:doctype: book

include::attributes.txt[]

// Attributes [.topic] [#migrate] = Migrate existing resources and {aws} CloudFormation templates to the {aws} CDK :info\_titleabbrev: Migrate to the {aws} CDK :keywords: {aws} CDK, {aws} CDK CLI, {aws} CloudFormation, Migrate, {aws} resources, Infrastructure as Code, IaC

== [abstract]

## Use the {aws} Cloud Development Kit ({aws} CDK) Command Line Interface ({aws} CDK CLI) to migrate deployed {aws} resources, deployed {aws} CloudFormation stacks, and local {aws} CloudFormation templates to {aws} CDK.

// Content start

[cols=“1”, frame=“all”] |===

|=== | The CDK Migrate feature is in preview release for {aws} CDK and is subject to change. |===

Use the {aws} Cloud Development Kit ({aws} CDK) Command Line Interface ({aws} CDK CLI) to migrate deployed {aws} resources, deployed {aws} CloudFormation stacks, and local {aws} CloudFormation templates to {aws} CDK.

[#migrate-intro] == How migration works

Use the {aws} CDK CLI cdk migrate command to migrate from the following sources:

–

* Deployed {aws} resources.
* Deployed {aws} CloudFormation stacks.
* {blank}
* == Local {aws} CloudFormation templates.

*Deployed {aws} resources*:: + You can migrate deployed {aws} resources from a specific environment ({aws} account and {aws} Region) that are not associated with an {aws} CloudFormation stack. + The {aws} CDK CLI utilizes the *IaC generator* service to scan for resources in your {aws} environment to gather resource details. To learn more about IaC generator, see https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/generate-IaC.html[Generating templates for existing resources] in the *{aws} CloudFormation User Guide*. + After gathering resource details, the {aws} CDK CLI creates a new CDK app that includes a single stack containing your migrated resources.

*Deployed {aws} CloudFormation stacks*:: + You can migrate a single {aws} CloudFormation stack into a new {aws} CDK app. The {aws} CDK CLI will retrieve the {aws} CloudFormation template of your stack and create a new CDK app. The CDK app will consist of a single stack that contains your migrated {aws} CloudFormation stack.

*Local {aws} CloudFormation templates*:: + You can migrate from a local {aws} CloudFormation template. Local templates may or may not contain deployed resources. The {aws} CDK CLI will create a new CDK app that contains a single stack with your resources. + After migrating, you can manage, modify, and deploy your CDK app to {aws} CloudFormation to provision or update your resources.

[#migrate-benefits] == Benefits of CDK Migrate

Migrating resources into {aws} CDK has historically been a manual process that requires time and expertise with {aws} CloudFormation and {aws} CDK to even begin. With CDK Migrate, the {aws} CDK CLI facilitates a majority of the migration effort for you in a fraction of the time. CDK Migrate will quickly get you started with using the {aws} CDK to develop and manage new and existing applications on {aws}.

[#migrate-considerations] == Considerations

[#migrate-considerations-general] === General considerations

*CDK Migrate vs. CDK Import*:: + The cdk import command can import deployed resources into a new or existing CDK app. When importing, each resource will have to manually be defined as an L1 construct in your app. We recommend using cdk import to import one or more resources at a time into a new or existing CDK app. To learn more, see xref:cli-import[Import existing resources into a stack]. + The cdk migrate command migrates from deployed resources, deployed {aws} CloudFormation stacks, or local {aws} CloudFormation templates into a new CDK app. During migration, the {aws} CDK CLI uses cdk import to import your resources into the new CDK app. The {aws} CDK CLI also generates L1 constructs for each resource for you. We recommend using cdk migrate when importing from a supported migration source into a new {aws} CDK app.

*CDK Migrate creates L1 constructs only*:: + The newly created CDK app will include L1 constructs only. You can add higher-level constructs to your app after migration.

*CDK Migrate creates CDK apps that contain a single stack*:: + The newly created CDK app will contain a single stack. + When migrating deployed resources, all migrated resources will be contained within a single stack in the new CDK app. + When migrating {aws} CloudFormation stacks, you can only migrate a single {aws} CloudFormation stack into a single stack in the new CDK app.

