keys for the {aws} CDK and jsii.

// Content start

This topic contains current and historical OpenPGP keys for the {aws} CDK and jsii.

[#pgp-keys-current,] == Current keys

These keys should be used to validate current releases of the {aws} CDK and jsii.

[#cdk-pgp-key] === {aws} CDK OpenPGP key

[cols=“1,1”] |===

|=== | Key ID: | 0x42B9CF2286CD987A |===

|=== | Type: | RSA |===

|=== | Size: | 4096/4096 |===

|=== | Created: | 2022-07-05 |===

|=== | Expires: | 2026-07-04 |===

|=== | User ID: | {aws} Cloud Development Kit link:mailto:aws-cdk@amazon.com[aws-cdk@amazon.com] |===

|=== | Key fingerprint: | 69B5 2D5B A295 1D11 FA65 413B 42B9 CF22 86CD 987A |===

Select the “Copy” icon to copy the following OpenPGP key:

== [source,none,subs=“verbatim,attributes”]

—-BEGIN PGP PUBLIC KEY BLOCK—-

mQINBGLEgOsBEADCoAMwvnszMLybJ+AD9cHhVyX6+rYIUEXYSgVnfkl6Z7qawIwv wgd/a5fEs9Kiz2XJmfwS9Rxb4d+0+Y1ls1A+gnpw9FMLcZlqkC9KLnS2MqvuXWLB t3z4kjZaL9fQ+58PoD4gy/M2hDg6gZrYqR3gtJuw8FcFpb/1KlkzRQUM8eAMFxf2 TyfjP0V0tSHwcB+84oushX7fUXVMyc3+OHsCPOe/WBFMIlWgKA+n33JKIQlUUC8f kCWBAsAFupil0lCveT6mZu5slNRlc1I3iBLjUZ3/MtLygfqAMKwUVXeawtDvRIZe PrAFc2NyODEhly2JG6K0FW7eIcvBqR3rg8U49t9Y74ELTM0kKnfd+flvq35xWqQC 0zghnk3kDppRTN4zWBgTKiCMxBcsHXGOoGn57t4B9VY9Zy3vkeySigeiwl/Tw9nJ PE0SRnwEc/HnjTTfX+GTG1aQVE0xSVyZ4m5ymRNCu6+rNH8lKwo5FujlXJ+GXPkp qT+Lx6Ix/Ny7PaoweWxwtZUkLRS4pWUsg0yotZrGyIbS+X3yMEG8WBTFI9hf6HTq 0ryfi5/TsBrdrGKqWB99EC9xYEGgtHp4fKO5X0ynOagVOhf0jSe8t1uyuJPGb2Gc MQagSys5xMhdG/ZnEY4Cb+JDtH/4jc3tca0+4Z5RQ7kF9IhCncFtrbjJbwARAQAB tC5BV1MgQ2xvdWQgRGV2ZWxvcG1lbnQgS2l0IDxhd3MtY2RrQGFtYXpvbi5jb20+ iQI/BBMBAgApBQJixIDrAhsvBQkHhM4ABwsJCAcDAgEGFQgCCQoLBBYCAwECHgEC F4AACgkQQrnPIobNmHo2qg//Zt9p/kN1DevflzxWKouUX0AS7UmUtRYXu5k/EEbu wkYNHpUr7+lZ+Me5YyjcIpt6UwuG9cW4SvwuxIfXucyKAWiwEbydCQauvnrYDxDa J6Yr/ntk7Sii6An9re99qic3IsvX+xlUXh+qJ/34ooP/1PHziCMqykvW/DwAIyhx 2qvTXy+9+OlOWSUbhkCnNz5XKb4XQGq73DqalZX1nH4dG6fckZmYRX+dpw2njfTw ZLdZ7bkrfiL84FI4A21RfSbEU4s4ngiV17lZ9ivilBKTbDv3da7+yc919M7C5N4J yrlxvtyYNDoqKAD2WYZAnpEbG/shu3f56RyOJd56tXGwl9nKPh+F9y+379XthSwA xZTURFtjWf7wWHaDZadU0DKi+Oeeszjg2f/VJaGmmS8PIg7q6GiSHHpqHqNvACHm ZXMw12QFd3qt3xu0JMmE11ZC5VBgblwpkQTrO04Sq1rOpJwXI9ODMS/ZEhAIoYmT OR7ouknlAx6mj9fwpavWDAAJHLdVUMYBZTXiQYFzDvx51ivvTRWkB1zTJcFdqShY B37+Jz2jLDNdMrcHk2yfVp/VvfbxKcexg8wEwrrtQUslTUenl5jBZJouoz/wW81s Y4U1nCPCdTK5/C7JCKzR2gVnCpe6uaxAWkkM2feQhjqJZkTC4cFVgBT+4M6WcT1r yq4= =ahbs —-END PGP PUBLIC KEY BLOCK—- —

[#jsii-pgp-key] === jsii OpenPGP key

[cols=“1,1”] |===

|=== | Key ID: | 0x056C4E15DAE3D8D9 |===

|=== | Type: | RSA |===

|=== | Size: | 4096/4096 |===

|=== | Created: | 2022-07-05 |===

|=== | Expires: | 2026-07-04 |===

|=== | User ID: | {aws} JSII Team link:mailto:aws-jsii@amazon.com[aws-jsii@amazon.com] |===

|=== | Key fingerprint: | 1E07 31D4 57E5 FE87 87E5 530A 056C 4E15 DAE3 D8D9 |===

Select the “Copy” icon to copy the following OpenPGP key:

== [source,none,subs=“verbatim,attributes”]

—-BEGIN PGP PUBLIC KEY BLOCK—-

mQINBGLEgOkBEAD27EPVG9g2mHQ3+M6tF6le+tfhARJ2EV7m7NKIrTdSlCZATLWn AVLlxG1unW34NlkKZbcbR86gAxRnnAhuEhPuloU/S5wAqPGbRiFl58YjYZDNJw6U 1SSMpE4O1sfjxv9yAbiRihLYtvksyHHZmaDhYner2aK1PdeWu+BKq/tjfm3Yzsd2 uuVEduJ72YoQk/29dEiGOHfT+2kUKxUX+0tJSJ9MGlEf4NtQE4WLzrT6Xqb2SG4+ alIiIVxIEi0XKDn7n8ZLjFwfJwOYxVYLtEUkqFWM8e8vgoc9/nYc+vDXZVED2g3Z FWrwSnDSXbQpnMa2cLhD4xLpDHUS3i2p7r3dkJQGLo/5JGOopLibrOAbYZ72izhu H/TuPFogSz0mNFPglrWdnLF04UIjIq420+06V4WQZC9n55Zjcbki/OhnC3B9pAdU tiy8zg070bWq45dPGf5STkPPn7G8A2zmKefyO51iLIi26ZzW78siB+FvcGRhdg25 39sHJ1cmrTeC+B+k4KeV5sQ/m3UucimrZnk1xdaiVp8mWzRqWb8bB6Rs8K9RMrMV tFBOKOBAT2QxOQtRGAantVgm193E1T1cmNpD0FKAKkDdPs64rKBEwFiHxccXHbah eMd1weVwn3AKFD6uAm8ZRMV+dyssfcQxqpo/kfT1XpA6cQeOmGDOcKBfdwARAQAB tCNBV1MgSlNJSSBUZWFtIDxhd3MtanNpaUBhbWF6b24uY29tPokCPwQTAQIAKQUC YsSA6QIbLwUJB4TOAAcLCQgHAwIBBhUIAgkKCwQWAgMBAh4BAheAAAoJEAVsThXa 49jZjU4QANoyqOJUT4gRrXshE3N0mW5Ad4i8Ke09GA62HyvTtfbsA+2nkNVGJpXm sFMzdaFO95Q65RkLS9vW4nhhjXBEc2XYNCt2AnARudA/41ykjDPwU112z9ZTB9he y4ItIeNGpHvMWr51fihl0y2nkpODOBeiv44jscLbHyOmZfki1f5fuIu2U2IbUGK3 5FtYyeHcgRHnpYkzLuzK4PfayOywqQPJ7M9DWrHf+v5Cu4ZCZDOIKfzF+ew7MWwc 6KaoWHCYbFpX8jxFppbGsSFOQ8Sl2quoP0TLz9Wsq70Khi6C2P8JI6lm0HRLO+1M jFbQxNOwAcN3k4HSwunAjXBlmT/6oc1RsdBdpXBaZ2AWseIXwSYZqNXp+5L179uZ vSiD3DSSUqLJbdQRVOsJi3/87V5QU59byq2dToHveRjtSbVnK0TkTx9ZlgkcpjvM BwHNqWhratV6af2Upjq2YQ0fdSB42f3pgopInxNJPMvlAb+cCfr0Pfwu7ge7UooQ WHTxbpCvwtn/HNctMGpWscO02WsWgoYVjnVFay/XphE77pQ9rRUkhMe6VKXfxj/n OCZJKrydluIIwR8vvONNqO+QwZ1xDEhO7MaSZlOm1AuUZIXFPgaWQkPZHKiiwFA/ QWnL/+shuRtMH2geTjkev198Jgb5HyXFm4SyYtZferQROyliEhik =BuGv —-END PGP PUBLIC KEY BLOCK—- —

[#pgp-keys-expired] == Historical keys

These keys may be used to validate releases of the {aws} CDK and jsii before 2022-07-05.

= [IMPORTANT]

New keys are created before the previous ones expire. As a result, at any given moment in time, more than one key may be valid. Keys are used to sign artifacts starting the day they are created, so use the more recently-issued key where keys’ validity overlaps.

====

[#cdk-pgp-key-2022-04-07] === {aws} CDK OpenPGP key (2022-04-07)

= [NOTE]

This key was not used to sign {aws} CDK artifacts after 2022-07-05.

====

[cols=“1,1”] |===

|=== | Key ID: | 0x015584281F44A3C3 |===

|=== | Type: | RSA |===

|=== | Size: | 4096/4096 |===

|=== | Created: | 2022-04-07 |===

|=== | Expires: | 2026-04-06 |===

|=== | User ID: | {aws} Cloud Development Kit link:mailto:aws-cdk@amazon.com[aws-cdk@amazon.com] |===

|=== | Key fingerprint: | EAE1 1A24 82B0 AA86 456E 6C67 0155 8428 1F44 A3C3 |===

Select the “Copy” icon to copy the following OpenPGP key:

== [source,none,subs=“verbatim,attributes”]

—-BEGIN PGP PUBLIC KEY BLOCK—-

mQINBGJPLgUBEADtlR5jQtxtBmROQvmWlPOViqqnJNhk0dULc3tXnq8NS/l6X81r wHk+/CHG5kBunwvM0qaqLFRC6z9NnnNDxEHcTi47n+OAjWyDM6unxxWOPz8Dfaps Uq/ZWa4by292ZeqRC9Ir2wdrizb69JbRjeshBwlJDAS/qtqCAqBRH/f7Zw7QSD6/ XTxyIy+KOVjZwFPFNHMRQ/NmgUc/Rfxsa0pUjk1YAj/AkvQlwwD8DEnASoBhO0DP QonZxouLqIpgp4LsGo8TZdQv30ocIj0C9DuYUiUXWlCPlYPgDj6IWf3rgpMQ6nB9 wC9lx4t/L3Zg1HUD52y8aymndmbdHVn90mzlNg4XWyc58rioYrEk57YwbDnea/Kk Hv4kVHZRfJ4/OFPyqs5ex1X3X6rb07VvA1tfLgPywO9XF2Xws8YWOWcEobaWTcnb AzyVC6wKya8rEQzxkYJ6UkJlhDB6g6bZwIpsI2zlimG+kSBsyFvE2oRYMS0cXPqU o+tX0+4TvxEyW3RrUQzQHIpqXrb0X1Q8Z2idPn5dwsipDEa4gsFXtrSXmbB/0Cee eJVvKWQAsxol3+NE9L/yozq3cz5PWh0SSbmCLRcs78lMJ23MmzbMWV7BWC9DXdY+ TywY5IkDUPjGCKlD8VlrI3TgC222bH6qaua6LYCiTtRtvpDYuJNAlUjhawARAQAB tC5BV1MgQ2xvdWQgRGV2ZWxvcG1lbnQgS2l0IDxhd3MtY2RrQGFtYXpvbi5jb20+ iQI/BBMBAgApBQJiTy4FAhsvBQkHhM4ABwsJCAcDAgEGFQgCCQoLBBYCAwECHgEC F4AACgkQAVWEKB9Eo8NpbxAAiBF0kR/lVw3vuam60mk4l0iGMVsP8Xq6g/buzbEO 2MEB4Ftk04qOnoa+93S0ZiLR9PqxrwsGSp4ADDX3Vtc4uxwzUlKUi1ywEhQ1cwyL YHQI3Hd75K1J81ozMEu6qJH+yF0TtTDZMeZHtH/XvuIYJW3Lx4o5ZFlsEegFPAgX YCCpUS+k9qC6M8g2VjcltQJpyjGswsKm6FWaKHW+B9dfjdOHlImB9E2jaknJ8eoY zb9zHgFANluMzpZ6rYVSiCuXiEgYmazQWCvlPcMOP7nX+1hq1z11LMqeSnfE09gX H+OYho9cMEJkb1dzx1H9MRpylFIn9tL+2iCp4UPJjnqi6uawWyLZ2tp4G11haqQq 1yAh69u233I8GZKFUySzjHwH5qWGRgBTjrZ6FdcjSS2w/wMkVKuCPkWtdvo/TJrm msCd1Reye8SEKYqrs0ujTwmlvWmUZmO06AdUjo1kWiBKeslTJrWEuG7Yk4pFOoA4 dsaq83gxpOJNVCh6M3y4DLNrvl7dhF95NwTWMROPj2otw7NIjF4/cdzve2+P7YNN pVAtyCtTJdD3eZbQPVaL3T8cf1VGqt6{pp}pnLGnWJ0+X3TyvfmTohdJvN3TE+tq7A 7cprDX/q9c56HaXdJzVpxEzuf/YC+JuYKeHwsX3QouDhyRg3PsigdZES/02Wr8so l6U= =MQI4 —-END PGP PUBLIC KEY BLOCK—- —

[#jsii-pgp-key-2022-04-07] === jsii OpenPGP key (2022-04-07)

= [NOTE]

This key was not used to sign jsii artifacts after 2022-07-05.

====

[cols=“1,1”] |===

|=== | Key ID: | 0x985F5BC974B79356 |===

|=== | Type: | RSA |===

|=== | Size: | 4096/4096 |===

|=== | Created: | 2022-04-07 |===

|=== | Expires: | 2026-04-06 |===

|=== | User ID: | {aws} JSII Team link:mailto:aws-jsii@amazon.com[aws-jsii@amazon.com] |===

|=== | Key fingerprint: | 35A7 1785 8FA6 282D C5AC CD95 985F 5BC9 74B7 9356 |===

Select the “Copy” icon to copy the following OpenPGP key:

== [source,none,subs=“verbatim,attributes”]

—-BEGIN PGP PUBLIC KEY BLOCK—-

mQINBGJPLewBEADHH4TXup/gOlHrKDZRbj8MvsMTdM6eDteA6/c32UYV/YsK9rDA jN8Jv/xlfosOebcHrfnFpHF9VTkmjuOpN695XdwMrW/NvlEPISTGEJf21x6ZTQ2r 1xWFYzC3sl3FZmvj9XAXTmygdv+XM3TqsFgZeCaBkZVdiLbQf+FhYrovUlgotb5D YiCQI3ofV5QTE+141jhO5Pkd3ZIoBG+P826LaT8NXhwS0o1XqVk39DCZNoFshNmR WFZpkVCTHyv5ZhVey1NWXnD8opO375htGNV4AeSmSIH9YkURD1g5F+2t7RiosKFo kJrfPmUjhHn8IFpReGc8qmMMZX0WaV3t+VAWfOHGGyrXDfQ4xz1VCot75C2+qypM +qhwOAOOP0zA7CfI96ULZzSH/j8HuQk3O0DsUCybpMuKEazEMxP3tgGtRerwDaFG jQvAlK8Rbq3v8buBI6YJuXTwSzJE8KLjleUiTFumE6WP4rsAvlP/5rBvubeMfa3n NIMm5Rkl36Z+jt3e2Z2ZqWDPpBRta8m7QHccrZhkvqu3YC3Gl6kdnm4Vio3Xfpg2 qtWhIQutQ6DmItewV+weQHas3hl88RPJtSrfWWIIMkpbF7Y4vbX9xcnsYCLlp2Mz tWbbnU+EWATNSsufml/Kdnu9iEEuLmeovE11I69nwjNOq9P+GJ3r/FUb2wARAQAB tCNBV1MgSlNJSSBUZWFtIDxhd3MtanNpaUBhbWF6b24uY29tPokCPwQTAQIAKQUC Yk8t7AIbLwUJB4TOAAcLCQgHAwIBBhUIAgkKCwQWAgMBAh4BAheAAAoJEJhfW8l0 t5NWo64P/2y7gcMRylLLW/wbrCjton2O4+YRocwQxKm1cBml9FVDUR5967YczNuu EwEOfH/Pu3UAlrBfKAfxPNhKchLwYiOBNh2Wk5UUxRcldNHTLb5jn5gxCeWNAsl/ Tc46qY+ObdBMdOf2Vu33UCOg83WLbg1bfBoA8Bm1cd0XObtLGucu606EBt1dBrKq 9UTcbJfuGivY2Xjy5r4kEiMHBoLKcFrSo2Mm7VtYlE4Mabjyj9+orqUio7qxOl60 aa7Psa6rMvs1Ip9IOrAdG7o5Y29tQpeINH0R1/u47BrlTEAgG63Dfy49w2h/1g0G c9KPXVuN55OWRIu0hsiySDMk/2ERsF348TU3NURZltnCOxp6pHlbPJIxRVTNa9Cn f8tbLB3y3HfA80516g+qwNYIYiqksDdV2bz+VbvmCWcO+FellDZli831gyMGa5JJ rq7d0lEr6nqjcnKiVwItTQXyFYmKTAXweQtVC72g1sd3oZIyqa7T8pvhWpKXxoJV WP+OPBhGg/JEVC9sguhuv53tzVwayrNwb54JxJsD2nemfhQm1Wyvb2bPTEaJ3mrv mhPUvXZj/I9rgsEq3L/sm2Xjy09nra4o3oe3bhEL8nOj11wkIodi17VaGP0y+H3s I5zB5UztS6dy+cH+J7DoRaxzVzq7qtH/ZY2quClt30wwqDHUX1ef =+iYX —-END PGP PUBLIC KEY BLOCK—- —

[#cdk-pgp-key-2018-06-19] === {aws} CDK OpenPGP key (2018-06-19)

[cols=“1,1”] |===

|=== | Key ID: | 0x0566A784E17F3870 |===

|=== | Type: | RSA |===

|=== | Size: | 4096/4096 |===

|=== | Created: | 2018-06-19 |===

|=== | Expires: | 2022-06-18 |===

|=== | User ID: | {aws} CDK Team link:mailto:aws-cdk@amazon.com[aws-cdk@amazon.com] |===

|=== | Key fingerprint: | E88B E3B6 F0B1 E350 9E36 4F96 0566 A784 E17F 3870 |===

Select the “Copy” icon to copy the following OpenPGP key:

== [source,none,subs=“verbatim,attributes”]

—-BEGIN PGP PUBLIC KEY BLOCK—-

mQINBFsovE8BEADEFVCHeAVPvoQgsjVu9FPUczxy9P+2zGIT/MLI3/vPLiULQwRy IN2oxyBNDtcDToNa/fTkW3Ev0NTP4V1h+uBoKDZD/p+dTmSDRfByECMI0sGZ3UsG Ohhyl2Of44s0sL8gdLtDnqSRLf+ZrfT3gpgUnplW7VitkwLxr78jDpW4QD8p8dZ9 WNm3JgB55jyPgaJKqA1Ln4Vduni/1XkrG42nxrrU71uUdZPvPZ2ELLJa6n0/raG8 jq3le+xQh45gAIs6PGaAgy7jAsfbwkGTBHjjujITAY1DwvQH5iS31OaCM9n4JNpc xGZeJAVYTLilznf2QtS/a50t+ZOmpq67Ssp2j6qYpiumm0Lo9q3K/R4/yF0FZ8SL 1TuNX0ecXEptiMVUfTiqrLsANg18EPtLZZOYW+ZkbcVytkDpiqj7bMwA7mI7zGCJ 1gjaTbcEmOmVdQYS1G6ZptwbTtvrgA6AfnZxX1HUxLRQ7tT/wvRtABfbQKAh85Ff a3U9W4oC3c1MP5IyhNV1Wo8Zm0flZiZc0iZnojTtSG6UbcxNNL4Q8e08FWjhungj yxSsIBnQO1Aeo1N4BbzlI+n9iaXVDUN7Kz1QEyS4PNpjvUyrUiQ+a9C5sRA7WP+x IEOaBBGpoAXB3oLsdTNO6AcwcDd9+r2NlXlhWC4/uH2YHQUIegPqHmPWxwARAQAB tCFBV1MgQ0RLIFRlYW0gPGF3cy1jZGtAYW1hem9uLmNvbT6JAj8EEwEIACkFAlso vE8CGy8FCQeEzgAHCwkIBwMCAQYVCAIJCgsEFgIDAQIeAQIXgAAKCRAFZqeE4X84 cLGxD/0XHnhoR2xvz38GM8HQlwlZy9W1wVhQKmNDQUavw8Zx7+iRR3m7nq3xM7Qq BDbcbKSg1lVLSBQ6H2V6vRpysOhkPSH1nN2dO8DtvSKIPcxK48+1x7lmO+ksSs/+ oo1UvOmTDaRzOitYh3kOGXHHXk/l11GtF2FGQzYssX5iM4PHcjBsK1unThs56IMh OJeZezEYzBaskTu/ytRJ236bPP2kZIEXfzAvhmTytuXWUXEftxOxc6fIAcYiKTha aofG7WyR+Fvb1j5gNLcbY552QMxa23NZd5cSZH7468WEW1SGJ3AdLA7k5xvsPPOC 2YvQFD+vUOZ1JJuu6B5rHkiEMhRTLklkvqXEShTxuXiCp7iTOo6TBCmrWAT4eQr7 htLmqlXrgKi8qPkWmRdXXG+MQBzI/UyZq2q8KC6cx2md1PhANmeefhiM7FZZfeNM WLonWfh8gVCsNH5h8WJ9fxsQCADd3Xxx3NelS2zDYBPRoaqZEEBbgUP6LnWFprA2 EkSlc/RoDqZCpBGgcoy1FFWvV/ZLgNU6OTQlYH6oYOWiylSJnaTDyurrktsxJI6d 4gdsFb6tqwTGecuUPvvZaEuvhWExLxAebhu780FdAPXgVTX+YCLI2zf+dWQvkFQf 80RE7ayn7BsiaLzFBVux/zz/WgvudsZX18r8tDiVQBL51ORmqw== =0wuQ —-END PGP PUBLIC KEY BLOCK—- —

[#jsii-pgp-key-2018-08-06] === jsii OpenPGP key (2018-08-06)

[cols=“1,1”] |===

|=== | Key ID: | 0x1C7ACE4CB2A1B93A |===

|=== | Type: | RSA |===

|=== | Size: | 4096/4096 |===

|=== | Created: | 2018-08-06 |===

|=== | Expires: | 2022-08-05 |===

|=== | User ID: | {aws} JSII Team link:mailto:aws-jsii@amazon.com[aws-jsii@amazon.com] |===

|=== | Key fingerprint: | 85EF 6522 4CE2 1E8C 72DB 28EC 1C7A CE4C B2A1 B93A |===

Select the “Copy” icon to copy the following OpenPGP key:

== [source,none,subs=“verbatim,attributes”]

—-BEGIN PGP PUBLIC KEY BLOCK—-

mQINBFtoSs0BEAD6WweLD0B26h0F7Jo9iR6tVQ4PgQBK1Va5H/eP+A2Iqw79UyxZ WNzHYhzQ5MjYYI1SgcPavXy5/LV1N8HJ7QzyKszybnLYpNTLPYArWE8ZM9ZmjvIR p1GzwnVBGQfoOlxyeutE9T5ZkAn45dTS5jlno4unji4gHjnwXKf2nP1APU2CZfdK 8vDpLOgj9LeeGlerYNbx+7xtY/I+csFIQvK09FPLSNMJQLlkBhY0r6Rt9ZQG+653 tJn+AUjyM237w0UIX1IqyYc5IONXu8HklPGu0NYuX9AY/63Ak2Cyfj0w/PZlvueQ noQNM3j0nkOEsTOEXCyaLQw9iBKpxvLnm5RjMSODDCkj8c9uu0LHr7J4EOtgt2S1 pem7Y/c/N+/Z+Ksg9fP8fVTfYwRPvdI1x2sCiRDfLoQSG9tdrN5VwPFi4sGV04sI x7Al8Vf/OBjAGZrDaJgM/gVvb9SKAQUA6t3ofeP14gDrS0eYodEXZ+lamnxFglxF Sn8NRC4JFNmkXSUaTNGUdFf//F0D69PRNT8CnFfmniGj0CphN5037PCA2LC/Buq2 3+K6mTPkCcCHYPC/SwItp/xIDAQsGuDc1i1SfDYXrjsK7uOuwC5jLA9X6wZ/jgXQ 4umRRJBAV1aW8b1+yfaYYCO2AfXXO6caObv8IvH7Pc4leC2DoqylD3KklQARAQAB tCNBV1MgSlNJSSBUZWFtIDxhd3MtanNpaUBhbWF6b24uY29tPokCPwQTAQgAKQUC W2hKzQIbLwUJB4TOAAcLCQgHAwIBBhUIAgkKCwQWAgMBAh4BAheAAAoJEBx6zkyy obk6B34P/iNb5QjKyhT0glZiq1wK7tuDDRpR6fC/sp6Jd/GhaNjO4BzlDbUPSjW5 950VT+qwaHXbIma/QVP7EIRztfwWy7m8eOodjpiu7JyJprhwG9nocXiNsLADcMoH BvabkDRWXWIWSurq2wbcFMlTVwxjHPIQs6kt2oojpzP985CDS/KTzyjow6/gfMim DLdhSSbDUM34STEgew79L2sQzL7cvM/N59k+AGyEMHZDXHkEw/Bge5Ovz50YOnsp lisH4BzPRIw7uWqPlkVPzJKwMuo2WvMjDfgbYLbyjfvs5mqDxT2GTwAx/rd2taU6 iSqP0QmLM54BtTVVdoVXZSmJyTmXAAGlITq8ECZ/coUW9K2pUSgVuWyu63lktFP6 MyCQYRmXPh9aSd4+ielteXM9Y39snlyLgEJBhMxioZXVO2oszwluPuhPoAp4ekwj /umVsBf6As6PoAchg7Qzr+lRZGmV9YTJOgDn2Z7jf/7tOes0g/mdiXTQMSGtp/Fp ggnifTBx3iXkrQhqHlwtam8XTHGHy3MvX17ZslNuB8Pjh+07hhCxv0VUVZPUHJqJ ZsLa398LMteQ8UMxwJ3t06jwDWAd7mbr2tatIilLHtWWBFoCwBh1XLe/03ENCpDp njZ7OsBsBK2nVVcN0H2v5ey0T1yE93o6r7xOwCwBiVp5skTCRUob =2Tag —-END PGP PUBLIC KEY BLOCK—- —

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cmd-ack] = cdk acknowledge :info\_titleabbrev: cdk ack :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation, cdk acknowledge, cdk ack

== [abstract]

## Acknowledge a notice by issue number and hide it from displaying again.

// Content start

Acknowledge a notice by issue number and hide it from displaying again.

This is useful to hide notices that have been addressed or do not apply to you.

Acknowledgements are saved at a CDK project level. If you acknowledge a notice in one CDK project, it will still display in other projects until acknowledged there.

[#ref-cli-cmd-ack-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk acknowledge ++++++++++++—-++++++++++++

[#ref-cli-cmd-ack-args] == Arguments

[#ref-cli-cmd-ack-args-notice-id] *Notice ID*:: The ID of the notice. + *Type*: String + *Required*: No

[#ref-cli-cmd-ack-options] == Options

For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-ack-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk acknowledge command.

[#ref-cli-cmd-ack-examples] == Examples

[#ref-cli-cmd-ack-examples-1] === Acknowledge and hide a notice that displays when running another CDK CLI command

== [source,none,subs=“verbatim,attributes”]

$ cdk deploy … # Normal output of the command

NOTICES

16603 Toggling off auto\_delete\_objects for Bucket empties the bucket

Overview: If a stack is deployed with an S3 bucket with  
 auto\_delete\_objects=True, and then re-deployed with  
 auto\_delete\_objects=False, all the objects in the bucket  
 will be deleted.  
  
 Affected versions: <1.126.0.  
  
 More information at: https://github.com/aws/aws-cdk/issues/16603

17061 Error when building EKS cluster with monocdk import

Overview: When using monocdk/aws-eks to build a stack containing  
 an EKS cluster, error is thrown about missing  
 lambda-layer-node-proxy-agent/layer/package.json.  
  
 Affected versions: >=1.126.0 <=1.130.0.  
  
 More information at: https://github.com/aws/aws-cdk/issues/17061

== $ cdk acknowledge 16603

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cmd-bootstrap] = cdk bootstrap :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation, bootstrap

== [abstract]

## Prepare an {aws} environment for CDK deployments by deploying the CDK bootstrap stack, named CDKToolkit, into the {aws} environment.

// Content start

Prepare an {aws} environment for CDK deployments by deploying the CDK bootstrap stack, named CDKToolkit, into the {aws} environment.

The bootstrap stack is a CloudFormation stack that provisions an Amazon S3 bucket and Amazon ECR repository in the {aws} environment. The {aws} CDK CLI uses these resources to store synthesized templates and related assets during deployment.

[#ref-cli-cmd-bootstrap-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk bootstrap ++++++++++++—-++++++++++++

[#ref-cli-cmd-bootstrap-args] == Arguments

[#ref-cli-cmd-bootstrap-args-env] *{aws} environment*:: The target {aws} environment to deploy the bootstrap stack to in the following format: aws://<account-id>/<region>. + Example: aws://123456789012/us-east-1 + This argument can be provided multiple times in a single command to deploy the bootstrap stack to multiple environments. + By default, the CDK CLI will bootstrap all environments referenced in the CDK app or will determine an environment from default sources. This could be an environment specified using the --profile option, from environment variables, or default {aws} CLI sources.

[#ref-cli-cmd-bootstrap-options] == Options

For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-bootstrap-options-bootstrap-bucket-name] --bootstrap-bucket-name, --toolkit-bucket-name, -b <STRING>:: The name of the Amazon S3 bucket that will be used by the CDK CLI. This bucket will be created and must not currently exist. + Provide this option to override the default name of the Amazon S3 bucket. + When you use this option, you may have to customize synthesis. To learn more, see xref:bootstrapping-custom-synth[Customize CDK stack synthesis]. + *Default value*: Undefined

[#ref-cli-cmd-bootstrap-options-bootstrap-customer-key] --bootstrap-customer-key <BOOLEAN>:: Create a Customer Master Key (CMK) for the bootstrap bucket (you will be charged but can customize permissions, modern bootstrapping only). + This option is not compatible with --bootstrap-kms-key-id. + *Default value*: Undefined

[#ref-cli-cmd-bootstrap-options-bootstrap-kms-key-id] --bootstrap-kms-key-id <STRING>:: The {aws} KMS master key ID to use for the SSE-KMS encryption. + Provide this option to override the default {aws} KMS key used to encrypt the Amazon S3 bucket. + This option is not compatible with --bootstrap-customer-key. + *Default value*: Undefined

[#ref-cli-cmd-bootstrap-options-cloudformation-execution-policies] --cloudformation-execution-policies <ARRAY>:: The managed IAM policy ARNs that should be attached to the deployment role assumed by {aws} CloudFormation during deployment of your stacks. + By default, stacks are deployed with full administrator permissions using the AdministratorAccess policy. + You can provide this option multiple times in a single command. You can also provide multiple ARNs as a single string, with the individual ARNs separated by commas. The following is an example: + [source,none,subs=“verbatim,attributes”] — $ cdk bootstrap –cloudformation-execution-policies “arn:aws:iam::aws:policy/AWSLambda\_FullAccess,arn:aws:iam::aws:policy/AWSCodeDeployFullAccess” — + To avoid deployment failures, be sure that the policies you specify are sufficient for any deployments that you will perform into the environment being bootstrapped. + This option applies to modern bootstrapping only. + [IMPORTANT] ==== The modern bootstrap template effectively grants the permissions implied by the --cloudformation-execution-policies to any {aws} account in the --trust list. By default, this extends permissions to read and write to any resource in the bootstrapped account. Make sure to xref:bootstrapping-customizing[configure the bootstrapping stack] with policies and trusted accounts that you are comfortable with. ==== + *Default value*: []

[#ref-cli-cmd-bootstrap-options-custom-permissions-boundary] --custom-permissions-boundary, -cpb <STRING>:: Specify the name of a permissions boundary to use. + This option is not compatible with --example-permissions-boundary. + *Default value*: Undefined

[#ref-cli-cmd-bootstrap-options-example-permissions-boundary] --example-permissions-boundary, -epb <BOOLEAN>:: Use the example permissions boundary, supplied by the {aws} CDK. + This option is not compatible with --custom-permissions-boundary. + The CDK supplied permissions boundary policy should be regarded as an example. Edit the content and reference the example policy if you are testing out the feature. Convert it into a new policy for actual deployments, if one does not already exist. The concern is to avoid drift. Most likely, a permissions boundary is maintained and has dedicated conventions, naming included. + For more information on configuring permissions, including using permissions boundaries, see the https://github.com/aws/aws-cdk/wiki/Security-And-Safety-Dev-Guide[Security and Safety Dev Guide]. + *Default value*: Undefined

[#ref-cli-cmd-bootstrap-options-execute] --execute <BOOLEAN>:: Configure whether to execute the change set. + *Default value*: true

[#ref-cli-cmd-bootstrap-options-force] --force, -f <BOOLEAN>:: Always bootstrap, even if it would downgrade the bootstrap template version. + *Default value*: false

[#ref-cli-cmd-bootstrap-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk bootstrap command.

[#ref-cli-cmd-bootstrap-options-previous-parameters] --previous-parameters <BOOLEAN>:: Use previous values for existing parameters. + Once a bootstrap template is deployed with a set of parameters, you must set this option to false to change any parameters on future deployments. When false, you must re-supply all previously supplied parameters. + *Default value*: true

[#ref-cli-cmd-bootstrap-options-public-access-block-configuration] --public-access-block-configuration <BOOLEAN>:: Block public access configuration on the Amazon S3 bucket that is created and used by the CDK CLI. + *Default value*: true

[#ref-cli-cmd-bootstrap-options-qualifier] --qualifier <STRING>:: Nine-digit string value that is unique for each bootstrap stack. This value is added to the physical ID of resources in the bootstrap stack. + By providing a qualifier, you avoid resource name clashes when provisioning multiple bootstrap stacks in the same environment. + When you change the qualifier, your CDK app must pass the changed value to the stack synthesizer. For more information, see xref:bootstrapping-custom-synth[Customize CDK stack synthesis]. + *Default value*: hnb659fds. This value has no significance.

[#ref-cli-cmd-bootstrap-options-show-template] --show-template <BOOLEAN>:: Instead of bootstrapping, print the current bootstrap template to the standard output (stdout). You can then copy and customize the template as necessary. + *Default value*: false

[#ref-cli-cmd-bootstrap-options-tags] --tags, -t <ARRAY>:: Tags to add to the bootstrap stack in the format of KEY=VALUE. + *Default value*: []

[#ref-cli-cmd-bootstrap-options-template] --template <STRING>:: Use the template from the given file instead of the built-in one.

[#ref-cli-cmd-bootstrap-options-termination-protection] --termination-protection <BOOLEAN>:: Toggle {aws} CloudFormation termination protection on the bootstrap stack. + When true, termination protection is enabled. This prevents the bootstrap stack from being accidentally deleted. + To learn more about termination protection, see https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-protect-stacks.html[Protecting a stack from being deleted] in the *{aws} CloudFormation User Guide*. + *Default value*: Undefined

[#ref-cli-cmd-bootstrap-options-toolkit-stack-name] --toolkit-stack-name <STRING>:: The name of the bootstrap stack to create. + By default, cdk bootstrap deploys a stack named CDKToolkit into the specified {aws} environment. Use this option to provide a different name for your bootstrap stack. + The CDK CLI uses this value to verify your bootstrap stack version. + *Default value*: CDKToolkit + *Required*: Yes

[#ref-cli-cmd-bootstrap-options-trust] --trust <ARRAY>:: The {aws} account IDs that should be trusted to perform deployments into this environment. + The account performing the bootstrapping will always be trusted. + This option requires that you also provide --cloudformation-execution-policies. + You can provide this option multiple times in a single command. + This option applies to modern bootstrapping only. + To add trusted accounts to an existing bootstrap stack, you must specify all of the accounts to trust, including those that you may have previously provided. If you only provide new accounts to trust, the previously trusted accounts will be removed. + The following is an example that trusts two accounts: + [source,none,subs=“verbatim,attributes”] — $ cdk bootstrap aws://123456789012/us-west-2 –trust 234567890123 –trust 987654321098 –cloudformation-execution-policies arn:aws:iam::aws:policy/AdministratorAccess ⏳ Bootstrapping environment aws://123456789012/us-west-2… Trusted accounts for deployment: 234567890123, 987654321098 Trusted accounts for lookup: (none) Execution policies: arn:aws:iam::aws:policy/AdministratorAccess CDKToolkit: creating CloudFormation changeset… ✅ Environment aws://123456789012/us-west-2 bootstrapped. — + [IMPORTANT] ==== The modern bootstrap template effectively grants the permissions implied by the --cloudformation-execution-policies to any {aws} account in the --trust list. By default, this extends permissions to read and write to any resource in the bootstrapped account. Make sure to xref:bootstrapping-customizing[configure the bootstrapping stack] with policies and trusted accounts that you are comfortable with. ==== + *Default value*: []

[#ref-cli-cmd-bootstrap-options-trust-for-lookup] --trust-for-lookup <ARRAY>:: The {aws} account IDs that should be trusted to look up values in this environment. + Use this option to give accounts permission to synthesize stacks that will be deployed into the environment, without actually giving them permission to deploy those stacks directly. + You can provide this option multiple times in a single command. + This option applies to modern bootstrapping only. + *Default value*: []

[#ref-cli-cmd-bootstrap-examples] == Examples

[#ref-cli-cmd-bootstrap-examples-1] === Bootstrap the {aws} environment specified in the prod profile

== [source,none,subs=“verbatim,attributes”]

## $ cdk bootstrap –profile prod

[#ref-cli-cmd-bootstrap-examples-2] === Deploy the bootstrap stack to environments foo and bar

== [source,none,subs=“verbatim,attributes”]

## $ cdk bootstrap –app=‘node bin/main.js’ foo bar

[#ref-cli-cmd-bootstrap-examples-3] === Export the bootstrap template to customize it

If you have specific requirements that are not met by the bootstrap template, you can customize it to fit your needs.

You can export the bootstrap template, modify it, and deploy it using {aws} CloudFormation. The following is an example of exporting the existing template:

== [source,none,subs=“verbatim,attributes”]

## $ cdk bootstrap –show-template > bootstrap-template.yaml

You can also tell the CDK CLI to use a custom template. The following is an example:

== [source,none,subs=“verbatim,attributes”]

## $ cdk bootstrap –template my-bootstrap-template.yaml

[#ref-cli-cmd-bootstrap-examples-4] === Bootstrap with a permissions boundary. Then, remove that permissions boundary

To bootstrap with a custom permissions boundary, we run the following:

== [source,none,subs=“verbatim,attributes”]

## $ cdk bootstrap –custom-permissions-boundary my-permissions-boundary

To remove the permissions boundary, we run the following:

== [source,none,subs=“verbatim,attributes”]

## $ cdk bootstrap –no-previous-parameters

[#ref-cli-cmd-bootstrap-examples-5] === Use a qualifier to distinguish resources that are created for a development environment

== [source,none,subs=“verbatim,attributes”]

## $ cdk bootstrap –qualifier dev2024

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cmd-context] = cdk context :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation, cdk context

== [abstract]

## Manage cached context values for your {aws} CDK application.

// Content start

Manage cached context values for your {aws} CDK application.

*Context* represents the configuration and environment information that can influence how your stacks are synthesized and deployed. Use cdk context to do the following:

* View your configured context values.
* Set and manage context values.
* Remove context values.

[#ref-cli-cmd-context-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk context ++++++—-++++++

[#ref-cli-cmd-context-options] == Options

For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-context-options-clear] --clear <BOOLEAN>:: Clear all context.

[#ref-cli-cmd-context-options-force] --force, -f <BOOLEAN>:: Ignore missing key error. + *Default value*: false

[#ref-cli-cmd-context-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk context command.

[#ref-cli-cmd-context-options-reset] --reset, -e <STRING>:: The context key, or its index, to reset.

include::attributes.txt[]

// Attributes [.topic] [#ref-cli-cmd-deploy] = cdk deploy :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation, cdk deploy

== [abstract]

## Deploy one or more {aws} CDK stacks into your {aws} environment.

// Content start

Deploy one or more {aws} CDK stacks into your {aws} environment.

During deployment, the CDK CLI will output progress indicators, similar to what can be observed from the {aws} CloudFormation console.

If the {aws} environment is not bootstrapped, only stacks without assets and with synthesized templates under 51,200 bytes will successfully deploy.

[#ref-cli-cmd-deploy-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk deploy ++++++++++++—-++++++++++++

[#ref-cli-cmd-deploy-args] == Arguments

[#ref-cli-cmd-deploy-args-stack-name] *CDK stack ID*:: The construct ID of the CDK stack from your app to deploy. + *Type*: String + *Required*: No

[#ref-cli-cmd-deploy-options] == Options

For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-deploy-options-all] --all <BOOLEAN>:: Deploy all stacks in your CDK app. + *Default value*: false

[#ref-cli-cmd-deploy-options-asset-parallelism] --asset-parallelism <BOOLEAN>:: Specify whether to build and publish assets in parallel.

[#ref-cli-cmd-deploy-options-asset-prebuild] --asset-prebuild <BOOLEAN>:: Specify whether to build all assets before deploying the first stack. This option is useful for failing Docker builds. + *Default value*: true

[#ref-cli-cmd-deploy-options-build-exclude] --build-exclude, -E <ARRAY>:: Do not rebuild asset with the given ID. + This option can be specified multiple times in a single command. + *Default value*: []

[#ref-cli-cmd-deploy-options-change-set-name] --change-set-name <STRING>:: The name of the {aws} CloudFormation change set to create. + This option is not compatible with --method='direct'.

[#ref-cli-cmd-deploy-options-concurrency] --concurrency <NUMBER>:: Deploy multiple stacks in parallel while accounting for inter-stack dependencies. Use this option to speed up deployments. You must still factor in {aws} CloudFormation and other {aws} account rate limiting. + Provide a number to specify the maximum number of simultaneous deployments (dependency permitting) to perform. + *Default value*: 1

[#ref-cli-cmd-deploy-options-exclusively] --exclusively, -e <BOOLEAN>:: Only deploy requested stacks and don’t include dependencies.

[#ref-cli-cmd-deploy-options-force] --force, -f <BOOLEAN>:: When you deploy to update an existing stack, the CDK CLI will compare the template and tags of the deployed stack to the stack about to be deployed. If no changes are detected, the CDK CLI will skip deployment. + To override this behavior and always deploy stacks, even if no changes are detected, use this option. + *Default value*: false

[#ref-cli-cmd-deploy-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk deploy command.

[#ref-cli-cmd-deploy-options-hotswap] --hotswap <BOOLEAN>:: Hotswap deployments for faster development. This option attempts to perform a faster, hotswap deployment if possible. For example, if you modify the code of a Lambda function in your CDK app, the CDK CLI will update the resource directly through service APIs instead of performing a CloudFormation deployment. + If the CDK CLI detects changes that don’t support hotswapping, those changes will be ignored and a message will display. If you prefer to perform a full CloudFormation deployment as a fall back, use --hotswap-fallback instead. + The CDK CLI uses your current {aws} credentials to perform the API calls. It does not assume the roles from your bootstrap stack, even if the @aws-cdk/core:newStyleStackSynthesis feature flag is set to true. Those roles do not have the necessary permissions to update {aws} resources directly, without using CloudFormation. For that reason, make sure that your credentials are for the same {aws} account of the stacks that you are performing hotswap deployments against and that they have the necessary IAM permissions to update the resources. + Hotswapping is currently supported for the following changes: + –

* Code assets (including Docker images and inline code), tag changes, and configuration changes (only description and environment variables are supported) of Lambda functions.
* Lambda versions and alias changes.
* Definition changes of {aws} Step Functions state machines.
* Container asset changes of Amazon ECS services.
* Website asset changes of Amazon S3 bucket deployments.
* Source and environment changes of {aws} CodeBuild projects.
* VTL mapping template changes for {aws} AppSync resolvers and functions.
* {blank}
* == Schema changes for {aws} AppSync GraphQL APIs.
* Usage of certain CloudFormation intrinsic functions are supported as part of a hotswapped deployment. These include:
* –
* Ref
* Fn::GetAtt – Only partially supported. Refer to link:https://github.com/aws/aws-cdk/blob/main/packages/aws-cdk/lib/api/evaluate-cloudformation-template.ts#L477-L492[this implementation] for supported resources and attributes.
* Fn::ImportValue
* Fn::Join
* Fn::Select
* Fn::Split
* {blank}
* == Fn::Sub
* This option is also compatible with nested stacks.