*Migrating assets*:: + Project assets, such as {aws} Lambda code, will not directly migrate into the new CDK app. After migration, you can specify asset values to include them in the CDK app.

*Migrating stateful resources*:: + When migrating stateful resources, such as databases and Amazon Simple Storage Service (Amazon S3) buckets, you’d most often want to migrate the existing resource instead of creating a new resource. + To migrate and preserve stateful resources, do the following: + – \*\* Verify that your stateful resource supports import. For more information, see https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/resource-import-supported-resources.html[Resource type support] in the *{aws} CloudFormation User Guide*. \*\* After migration, verify that the migrated resource`’s logical ID in the new CDK app matches the logical ID of the deployed resource. \*\* If migrating from an {aws} CloudFormation stack, verify that the stack name in the new CDK app matches the {aws} CloudFormation stack. \*\* Deploy the CDK app using the same {aws} account and {aws} Region of the migrated resource. –

[#migrate-considerations-template] === Considerations when migrating from an {aws} CloudFormation template

*CDK Migrate supports single template migration*:: + When migrating {aws} CloudFormation templates, you can select a single template to migrate. Nested templates are not supported.

*Migrating templates with intrinsic functions*:: + When migrating from an {aws} CloudFormation template that uses intrinsic functions, the {aws} CDK CLI will attempt to migrate your logic into the CDK app with the Fn class. To learn more, see https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.Fn.html[class Fn] in the *{aws} Cloud Development Kit ({aws} CDK) API Reference*.

[#migrate-considerations-resources] === Considerations when migrating from deployed resources

*Scan limitations*:: + When scanning your environment for resources, IaC generator has specific limitations on the data it can retrieve and quota limitations when scanning. To learn more, see link:https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/generate-IaC.html#generate-template-considerations[Considerations] in the *{aws} CloudFormation User Guide*.

[#migrate-prerequisites] == Prerequisites

Before using the cdk migrate command, complete all set up steps in xref:getting-started[Getting started with the {aws} CDK].

[#migrate-gs] == Get started with CDK Migrate

To begin, run the {aws} CDK CLI cdk migrate command from a directory of your choice. Provide required and optional options, depending on the type of migration you are performing.

For a full list and description of options that you can use with cdk migrate, see xref:ref-cli-cdk-migrate[cdk migrate].

The following are some important options that you may want to provide.

*Stack name*:: + The only required option is --stack-name. Use this option to specify a name for the stack that will be created within the {aws} CDK app after migration. The stack name will also be used as the name of your {aws} CloudFormation stack at deployment.

*Language*:: + Use --language to specify the programming language of the new CDK app.

*{aws} account and {aws} Region*:: + The {aws} CDK CLI retrieves {aws} account and {aws} Region information from default sources. For more information, see xref:environments[Environments for the {aws} CDK]. You can use --account and --region options with cdk migrate to provide other values.

*Output directory of your new CDK project*:: + By default, the {aws} CDK CLI will create a new CDK project in your working directory and use the value you provide with --stack-name to name the project folder. If a folder with the same name currently exists, the {aws} CDK CLI will overwrite that folder. + You can specify a different output path for the new CDK project folder with the --output-path option.

*Migration source*:: + Provide an option to specify the source you are migrating from. + –

* --from-path – Migrate from a local {aws} CloudFormation template.
* --from-scan – Migrate from deployed resources in an {aws} account and {aws} Region.
* {blank}
* == --from-stack – Migrate from an {aws} CloudFormation stack.
* Depending on your migration source, you can provide additional options to customize the cdk migrate command.

[#migrate-stack] == Migrate from an {aws} CloudFormation stack

To migrate from a deployed {aws} CloudFormation stack, provide the --from-stack option. Provide the name of your deployed {aws} CloudFormation stack with --stack-name. The following is an example:

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

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

The {aws} CDK CLI will do the following:

. Retrieve the {aws} CloudFormation template of your deployed stack. . Run cdk init to initialize a new CDK app. . Create a stack within the CDK app that contains your migrated {aws} CloudFormation stack.