# [NOTE]

* \*\* This option deliberately introduces drift in CloudFormation stacks in order to speed up deployments. For this reason, only use it for development purposes. Do not use this option for your production deployments. \*\* This option is considered experimental and may have breaking changes in the future. \*\* Defaults for certain parameters may be different with the hotswap parameter. For example, an Amazon ECS service``’s minimum healthy percentage will currently be set at 0. Review the source accordingly if this occurs. ====
* *Default value*: `false

[#ref-cli-cmd-deploy-options-hotswap-fallback] --hotswap-fallback <BOOLEAN>:: This option is is similar to --hotswap. The difference being that --hotswap-fallback will fall back to perform a full CloudFormation deployment if a change is detected that requires it. + For more information about this option, see --hotswap. + *Default value*: false

[#ref-cli-cmd-deploy-options-ignore-no-stacks] --ignore-no-stacks <BOOLEAN>:: Perform a deployment even if your CDK app doesn’t contain any stacks. + This option is helpful in the following scenario: You may have an app with multiple environments, such as dev and prod. When starting development, your prod app may not have any resources, or the resources may be commented out. This will result in a deployment error with a message stating that the app has no stacks. Use --ignore-no-stacks to bypass this error. + *Default value*: false

[#ref-cli-cmd-deploy-options-import-existing-resources] --import-existing-resources <BOOLEAN>:: Import existing, unmanaged {aws} CloudFormation resources from your {aws} account. + When you use this option, resources from your synthesized {aws} CloudFormation template with the same custom name as existing unmanaged resources in the same account will be imported into your stack. + You can use this option to import existing resources into new or existing stacks. + You can import existing resources and deploy new resources in the same cdk deploy command. + To learn more about custom names, see https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-name.html[Name type] in the *{aws} CloudFormation User Guide*. + To learn more about the ImportExistingResources CloudFormation parameter, see https://aws.amazon.com/about-aws/whats-new/2023/11/aws-cloudformation-import-parameter-changesets/[{aws} CloudFormation simplifies resource import with a new parameter for ChangeSets]. + For more information on using this option, see link:https://github.com/aws/aws-cdk-cli/tree/main/packages/aws-cdk#import-existing-resources[Import existing resources] in the *aws-cdk-cli GitHub repository*.

[#ref-cli-cmd-deploy-options-logs] --logs <BOOLEAN>:: Show Amazon CloudWatch log in the standard output (stdout) for all events from all resources in the selected stacks. + This option is only compatible with --watch. + *Default value*: true

[#ref-cli-cmd-deploy-options-method] --method, -m <STRING>:: Configure the method to perform a deployment. + –

* change-set – Default method. The CDK CLI creates a CloudFormation change set with the changes that will be deployed, then performs deployment.
* direct – Do not create a change set. Instead, apply the change immediately. This is typically faster than creating a change set, but you lose deployment progress details in the CLI output.
* {blank}
* == prepare-change-set – Create change set but don’t perform deployment. This is useful if you have external tools that will inspect the change set or if you have an approval process for change sets.
* *Valid values*: change-set, direct, prepare-change-set
* *Default value*: change-set

[#ref-cli-cmd-deploy-options-notification-arns] --notification-arns <ARRAY>:: The ARNs of Amazon SNS topics that CloudFormation will notify for stack related events.

[#ref-cli-cmd-deploy-options-outputs-file] --outputs-file, -O <STRING>:: The path to where stack outputs from deployments are written to. + After deployment, stack outputs will be written to the specified output file in JSON format. + You can configure this option in the project’s cdk.json file or at ~/.cdk.json on your local development machine: + [source,json,subs=“verbatim,attributes”] — { “app”: “npx ts-node bin/myproject.ts”, // … “outputsFile”: “outputs.json” } — + If multiple stacks are deployed, outputs are written to the same output file, organized by keys representing the stack name.

[#ref-cli-cmd-deploy-options-parameters] --parameters <ARRAY>:: Pass additional parameters to CloudFormation during deployment. + This option accepts an array in the following format: STACK:KEY=VALUE. + –

* STACK – The name of the stack to associate the parameter with.
* KEY – The name of the parameter from your stack.
* {blank}
* == VALUE – The value to pass at deployment.
* If a stack name is not provided, or if \* is provided as the stack name, parameters will be applied to all stacks being deployed. If a stack does not make use of the parameter, deployment will fail.
* Parameters do not propagate to nested stacks. To pass parameters to nested stacks, use the +link:https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.NestedStack.html[NestedStack]+ construct.
* *Default value*: {}

[#ref-cli-cmd-deploy-options-previous-parameters] --previous-parameters <BOOLEAN>:: Use previous values for existing parameters. + When this option is set to false, you must specify all parameters on every deployment. + *Default value*: true

[#ref-cli-cmd-deploy-options-progress] --progress <STRING>:: Configure how the CDK CLI displays deployment progress. + –

* bar – Display stack deployment events as a progress bar, with the events for the resource currently being deployed.
* {blank}
* == events – Provide a complete history, including all CloudFormation events.
* You can also configure this option in the project’s cdk.json file or at ~/.cdk.json on your local development machine:

## [source,json,subs=“verbatim,attributes”]

* { “progress”: “events” } —
* *Valid values*: bar, events
* *Default value*: bar

[#ref-cli-cmd-deploy-options-require-approval] --require-approval <STRING>:: Specify what security-sensitive changes require manual approval. + –

* any-change – Manual approval required for any change to the stack.
* broadening – Manual approval required if changes involve a broadening of permissions or security group rules.
* {blank}
* == never – Approval is not required.
* *Valid values*: any-change, broadening, never
* *Default value*: broadening

[#ref-cli-cmd-deploy-options-rollback] --rollback | --no-rollback, -R:: During deployment, if a resource fails to be created or updated, the deployment will roll back to the latest stable state before the CDK CLI returns. All changes made up to that point will be undone. Resources that were created will be deleted and updates that were made will be rolled back. + Specify --no-rollback to turn off this behavior. If a resource fails to be created or updated, the CDK CLI will leave changes made up to that point in place and return. This will leave your deployment in a failed, paused state. From here, you can update your code and try the deployment again. This may be helpful in development environments where you are iterating quickly. + If a deployment performed with --no-rollback fails, and you decide that you want to rollback the deployment, you can use the cdk rollback command. For more information, see xref:ref-cli-cmd-rollback[cdk rollback]. + [NOTE] ==== With --no-rollback, deployments that cause resource replacements will always fail. You can only use this option value for deployments that update or create new resources. ==== + *Default value*: --rollback

[#ref-cli-cmd-deploy-options-toolkit-stack-name] --toolkit-stack-name <STRING>:: The name of the existing CDK Toolkit stack. + By default, cdk bootstrap deploys a stack named CDKToolkit into the specified {aws} environment. Use this option to provide a different name for your bootstrap stack. + The CDK CLI uses this value to verify your bootstrap stack version.

[#ref-cli-cmd-deploy-options-watch] --watch <BOOLEAN>:: Continuously observe CDK project files, and deploy the specified stacks automatically when changes are detected. + This option implies --hotswap by default. + This option has an equivalent CDK CLI command. For more information, see xref:ref-cli-cmd-watch[cdk watch].

[#ref-cli-cmd-deploy-examples] == Examples

[#ref-cli-cmd-deploy-examples-1] === Deploy the stack named MyStackName

== [source,none,subs=“verbatim,attributes”]

## $ cdk deploy MyStackName –app=‘node bin/main.js’

[#ref-cli-cmd-deploy-examples-2] === Deploy multiple stacks in an app

Use cdk list to list your stacks:

== [source,none,subs=“verbatim,attributes”]

$ cdk list CdkHelloWorldStack CdkStack2 CdkStack3 —

To deploy all stacks, use the --all option:

== [source,none,subs=“verbatim,attributes”]

## $ cdk deploy –all

To choose which stacks to deploy, provide stack names as arguments:

== [source,none,subs=“verbatim,attributes”]

## $ cdk deploy CdkHelloWorldStack CdkStack3

[#ref-cli-cmd-deploy-examples-3] === Deploy pipeline stacks

Use cdk list to show stack names as paths, showing where they are in the pipeline hierarchy:

== [source,none,subs=“verbatim,attributes”]

$ cdk list PipelineStack PiplelineStack/Prod PipelineStack/Prod/MyService —

Use the --all option or the wildcard \\* to deploy all stacks. If you have a hierarchy of stacks as described above, --all and \* will only match stacks on the top level. To match all stacks in the hierarchy, use +\*\*+.

You can combine these patterns. The following deploys all stacks in the Prod stage:

== [source,none,subs=“verbatim,attributes”]

## $ cdk deploy PipelineStack/Prod/\*\*

[#ref-cli-cmd-deploy-examples-4] === Pass parameters at deployment

Define parameters in your CDK stack. The following is an example that creates a parameter named TopicNameParam for an Amazon SNS topic:

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

new sns.Topic(this, ‘TopicParameter’, { topicName: new cdk.CfnParameter(this, ‘TopicNameParam’).value.toString() }); —

To provide a parameter value of parameterized, run the following:

== [source,none,subs=“verbatim,attributes”]

## $ cdk deploy –parameters “MyStackName:TopicNameParam=parameterized”

You can override parameter values by using the --force option. The following is an example of overriding the topic name from a previous deployment:

== [source,none,subs=“verbatim,attributes”]

## $ cdk deploy –parameters “MyStackName:TopicNameParam=parameterName” –force

[#ref-cli-cmd-deploy-examples-5] === Write stack outputs to a file after deployment

Define outputs in your CDK stack file. The following is an example that creates an output for a function ARN:

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

const fn = new lambda.Function(this, “fn”, { handler: “index.handler”, code: lambda.Code.fromInline(exports.handler = \${handler.toString()}), runtime: lambda.Runtime.NODEJS\_LATEST });

new cdk.CfnOutput(this, ‘FunctionArn’, { value: fn.functionArn, }); —

Deploy the stack and write outputs to outputs.json:

== [source,none,subs=“verbatim,attributes”]

## $ cdk deploy –outputs-file outputs.json

The following is an example of outputs.json after deployment:

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

{ “MyStack”: { “FunctionArn”: “arn:aws:lambda:us-east-1:123456789012:function:MyStack-fn5FF616E3-G632ITHSP5HK” } } —

From this example, the key FunctionArn corresponds to the logical ID of the CfnOutput instance.

The following is an example of outputs.json after deployment when multiple stacks are deployed:

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

{ “MyStack”: { “FunctionArn”: “arn:aws:lambda:us-east-1:123456789012:function:MyStack-fn5FF616E3-G632ITHSP5HK” }, “AnotherStack”: { “VPCId”: “vpc-z0mg270fee16693f” } } —

[#ref-cli-cmd-deploy-examples-6] === Modify the deployment method

To deploy faster, without using change sets, use --method='direct':

== [source,none,subs=“verbatim,attributes”]

## $ cdk deploy –method=‘direct’

To create a change set but don’t deploy, use --method='prepare-change-set'. By default, a change set named cdk-deploy-change-set will be created. If a previous change set with this name exists, it will be overwritten. If no changes are detected, an empty change set is still created.

You can also name your change set. The following is an example:

== [source,none,subs=“verbatim,attributes”]

## $ cdk deploy –method=‘prepare-change-set’ –change-set-name=‘MyChangeSetName’

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cmd-destroy] = cdk destroy :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation, cdk destroy

== [abstract]

## Delete one or more {aws} CDK stacks from your {aws} environment.

// Content start

Delete one or more {aws} CDK stacks from your {aws} environment.

When you delete a stack, resources in the stack will be destroyed, unless they were configured with a DeletionPolicy of Retain.

During stack deletion, this command will output progress information similar to cdk deploy behavior.

[#ref-cli-cmd-destroy-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk destroy ++++++++++++—-++++++++++++

[#ref-cli-cmd-destroy-args] == Arguments

[#ref-cli-cmd-destroy-args-stack-name] *CDK stack ID*:: The construct ID of the CDK stack from your app to delete. + *Type*: String + *Required*: No

[#ref-cli-cmd-destroy-options] == Options

For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-destroy-options-all] --all <BOOLEAN>:: Destroy all available stacks. + *Default value*: false

[#ref-cli-cmd-destroy-options-exclusively] --exclusively, -e <BOOLEAN>:: Only destroy requested stacks and don’t include dependencies.

[#ref-cli-cmd-destroy-options-force] --force, -f <BOOLEAN>:: Do not ask for confirmation before destroying the stacks.

[#ref-cli-cmd-destroy-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk destroy command.

[#ref-cli-cmd-destroy-examples] == Examples

[#ref-cli-cmd-destroy-examples-1] === Delete a stack named MyStackName

== [source,none,subs=“verbatim,attributes”]

$ cdk destroy –app=‘node bin/main.js’ ++++++—-++++++

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cmd-diff] = cdk diff :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation, cdk diff

== [abstract]

## Perform a diff to see infrastructure changes between {aws} CDK stacks.

// Content start

Perform a diff to see infrastructure changes between {aws} CDK stacks.

This command is typically used to compare differences between the current state of stacks in your local CDK app against deployed stacks. However, you can also compare a deployed stack with any local {aws} CloudFormation template.

[#ref-cli-cmd-diff-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk diff ++++++++++++—-++++++++++++

[#ref-cli-cmd-diff-args] == Arguments

[#ref-cli-cmd-diff-args-stack-name] *CDK stack ID*:: The construct ID of the CDK stack from your app to perform a diff. + *Type*: String + *Required*: No

[#ref-cli-cmd-diff-options] == Options

For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-diff-options-change-set] --change-set <BOOLEAN>:: Specify whether to create a change set to analyze resource replacements. + When true, the CDK CLI will create an {aws} CloudFormation change set to display the exact changes that will be made to your stack. This output includes whether resources will be updated or replaced. The CDK CLI uses the deploy role instead of the lookup role to perform this action. + When false, a quicker, but less-accurate diff is performed by comparing CloudFormation templates. Any change detected to properties that require resource replacement will be displayed as a resource replacement, even if the change is purely cosmetic, like replacing a resource reference with a hard-coded ARN. + *Default value*: true

[#ref-cli-cmd-diff-options-context-lines] --context-lines <NUMBER>:: Number of context lines to include in arbitrary JSON diff rendering. + *Default value*: 3

[#ref-cli-cmd-diff-options-exclusively] --exclusively, -e <BOOLEAN>:: Only diff requested stacks and don’t include dependencies.

[#ref-cli-cmd-diff-options-fail] --fail <BOOLEAN>:: Fail and exit with a code of 1 if differences are detected.

[#ref-cli-cmd-diff-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk diff command.

[#ref-cli-cmd-diff-options-processed] --processed <BOOLEAN>:: Specify whether to compare against the template with CloudFormation transforms already processed. + *Default value*: false

[#ref-cli-cmd-diff-options-quiet] --quiet, -q <BOOLEAN>:: Do not print the CDK stack name and default cdk diff message to stdout when no changes are detected. + *Default value*: false

[#ref-cli-cmd-diff-options-security-only] --security-only <BOOLEAN>:: Only diff for broadened security changes. + *Default value*: false

[#ref-cli-cmd-diff-options-strict] --strict <BOOLEAN>:: Modify cdk diff behavior to be more precise or stringent. When true, the CDK CLI will not filter out +{aws}::CDK::Metadata+ resources or unreadable non-ASCII characters. + *Default value*: false

[#ref-cli-cmd-diff-options-template] --template <STRING>:: The path to the CloudFormation template to compare a CDK stack with.

[#ref-cli-cmd-diff-examples] == Examples

[#ref-cli-cmd-diff-examples-1] === Diff against the currently deployed stack named MyStackName

The CDK CLI uses the following symbols in the diff output:

* [+] – Identifies code or resources that will be added if you deploy your changes.
* [-] – Identifies code or resources that will be removed if you deploy your changes.
* [~] – Identifies a resource or property that will be modified if you deploy your changes.

The following is an example that shows a diff of local changes to a Lambda function:

== [source,none,subs=“verbatim,attributes”]

$ cdk diff MyStackName start: Building ++++++:++++++success: Built ++++++:++++++start: Publishing ++++++:++++++success: Published ++++++:++++++Hold on while we create a read-only change set to get a diff with accurate replacement information (use –no-change-set to use a less accurate but faster template-only diff) Stack MyStackName Resources [~] {aws}::Lambda::Function HelloWorldFunction ++++++└─ [~] Code └─ [~] .ZipFile: ├─ [-] exports.handler = async function(event) { return { statusCode: 200, body: JSON.stringify(‘Hello World!’), }; };++++++++++++++++++++++++++++++++++++++++++++++++++++++

└─ [+]  
 exports.handler = async function(event) {  
 return {  
 statusCode: 200,  
 body: JSON.stringify('Hello from CDK!'),  
 };  
 };

== ✨ Number of stacks with differences: 1

A [~] indicator for resources that will be modified does not always mean a full resource replacement:

* Some resource properties, like Code, will update the resource.
* Some resource properties, like FunctionName, may cause a full resource replacement.

[#ref-cli-cmd-diff-examples-2] === Diff against a specific CloudFormation template

== [source,none,subs=“verbatim,attributes”]

## $ cdk diff MyStackName –app=‘node bin/main.js’ –template-path=‘./MyStackNameTemplate.yaml’

[#ref-cli-cmd-diff-examples-3] === Diff a local stack with its deployed stack. don’t print to stdout if no changes are detected

== [source,none,subs=“verbatim,attributes”]

## $ cdk diff MyStackName –app=‘node bin/main.js’ –quiet

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cmd-docs] = cdk docs :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation

== [abstract]

## Open {aws} CDK documentation in your browser.

// Content start

Open {aws} CDK documentation in your browser.

[#ref-cli-cmd-docs-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk docs ++++++—-++++++

[#ref-cli-cmd-docs-options] == Options

For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-docs-options-browser] --browser, -b <STRING>:: The command to use to open the browser, using %u as a placeholder for the path of the file to open. + *Default value*: open %u

[#ref-cli-cmd-docs-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk docs command.

[#ref-cli-cmd-docs-examples] == Examples

[#ref-cli-cmd-docs-examples-1] === Open {aws} CDK documentation in Google Chrome

== [source,none,subs=“verbatim,attributes”]

## $ cdk docs –browser=‘chrome %u’

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cmd-doctor] = cdk doctor :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation

== [abstract]

## Inspect and display useful information about your local {aws} CDK project and development environment.

// Content start

Inspect and display useful information about your local {aws} CDK project and development environment.

This information can help with troubleshooting CDK issues and should be provided when submitting bug reports.

[#ref-cli-cmd-doctor-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk doctor ++++++—-++++++

[#ref-cli-cmd-doctor-options] == Options

For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-doctor-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk doctor command.

[#ref-cli-cmd-doctor-examples] == Examples

[#ref-cli-cmd-doctor-examples-1] === Simple example of the cdk doctor command

== [source,none,subs=“verbatim,attributes”]

$ cdk doctor ℹ️ CDK Version: 1.0.0 (build e64993a) ℹ️ {aws} environment variables:

* AWS\_EC2\_METADATA\_DISABLED = 1
* {blank}
* == AWS\_SDK\_LOAD\_CONFIG = 1

:doctype: book

include::attributes.txt[]

// Attributes [.topic] [#ref-cli-cmd-drift] = cdk drift :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit CLI, cdk drift

== [abstract]

## Detect and report drift in deployed {aws} CloudFormation stacks that are defined in your CDK app.

// Content start

Detect configuration drift for resources that you define, manage, and deploy using the {aws} Cloud Development Kit ({aws} CDK). Drift occurs when a stack’s actual configuration differs from its expected configuration, which happens when resources are modified outside of {aws} CloudFormation.

This command identifies resources that have been modified (for example, through the {aws} Console or {aws} CLI) by comparing their current state against their expected configuration. These modifications can cause unexpected behavior in your infrastructure.

During drift detection, the CDK CLI will output progress indicators and results, showing:

* Resources that have drifted from their expected configuration.
* The total number of resources with drift.
* A summary indicating whether drift was detected in the stack.

= [IMPORTANT]

The cdk drift and cdk diff commands work differently:

* cdk drift calls CloudFormation’s drift detection operation to compare the actual state of resources in {aws} (“reality”) against their expected configuration in CloudFormation. Not all {aws} resources support drift detection. For a list of supported resources, see link:https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/resource-import-supported-resources.html[Resource type support] in the *{aws} CloudFormation User Guide*.
* cdk diff compares the CloudFormation template synthesized from your local CDK code against the template of the deployed CloudFormation stack.

= Use cdk drift when you need to verify if resources have been modified outside of CloudFormation (for example, through the {aws} Console or {aws} CLI). Use cdk diff when you want to preview how your local code changes would affect your infrastructure before deployment.

[#ref-cli-cmd-drift-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk drift ++++++++++++—-++++++++++++

[#ref-cli-cmd-drift-args] == Arguments

[#ref-cli-cmd-drift-args-stack-name] *Stack name*:: The name of the stack that you want to check for drift. The stack must be previously deployed to CloudFormation to perform drift detection. + *Type*: String + *Required*: No + If no stack is specified, drift detection will be performed on all stacks defined in your CDK app.

[#ref-cli-cmd-drift-options] == Options

For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-drift-options-fail] --fail <BOOLEAN>:: Return with exit code 1 if drift is detected. + *Default value*: false

[#ref-cli-cmd-drift-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk drift command.

[#ref-cli-cmd-drift-examples] == Examples

[#ref-cli-cmd-drift-examples-stack] === Check drift for a specific stack

== [source,none,subs=“verbatim,attributes”]

## $ cdk drift MyStackName

The command will output results similar to:

== [source,none,subs=“verbatim,attributes”]

Stack MyStackName Modified Resources [~] AWS::Lambda::Function MyFunction MyLambdaFunc1234ABCD └─ [~] /Description ├─ [-] My original hello world Lambda function └─ [+] My drifted hello world Lambda function

1 resource has drifted from their expected configuration

== ✨ Number of resources with drift: 1

[#ref-cli-cmd-drift-examples-deleted] === Check drift when resources have been deleted

The following example shows what the output looks like when resources have been both modified and deleted:

== [source,none,subs=“verbatim,attributes”]

Stack MyStackName Modified Resources [~] AWS::Lambda::Function MyFunction MyLambdaFunc1234ABCD └─ [~] /Description ├─ [-] My original hello world Lambda function └─ [+] My drifted hello world Lambda function Deleted Resources [-] AWS::CloudWatch::Alarm MyAlarm MyCWAlarmABCD1234

2 resources have drifted from their expected configuration

== ✨ Number of resources with drift: 2

[#ref-cli-cmd-drift-examples-fail] === Check drift with exit code

To have the command return a non-zero exit code if drift is detected:

== [source,none,subs=“verbatim,attributes”]

## $ cdk drift MyStackName –fail

This is useful in CI/CD pipelines to automatically detect and respond to infrastructure drift.

:doctype: book

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cmd-gc] = cdk gc :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation, cdk gc, garbage collection, assets, Amazon S3

== [abstract]

## Use the {aws} Cloud Development Kit ({aws} CDK) command line interface (CLI) cdk gc command to perform garbage collection on unused assets stored in the resources of your bootstrap stack. Use this command to view, manage, and delete assets that you no longer need.

// Content start

Use the {aws} Cloud Development Kit ({aws} CDK) command line interface (CLI) cdk gc command to perform garbage collection on unused assets stored in the resources of your bootstrap stack. Use this command to view, manage, and delete assets that you no longer need.

For Amazon Simple Storage Service (Amazon S3) assets, the CDK CLI will check existing {aws} CloudFormation templates in the same environment to see if they are referenced. If not referenced, they will be considered unused and eligible for garbage collection actions.

= [WARNING]

The cdk gc command is in development for the {aws} CDK. The current features of this command are considered production-ready and safe to use. However, the scope of this command and its features are subject to change. Therefore, you must opt in by providing the unstable=gc option to use this command.

====

[#ref-cli-cmd-gc-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk gc ++++++++++++—-++++++++++++

[#ref-cli-cmd-gc-args] == Arguments

[#ref-cli-cmd-gc-args-env] *{aws} environment*:: The target {aws} environment to perform garbage collection actions on. + When providing an environment, use the following format: aws://<account-id>/<region>. For example, aws://<123456789012>/<us-east-1>. + This argument can be provided multiple times in a single command to perform garbage collection actions on multiple environments. + By default, the CDK CLI will perform garbage collection actions on all environments that you reference in your CDK app or provide as arguments. If you don’t provide an environment, the CDK CLI will determine the environment from default sources. These sources include environments that you specify using the --profile option, environment variables, or default {aws} CLI sources.

[#ref-cli-cmd-gc-options] == Options For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-gc-options-action] --action <STRING>:: The action that cdk gc performs on your assets during a garbage collection. + –

* delete-tagged – Deletes assets that have been tagged with a date within the range of buffer days that you provide, but does not tag newly identified unused assets.
* full – Perform all garbage collection actions. This includes deleting assets within the range of buffer days that you provide and tagging newly identified unused assets.
* print – Outputs the number of unused assets at the command prompt but doesn’t make any actual changes within your {aws} environment.
* {blank}
* == tag – Tags any newly identified unused assets, but doesn’t delete any assets within the range of buffer days that you provide.
* *Accepted values*: delete-tagged, full, print, tag
* *Default value*: full

[#ref-cli-cmd-gc-options-bootstrap-stack-name] --bootstrap-stack-name <STRING>:: The name of the CDK bootstrap stack in your {aws} environment. Provide this option if you customized your bootstrap stack name. If you are using the default CDKToolkit stack name, you don’t have to provide this option. + *Default value*: CDKToolkit

[#ref-cli-cmd-gc-options-confirm] --confirm <BOOLEAN>:: Specify whether the CDK CLI will request manual confirmation from you before deleting any assets. + Specify false to automatically delete assets without prompting you for manual confirmation. + *Default value*: true

[#ref-cli-cmd-gc-options-created-buffer-days] --created-buffer-days <NUMBER>:: The number of days an asset must exist before it is eligible for garbage collection actions. + When you provide a number, assets that have not existed beyond your specified number of days are filtered out from garbage collection actions. + *Default value*: 1

[#ref-cli-cmd-gc-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk gc command.

[#ref-cli-cmd-gc-options-rollback-buffer-days] --rollback-buffer-days <NUMBER>:: The number of days an asset must be tagged as isolated before it is eligible for garbage collection actions. + When you provide a number, the CDK CLI will tag unused assets with the current date instead of deleting them. The CDK CLI will also check if any assets have been tagged during previous runs of the cdk gc command. Previously tagged assets that fall within the range of buffer days you provide will be deleted. + *Default value*: 0

[#ref-cli-cmd-gc-options-type] --type <STRING>:: The bootstrap resource type within your bootstrap stack to perform garbage collection actions on. + –

* all – Perform garbage collection actions on all bootstrapped resources.
* ecr – Perform garbage collection actions on assets in the Amazon Elastic Container Registry (Amazon ECR) repository of your bootstrap stack.
* {blank}
* == s3 – Perform garbage collection actions on assets in the Amazon S3 bucket of your bootstrap stack.
* *Accepted values*: all, ecr, s3
* *Default value*: all

[#ref-cli-cmd-gc-options-unstable] --unstable <STRING>:: Allow the usage of CDK CLI commands that are still in development. + This option is required to use any CDK CLI command that is still in development and subject to change. + This option can be provided multiple times in a single command. + To use cdk gc, provide --unstable=gc.

[#ref-cli-cmd-gc-examples] == Examples

[#ref-cli-cmd-gc-examples-basic] === Basic examples

The following example prompts you for manual confirmation to perform default garbage collection actions on assets in the Amazon S3 bucket of your bootstrap stack:

== [source,none,subs=“verbatim,attributes”]

$ cdk gc –unstable=gc –type=s3

⏳ Garbage Collecting environment aws://++++++/++++++... Found 99 assets to delete based off of the following criteria:++++++++++++

* assets have been isolated for > 0 days
* assets were created > 1 days ago

== Delete this batch (yes/no/delete-all)?

The following example performs garbage collection actions on a range of assets in the Amazon S3 bucket of your bootstrap stack. This range includes assets that have been previously tagged by cdk gc for over 30 days and have been created 10 days or older. This command will prompt for manual confirmation before deleting any assets:

== [source,none,subs=“verbatim,attributes”]

## $ cdk gc –unstable=gc –type=s3 –rollback-buffer-days=30 –created-buffer-days=10

The following example performs the action of deleting previously tagged assets in the Amazon S3 bucket of your bootstrap stack that have been unused for longer than 30 days:

== [source,none,subs=“verbatim,attributes”]

## $ cdk gc –unstable=gc –type=s3 –action=delete-tagged –rollback-buffer-days=30

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cmd-import] = cdk import :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation, cdk import

== [abstract]

## Use https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/resource-import.html[{aws} CloudFormation resource imports] to import existing {aws} resources into a CDK stack.

// Content start

Use https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/resource-import.html[{aws} CloudFormation resource imports] to import existing {aws} resources into a CDK stack.

With this command, you can take existing resources that were created using other methods and start managing them using the {aws} CDK.

When considering moving resources into CDK management, sometimes creating new resources is acceptable, such as with IAM roles, Lambda functions, and event rules. For other resources, such as stateful resources like Amazon S3 buckets and DynamoDB tables, creating new resources can cause impacts to your service. You can use cdk import to import existing resources with minimal disruption to your services. For a list of supported {aws} resources, see https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/resource-import-supported-resources.html[Resource type support] in the *{aws} CloudFormation User Guide*.

*To import an existing resource to a CDK stack*:: + . Run a cdk diff to make sure your CDK stack has no pending changes. When performing a cdk import, the only changes allowed in an import operation are the addition of new resources being imported. + . Add constructs for the resources you want to import to your stack. For example, add the following for an Amazon S3 bucket: + [source,javascript,subs=“verbatim,attributes”] — new s3.Bucket(this, ‘ImportedS3Bucket’, {}); — + Do not add any other changes. You must also make sure to exactly model the state that the resource currently has. For the bucket example, be sure to include {aws} KMS keys, lifecycle policies, and anything else that is relevant about the bucket. Otherwise, subsequent update operations may not do what you expect. + . Run cdk import. If there are multiple stacks in the CDK app, pass a specific stack name as an argument. + . The CDK CLI will prompt you to pass in the actual names of the resources you are importing. After you provide this information, import will begin. + . When cdk import reports success, the resource will be managed by the CDK. Any subsequent changes in the construct configuration will be reflected on the resource.

This feature currently has the following limitations:

* Importing resources into nested stacks isn’t possible.
* There is no check on whether the properties you specify are correct and complete for the imported resource. Try starting a drift detection operation after importing.
* Resources that depend on other resources must all be imported together, or individually, in the right order. Otherwise, the CloudFormation deployment will fail with unresolved references.
* This command uses the deploy role credentials, which is necessary to read the encrypted staging bucket. This requires version 12 of the bootstrap template, which includes the necessary IAM permissions for the deploy role.

[#ref-cli-cmd-import-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk import ++++++++++++—-++++++++++++

[#ref-cli-cmd-import-args] == Arguments

[#ref-cli-cmd-import-args-stack-name] *CDK stack ID*:: The construct ID of the CDK stack from your app to import resources to. This argument can be provided multiple times in a single command. + *Type*: String + *Required*: No

[#ref-cli-cmd-import-options] == Options

For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-import-options-change-set-name] --change-set-name <STRING>:: The name of the CloudFormation change set to create.

[#ref-cli-cmd-import-options-execute] --execute <BOOLEAN>:: Specify whether to execute change set. + *Default value*: true

[#ref-cli-cmd-import-options-force] --force, -f <BOOLEAN>:: By default, the CDK CLI exits the process if the template diff includes updates or deletions. Specify true to override this behavior and always continue with importing.

[#ref-cli-cmd-import-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk import command.

[#ref-cli-cmd-import-options-record-resource-mapping] --record-resource-mapping, -r <STRING>:: Use this option to generate a mapping of existing physical resources to the CDK resources that will be imported. The mapping will be written to the file path that you provide. No actual import operations will be performed.

[#ref-cli-cmd-import-options-resource-mapping] --resource-mapping, -m <STRING>:: Use this option to specify a file that defines your resource mapping. The CDK CLI will use this file to map physical resources to resources for import instead of interactively asking you. + This option can be run from scripts.

[#ref-cli-cmd-import-options-rollback] --rollback <BOOLEAN>:: Roll back the stack to stable state on failure. + To specify false, you can use --no-rollback or -R. + Specify false to iterate more rapidly. Deployments containing resource replacements will always fail. + *Default value*: true

[#ref-cli-cmd-import-options-toolkit-stack-name] --toolkit-stack-name <STRING>:: The name of the CDK Toolkit stack to create. + By default, cdk bootstrap deploys a stack named CDKToolkit into the specified {aws} environment. Use this option to provide a different name for your bootstrap stack. + The CDK CLI uses this value to verify your bootstrap stack version.

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cmd-init] = cdk init :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation, cdk init

== [abstract]

## Create a new {aws} CDK project from a template.

// Content start

Create a new {aws} CDK project from a template.

[#ref-cli-cmd-init-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk init ++++++++++++—-++++++++++++

[#ref-cli-cmd-init-args] == Arguments

[#ref-cli-cmd-init-args-template-type] *Template type*:: The CDK template type to initialize a new CDK project from. + –

* app – Template for a CDK application.
* lib – Template for an {aws} Construct Library.
* {blank}
* == sample-app – Example CDK application that includes some constructs.
* *Valid values*: app, lib, sample-app

[#ref-cli-cmd-init-options] == Options

For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-init-options-generate-only] --generate-only <BOOLEAN>:: Specify this option to generate project files without initiating additional operations such as setting up a git repository, installing dependencies, or compiling the project. + *Default value*: false

[#ref-cli-cmd-init-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk init command.

[#ref-cli-cmd-init-options-language] --language, -l <STRING>:: The language to be used for the new project. This option can be configured in the project’s cdk.json configuration file or at ~/.cdk.json on your local development machine. + *Valid values*: csharp, fsharp, go, java, javascript, python, typescript

[#ref-cli-cmd-init-options-list] --list <BOOLEAN>:: List the available template types and languages.

[#ref-cli-cmd-init-examples] == Examples

[#ref-cli-cmd-init-examples-1] === List the available template types and languages

== [source,none,subs=“verbatim,attributes”]

$ cdk init –list Available templates:

* app: Template for a CDK Application └─ cdk init app –language=[csharp|fsharp|go|java|javascript|python|typescript]
* lib: Template for a CDK Construct Library └─ cdk init lib –language=typescript
* sample-app: Example CDK Application with some constructs └─ cdk init sample-app –language=[csharp|fsharp|go|java|javascript|python|typescript] —

[#ref-cli-cmd-init-examples-2] === Create a new CDK app in TypeScript from the library template

== [source,none,subs=“verbatim,attributes”]

## $ cdk init lib –language=typescript

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cmd-list] = cdk list :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation, cdk list

== [abstract]

## List all {aws} CDK stacks and their dependencies from a CDK app.

// Content start

List all {aws} CDK stacks and their dependencies from a CDK app.

[#ref-cli-cmd-list-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk list ++++++++++++—-++++++++++++

[#ref-cli-cmd-list-args] == Arguments

[#ref-cli-cmd-list-args-stack-name] *CDK stack ID*:: The construct ID of the CDK stack from your app to perform this command against. + *Type*: String + *Required*: No

[#ref-cli-cmd-list-options] == Options

For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-list-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk list command.

[#ref-cli-cmd-list-options-long] --long, -l <BOOLEAN>:: Display {aws} environment information for each stack. + *Default value*: false

[#ref-cli-cmd-list-options-show-dependencies] --show-dependencies, -d <BOOLEAN>:: Display stack dependency information for each stack. + *Default value*: false

[#ref-cli-cmd-list-examples] == Examples

[#ref-cli-cmd-list-examples-1] === List all stacks in the CDK app ‘[.code]node bin/main.js’

== [source,none,subs=“verbatim,attributes”]

$ cdk list –app=‘node bin/main.js’ Foo Bar Baz —

[#ref-cli-cmd-list-examples-] === List all stacks, including {aws} environment details for each stack

== [source,none,subs=“verbatim,attributes”]

## $ cdk list –app=‘node bin/main.js’ –long

name: Foo  
environment:  
 name: 000000000000/bermuda-triangle-1  
 account: '000000000000'  
 region: bermuda-triangle-1

* name: Bar environment: name: 111111111111/bermuda-triangle-2 account: ‘111111111111’ region: bermuda-triangle-2
* name: Baz environment: name: 333333333333/bermuda-triangle-3 account: ‘333333333333’ region: bermuda-triangle-3 —

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cmd-metadata] = cdk metadata :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation, cdk metadata

== [abstract]

## Display metadata associated with a CDK stack.

// Content start

Display metadata associated with a CDK stack.

[#ref-cli-cmd-metadata-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk metadata ++++++++++++—-++++++++++++

[#ref-cli-cmd-metadata-args] == Arguments

[#ref-cli-cmd-metadata-args-stack-name] *CDK stack ID*:: The construct ID of the CDK stack from your app to display metadata for. + *Type*: String + *Required*: No

[#ref-cli-cmd-metadata-options] == Options

For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-metadata-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk metadata command.

:doctype: book

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cdk-migrate] = cdk migrate :keywords: {aws} CDK, {aws} CDK CLI, cdk migrate

== [abstract]

## Migrate deployed {aws} resources, {aws} CloudFormation stacks, and CloudFormation templates into a new {aws} CDK project.

// Content start

Migrate deployed {aws} resources, {aws} CloudFormation stacks, and CloudFormation templates into a new {aws} CDK project.

This command creates a new CDK app that includes a single stack that is named with the value you provide using --stack-name. You can configure the migration source using --from-scan, --from-stack, or --from-path.

For more information on using cdk migrate, see xref:migrate[Migrate existing resources and {aws} CloudFormation templates to the {aws} CDK].

= [NOTE]

The cdk migrate command is experimental and may have breaking changes in the future.

====

[#ref-cli-cdk-migrate-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk migrate ++++++—-++++++

[#ref-cli-cdk-migrate-options] == Options

For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cdk-migrate-options-required] === Required options

[#ref-cli-cdk-migrate-options-stack-name] --stack-name <STRING>:: The name of the {aws} CloudFormation stack that will be created within the CDK app after migrating. + *Required*: Yes

[#ref-cli-cdk-migrate-options-conditional] === Conditional options

[#ref-cli-cdk-migrate-options-from-path] --from-path <PATH>:: The path to the {aws} CloudFormation template to migrate. Provide this option to specify a local template. + *Required*: Conditional. Required if migrating from a local {aws} CloudFormation template.

[#ref-cli-cdk-migrate-options-from-scan] --from-scan <STRING>:: When migrating deployed resources from an {aws} environment, use this option to specify whether a new scan should be started or if the {aws} CDK CLI should use the last successful scan. + *Required*: Conditional. Required when migrating from deployed {aws} resources. + *Accepted values*: most-recent, new

[#ref-cli-cdk-migrate-options-from-stack] --from-stack <BOOLEAN>:: Provide this option to migrate from a deployed {aws} CloudFormation stack. Use --stack-name to specify the name of the deployed {aws} CloudFormation stack. + *Required*: Conditional. Required if migrating from a deployed {aws} CloudFormation stack.

[#ref-cli-cdk-migrate-options-optional] === Optional options

[#ref-cli-cdk-migrate-options-account] --account <STRING>:: The account to retrieve the {aws} CloudFormation stack template from. + *Required*: No + *Default*: The {aws} CDK CLI obtains account information from default sources.

[#ref-cli-cdk-migrate-options-compress] --compress <BOOLEAN>:: Provide this option to compress the generated CDK project into a ZIP file. + *Required*: No

[#ref-cli-cdk-migrate-options-filter] --filter <ARRAY>:: Use when migrating deployed resources from an {aws} account and {aws} Region. This option specifies a filter to determine which deployed resources to migrate. + This option accepts an array of key-value pairs, where *key* represents the filter type and *value* represents the value to filter. + The following are accepted keys: + –

* resource-identifier – An identifier for the resource. Value can be the resource logical or physical ID. For example, resource-identifier="ClusterName".
* resource-type-prefix – The {aws} CloudFormation resource type prefix. For example, specify +resource-type-prefix="{aws}::DynamoDB::"+ to filter all Amazon DynamoDB resources.
* tag-key – The key of a resource tag. For example, tag-key="myTagKey".
* {blank}
* == tag-value – The value of a resource tag. For example, tag-value="myTagValue".
* Provide multiple key-value pairs for AND conditional logic. The following example filters for any DynamoDB resource that is tagged with myTagKey as the tag key: +--filter resource-type-prefix="{aws}::DynamoDB::", tag-key="myTagKey"+.
* Provide the --filter option multiple times in a single command for OR conditional logic. The following example filters for any resource that is a DynamoDB resource or is tagged with myTagKey as the tag key: +--filter resource-type-prefix="{aws}::DynamoDB::" --filter tag-key="myTagKey"+.
* *Required*: No

[#ref-cli-cdk-migrate-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk migrate command.

[#ref-cli-cdk-migrate-options-language] --language <STRING>:: The programming language to use for the CDK project created during migration. + *Required*: No + *Valid values*: typescript, python, java, csharp, go. + *Default*: typescript

[#ref-cli-cdk-migrate-options-output-path] --output-path <PATH>:: The output path for the migrated CDK project. + *Required*: No + *Default*: By default, the {aws} CDK CLI will use your current working directory.

[#ref-cli-cdk-migrate-options-region] --region <STRING>:: The {aws} Region to retrieve the {aws} CloudFormation stack template from. + *Required*: No + *Default*: The {aws} CDK CLI obtains {aws} Region information from default sources.

[#ref-cli-cdk-migrate-examples] == Examples

[#ref-cli-cdk-migrate-examples-1] === Simple example of migrating from a CloudFormation stack

Migrate from a deployed CloudFormation stack in a specific {aws} environment using --from-stack. Provide --stack-name to name your new CDK stack. The following is an example that migrates myCloudFormationStack to a new CDK app that is using TypeScript:

== [source,none,subs=“verbatim,attributes”]

## $ cdk migrate –language typescript –from-stack –stack-name ‘myCloudFormationStack’

[#ref-cli-cdk-migrate-examples-2] === Simple example of migrating from a local CloudFormation template

Migrate from a local JSON or YAML CloudFormation template using --from-path. Provide --stack-name to name your new CDK stack. The following is an example that creates a new CDK app in TypeScript that includes a myCloudFormationStack stack from a local template.json file:

== [source,none,subs=“verbatim,attributes”]

## $ cdk migrate –stack-name “myCloudFormationStack” –language typescript –from-path “./template.json”

[#ref-cli-cdk-migrate-examples-3] === Simple example of migrating from deployed {aws} resources

Migrate deployed {aws} resources from a specific {aws} environment that are not associated with a CloudFormation stack using --from-scan. The CDK CLI utilizes the [.noloc]IaC generator service to scan for resources and generate a template. Then, the CDK CLI references the template to create the new CDK app. The following is an example that creates a new CDK app in TypeScript with a new myCloudFormationStack stack containing migrated {aws} resources:

== [source,none,subs=“verbatim,attributes”]

## $ cdk migrate –language typescript –from-scan –stack-name “myCloudFormationStack”

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cmd-notices] = cdk notices :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation, cdk notices

== [abstract]

## Display notices for your CDK application.

// Content start

Display notices for your CDK application.

Notices can include important messages regarding security vulnerabilities, regressions, and usage of unsupported versions.

This command displays relevant notices, regardless of whether they have been acknowledged or not. Relevant notices may also appear after every command by default.

You can suppress notices in the following ways:

* Through command options. The following is an example:

## [source,none,subs=“verbatim,attributes”]

## $ cdk deploy –no-notices

* Suppress all notices indefinitely through context in the project’s cdk.json file:

## [source,json,subs=“verbatim,attributes”]

* { “notices”: false, “context”: { // … } } —
* Acknowledge each notice with the cdk acknowledge command.

[#ref-cli-cmd-notices-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk notices ++++++—-++++++

[#ref-cli-cmd-notices-options] == Options

For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-notices-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk notices command.

[#ref-cli-cmd-notices-examples] == Examples

[#ref-cli-cmd-notices-examples-1] === Example of a default notice that displays after running the cdk deploy command

== [source,none,subs=“verbatim,attributes”]

$ cdk deploy

… # Normal output of the command

NOTICES

16603 Toggling off auto\_delete\_objects for Bucket empties the bucket

…. Overview: If a stack is deployed with an S3 bucket with auto\_delete\_objects=True, and then re-deployed with auto\_delete\_objects=False, all the objects in the bucket will be deleted.

Affected versions: <1.126.0.  
  
More information at: https://github.com/aws/aws-cdk/issues/16603

….

17061 Error when building EKS cluster with monocdk import

…. Overview: When using monocdk/aws-eks to build a stack containing an EKS cluster, error is thrown about missing lambda-layer-node-proxy-agent/layer/package.json.

Affected versions: >=1.126.0 <=1.130.0.  
  
More information at: https://github.com/aws/aws-cdk/issues/17061

….

== If you don’t want to see an notice anymore, use “cdk acknowledge ID”. For example, “cdk acknowledge 16603”

[#ref-cli-cmd-notices-examples-2] === Simple example of running the cdk notices command

== [source,none,subs=“verbatim,attributes”]

$ cdk notices

NOTICES

16603 Toggling off auto\_delete\_objects for Bucket empties the bucket

…. Overview: if a stack is deployed with an S3 bucket with auto\_delete\_objects=True, and then re-deployed with auto\_delete\_objects=False, all the objects in the bucket will be deleted.

Affected versions: framework: <=2.15.0 >=2.10.0  
  
More information at: https://github.com/aws/aws-cdk/issues/16603

….

== If you don’t want to see a notice anymore, use “cdk acknowledge ++++++”. For example, “cdk acknowledge 16603”++++++

:doctype: book

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cmd-rollback] = cdk rollback :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation, cdk deploy, rollback

== [abstract]

## Use the {aws} Cloud Development Kit ({aws} CDK) Command Line Interface (CLI) cdk rollback command to rollback a failed or paused stack from an {aws} CloudFormation deployment to its last stable state.

// Content start

Use the {aws} Cloud Development Kit ({aws} CDK) Command Line Interface (CLI) cdk rollback command to rollback a failed or paused stack from an {aws} CloudFormation deployment to its last stable state.

= [NOTE]

To use this command, you must have v23 of the bootstrap template deployed to your environment. For more information, see xref:bootstrap-template-history[Bootstrap template version history].

====

When you deploy using cdk deploy, the CDK CLI will rollback a failed deployment by default. If you specify --no-rollback with cdk deploy, you can then use the cdk rollback command to manually rollback a failed deployment. This will initiate a rollback to the last stable state of your stack.

[#ref-cli-cmd-rollback-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk rollback ++++++++++++—-++++++++++++

[#ref-cli-cmd-rollback-args] == Arguments

[#ref-cli-cmd-rollback-args-stack-name] *CDK stack ID*:: The construct ID of the CDK stack from your app to rollback. + *Type*: String + *Required*: No

[#ref-cli-cmd-rollback-options] == Options

For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-rollback-options-all] --all <BOOLEAN>:: Rollback all stacks in your CDK app. + *Default value*: false

[#ref-cli-cmd-rollback-options-force] --force, -f <BOOLEAN>:: When you use cdk rollback, some resources may fail to rollback. Provide this option to force the rollback of all resources. This is the same behavior as providing the --orphan option for each resource in your stack. + *Default value*: false

[#ref-cli-cmd-rollback-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk rollback command.

[#ref-cli-cmd-rollback-options-orphan] --orphan <LogicalId>:: When you use cdk rollback, some resources may fail to rollback. When this happens, you can try to force the rollback of a resource by using this option and providing the logical ID of the resource that failed to rollback. + This option can be provided multiple times in a single command The following is an example: + [source,none,subs=“verbatim,attributes”] — $ cdk rollback MyStack –orphan MyLambdaFunction –orphan MyLambdaFunction2 — + To force the rollback of all resources, use the --force option instead.

[#ref-cli-cmd-rollback-options-toolkit-stack-name] --toolkit-stack-name <STRING>:: The name of the existing CDK Toolkit stack that the environment is bootstrapped with. + By default, cdk bootstrap deploys a stack named CDKToolkit into the specified {aws} environment. Use this option to provide a different name for your bootstrap stack. + The CDK CLI uses this value to verify your bootstrap stack version.

[#ref-cli-cmd-rollback-options-validate-bootstrap-version] --validate-bootstrap-version <BOOLEAN>:: Specify whether to validate the bootstrap stack version. Provide --validate-bootstrap-version=false or --no-validate-bootsrap-version to turn off this behavior. + *Default value*: true

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cmd-synth] = cdk synthesize :info\_titleabbrev: cdk synth :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation, cdk synthesize, cdk synth

== [abstract]

## Synthesize a CDK app to produce a cloud assembly, including an {aws} CloudFormation template for each stack.

// Content start

Synthesize a CDK app to produce a cloud assembly, including an {aws} CloudFormation template for each stack.

Cloud assemblies are files that include everything needed to deploy your app to your {aws} environment. For example, it includes a CloudFormation template for each stack in your app, and a copy of the file assets or Docker images that you reference in your app.

If your app contains a single stack or if a single stack is provided as an argument, the CloudFormation template will also be displayed in the standard output (stdout) in YAML format.

If your app contains multiple stacks, cdk synth will synthesize the cloud assembly to cdk.out.

[#ref-cli-cmd-synth-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk synthesize ++++++++++++—-++++++++++++

[#ref-cli-cmd-synth-args] == Arguments

[#ref-cli-cmd-synth-args-stack-name] *CDK stack ID*:: The construct ID of the CDK stack from your app to synthesize. + *Type*: String + *Required*: No

[#ref-cli-cmd-synth-options] == Options

For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-synth-options-exclusively] --exclusively, -e <BOOLEAN>:: Only synthesize requested stacks, don’t include dependencies.

[#ref-cli-cmd-synth-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk synthesize command.

[#ref-cli-cmd-synth-options-quiet] --quiet, -q <BOOLEAN>:: Do not output the CloudFormation template to stdout. + This option can be configured in the CDK project’s cdk.json file. The following is an example: + [source,json,subs=“verbatim,attributes”] — { “quiet”: true } — + *Default value*: false

[#ref-cli-cmd-synth-options-validation] --validation <BOOLEAN>:: Validate the generated CloudFormation templates after synthesis by performing additional checks. + You can also configure this option through the validateOnSynth attribute or CDK\_VALIDATION environment variable. + *Default value*: true

[#ref-cli-cmd-synth-examples] == Examples

[#ref-cli-cmd-synth-examples-1] === Synthesize the cloud assembly for a CDK stack with logial ID MyStackName and output the CloudFormation template to stdout

== [source,none,subs=“verbatim,attributes”]

## $ cdk synth MyStackName

[#ref-cli-cmd-synth-examples-2] === Synthesize the cloud assembly for all stacks in a CDK app and save them into cdk.out

== [source,none,subs=“verbatim,attributes”]

## $ cdk synth

[#ref-cli-cmd-synth-examples-3] === Synthesize the cloud assembly for MyStackName, but don’t include dependencies

== [source,none,subs=“verbatim,attributes”]

## $ cdk synth MyStackName –exclusively

[#ref-cli-cmd-synth-examples-4] === Synthesize the cloud assembly for MyStackName, but don’t output the CloudFormation template to stdout

== [source,none,subs=“verbatim,attributes”]

## $ cdk synth MyStackName –quiet

:doctype: book

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cmd-watch] = cdk watch :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation, cdk watch

== [abstract]

## Continuously watch a local {aws} CDK project for changes to perform deployments and hotswaps.

// Content start

Continuously watch a local {aws} CDK project for changes to perform deployments and hotswaps.

This command is similar to cdk deploy, except that it can perform continuous deployments and hotswaps through a single command.

This command is a shortcut for cdk deploy --watch.

To end a cdk watch session, interrupt the process by pressing Ctrl+C.

The files that are observed is determined by the "watch" setting in your cdk.json file. It has two sub-keys, "include" and "exclude", that accepts a single string or an array of strings. Each entry is interpreted as a path relative to the location of the cdk.json file. Both \\* and +\*\*+ are accepted.

If you create a project using the cdk init command, the following default behavior is configured for cdk watch in your project’s cdk.json file:

* "include" is set to +"\\*\*/\*"+, which includes all files and directories in the root of the project.
* "exclude" is optional, except for files and folders already ignored by default. This consists of files and directories starting with ., the CDK output directory, and the node\_modules directory.

The minimal setting to configure watch is "watch": {}.

If either your CDK code or application code requires a build step before deployment, cdk watch works with the "build" key in the cdk.json file.

= [NOTE]

This command is considered experimental and may have breaking changes in the future.

====

The same limitations of cdk deploy --hotswap applies to cdk watch. For more information, see xref:ref-cli-cmd-deploy-options-hotswap[cdk deploy --hotswap].

[#ref-cli-cmd-watch-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk watch ++++++++++++—-++++++++++++

[#ref-cli-cmd-watch-args] == Arguments

[#ref-cli-cmd-watch-args-stack-name] *CDK stack ID*:: The construct ID of the CDK stack from your app to watch. + *Type*: String + *Required*: No

[#ref-cli-cmd-watch-options] == Options For a list of global options that work with all CDK CLI commands, see xref:ref-cli-cmd-options[Global options].

[#ref-cli-cmd-watch-options-build-exclude] --build-exclude, -E <ARRAY>:: Do not rebuild asset with the given ID. + This option can be specified multiple times in a single command. + *Default value*: []

[#ref-cli-cmd-watch-options-change-set-name] --change-set-name <STRING>:: The name of the CloudFormation change set to create.

[#ref-cli-cmd-watch-options-concurrency] --concurrency <NUMBER>:: Deploy and hotswap multiple stacks in parallel while accounting for inter-stack dependencies. Use this option to speed up deployments. You must still factor in CloudFormation and other {aws} account rate limiting. + Provide a number to specify the maximum number of simultaneous deployments (dependency permitting) to perform. + *Default value*: 1

[#ref-cli-cmd-watch-options-exclusively] --exclusively, -e <BOOLEAN>:: Only deploy requested stacks and don’t include dependencies.

[#ref-cli-cmd-watch-options-force] --force, -f <BOOLEAN>:: Always deploy stacks, even if templates are identical. + *Default value*: false

[#ref-cli-cmd-watch-options-help] --help, -h <BOOLEAN>:: Show command reference information for the cdk watch command.

[#ref-cli-cmd-watch-options-hotswap] --hotswap <BOOLEAN>:: By default, cdk watch uses hotswap deployments when possible to update your resources. The CDK CLI will attempt to perform a hotswap deployment and will not fall back to a full CloudFormation deployment if unsuccessful. Any changes detected that cannot be updated through a hotswap are ignored. + *Default value*: true

[#ref-cli-cmd-watch-options-hotswap-fallback] --hotswap-fallback <BOOLEAN>:: By default, cdk watch attempts to perform hotswap deployments and ignores changes that require CloudFormation deployments. Provide --hotswap-fallback to fall back and perform a full CloudFormation deployment if the hotswap deployment is unsuccessful.

[#ref-cli-cmd-watch-options-logs] --logs <BOOLEAN>:: By default, cdk watch monitors all CloudWatch log groups in your application and streams the log events locally to stdout. + *Default value*: true

[#ref-cli-cmd-watch-options-progress] --progress <STRING>:: Configure how the CDK CLI displays deployment progress. + –

* bar – Display stack deployment events as a progress bar, with the events for the resource currently being deployed.
* {blank}
* == events – Provide a complete history, including all CloudFormation events.
* You can also configure this option in the project’s cdk.json file or at ~/.cdk.json on your local development machine:

## [source,json,subs=“verbatim,attributes”]

* { “progress”: “events” } —
* *Valid values*: bar, events
* *Default value*: bar

[#ref-cli-cmd-watch-options-rollback] --rollback <BOOLEAN>:: During deployment, if a resource fails to be created or updated, the deployment will roll back to the latest stable state before the CDK CLI returns. All changes made up to that point will be undone. Resources that were created will be deleted and updates that were made will be rolled back. + Use --no-rollback or -R to deactivate this behavior. If a resource fails to be created or updated, the CDK CLI will leave changes made up to that point in place and return. This may be helpful in development environments where you are iterating quickly. + [NOTE] ==== When false, deployments that cause resource replacements will always fail. You can only use this value for deployments that update or create new resources. ==== + *Default value*: true

[#ref-cli-cmd-watch-options-toolkit-stack-name] --toolkit-stack-name <STRING>:: The name of the existing CDK Toolkit stack. + By default, cdk bootstrap deploys a stack named CDKToolkit into the specified {aws} environment. Use this option to provide a different name for your bootstrap stack. + The CDK CLI uses this value to verify your bootstrap stack version.

[#ref-cli-cmd-watch-examples] == Examples

[#ref-cli-cmd-watch-examples-1] === Watch a CDK stack with logical ID DevelopmentStack for changes

== [source,none,subs=“verbatim,attributes”]

$ cdk watch DevelopmentStack Detected change to ‘lambda-code/index.js’ (type: change). Triggering ‘cdk deploy’ DevelopmentStack: deploying…

== ✅ DevelopmentStack

[#ref-cli-cmd-watch-examples-2] === Configure a cdk.json file for what to include and exclude from being watched for changes

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

{ “app”: “mvn -e -q compile exec:java”, “watch”: { “include”: “src/main/\_\*“,”exclude”: “target/\_” } } —

[#ref-cli-cmd-watch-examples-3] === Build a CDK project using Java before deployment by configuring the cdk.json file

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

{ “app”: “mvn -e -q exec:java”, “build”: “mvn package”, “watch”: { “include”: “src/main/\_\*“,”exclude”: “target/\_” } } —

include::attributes.txt[]

// Attributes

[.topic] [#ref-cli-cmd] = {aws} CDK CLI command reference :keywords: {aws} CDK, {aws} CDK CLI, CDK Toolkit, IaC, Infrastructure as code, {aws} Cloud, {aws} CloudFormation

== [abstract]

## This section contains command reference information for the {aws} Cloud Development Kit ({aws} CDK) Command Line Interface (CLI). The CDK CLI is also referred to as the CDK Toolkit.

// Content start

This section contains command reference information for the {aws} Cloud Development Kit ({aws} CDK) Command Line Interface (CLI). The CDK CLI is also referred to as the CDK Toolkit.

[#ref-cli-cmd-usage] == Usage

== [source,none,subs=“verbatim,attributes”]

$ cdk ++++++++++++ ++++++++++++—-++++++++++++

[#ref-cli-cmd-commands] == Commands

[#ref-cli-cmd-commands-acknowledge] xref:ref-cli-cmd-ack[acknowledge ack]:: Acknowledge a notice by issue number and hide it from displaying again.

[#ref-cli-cmd-commands-bootstrap] xref:ref-cli-cmd-bootstrap[bootstrap]:: Prepare an {aws} environment for CDK deployments by deploying the CDK bootstrap stack, named CDKToolkit, into the {aws} environment.

[#ref-cli-cmd-commands-context] xref:ref-cli-cmd-context[context]:: Manage cached context values for your CDK application.

[#ref-cli-cmd-commands-deploy] xref:ref-cli-cmd-deploy[deploy]:: Deploy one or more CDK stacks into your {aws} environment.

[#ref-cli-cmd-commands-destroy] xref:ref-cli-cmd-destroy[destroy]:: Delete one or more CDK stacks from your {aws} environment.

[#ref-cli-cmd-commands-diff] xref:ref-cli-cmd-diff[diff]:: Perform a diff to see infrastructure changes between CDK stacks.

[#ref-cli-cmd-commands-docs] xref:ref-cli-cmd-docs[docs doc]:: Open CDK documentation in your browser.

[#ref-cli-cmd-commands-doctor] xref:ref-cli-cmd-doctor[doctor]:: Inspect and display useful information about your local CDK project and development environment.

[#ref-cli-cmd-commands-drift] xref:ref-cli-cmd-drift[drift]:: Detect configuration drift for resources that you define, manage, and deploy using CDK.

[#ref-cli-cmd-commands-import] xref:ref-cli-cmd-import[import]:: Use {aws} CloudFormation resource imports to import existing {aws} resources into a CDK stack.

[#ref-cli-cmd-commands-init] xref:ref-cli-cmd-init[init]:: Create a new CDK project from a template.

[#ref-cli-cmd-commands-list] xref:ref-cli-cmd-list[list, ls]:: List all CDK stacks and their dependencies from a CDK app.

[#ref-cli-cmd-commands-metadata] xref:ref-cli-cmd-metadata[metadata]:: Display metadata associated with a CDK stack.

[#ref-cli-cmd-commands-migrate] xref:ref-cli-cdk-migrate[migrate]:: Migrate {aws} resources, {aws} CloudFormation stacks, and {aws} CloudFormation templates into a new CDK project.

[#ref-cli-cmd-commands-notices] xref:ref-cli-cmd-notices[notices]:: Display notices for your CDK application.

[#ref-cli-cmd-commands-synthesize] xref:ref-cli-cmd-synth[synthesize, synth]:: Synthesize a CDK app to produce a cloud assembly, including an {aws} CloudFormation template for each stack.

[#ref-cli-cmd-commands-watch] xref:ref-cli-cmd-watch[watch]:: Continuously watch a local CDK project for changes to perform deployments and hotswaps.

[#ref-cli-cmd-options] == Global options

The following options are compatible with all CDK CLI commands.

[#ref-cli-cmd-options-app] --app, -a <STRING>:: Provide the command for running your app or cloud assembly directory. + *Required*: Yes

[#ref-cli-cmd-options-asset-metadata] --asset-metadata <BOOLEAN>:: Include aws:asset:\* {aws} CloudFormation metadata for resources that use assets. + *Required*: No + *Default value*: true

[#ref-cli-cmd-options-build] --build <STRING>:: Command for running a pre-synthesis build. + *Required*: No

[#ref-cli-cmd-options-ca-bundle-path] --ca-bundle-path <STRING>:: Path to a CA certificate to use when validating HTTPS requests. + If this option is not provided, the CDK CLI will read from the AWS\_CA\_BUNDLE environment variable. + *Required*: Yes

[#ref-cli-cmd-options-ci] --ci <BOOLEAN>:: Indicate that CDK CLI commands are being run in a continuous integration (CI) environment. + This option modifies the behavior of the CDK CLI to better suit automated operations that are typical in CI pipelines. + When you provide this option, logs are sent to stdout instead of stderr. + *Required*: No + *Default value*: false

[#ref-cli-cmd-options-context] --context, -c <ARRAY>:: Add contextual string parameters as key-value pairs.

[#ref-cli-cmd-options-debug] --debug <BOOLEAN>:: Enable detailed debugging information. This option produces a verbose output that includes a lot more detail about what the CDK CLI is doing behind the scenes. + *Required*: No + *Default value*: false

[#ref-cli-cmd-options-ec2creds] --ec2creds, -i <BOOLEAN>:: Force the CDK CLI to try and fetch Amazon EC2 instance credentials. + By default, the CDK CLI guesses the Amazon EC2 instance status. + *Required*: No + *Default value*: false

[#ref-cli-cmd-options-help] --help, -h <BOOLEAN>:: Show command reference information for the CDK CLI. + *Required*: No + *Default value*: false

[#ref-cli-cmd-options-ignore-errors] --ignore-errors <BOOLEAN>:: Ignore synthesis errors, which will likely produce an output that is not valid. + *Required*: No + *Default value*: false

[#ref-cli-cmd-options-json] --json, -j <BOOLEAN>:: Use JSON instead of YAML for {aws} CloudFormation templates that are printed to standard output (stdout). + *Required*: No + *Default value*: false

[#ref-cli-cmd-options-lookups] --lookups <BOOLEAN>:: Perform context lookups. + Synthesis will fail if this value is false and context lookups need to be performed. + *Required*: No + *Default value*: true

[#ref-cli-cmd-options-no-color] --no-color <BOOLEAN>:: Remove color and other styling from the console output. + *Required*: No + *Default value*: false

[#ref-cli-cmd-options-notices] --notices <BOOLEAN>:: Show relevant notices. + *Required*: No + *Default value*: false

[#ref-cli-cmd-options-output] --output, -o <STRING>:: Specify the directory to output the synthesized cloud assembly to. + *Required*: Yes + *Default value*: cdk.out

[#ref-cli-cmd-options-path-metadata] --path-metadata <BOOLEAN>:: Include aws::cdk::path {aws} CloudFormation metadata for each resource. + *Required*: No + *Default value*: true

[#ref-cli-cmd-options-plugin] --plugin, -p <ARRAY>:: Name or path of a node package that extends CDK features. This option can be provided multiple times in a single command. + You can configure this option in the project’s cdk.json file or at ~/.cdk.json on your local development machine: + [source,json,subs=“verbatim,attributes”] — { // … “plugin”: [ “module\_1”, “module\_2” ], // … } — + *Required*: No

[#ref-cli-cmd-options-profile] --profile <STRING>:: Specify the name of the {aws} profile, containing your {aws} environment information, to use with the CDK CLI. + *Required*: Yes

[#ref-cli-cmd-options-proxy] --proxy <STRING>:: Use the indicated proxy. + If this option is not provided, the CDK CLI will read from the HTTPS\_PROXY environment variable. + *Required*: Yes + *Default value*: Read from HTTPS\_PROXY environment variable.

[#ref-cli-cmd-options-role-arn] --role-arn, -r <STRING>:: The ARN of the IAM role that the CDK CLI will assume when interacting with {aws} CloudFormation. + *Required*: No

[#ref-cli-cmd-options-staging] --staging <BOOLEAN>:: Copy assets to the output directory. + Specify false to prevent the copying of assets to the output directory. This allows the {aws} SAM CLI to reference the original source files when performing local debugging. + *Required*: No + *Default value*: true

[#ref-cli-cmd-options-strict] --strict <BOOLEAN>:: Do not construct stacks that contain warnings. + *Required*: No + *Default value*: false

[#ref-cli-cmd-options-trace] --trace <BOOLEAN>:: Print trace for stack warnings. + *Required*: No + *Default value*: false

[#ref-cli-cmd-options-verbose] --verbose, -v <COUNT>:: Show debug logs. You can specify this option multiple times to increase verbosity. + *Required*: No

[#ref-cli-cmd-options-version] --version <BOOLEAN>:: Show the CDK CLI version number. + *Required*: No + *Default value*: false

[#ref-cli-cmd-options-version-reporting] --version-reporting <BOOLEAN>:: Include the +{aws}::CDK::Metadata+ resource in synthesized {aws} CloudFormation templates. + *Required*: No + *Default value*: true

[#ref-cli-cmd-configure] == Providing and configuring options

You can pass options through command-line arguments. For most options, you can configure them in a cdk.json configuration file. When you use multiple configuration sources, the CDK CLI adheres to the following precedence:

. *Command-line values* – Any option provided at the command-line overrides options configured in cdk.json files. . *Project configuration file* – The cdk.json file in your CDK project’s directory. . *User configuration file* – The cdk.json file located at ~/.cdk.json on your local machine.

[#ref-cli-cmd-pass] == Passing options at the command line

[#ref-cli-cmd-pass-bool] *Passing boolean values*:: + For options that accept a boolean value, you can specify them in the following ways: +

* Use true and false values – Provide the boolean value with the command. The following is an example:

## [source,none,subs=“verbatim,attributes”]

* $ cdk deploy –watch=true $ cdk deploy –watch=false —
* Provide the option’s counterpart – Modify the option name by adding no to specify a false value. The following is an example:

## [source,none,subs=“verbatim,attributes”]

* $ cdk deploy –watch $ cdk deploy –no-watch —
* For options that default to true or false, you don’t have to provide the option unless you want to change from the default.

include::ref-cli-cmd-ack.adoc[leveloffset=+1]

include::ref-cli-cmd-bootstrap.adoc[leveloffset=+1]

include::ref-cli-cmd-context.adoc[leveloffset=+1]

include::ref-cli-cmd-deploy.adoc[leveloffset=+1]

include::ref-cli-cmd-destroy.adoc[leveloffset=+1]

include::ref-cli-cmd-diff.adoc[leveloffset=+1]

include::ref-cli-cmd-docs.adoc[leveloffset=+1]

include::ref-cli-cmd-doctor.adoc[leveloffset=+1]

include::ref-cli-cmd-drift.adoc[leveloffset=+1]

include::ref-cli-cmd-gc.adoc[leveloffset=+1]

include::ref-cli-cmd-import.adoc[leveloffset=+1]

include::ref-cli-cmd-init.adoc[leveloffset=+1]

include::ref-cli-cmd-list.adoc[leveloffset=+1]

include::ref-cli-cmd-metadata.adoc[leveloffset=+1]

include::ref-cli-cmd-migrate.adoc[leveloffset=+1]

include::ref-cli-cmd-notices.adoc[leveloffset=+1]

include::ref-cli-cmd-rollback.adoc[leveloffset=+1]

include::ref-cli-cmd-synth.adoc[leveloffset=+1]

include::ref-cli-cmd-watch.adoc[leveloffset=+1]

include::attributes.txt[]

// Attributes

[#reference] = {aws} CDK reference

// Content start

This section contains reference information for the {aws} Cloud Development Kit ({aws} CDK).

[#reference-api] == API reference

The https://docs.aws.amazon.com/cdk/api/v2[API Reference] contains information about the {aws} Construct Library and other APIs provided by the {aws} Cloud Development Kit ({aws} CDK). Most of the {aws} Construct Library is contained in a single package called by its TypeScript name: aws-cdk-lib. The actual package name varies by language. Separate versions of the API reference are provided for each supported programming language.

The CDK API reference is organized into sub-modules. There are one or more sub-modules for each {aws} service.

Each sub-module has an overview that includes information about how to use its APIs. For example, the link:https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.aws\_s3-readme.html[S3] overview demonstrates how to set default encryption on an Amazon Simple Storage Service (Amazon S3) bucket.

include::versioning.adoc[leveloffset=+1]

include::node-versions.adoc[leveloffset=+1]

include::videos.adoc[leveloffset=+1]

:doctype: book

include::attributes.txt[]

// Attributes [.topic] :info\_titleabbrev: Resources :keywords: {aws} CDK, {aws} Cloud Development Kit ({aws} CDK), {aws} CloudFormation, IaC, Infrastructure as code, {aws}, {aws} resources

[#resources] = Resources and the {aws} CDK

== [abstract]

## *Resources* are what you configure to use {aws} services in your applications. Resources are a feature of {aws} CloudFormation. By configuring resources and their properties in a CloudFormation template, you can deploy to CloudFormation to provision your resources. With the {aws} Cloud Development Kit ({aws} CDK), you can configure resources through constructs. You then deploy your CDK app, which involves synthesizing a {aws} CloudFormation template and deploying to {aws} CloudFormation to provision your resources.

// Content start

*Resources* are what you configure to use {aws} services in your applications. Resources are a feature of {aws} CloudFormation. By configuring resources and their properties in a {aws} CloudFormation template, you can deploy to {aws} CloudFormation to provision your resources. With the {aws} Cloud Development Kit ({aws} CDK), you can configure resources through constructs. You then deploy your CDK app, which involves synthesizing a {aws} CloudFormation template and deploying to {aws} CloudFormation to provision your resources.

[#resources-configure] == Configuring resources using constructs

As described in xref:constructs[{aws} CDK Constructs], the {aws} CDK provides a rich class library of constructs, called *constructs*, that represent all {aws} resources.

To create an instance of a resource using its corresponding construct, pass in the scope as the first argument, the logical ID of the construct, and a set of configuration properties (props). For example, here’s how to create an Amazon SQS queue with {aws} KMS encryption using the https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.aws\_sqs.Queue.html[sqs.Queue] construct from the {aws} Construct Library.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — import \* as sqs from ‘@aws-cdk/aws-sqs’;

new sqs.Queue(this, ‘MyQueue’, { encryption: sqs.QueueEncryption.KMS\_MANAGED }); —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const sqs = require(‘@aws-cdk/aws-sqs’);

new sqs.Queue(this, ‘MyQueue’, { encryption: sqs.QueueEncryption.KMS\_MANAGED }); —

Python:: + [source,python,subs=“verbatim,attributes”] — import aws\_cdk.aws\_sqs as sqs

== sqs.Queue(self, “MyQueue”, encryption=sqs.QueueEncryption.KMS\_MANAGED)

Java:: + [source,java,subs=“verbatim,attributes”] — import software.amazon.awscdk.services.sqs.\*;

Queue.Builder.create(this, “MyQueue”).encryption( QueueEncryption.KMS\_MANAGED).build(); —

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

new Queue(this, “MyQueue”, new QueueProps { Encryption = QueueEncryption.KMS\_MANAGED }); —

Go:: + [source,go,subs=“verbatim,attributes”] — import ( “github.com/aws/aws-cdk-go/awscdk/v2” “github.com/aws/jsii-runtime-go”  
sqs “github.com/aws/aws-cdk-go/awscdk/v2/awssqs” )

sqs.NewQueue(stack, jsii.String(“MyQueue”), &sqs.QueueProps{ Encryption: sqs.QueueEncryption\_KMS\_MANAGED, }) — ====

Some configuration props are optional, and in many cases have default values. In some cases, all props are optional, and the last argument can be omitted entirely.

[#resources-attributes] === Resource attributes

Most resources in the {aws} Construct Library expose attributes, which are resolved at deployment time by {aws} CloudFormation. Attributes are exposed in the form of properties on the resource classes with the type name as a prefix. The following example shows how to get the URL of an Amazon SQS queue using the queueUrl (Python: queue\_url) property.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — import \* as sqs from ‘@aws-cdk/aws-sqs’;

const queue = new sqs.Queue(this, ‘MyQueue’); const url = queue.queueUrl; // => A string representing a deploy-time value —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const sqs = require(‘@aws-cdk/aws-sqs’);

const queue = new sqs.Queue(this, ‘MyQueue’); const url = queue.queueUrl; // => A string representing a deploy-time value —

Python:: + [source,python,subs=“verbatim,attributes”] — import aws\_cdk.aws\_sqs as sqs

queue = sqs.Queue(self, “MyQueue”) url = queue.queue\_url # => A string representing a deploy-time value —

Java:: + [source,java,subs=“verbatim,attributes”] — Queue queue = new Queue(this, “MyQueue”); String url = queue.getQueueUrl(); // => A string representing a deploy-time value —

C#:: + [source,csharp,subs=“verbatim,attributes”] — var queue = new Queue(this, “MyQueue”); var url = queue.QueueUrl; // => A string representing a deploy-time value —

Go:: + [source,go,subs=“verbatim,attributes”] — import ( “github.com/aws/aws-cdk-go/awscdk/v2” “github.com/aws/jsii-runtime-go”  
sqs “github.com/aws/aws-cdk-go/awscdk/v2/awssqs” )

queue := sqs.NewQueue(stack, jsii.String(“MyQueue”), &sqs.QueueProps{}) url := queue.QueueUrl() // => A string representing a deploy-time value — ====

See xref:tokens[Tokens and the {aws} CDK] for information about how the {aws} CDK encodes deploy-time attributes as strings.

[#resources-referencing] == Referencing resources

When configuring resources, you will often have to reference properties of another resource. The following are examples:

* An Amazon Elastic Container Service (Amazon ECS) resource requires a reference to the cluster on which it runs.
* An Amazon CloudFront distribution requires a reference to the Amazon Simple Storage Service (Amazon S3) bucket containing the source code.

You can reference resources in any of the following ways:

* By passing a resource defined in your CDK app, either in the same stack or in a different one
* By passing a proxy object referencing a resource defined in your {aws} account, created from a unique identifier of the resource (such as an ARN)

If the property of a construct represents a construct for another resource, its type is that of the interface type of the construct. For example, the Amazon ECS construct takes a property cluster of type ecs.ICluster. Another example, is the CloudFront distribution construct that takes a property sourceBucket (Python: source\_bucket) of type s3.IBucket.

You can directly pass any resource object of the proper type defined in the same {aws} CDK app. The following example defines an Amazon ECS cluster and then uses it to define an Amazon ECS service.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — const cluster = new ecs.Cluster(this, ‘Cluster’, { /*…*/ });

== const service = new ecs.Ec2Service(this, ‘Service’, { cluster: cluster });

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const cluster = new ecs.Cluster(this, ‘Cluster’, { /*…*/ });

== const service = new ecs.Ec2Service(this, ‘Service’, { cluster: cluster });

Python:: + [source,python,subs=“verbatim,attributes”] — cluster = ecs.Cluster(self, “Cluster”)

== service = ecs.Ec2Service(self, “Service”, cluster=cluster)

Java:: + [source,java,subs=“verbatim,attributes”] — Cluster cluster = new Cluster(this, “Cluster”); Ec2Service service = new Ec2Service(this, “Service”, new Ec2ServiceProps.Builder().cluster(cluster).build()); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — var cluster = new Cluster(this, “Cluster”); var service = new Ec2Service(this, “Service”, new Ec2ServiceProps { Cluster = cluster }); —

Go:: + [source,go,subs=“verbatim,attributes”] — import ( “github.com/aws/aws-cdk-go/awscdk/v2” “github.com/aws/jsii-runtime-go” + ecs “github.com/aws/aws-cdk-go/awscdk/v2/awsecs” )

cluster := ecs.NewCluster(stack, jsii.String(“MyCluster”), &ecs.ClusterProps{}) service := ecs.NewEc2Service(stack, jsii.String(“MyService”), &ecs.Ec2ServiceProps{ Cluster: cluster, }) — ====

[#resource-stack] === Referencing resources in a different stack

You can refer to resources in a different stack as long as they are defined in the same app and are in the same {aws} environment. The following pattern is generally used:

* Store a reference to the construct as an attribute of the stack that produces the resource. (To get a reference to the current construct’s stack, use Stack.of(this).)
* Pass this reference to the constructor of the stack that consumes the resource as a parameter or a property. The consuming stack then passes it as a property to any construct that needs it.

The following example defines a stack stack1. This stack defines an Amazon S3 bucket and stores a reference to the bucket construct as an attribute of the stack. Then the app defines a second stack, stack2, which accepts a bucket at instantiation. stack2 might, for example, define an {aws} Glue Table that uses the bucket for data storage.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — const prod = { account: ‘123456789012’, region: ‘us-east-1’ };

const stack1 = new StackThatProvidesABucket(app, ‘Stack1’, { env: prod });

// stack2 will take a property { bucket: IBucket } const stack2 = new StackThatExpectsABucket(app, ‘Stack2’, { bucket: stack1.bucket, env: prod }); —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const prod = { account: ‘123456789012’, region: ‘us-east-1’ };

const stack1 = new StackThatProvidesABucket(app, ‘Stack1’, { env: prod });

// stack2 will take a property { bucket: IBucket } const stack2 = new StackThatExpectsABucket(app, ‘Stack2’, { bucket: stack1.bucket, env: prod }); —

Python:: + [source,python,subs=“verbatim,attributes”] — prod = core.Environment(account=“123456789012”, region=“us-east-1”)

stack1 = StackThatProvidesABucket(app, “Stack1”, env=prod)

= stack2 will take a property “bucket”

## stack2 = StackThatExpectsABucket(app, “Stack2”, bucket=stack1.bucket, env=prod)

Java:: + [source,java,subs=“verbatim,attributes”] — // Helper method to build an environment static Environment makeEnv(String account, String region) { return Environment.builder().account(account).region(region) .build(); }

App app = new App();

Environment prod = makeEnv(“123456789012”, “us-east-1”);

StackThatProvidesABucket stack1 = new StackThatProvidesABucket(app, “Stack1”, StackProps.builder().env(prod).build());

// stack2 will take an argument “bucket” StackThatExpectsABucket stack2 = new StackThatExpectsABucket(app, “Stack,”, StackProps.builder().env(prod).build(), stack1.bucket); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — Amazon.CDK.Environment makeEnv(string account, string region) { return new Amazon.CDK.Environment { Account = account, Region = region }; }

var prod = makeEnv(account: “123456789012”, region: “us-east-1”);

var stack1 = new StackThatProvidesABucket(app, “Stack1”, new StackProps { Env = prod });

// stack2 will take a property “bucket” var stack2 = new StackThatExpectsABucket(app, “Stack2”, new StackProps { Env = prod, bucket = stack1.Bucket}); — ====

If the {aws} CDK determines that the resource is in the same environment, but in a different stack, it automatically synthesizes {aws} CloudFormation https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-stack-exports.html[exports] in the producing stack and an https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-importvalue.html[Fn::ImportValue] in the consuming stack to transfer that information from one stack to the other.

[#resources-deadlock] ==== Resolving dependency deadlocks

Referencing a resource from one stack in a different stack creates a dependency between the two stacks. This makes sure that they’re deployed in the right order. After the stacks are deployed, this dependency is concrete. After that, removing the use of the shared resource from the consuming stack can cause an unexpected deployment failure. This happens if there is another dependency between the two stacks that force them to be deployed in the same order. It can also happen without a dependency if the producing stack is simply chosen by the CDK Toolkit to be deployed first. The {aws} CloudFormation export is removed from the producing stack because it’s no longer needed, but the exported resource is still being used in the consuming stack because its update is not yet deployed. Therefore, deploying the producer stack fails.

To break this deadlock, remove the use of the shared resource from the consuming stack. (This removes the automatic export from the producing stack.) Next, manually add the same export to the producing stack using exactly the same logical ID as the automatically generated export. Remove the use of the shared resource in the consuming stack and deploy both stacks. Then, remove the manual export (and the shared resource if it’s no longer needed) and deploy both stacks again. The stack’s https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.Stack.html#exportwbrvalueexportedvalue-options[exportValue()] method is a convenient way to create the manual export for this purpose. (See the example in the linked method reference.)

[#resources-external] === Referencing resources in your {aws} account

Suppose you want to use a resource already available in your {aws} account in your {aws} CDK app. This might be a resource that was defined through the console, an {aws} SDK, directly with {aws} CloudFormation, or in a different {aws} CDK application. You can turn the resource’s ARN (or another identifying attribute, or group of attributes) into a proxy object. The proxy object serves as a reference to the resource by calling a static factory method on the resource’s class.

When you create such a proxy, the external resource *does not* become a part of your {aws} CDK app. Therefore, changes you make to the proxy in your {aws} CDK app do not affect the deployed resource. The proxy can, however, be passed to any {aws} CDK method that requires a resource of that type.

The following example shows how to reference a bucket based on an existing bucket with the ARN arn:aws:s3:::amzn-s3-demo-bucket1, and an Amazon Virtual Private Cloud based on an existing VPC having a specific ID.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — // Construct a proxy for a bucket by its name (must be same account) s3.Bucket.fromBucketName(this, ‘MyBucket’, ‘amzn-s3-demo-bucket1’);

// Construct a proxy for a bucket by its full ARN (can be another account) s3.Bucket.fromBucketArn(this, ‘MyBucket’, ‘arn:aws:s3:::amzn-s3-demo-bucket1’);

// Construct a proxy for an existing VPC from its attribute(s) ec2.Vpc.fromVpcAttributes(this, ‘MyVpc’, { vpcId: ‘vpc-1234567890abcde’, }); —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — // Construct a proxy for a bucket by its name (must be same account) s3.Bucket.fromBucketName(this, ‘MyBucket’, ‘amzn-s3-demo-bucket1’);

// Construct a proxy for a bucket by its full ARN (can be another account) s3.Bucket.fromBucketArn(this, ‘MyBucket’, ‘arn:aws:s3:::amzn-s3-demo-bucket1’);

// Construct a proxy for an existing VPC from its attribute(s) ec2.Vpc.fromVpcAttributes(this, ‘MyVpc’, { vpcId: ‘vpc-1234567890abcde’ }); —

Python:: + [source,python,subs=“verbatim,attributes”] —

= Construct a proxy for a bucket by its name (must be same account)

s3.Bucket.from\_bucket\_name(self, “MyBucket”, “amzn-s3-demo-bucket1”)

= Construct a proxy for a bucket by its full ARN (can be another account)

s3.Bucket.from\_bucket\_arn(self, “MyBucket”, “arn:aws:s3:::amzn-s3-demo-bucket1”)

= Construct a proxy for an existing VPC from its attribute(s)

## ec2.Vpc.from\_vpc\_attributes(self, “MyVpc”, vpc\_id=“vpc-1234567890abcdef”)

Java:: + [source,java,subs=“verbatim,attributes”] — // Construct a proxy for a bucket by its name (must be same account) Bucket.fromBucketName(this, “MyBucket”, “amzn-s3-demo-bucket1”);

// Construct a proxy for a bucket by its full ARN (can be another account) Bucket.fromBucketArn(this, “MyBucket”, “arn:aws:s3:::amzn-s3-demo-bucket1”);

// Construct a proxy for an existing VPC from its attribute(s) Vpc.fromVpcAttributes(this, “MyVpc”, VpcAttributes.builder() .vpcId(“vpc-1234567890abcdef”).build()); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — // Construct a proxy for a bucket by its name (must be same account) Bucket.FromBucketName(this, “MyBucket”, “amzn-s3-demo-bucket1”);

// Construct a proxy for a bucket by its full ARN (can be another account) Bucket.FromBucketArn(this, “MyBucket”, “arn:aws:s3:::amzn-s3-demo-bucket1”);

// Construct a proxy for an existing VPC from its attribute(s) Vpc.FromVpcAttributes(this, “MyVpc”, new VpcAttributes { VpcId = “vpc-1234567890abcdef” }); —

Go:: + [source,go,subs=“verbatim,attributes”] — // Define a proxy for a bucket by its name (must be same account) s3.Bucket\_FromBucketName(stack, jsii.String(“MyBucket”), jsii.String(“amzn-s3-demo-bucket1”))

// Define a proxy for a bucket by its full ARN (can be another account) s3.Bucket\_FromBucketArn(stack, jsii.String(“MyBucket”), jsii.String(“arn:aws:s3:::amzn-s3-demo-bucket1”))

// Define a proxy for an existing VPC from its attributes ec2.Vpc\_FromVpcAttributes(stack, jsii.String(“MyVpc”), &ec2.VpcAttributes{ VpcId: jsii.String(“vpc-1234567890abcde”), }) — ====

Let’s take a closer look at the https://docs.aws.amazon.com/cdk/api/v1/docs/@aws-cdk\_aws-ec2.Vpc.html#static-fromwbrlookupscope-id-options [`Vpc.fromLookup()`] method. Because the ec2.Vpc construct is complex, there are many ways you might want to select the VPC to be used with your CDK app. To address this, the VPC construct has a fromLookup static method (Python: from\_lookup) that lets you look up the desired Amazon VPC by querying your {aws} account at synthesis time.

To use Vpc.fromLookup(), the system that synthesizes the stack must have access to the account that owns the Amazon VPC. This is because the CDK Toolkit queries the account to find the right Amazon VPC at synthesis time.

Furthermore, Vpc.fromLookup() works only in stacks that are defined with an explicit *account* and *region* (see xref:environments[Environments for the {aws} CDK]). If the {aws} CDK tries to look up an Amazon VPC from an xref:stack-api[environment-agnostic stack], the CDK Toolkit doesn’t know which environment to query to find the VPC.

You must provide Vpc.fromLookup() attributes sufficient to uniquely identify a VPC in your {aws} account. For example, there can only ever be one default VPC, so it’s sufficient to specify the VPC as the default.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — ec2.Vpc.fromLookup(this, ‘DefaultVpc’, { isDefault: true }); —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — ec2.Vpc.fromLookup(this, ‘DefaultVpc’, { isDefault: true }); —

Python:: + [source,python,subs=“verbatim,attributes”] — ec2.Vpc.from\_lookup(self, “DefaultVpc”, is\_default=True) —

Java:: + [source,java,subs=“verbatim,attributes”] — Vpc.fromLookup(this, “DefaultVpc”, VpcLookupOptions.builder() .isDefault(true).build()); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — Vpc.FromLookup(this, id = “DefaultVpc”, new VpcLookupOptions { IsDefault = true }); —

Go:: + [source,go,subs=“verbatim,attributes”] — ec2.Vpc\_FromLookup(this, jsii.String(“DefaultVpc”), &ec2.VpcLookupOptions{ IsDefault: jsii.Bool(true), }) — ====

You can also use the tags property to query for VPCs by tag. You can add tags to the Amazon VPC at the time of its creation by using {aws} CloudFormation or the {aws} CDK. You can edit tags at any time after creation by using the {aws} Management Console, the {aws} CLI, or an {aws} SDK. In addition to any tags you add yourself, the {aws} CDK automatically adds the following tags to all VPCs it creates.

* *Name* – The name of the VPC.
* *aws-cdk:subnet-name* – The name of the subnet.
* *aws-cdk:subnet-type* – The type of the subnet: Public, Private, or Isolated.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — ec2.Vpc.fromLookup(this, ‘PublicVpc’, {tags: {‘aws-cdk:subnet-type’: “Public”}}); —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — ec2.Vpc.fromLookup(this, ‘PublicVpc’, {tags: {‘aws-cdk:subnet-type’: “Public”}}); —

Python:: + [source,python,subs=“verbatim,attributes”] — ec2.Vpc.from\_lookup(self, “PublicVpc”, tags={“aws-cdk:subnet-type”: “Public”}) —

Java:: + [source,java,subs=“verbatim,attributes”] — Vpc.fromLookup(this, “PublicVpc”, VpcLookupOptions.builder() .tags(java.util.Map.of(“aws-cdk:subnet-type”, “Public”)) // Java 9 or later .build()); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — Vpc.FromLookup(this, id: “PublicVpc”, new VpcLookupOptions { Tags = new Dictionary<string, string> { [“aws-cdk:subnet-type”] = “Public” } }); —

Go:: + [source,go,subs=“verbatim,attributes”] — ec2.Vpc\_FromLookup(this, jsii.String(“DefaultVpc”), &ec2.VpcLookupOptions{ Tags: &map[string]\*string{“aws-cdk:subnet-type”: jsii.String(“Public”)}, }) — ====

Results of Vpc.fromLookup() are cached in the project’s cdk.context.json file. (See xref:context[Context values and the {aws} CDK].) Commit this file to version control so that your app will continue to refer to the same Amazon VPC. This works even if you later change the attributes of your VPCs in a way that would result in a different VPC being selected. This is particularly important if you’re deploying the stack in an environment that doesn’t have access to the {aws} account that defines the VPC, such as xref:cdk-pipeline[CDK Pipelines].

Although you can use an external resource anywhere you’d use a similar resource defined in your {aws} CDK app, you cannot modify it. For example, calling addToResourcePolicy (Python: add\_to\_resource\_policy) on an external s3.Bucket does nothing.

[#resources-physical-names] == Resource physical names

The logical names of resources in {aws} CloudFormation are different from the names of resources that are shown in the {aws} Management Console after they’re deployed by {aws} CloudFormation. The {aws} CDK calls these final names *physical names*.

For example, {aws} CloudFormation might create the Amazon S3 bucket with the logical ID Stack2MyBucket4DD88B4F and the physical name stack2MyBucket4dd88b4f-iuv1rbv9z3to.

You can specify a physical name when creating constructs that represent resources by using the property <resourceType>Name. The following example creates an Amazon S3 bucket with the physical name amzn-s3-demo-bucket.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — const bucket = new s3.Bucket(this, ‘MyBucket’, { bucketName: ‘amzn-s3-demo-bucket’, }); —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const bucket = new s3.Bucket(this, ‘MyBucket’, { bucketName: ‘amzn-s3-demo-bucket’ }); —

Python:: + [source,python,subs=“verbatim,attributes”] — bucket = s3.Bucket(self, “MyBucket”, bucket\_name=“amzn-s3-demo-bucket”) —

Java:: + [source,java,subs=“verbatim,attributes”] — Bucket bucket = Bucket.Builder.create(this, “MyBucket”) .bucketName(“amzn-s3-demo-bucket”).build(); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — var bucket = new Bucket(this, “MyBucket”, new BucketProps { BucketName = “amzn-s3-demo-bucket” }); —

Go:: + [source,go,subs=“verbatim,attributes”] — bucket := s3.NewBucket(this, jsii.String(“MyBucket”), &s3.BucketProps{ BucketName: jsii.String(“amzn-s3-demo-bucket”), }) — ====

Assigning physical names to resources has some disadvantages in {aws} CloudFormation. Most importantly, any changes to deployed resources that require a resource replacement, such as changes to a resource’s properties that are immutable after creation, will fail if a resource has a physical name assigned. If you end up in that state, the only solution is to delete the {aws} CloudFormation stack, then deploy the {aws} CDK app again. See the https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-name.html[{aws} CloudFormation documentation] for details.

In some cases, such as when creating an {aws} CDK app with cross-environment references, physical names are required for the {aws} CDK to function correctly. In those cases, if you don’t want to bother with coming up with a physical name yourself, you can let the {aws} CDK name it for you. To do so, use the special value PhysicalName.GENERATE\_IF\_NEEDED, as follows.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — const bucket = new s3.Bucket(this, ‘MyBucket’, { bucketName: core.PhysicalName.GENERATE\_IF\_NEEDED, }); —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const bucket = new s3.Bucket(this, ‘MyBucket’, { bucketName: core.PhysicalName.GENERATE\_IF\_NEEDED }); —

Python:: + [source,python,subs=“verbatim,attributes”] — bucket = s3.Bucket(self, “MyBucket”, bucket\_name=core.PhysicalName.GENERATE\_IF\_NEEDED) —

Java:: + [source,java,subs=“verbatim,attributes”] — Bucket bucket = Bucket.Builder.create(this, “MyBucket”) .bucketName(PhysicalName.GENERATE\_IF\_NEEDED).build(); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — var bucket = new Bucket(this, “MyBucket”, new BucketProps { BucketName = PhysicalName.GENERATE\_IF\_NEEDED }); —

Go:: + [source,go,subs=“verbatim,attributes”] — bucket := s3.NewBucket(this, jsii.String(“MyBucket”), &s3.BucketProps{ BucketName: awscdk.PhysicalName\_GENERATE\_IF\_NEEDED(), }) — ====

[#resources-identifiers] == Passing unique resource identifiers

Whenever possible, you should pass resources by reference, as described in the previous section. However, there are cases where you have no other choice but to refer to a resource by one of its attributes. Example use cases include the following:

* When you are using low-level {aws} CloudFormation resources.
* When you need to expose resources to the runtime components of an {aws} CDK application, such as when referring to Lambda functions through environment variables.

These identifiers are available as attributes on the resources, such as the following.

==== [role=“tablist”] TypeScript:: + [source,none,subs=“verbatim,attributes”] — bucket.bucketName lambdaFunc.functionArn securityGroup.groupArn —

JavaScript:: + [source,none,subs=“verbatim,attributes”] — bucket.bucketName lambdaFunc.functionArn securityGroup.groupArn —

Python:: + [source,none,subs=“verbatim,attributes”] — bucket.bucket\_name lambda\_func.function\_arn security\_group\_arn —

Java:: The Java {aws} CDK binding uses getter methods for attributes. + [source,java,subs=“verbatim,attributes”] — bucket.getBucketName() lambdaFunc.getFunctionArn() securityGroup.getGroupArn() —

C#:: + [source,none,subs=“verbatim,attributes”] — bucket.BucketName lambdaFunc.FunctionArn securityGroup.GroupArn —

Go:: + [source,none,subs=“verbatim,attributes”] — bucket.BucketName() fn.FunctionArn() — ====

The following example shows how to pass a generated bucket name to an {aws} Lambda function.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — const bucket = new s3.Bucket(this, ‘Bucket’);

new lambda.Function(this, ‘MyLambda’, { // … environment: { BUCKET\_NAME: bucket.bucketName, }, }); —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const bucket = new s3.Bucket(this, ‘Bucket’);

new lambda.Function(this, ‘MyLambda’, { // … environment: { BUCKET\_NAME: bucket.bucketName } }); —

Python:: + [source,python,subs=“verbatim,attributes”] — bucket = s3.Bucket(self, “Bucket”)

== lambda.Function(self, “MyLambda”, environment=dict(BUCKET\_NAME=bucket.bucket\_name))

Java:: + [source,java,subs=“verbatim,attributes”] — final Bucket bucket = new Bucket(this, “Bucket”);

Function.Builder.create(this, “MyLambda”) .environment(java.util.Map.of( // Java 9 or later “BUCKET\_NAME”, bucket.getBucketName())) .build(); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — var bucket = new Bucket(this, “Bucket”);

new Function(this, “MyLambda”, new FunctionProps { Environment = new Dictionary<string, string> { [“BUCKET\_NAME”] = bucket.BucketName } }); —

Go:: + [source,go,subs=“verbatim,attributes”] — bucket := s3.NewBucket(this, jsii.String(“Bucket”), &s3.BucketProps{}) lambda.NewFunction(this, jsii.String(“MyLambda”), &lambda.FunctionProps{ Environment: &map[string]\*string{“BUCKET\_NAME”: bucket.BucketName()}, }) — ====

[#resources-grants] == Granting permissions between resources

Higher-level constructs make least-privilege permissions achievable by offering simple, intent-based APIs to express permission requirements. For example, many L2 constructs offer grant methods that you can use to grant an entity (such as an IAM role or user) permission to work with the resource, without having to manually create IAM permission statements.

The following example creates the permissions to allow a Lambda function’s execution role to read and write objects to a particular Amazon S3 bucket. If the Amazon S3 bucket is encrypted with an {aws} KMS key, this method also grants permissions to the Lambda function’s execution role to decrypt with the key.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — if (bucket.grantReadWrite(func).success) { // … } —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — if ( bucket.grantReadWrite(func).success) { // … } —

Python:: + [source,python,subs=“verbatim,attributes”] — if bucket.grant\_read\_write(func).success: # … —

Java:: + [source,java,subs=“verbatim,attributes”] — if (bucket.grantReadWrite(func).getSuccess()) { // … } —

C#:: + [source,csharp,subs=“verbatim,attributes”] — if (bucket.GrantReadWrite(func).Success) { // … } —

Go:: + [source,go,subs=“verbatim,attributes”] — if \*bucket.GrantReadWrite(function, nil).Success() { // … } — ====

The grant methods return an iam.Grant object. Use the success attribute of the Grant object to determine whether the grant was effectively applied (for example, it may not have been applied on xref:resources-referencing[external resources]). You can also use the assertSuccess (Python: assert\_success) method of the Grant object to enforce that the grant was successfully applied.

If a specific grant method isn’t available for the particular use case, you can use a generic grant method to define a new grant with a specified list of actions.

The following example shows how to grant a Lambda function access to the Amazon DynamoDB CreateBackup action.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — table.grant(func, ‘dynamodb:CreateBackup’); —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — table.grant(func, ‘dynamodb:CreateBackup’); —

Python:: + [source,python,subs=“verbatim,attributes”] — table.grant(func, “dynamodb:CreateBackup”) —

Java:: + [source,java,subs=“verbatim,attributes”] — table.grant(func, “dynamodb:CreateBackup”); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — table.Grant(func, “dynamodb:CreateBackup”); —

Go:: + [source,go,subs=“verbatim,attributes”] — table := dynamodb.NewTable(this, jsii.String(“MyTable”), &dynamodb.TableProps{}) table.Grant(function, jsii.String(“dynamodb:CreateBackup”)) — ====

Many resources, such as Lambda functions, require a role to be assumed when executing code. A configuration property enables you to specify an iam.IRole. If no role is specified, the function automatically creates a role specifically for this use. You can then use grant methods on the resources to add statements to the role.

The grant methods are built using lower-level APIs for handling with IAM policies. Policies are modeled as https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.aws\_iam.PolicyDocument.html[PolicyDocument] objects. Add statements directly to roles (or a construct’s attached role) using the addToRolePolicy method (Python: add\_to\_role\_policy), or to a resource’s policy (such as a Bucket policy) using the addToResourcePolicy (Python: add\_to\_resource\_policy) method.

[#resources-metrics] == Resource metrics and alarms

Many resources emit CloudWatch metrics that can be used to set up monitoring dashboards and alarms. Higher-level constructs have metric methods that let you access the metrics without looking up the correct name to use.

The following example shows how to define an alarm when the ApproximateNumberOfMessagesNotVisible of an Amazon SQS queue exceeds 100.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — import \* as cw from ‘@aws-cdk/aws-cloudwatch’; import \* as sqs from ‘@aws-cdk/aws-sqs’; import { Duration } from ‘@aws-cdk/core’;

const queue = new sqs.Queue(this, ‘MyQueue’);

const metric = queue.metricApproximateNumberOfMessagesNotVisible({ label: ‘Messages Visible (Approx)’, period: Duration.minutes(5), // … }); metric.createAlarm(this, ‘TooManyMessagesAlarm’, { comparisonOperator: cw.ComparisonOperator.GREATER\_THAN\_THRESHOLD, threshold: 100, // … }); —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const cw = require(‘@aws-cdk/aws-cloudwatch’); const sqs = require(‘@aws-cdk/aws-sqs’); const { Duration } = require(‘@aws-cdk/core’);

const queue = new sqs.Queue(this, ‘MyQueue’);

const metric = queue.metricApproximateNumberOfMessagesNotVisible({ label: ‘Messages Visible (Approx)’, period: Duration.minutes(5) // … }); metric.createAlarm(this, ‘TooManyMessagesAlarm’, { comparisonOperator: cw.ComparisonOperator.GREATER\_THAN\_THRESHOLD, threshold: 100 // … }); —

Python:: + [source,python,subs=“verbatim,attributes”] — import aws\_cdk.aws\_cloudwatch as cw import aws\_cdk.aws\_sqs as sqs from aws\_cdk.core import Duration

queue = sqs.Queue(self, “MyQueue”) metric = queue.metric\_approximate\_number\_of\_messages\_not\_visible( label=“Messages Visible (Approx)”, period=Duration.minutes(5), # … ) metric.create\_alarm(self, “TooManyMessagesAlarm”, comparison\_operator=cw.ComparisonOperator.GREATER\_THAN\_THRESHOLD, threshold=100, # … ) —

Java:: + [source,java,subs=“verbatim,attributes”] — import software.amazon.awscdk.core.Duration; import software.amazon.awscdk.services.sqs.Queue; import software.amazon.awscdk.services.cloudwatch.Metric; import software.amazon.awscdk.services.cloudwatch.MetricOptions; import software.amazon.awscdk.services.cloudwatch.CreateAlarmOptions; import software.amazon.awscdk.services.cloudwatch.ComparisonOperator;

Queue queue = new Queue(this, “MyQueue”);

Metric metric = queue .metricApproximateNumberOfMessagesNotVisible(MetricOptions.builder() .label(“Messages Visible (Approx)”) .period(Duration.minutes(5)).build());

metric.createAlarm(this, “TooManyMessagesAlarm”, CreateAlarmOptions.builder() .comparisonOperator(ComparisonOperator.GREATER\_THAN\_THRESHOLD) .threshold(100) // … .build()); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — using cdk = Amazon.CDK; using cw = Amazon.CDK.{aws}.CloudWatch; using sqs = Amazon.CDK.{aws}.SQS;

var queue = new sqs.Queue(this, “MyQueue”); var metric = queue.MetricApproximateNumberOfMessagesNotVisible(new cw.MetricOptions { Label = “Messages Visible (Approx)”, Period = cdk.Duration.Minutes(5), // … }); metric.CreateAlarm(this, “TooManyMessagesAlarm”, new cw.CreateAlarmOptions { ComparisonOperator = cw.ComparisonOperator.GREATER\_THAN\_THRESHOLD, Threshold = 100, // .. }); —

Go:: + [source,go,subs=“verbatim,attributes”] — import ( “github.com/aws/aws-cdk-go/awscdk/v2” “github.com/aws/jsii-runtime-go” cw “github.com/aws/aws-cdk-go/awscdk/v2/awscloudwatch” sqs “github.com/aws/aws-cdk-go/awscdk/v2/awssqs” )

queue := sqs.NewQueue(this, jsii.String(“MyQueue”), &sqs.QueueProps{}) metric := queue.MetricApproximateNumberOfMessagesNotVisible(&cw.MetricOptions{ Label: jsii.String(“Messages Visible (Approx)”), Period: awscdk.Duration\_Minutes(jsii.Number(5)), })

metric.CreateAlarm(this, jsii.String(“TooManyMessagesAlarm”), &cw.CreateAlarmOptions{ ComparisonOperator: cw.ComparisonOperator\_GREATER\_THAN\_THRESHOLD, Threshold: jsii.Number(100), }) — ====

If there is no method for a particular metric, you can use the general metric method to specify the metric name manually.

Metrics can also be added to CloudWatch dashboards. See https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.aws\_cloudwatch-readme.html[CloudWatch].

[#resources-traffic] == Network traffic

In many cases, you must enable permissions on a network for an application to work, such as when the compute infrastructure needs to access the persistence layer. Resources that establish or listen for connections expose methods that enable traffic flows, including setting security group rules or network ACLs.

https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.aws\_ec2.IConnectable.html[IConnectable] resources have a connections property that is the gateway to network traffic rules configuration.

You enable data to flow on a given network path by using allow methods. The following example enables HTTPS connections to the web and incoming connections from the Amazon EC2 Auto Scaling group fleet2.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — import \* as asg from ‘@aws-cdk/aws-autoscaling’; import \* as ec2 from ‘@aws-cdk/aws-ec2’;

const fleet1: asg.AutoScalingGroup = asg.AutoScalingGroup(/*…*/);

// Allow surfing the (secure) web fleet1.connections.allowTo(new ec2.Peer.anyIpv4(), new ec2.Port({ fromPort: 443, toPort: 443 }));

const fleet2: asg.AutoScalingGroup = asg.AutoScalingGroup(/*…*/); fleet1.connections.allowFrom(fleet2, ec2.Port.AllTraffic()); —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const asg = require(‘@aws-cdk/aws-autoscaling’); const ec2 = require(‘@aws-cdk/aws-ec2’);

const fleet1 = asg.AutoScalingGroup();

// Allow surfing the (secure) web fleet1.connections.allowTo(new ec2.Peer.anyIpv4(), new ec2.Port({ fromPort: 443, toPort: 443 }));

const fleet2 = asg.AutoScalingGroup(); fleet1.connections.allowFrom(fleet2, ec2.Port.AllTraffic()); —

Python:: + [source,python,subs=“verbatim,attributes”] — import aws\_cdk.aws\_autoscaling as asg import aws\_cdk.aws\_ec2 as ec2

fleet1 = asg.AutoScalingGroup( … )

= Allow surfing the (secure) web

fleet1.connections.allow\_to(ec2.Peer.any\_ipv4(), ec2.Port(PortProps(from\_port=443, to\_port=443)))

fleet2 = asg.AutoScalingGroup( … ) fleet1.connections.allow\_from(fleet2, ec2.Port.all\_traffic()) —

Java:: + [source,java,subs=“verbatim,attributes”] — import software.amazon.awscdk.services.autoscaling.AutoScalingGroup; import software.amazon.awscdk.services.ec2.Peer; import software.amazon.awscdk.services.ec2.Port;

AutoScalingGroup fleet1 = AutoScalingGroup.Builder.create(this, “MyFleet”) /\* … \*/.build();

// Allow surfing the (secure) Web fleet1.getConnections().allowTo(Peer.anyIpv4(), Port.Builder.create().fromPort(443).toPort(443).build());

AutoScalingGroup fleet2 = AutoScalingGroup.Builder.create(this, “MyFleet2”) /\* … \*/.build(); fleet1.getConnections().allowFrom(fleet2, Port.allTraffic()); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — using cdk = Amazon.CDK; using asg = Amazon.CDK.AWS.AutoScaling; using ec2 = Amazon.CDK.AWS.EC2;

// Allow surfing the (secure) Web var fleet1 = new asg.AutoScalingGroup(this, “MyFleet”, new asg.AutoScalingGroupProps { /\* … \*/ }); fleet1.Connections.AllowTo(ec2.Peer.AnyIpv4(), new ec2.Port(new ec2.PortProps { FromPort = 443, ToPort = 443 }));

var fleet2 = new asg.AutoScalingGroup(this, “MyFleet2”, new asg.AutoScalingGroupProps { /\* … \*/ }); fleet1.Connections.AllowFrom(fleet2, ec2.Port.AllTraffic()); —

Go:: + [source,go,subs=“verbatim,attributes”] — import ( “github.com/aws/aws-cdk-go/awscdk/v2” “github.com/aws/jsii-runtime-go” autoscaling “github.com/aws/aws-cdk-go/awscdk/v2/awsautoscaling” ec2 “github.com/aws/aws-cdk-go/awscdk/v2/awsec2” )

fleet1 := autoscaling.NewAutoScalingGroup(this, jsii.String(“MyFleet1”), &autoscaling.AutoScalingGroupProps{}) fleet1.Connections().AllowTo(ec2.Peer\_AnyIpv4(),ec2.NewPort(&ec2.PortProps{ FromPort: jsii.Number(443), ToPort: jsii.Number(443) }),jsii.String(“secure web”))

fleet2 := autoscaling.NewAutoScalingGroup(this, jsii.String(“MyFleet2”), &autoscaling.AutoScalingGroupProps{}) fleet1.Connections().AllowFrom(fleet2, ec2.Port\_AllTraffic(),jsii.String(“all traffic”)) — ====

Certain resources have default ports associated with them. Examples include the listener of a load balancer on the public port, and the ports on which the database engine accepts connections for instances of an Amazon RDS database. In such cases, you can enforce tight network control without having to manually specify the port. To do so, use the allowDefaultPortFrom and allowToDefaultPort methods (Python: allow\_default\_port\_from, allow\_to\_default\_port).

The following example shows how to enable connections from any IPV4 address, and a connection from an Auto Scaling group to access a database.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — listener.connections.allowDefaultPortFromAnyIpv4(‘Allow public access’);

== fleet.connections.allowToDefaultPort(rdsDatabase, ‘Fleet can access database’);

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — listener.connections.allowDefaultPortFromAnyIpv4(‘Allow public access’);

== fleet.connections.allowToDefaultPort(rdsDatabase, ‘Fleet can access database’);

Python:: + [source,python,subs=“verbatim,attributes”] — listener.connections.allow\_default\_port\_from\_any\_ipv4(“Allow public access”)

== fleet.connections.allow\_to\_default\_port(rds\_database, “Fleet can access database”)

Java:: + [source,java,subs=“verbatim,attributes”] — listener.getConnections().allowDefaultPortFromAnyIpv4(“Allow public access”);

== fleet.getConnections().AllowToDefaultPort(rdsDatabase, “Fleet can access database”);

C#:: + [source,csharp,subs=“verbatim,attributes”] — listener.Connections.AllowDefaultPortFromAnyIpv4(“Allow public access”);

== fleet.Connections.AllowToDefaultPort(rdsDatabase, “Fleet can access database”);

Go:: + [source,go,subs=“verbatim,attributes”] — listener.Connections().AllowDefaultPortFromAnyIpv4(jsii.String(“Allow public Access”)) fleet.Connections().AllowToDefaultPort(rdsDatabase, jsii.String(“Fleet can access database”)) — ====

[#resources-events] == Event handling

Some resources can act as event sources. Use the addEventNotification method (Python: add\_event\_notification) to register an event target to a particular event type emitted by the resource. In addition to this, addXxxNotification methods offer a simple way to register a handler for common event types.

The following example shows how to trigger a Lambda function when an object is added to an Amazon S3 bucket.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — import \* as s3nots from ‘@aws-cdk/aws-s3-notifications’;

const handler = new lambda.Function(this, ‘Handler’, { /*…*/ }); const bucket = new s3.Bucket(this, ‘Bucket’); bucket.addObjectCreatedNotification(new s3nots.LambdaDestination(handler)); —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const s3nots = require(‘@aws-cdk/aws-s3-notifications’);

const handler = new lambda.Function(this, ‘Handler’, { /*…*/ }); const bucket = new s3.Bucket(this, ‘Bucket’); bucket.addObjectCreatedNotification(new s3nots.LambdaDestination(handler)); —

Python:: + [source,python,subs=“verbatim,attributes”] — import aws\_cdk.aws\_s3\_notifications as s3\_nots

handler = lambda\_.Function(self, “Handler”, …) bucket = s3.Bucket(self, “Bucket”) bucket.add\_object\_created\_notification(s3\_nots.LambdaDestination(handler)) —

Java:: + [source,java,subs=“verbatim,attributes”] — import software.amazon.awscdk.services.s3.Bucket; import software.amazon.awscdk.services.lambda.Function; import software.amazon.awscdk.services.s3.notifications.LambdaDestination;

Function handler = Function.Builder.create(this, “Handler”)/\* … \*/.build(); Bucket bucket = new Bucket(this, “Bucket”); bucket.addObjectCreatedNotification(new LambdaDestination(handler)); —

C#:: + [source,csharp,subs=“verbatim,attributes”] — using lambda = Amazon.CDK.{aws}.Lambda; using s3 = Amazon.CDK.{aws}.S3; using s3Nots = Amazon.CDK.{aws}.S3.Notifications;

var handler = new lambda.Function(this, “Handler”, new lambda.FunctionProps { .. }); var bucket = new s3.Bucket(this, “Bucket”); bucket.AddObjectCreatedNotification(new s3Nots.LambdaDestination(handler)); —

Go:: + [source,go,subs=“verbatim,attributes”] — import ( “github.com/aws/aws-cdk-go/awscdk/v2” “github.com/aws/jsii-runtime-go” s3 “github.com/aws/aws-cdk-go/awscdk/v2/awss3” s3nots “github.com/aws/aws-cdk-go/awscdk/v2/awss3notifications”  
)

handler := lambda.NewFunction(this, jsii.String(“MyFunction”), &lambda.FunctionProps{}) bucket := s3.NewBucket(this, jsii.String(“Bucket”), &s3.BucketProps{}) bucket.AddObjectCreatedNotification(s3nots.NewLambdaDestination(handler), nil) — ====

[#resources-removal] == Removal policies

Resources that maintain persistent data, such as databases, Amazon S3 buckets, and Amazon ECR registries, have a *removal policy*. The removal policy indicates whether to delete persistent objects when the {aws} CDK stack that contains them is destroyed. The values specifying the removal policy are available through the RemovalPolicy enumeration in the {aws} CDK core module.

= [NOTE]

Resources besides those that store data persistently might also have a removalPolicy that is used for a different purpose. For example, a Lambda function version uses a removalPolicy attribute to determine whether a given version is retained when a new version is deployed. These have different meanings and defaults compared to the removal policy on an Amazon S3 bucket or DynamoDB table.

====

[cols=“1,1”, options=“header”] |=== |Value |Meaning

|=== | RemovalPolicy.RETAIN | Keep the contents of the resource when destroying the stack (default). The resource is orphaned from the stack and must be deleted manually. If you attempt to re-deploy the stack while the resource still exists, you will receive an error message due to a name conflict. |===

|=== | RemovalPolicy.DESTROY | The resource will be destroyed along with the stack. |===

{aws} CloudFormation does not remove Amazon S3 buckets that contain files even if their removal policy is set to DESTROY. Attempting to do so is an {aws} CloudFormation error. To have the {aws} CDK delete all files from the bucket before destroying it, set the bucket’s autoDeleteObjects property to true.

Following is an example of creating an Amazon S3 bucket with RemovalPolicy of DESTROY and autoDeleteOjbects set to true.

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — import \* as cdk from ‘@aws-cdk/core’; import \* as s3 from ‘@aws-cdk/aws-s3’;

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

const bucket = new s3.Bucket(this, ‘Bucket’, { removalPolicy: cdk.RemovalPolicy.DESTROY, autoDeleteObjects: true }); } } —-

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const cdk = require(‘@aws-cdk/core’); const s3 = require(‘@aws-cdk/aws-s3’);

class CdkTestStack extends cdk.Stack { constructor(scope, id, props) { super(scope, id, props);

const bucket = new s3.Bucket(this, ‘Bucket’, { removalPolicy: cdk.RemovalPolicy.DESTROY, autoDeleteObjects: true }); } }

== module.exports = { CdkTestStack }

Python:: + [source,python,subs=“verbatim,attributes”] — import aws\_cdk.core as cdk import aws\_cdk.aws\_s3 as s3

class CdkTestStack(cdk.stack): def *init*(self, scope: cdk.Construct, id: str, **kwargs): super().*init*(scope, id,** kwargs)

bucket = s3.Bucket(self, "Bucket",  
 removal\_policy=cdk.RemovalPolicy.DESTROY,  
 auto\_delete\_objects=True) ----

Java:: + [source,java,subs=“verbatim,attributes”] — software.amazon.awscdk.core.\_; import software.amazon.awscdk.services.s3.\_;

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

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

Bucket.Builder.create(this, "Bucket")  
 .removalPolicy(RemovalPolicy.DESTROY)  
 .autoDeleteObjects(true).build();

} } —- ….

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

public CdkTestStack(Construct scope, string id, IStackProps props) : base(scope, id, props) { new Bucket(this, “Bucket”, new BucketProps { RemovalPolicy = RemovalPolicy.DESTROY, AutoDeleteObjects = true }); } —

Go:: + [source,go,subs=“verbatim,attributes”] — import ( “github.com/aws/aws-cdk-go/awscdk/v2” “github.com/aws/jsii-runtime-go” s3 “github.com/aws/aws-cdk-go/awscdk/v2/awss3” )

s3.NewBucket(this, jsii.String(“Bucket”), &s3.BucketProps{ RemovalPolicy: awscdk.RemovalPolicy\_DESTROY, AutoDeleteObjects: jsii.Bool(true), }) — ====

You can also apply a removal policy directly to the underlying {aws} CloudFormation resource via the applyRemovalPolicy() method. This method is available on some stateful resources that do not have a removalPolicy property in their L2 resource’s props. Examples include the following:

* {aws} CloudFormation stacks
* Amazon Cognito user pools
* Amazon DocumentDB database instances
* Amazon EC2 volumes
* Amazon OpenSearch Service domains
* Amazon FSx file systems
* Amazon SQS queues

==== [role=“tablist”] TypeScript:: + [source,javascript,subs=“verbatim,attributes”] — const resource = bucket.node.findChild(‘Resource’) as cdk.CfnResource; resource.applyRemovalPolicy(cdk.RemovalPolicy.DESTROY); —

JavaScript:: + [source,javascript,subs=“verbatim,attributes”] — const resource = bucket.node.findChild(‘Resource’); resource.applyRemovalPolicy(cdk.RemovalPolicy.DESTROY); —

Python:: + [source,python,subs=“verbatim,attributes”] — resource = bucket.node.find\_child(‘Resource’) resource.apply\_removal\_policy(cdk.RemovalPolicy.DESTROY); —

Java:: + [source,java,subs=“verbatim,attributes”] — CfnResource resource = (CfnResource)bucket.node.findChild(“Resource”); resource.applyRemovalPolicy(cdk.RemovalPolicy.DESTROY); —

C#:: + [source,csharp,subs=“verbatim,attributes”] —

var resource = (CfnResource)bucket.node.findChild(‘Resource’); resource.ApplyRemovalPolicy(cdk.RemovalPolicy.DESTROY); — ====

= [NOTE]

The {aws} CDK’s RemovalPolicy translates to {aws} CloudFormation’s DeletionPolicy. However, the default in {aws} CDK is to retain the data, which is the opposite of the {aws} CloudFormation default.

====

include::attributes.txt[]

// Attributes [.topic] [#security-iam] = Identity and access management for the {aws} Cloud Development Kit ({aws} CDK) :info\_titleabbrev: Identity and access management

== [abstract]

## Provides information about identity and access management for the {aws} CDK.

// Content start

{aws} Identity and Access Management (IAM) is an {aws} service that helps an administrator securely control access to {aws} resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use {aws} resources. IAM is an {aws} service that you can use with no additional charge.

[#security-iam-audience] == Audience

How you use {aws} Identity and Access Management (IAM) differs, depending on the work that you do in {aws}.

*Service user* – If you use {aws} services to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more {aws} features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator.

*Service administrator* – If you’re in charge of {aws} resources at your company, you probably have full access to {aws} resources. It’s your job to determine which {aws} services and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM.

*IAM administrator* – If you’re an IAM administrator, you might want to learn details about how you can write policies to manage access to {aws} services.

[#security-iam-authentication] == Authenticating with identities

Authentication is how you sign in to {aws} using your identity credentials. You must be *authenticated* (signed in to {aws}) as the {aws} account root user, as an IAM user, or by assuming an IAM role.

You can sign in to {aws} as a federated identity by using credentials provided through an identity source. {aws} IAM Identity Center (IAM Identity Center) users, your company’s single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access {aws} by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the {aws} Management Console or the {aws} access portal. For more information about signing in to {aws}, see link:https://docs.aws.amazon.com/signin/latest/userguide/how-to-sign-in.html[How to sign in to your {aws} account] in the *{aws} Sign-In User Guide*.

To access {aws} programmatically, {aws} provides the {aws} CDK, software development kits (SDKs), and a command line interface (CLI) to cryptographically sign your requests using your credentials. If you don’t use {aws} tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see link:https://docs.aws.amazon.com/general/latest/gr/signature-version-4.html[Signature Version 4 signing process] in the *{aws} General Reference*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, {aws} recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see https://docs.aws.amazon.com/singlesignon/latest/userguide/enable-mfa.html[Multi-factor authentication] in the *{aws} IAM Identity Center User Guide* and link:https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_credentials\_mfa.html[Using multi-factor authentication (MFA) in {aws}] in the *IAM User Guide*.

[#security-iam-authentication-rootuser] === {aws} account root user

When you create an {aws} account, you begin with one sign-in identity that has complete access to all {aws} services and resources in the account. This identity is called the {aws} account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don’t use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see link:https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_root-user.html#root-user-tasks[Tasks that require root user credentials] in the *IAM User Guide*.

[#security-iam-authentication-federated] === Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access {aws} services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the {aws} Directory Service, the Identity Center directory, or any user that accesses {aws} services by using credentials provided through an identity source. When federated identities access {aws} accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use {aws} IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your {aws} accounts and applications. For information about IAM Identity Center, see link:https://docs.aws.amazon.com/singlesignon/latest/userguide/what-is.html[What is IAM Identity Center?] in the *{aws} IAM Identity Center User Guide*.

[#security-iam-authentication-iamuser] === IAM users and groups

An *link:https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_users.html[IAM user]* is an identity within your {aws} account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see link:https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#rotate-credentials[Rotate access keys regularly for use cases that require long-term credentials] in the *IAM User Guide*.

An link:https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_groups.html[IAM group] is an identity that specifies a collection of IAM users. You can’t sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see link:https://docs.aws.amazon.com/IAM/latest/UserGuide/gs-identities-iam-users.html[Use cases for IAM users] in the *IAM User Guide*.

[#security-iam-authentication-iamrole] === IAM roles

An *link:https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_roles.html[IAM role]* is an identity within your {aws} account that has specific permissions. It’s similar to an IAM user, but isn’t associated with a specific person. You can temporarily assume an IAM role in the {aws} Management Console by link:https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_roles\_use\_switch-role-console.html[switching roles]. You can assume a role by calling an {aws} CLI or {aws} API operation or by using a custom URL. For more information about methods for using roles, see link:https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_roles\_use.html[Using IAM roles] in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

* *Federated user access* – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see link:https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_roles\_create\_for-idp.html[Create a role for a third-party identity provider (federation)] in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see https://docs.aws.amazon.com/singlesignon/latest/userguide/permissionsetsconcept.html[Permission sets] in the *{aws} IAM Identity Center User Guide*.
* *Temporary IAM user permissions* – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
* *Cross-account access* – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some {aws} services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see link:https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_roles\_compare-resource-policies.html[How IAM roles differ from resource-based policies] in the *IAM User Guide*.
* *Cross-service access* – Some {aws} services use features in other {aws} services. For example, when you make a call in a service, it’s common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal’s permissions, using a service role, or using a service-linked role.

|  |
| --- |
| \*\* *Service role* – A service role is an link:https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_roles.html[IAM role] that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see link:https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_roles\_create\_for-service.html[Create a role to delegate permissions to an {aws} service] in the *IAM User Guide*. |
| \*\* *Service-linked role* – A service-linked role is a type of service role that is linked to an {aws} service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your {aws} account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles. |

* *Applications running on Amazon EC2* – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making {aws} CLI or {aws} API requests. This is preferable to storing access keys within the EC2 instance. To assign an {aws} role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see link:https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_roles\_use\_switch-role-ec2.html[Use an IAM role to grant permissions to applications running on Amazon EC2 instances] in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see link:https://docs.aws.amazon.com/IAM/latest/UserGuide/id.html#id\_which-to-choose\_role[When to create an IAM role (instead of a user)] in the *IAM User Guide*.

include::attributes.txt[]

// Attributes [.topic] [#security] = Security for the {aws} Cloud Development Kit ({aws} CDK) :info\_titleabbrev: Security

== [abstract]

## Provides security-related information for the {aws} CDK.

// Content start

Cloud security at Amazon Web Services ({aws}) is the highest priority. As an {aws} customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations. Security is a shared responsibility between {aws} and you. The link:https://aws.amazon.com/compliance/shared-responsibility-model/[Shared Responsibility Model] describes this as Security of the Cloud and Security in the Cloud.

*Security of the Cloud* – {aws} is responsible for protecting the infrastructure that runs all of the services offered in the {aws} Cloud and providing you with services that you can use securely. Our security responsibility is the highest priority at {aws}, and the effectiveness of our security is regularly tested and verified by third-party auditors as part of the https://aws.amazon.com/compliance/programs/[{aws} Compliance Programs].

*Security in the Cloud* – Your responsibility is determined by the {aws} service you are using, and other factors including the sensitivity of your data, your organization’s requirements, and applicable laws and regulations.

The {aws} CDK follows the link:https://aws.amazon.com/compliance/shared-responsibility-model/[shared responsibility model] through the specific Amazon Web Services ({aws}) services it supports. For {aws} service security information, see the link:https://docs.aws.amazon.com/security/?id=docs\_gateway#aws-security[{aws} service security documentation page] and https://aws.amazon.com/compliance/services-in-scope/[{aws} services that are in scope of {aws} compliance efforts by compliance program].

include::security-iam.adoc[leveloffset=+1]

include::compliance-validation.adoc[leveloffset=+1]

include::disaster-recovery-resiliency.adoc[leveloffset=+1]

include::infrastructure-security.adoc[leveloffset=+1]

include::attributes.txt[]

// Attributes

[.topic] [#tools] = Use other tools with the {aws} CDK :doctype: book :info\_titleabbrev: Use tools with the CDK

== [abstract]

## Tools for the {aws} CDK

// Content start

You can use other tools with the {aws} CDK to improve your development workflows.

[#vscode] == {aws} Toolkit for Visual Studio Code

The https://aws.amazon.com/visualstudiocode/[{aws} Toolkit for Visual Studio Code] is an open source plugin for Visual Studio Code that makes it easier to create, debug, and deploy applications on {aws}. The toolkit provides an integrated experience for developing {aws} CDK applications. It includes the {aws} CDK Explorer feature to list your {aws} CDK projects and browse the various components of the CDK application. https://docs.aws.amazon.com/toolkit-for-vscode/latest/userguide/setup-toolkit.html[Install the {aws} Toolkit] and learn more about https://docs.aws.amazon.com/toolkit-for-vscode/latest/userguide/cdk-explorer.html[using the {aws} CDK Explorer].

[#sam] == {aws} SAM integration

Use the {aws} CDK and the {aws} Serverless Application Model ({aws} SAM) together to locally build and test serverless applications defined in the CDK. For complete information, see https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-cdk.html[{aws} Cloud Development Kit ({aws} CDK)] in the *{aws} Serverless Application Model Developer Guide*. To install the {aws} SAM CLI, see https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-sam-cli-install.html[Install the {aws} SAM CLI].

:doctype: book

include::attributes.txt[]

// Attributes

[.topic] [#troubleshooting] = Troubleshooting common {aws} CDK issues :info\_titleabbrev: {aws} CDK troubleshooting

// Content start

This topic describes how to troubleshoot the following issues with the {aws} CDK.

* xref:troubleshooting-toolkit[After updating the {aws} CDK the {aws} CDK Toolkit (CLI) reports a mismatch with the {aws} Construct Library]
* xref:troubleshooting-nobucket[When deploying my {aws} CDK stack I receive a NoSuchBucket error]
* xref:troubleshooting-forbidden-null[When deploying my {aws} CDK stack I receive a forbidden: null message]
* xref:troubleshooting-app-required[When synthesizing an {aws} CDK stack I get the message –app is required either in command-line in cdk.json or in ~/.cdk.json]
* xref:troubleshooting-resource-count[When synthesizing an {aws} CDK stack I receive an error because the {aws} CloudFormation template contains too many resources]
* xref:troubleshooting-availability-zones[I specified three (or more) Availability Zones for my Auto Scaling group or VPC but it was only deployed in two]
* xref:troubleshooting-resource-not-deleted[My S3 bucket DynamoDB table or other resource is not deleted when I issue cdk destroy]

[#troubleshooting-toolkit] == After updating the {aws} CDK, the {aws} CDK Toolkit (CLI) reports a mismatch with the {aws} Construct Library

The version of the {aws} CDK Toolkit (which provides the cdk command) must be at least equal to the version of the main {aws} Construct Library module, aws-cdk-lib. The Toolkit is intended to be backward compatible. The latest 2.x version of the toolkit can be used with any 1.x or 2.x release of the library. For this reason, we recommend you install this component globally and keep it up to date.

== [source,none,subs=“verbatim,attributes”]

## npm update -g aws-cdk

If you need to work with multiple versions of the {aws} CDK Toolkit, install a specific version of the toolkit locally in your project folder.

If you are using TypeScript or JavaScript, your project directory already contains a versioned local copy of the CDK Toolkit.

If you are using another language, use npm to install the {aws} CDK Toolkit, omitting the -g flag and specifying the desired version. For example:

== [source,none,subs=“verbatim,attributes”]

## npm install aws-cdk@2.0

To run a locally installed {aws} CDK Toolkit, use the command npx aws-cdk instead of only cdk. For example:

== [source,none,subs=“verbatim,attributes”]

## npx aws-cdk deploy MyStack

npx aws-cdk runs the local version of the {aws} CDK Toolkit if one exists. It falls back to the global version when a project doesn’t have a local installation. You may find it convenient to set up a shell alias to make sure cdk is always invoked this way.

==== [role=“tablist”] macOS/Linux:: + [source,none,subs=“verbatim,attributes”] — alias cdk=“npx aws-cdk” —

Windows:: + [source,none,subs=“verbatim,attributes”] — doskey cdk=npx aws-cdk $\* — ====

[#troubleshooting-nobucket] == When deploying my {aws} CDK stack, I receive a NoSuchBucket error

Your {aws} environment has not been bootstrapped, and so does not have an Amazon S3 bucket to hold resources during deployment. You can create the staging bucket and other required resources with the following command:

== [source,none,subs=“verbatim,attributes”]

## cdk bootstrap aws://ACCOUNT-NUMBER/REGION

To avoid generating unexpected {aws} charges, the {aws} CDK does not automatically bootstrap any environment. You must explicitly bootstrap each environment into which you will deploy.

By default, the bootstrap resources are created in the Region or Regions that are used by stacks in the current {aws} CDK application. Alternatively, they are created in the Region specified in your local {aws} profile (set by aws configure), using that profile’s account. You can specify a different account and Region on the command line as follows. (You must specify the account and Region if you are not in an app’s directory.)

== [source,none,subs=“verbatim,attributes”]

## cdk bootstrap aws://ACCOUNT-NUMBER/REGION

For more information, see xref:bootstrapping[{aws} CDK bootstrapping].

[#troubleshooting-forbidden-null] == When deploying my {aws} CDK stack, I receive a forbidden: null message

You are deploying a stack that requires bootstrap resources, but are using an IAM role or account that lacks permission to write to it. (The staging bucket is used when deploying stacks that contain assets or that synthesize an {aws} CloudFormation template larger than 50K.) Use an account or role that has permission to perform the action s3:\* against the bucket mentioned in the error message.

[#troubleshooting-app-required] == When synthesizing an {aws} CDK stack, I get the message --app is required either in command-line, in cdk.json or in ~/.cdk.json

This message usually means that you aren’t in the main directory of your {aws} CDK project when you issue cdk synth. The file cdk.json in this directory, created by the cdk init command, contains the command line needed to run (and thereby synthesize) your {aws} CDK app. For a TypeScript app, for example, the default cdk.json looks something like this:

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

{ “app”: “npx ts-node bin/my-cdk-app.ts” } —

We recommend issuing cdk commands only in your project’s main directory, so the {aws} CDK toolkit can find cdk.json there and successfully run your app.

If this isn’t practical for some reason, the {aws} CDK Toolkit looks for the app’s command line in two other locations:

* In cdk.json in your home directory
* On the cdk synth command itself using the -a option

For example, you might synthesize a stack from a TypeScript app as follows.

== [source,none,subs=“verbatim,attributes”]

## cdk synth –app “npx ts-node my-cdk-app.ts” MyStack

[#troubleshooting-resource-count] == When synthesizing an {aws} CDK stack, I receive an error because the {aws} CloudFormation template contains too many resources

The {aws} CDK generates and deploys {aws} CloudFormation templates. {aws} CloudFormation has a hard limit on the number of resources a stack can contain. With the {aws} CDK, you can run up against this limit more quickly than you might expect.

= [NOTE]

The {aws} CloudFormation resource limit is 500 at this writing. See https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cloudformation-limits.html[{aws} CloudFormation quotas] for the current resource limit.

====

The {aws} Construct Library’s higher-level, intent-based constructs automatically provision any auxiliary resources that are needed for logging, key management, authorization, and other purposes. For example, granting one resource access to another generates any IAM objects needed for the relevant services to communicate.

In our experience, real-world use of intent-based constructs results in 1–5 {aws} CloudFormation resources per construct, though this can vary. For serverless applications, 5–8 {aws} resources per API endpoint is typical.

Patterns, which represent a higher level of abstraction, let you define even more {aws} resources with even less code. The {aws} CDK code in xref:ecs-example[Example: Create an {aws} Fargate service using the {aws} CDK], for example, generates more than 50 {aws} CloudFormation resources while defining only three constructs!

Exceeding the {aws} CloudFormation resource limit is an error during {aws} CloudFormation synthesis. The {aws} CDK issues a warning if your stack exceeds 80% of the limit. You can use a different limit by setting the maxResources property on your stack, or disable validation by setting maxResources to 0.

= [TIP]

You can get an exact count of the resources in your synthesized output using the following utility script. (Since every {aws} CDK developer needs Node.js, the script is written in JavaScript.)

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

// rescount.js - count the resources defined in a stack // invoke with: node rescount.js ++++++// e.g. node rescount.js cdk.out/MyStack.template.json++++++

import \* as fs from ‘fs’; const path = process.argv[2];

if (path) fs.readFile(path, ‘utf8’, function(err, contents) { console.log(err ? +${err}+ : +${Object.keys(JSON.parse(contents).Resources).length} resources defined in ${path}+); }); else console.log(“Please specify the path to the stack’s output .json file”); — ====

As your stack’s resource count approaches the limit, consider re-architecting to reduce the number of resources your stack contains: for example, by combining some Lambda functions, or by breaking your stack into multiple stacks. The CDK supports xref:resource-stack[references between stacks], so you can separate your app’s functionality into different stacks in whatever way makes the most sense to you.

= [NOTE]

{aws} CloudFormation experts often suggest the use of nested stacks as a solution to the resource limit. The {aws} CDK supports this approach via the xref:stack-nesting[NestedStack] construct.

====

[#troubleshooting-availability-zones] == I specified three (or more) Availability Zones for my Auto Scaling group or VPC, but it was only deployed in two

To get the number of Availability Zones that you request, specify the account and Region in the stack’s env property. If you do not specify both, the {aws} CDK, by default, synthesizes the stack as environment-agnostic. You can then deploy the stack to a specific Region using {aws} CloudFormation. Because some Regions have only two Availability Zones, an environment-agnostic template doesn’t use more than two.

= [NOTE]

In the past, Regions have occasionally launched with only one Availability Zone. Environment-agnostic {aws} CDK stacks cannot be deployed to such Regions. At this writing, however, all {aws} Regions have at least two AZs.

====

You can change this behavior by overriding your stack’s link:https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.Stack.html#availabilityzones[availabilityZones] (Python: availability\_zones) property to explicitly specify the zones that you want to use.

For more information about specifying a stack’s account and region at synthesis time, while retaining the flexibility to deploy to any region, see xref:environments[Environments for the {aws} CDK].

[#troubleshooting-resource-not-deleted] == My S3 bucket, DynamoDB table, or other resource is not deleted when I issue cdk destroy

By default, resources that can contain user data have a removalPolicy (Python: removal\_policy) property of RETAIN, and the resource is not deleted when the stack is destroyed. Instead, the resource is orphaned from the stack. You must then delete the resource manually after the stack is destroyed. Until you do, redeploying the stack fails. This is because the name of the new resource being created during deployment conflicts with the name of the orphaned resource.

If you set a resource’s removal policy to DESTROY, that resource will be deleted when the stack is destroyed.

==== [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 CdkTestStack extends cdk.Stack { constructor(scope: Construct, id: string, props?: cdk.StackProps) { super(scope, id, props);

const bucket = new s3.Bucket(this, ‘Bucket’, { removalPolicy: cdk.RemovalPolicy.DESTROY, }); } } —-

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

class CdkTestStack extends cdk.Stack { constructor(scope, id, props) { super(scope, id, props);

const bucket = new s3.Bucket(this, ‘Bucket’, { removalPolicy: cdk.RemovalPolicy.DESTROY }); } }

== module.exports = { CdkTestStack }

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

class CdkTestStack(cdk.stack): def *init*(self, scope: Construct, id: str, **kwargs): super().*init*(scope, id,** kwargs)

bucket = s3.Bucket(self, "Bucket",  
 removal\_policy=cdk.RemovalPolicy.DESTROY) ----

Java:: + [source,java,subs=“verbatim,attributes”] — software.amazon.awscdk.\_; import software.amazon.awscdk.services.s3.\_; import software.constructs;

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

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

Bucket.Builder.create(this, "Bucket")  
 .removalPolicy(RemovalPolicy.DESTROY).build();

} } —- ….

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

public CdkTestStack(Construct scope, string id, IStackProps props) : base(scope, id, props) { new Bucket(this, “Bucket”, new BucketProps { RemovalPolicy = RemovalPolicy.DESTROY }); } — ====

= [NOTE]

{aws} CloudFormation cannot delete a non-empty Amazon S3 bucket. If you set an Amazon S3 bucket’s removal policy to DESTROY, and it contains data, attempting to destroy the stack will fail because the bucket cannot be deleted. You can have the {aws} CDK delete the objects in the bucket before attempting to destroy it by setting the bucket’s autoDeleteObjects prop to true.

====