When you migrate from a deployed {aws} CloudFormation stack, the {aws} CDK CLI attempts to match deployed resource logical IDs and the deployed {aws} CloudFormation stack name to the migrated resources and stack in the new CDK app.

After migration, you can manage and modify your CDK app normally. When you deploy, {aws} CloudFormation will identify the deployment as an {aws} CloudFormation stack update due to the matching {aws} CloudFormation stack name. Resources with matching logical IDs will be updated. For more information on deploying, see xref:migrate-manage[Manage and deploy your CDK app].

[#migrate-template] == Migrate from an {aws} CloudFormation template

CDK Migrate supports migrating from {aws} CloudFormation templates formatted in JSON or YAML.

To migrate from a local {aws} CloudFormation template, use the --from-path option and provide a path to the local template. You must also provide the required --stack-name option. The following is an example:

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

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

The {aws} CDK CLI will do the following:

. Retrieve your local {aws} CloudFormation template. . Run cdk init to initialize a new CDK app. . Create a stack within the CDK app that contains your migrated {aws} CloudFormation template.

After migration, you can manage and modify your CDK app normally. At deployment, you have the following options:

* *Update an {aws} CloudFormation stack* – If the local {aws} CloudFormation template was previously deployed, you can update the deployed {aws} CloudFormation stack.
* *Deploy a new {aws} CloudFormation stack* – If the local {aws} CloudFormation template was never deployed, or if you want to create a new stack from a previously deployed template, you can deploy a new {aws} CloudFormation stack.

[#migrate-template-sam] === Migrate from an {aws} SAM template

To migrate from an {aws} Serverless Application Model ({aws} SAM) template, you must first convert it to an {aws} CloudFormation template or deploy to create an {aws} CloudFormation stack.

To convert an {aws} SAM template to {aws} CloudFormation, you can use the {aws} SAM CLI sam validate --debug command. You may have to set lint to false in your samconfig.toml file before running this command.

To convert to an {aws} CloudFormation stack, deploy the {aws} SAM template using the {aws} SAM CLI. Then migrate from the deployed stack.

[#migrate-resources] == Migrate from deployed resources

To migrate from deployed {aws} resources, provide the --from-scan option. You must also provide the required --stack-name option. The following is an example:

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

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

The {aws} CDK CLI will do the following:

. *Scan your account for resource and property details* – The {aws} CDK CLI utilizes IaC generator to scan your account and gather details. . *Generate an {aws} CloudFormation template* – After scanning, the {aws} CDK CLI utilizes IaC generator to create an {aws} CloudFormation template. . *Initialize a new CDK app and migrate your template* – The {aws} CDK CLI runs cdk init to initialize a new {aws} CDK app and migrates your {aws} CloudFormation template into the CDK app as a single stack.

[#migrate-resources-filters] === Use filters

By default, the {aws} CDK CLI will scan the entire {aws} environment and migrate resources up to the maximum quota limit of IaC generator. You can provide filters with the {aws} CDK CLI to specify a criteria for which resources get migrated from your account to the new CDK app. To learn more, see xref:ref-cli-cdk-migrate-options-filter[--filter].

[#migrate-resources-scan] === Scanning for resources with IaC generator

Depending on the number of resources in your account, scanning may take a few minutes. A progress bar will display during the scanning process.

[#migrate-resources-supported] *Supported resource types*:: + The {aws} CDK CLI will migrate resources supported by the IaC generator. For a full list, see https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/resource-import-supported-resources.html[Resource type support] in the *{aws} CloudFormation User Guide*.

[#migrate-resources-writeonly] === Resolve write-only properties

Some supported resources contain write-only properties. These properties can be written to, to configure the property, but can’t be read by IaC generator or {aws} CloudFormation to obtain the value. For example, a property used to specify a database password may be write-only for security reasons.

When scanning resources during migration, IaC generator will detect resources that may contain write-only properties and will categorize them into any of the following types:

* MUTUALLY\_EXCLUSIVE\_PROPERTIES – These are write-only properties for a specific resource that are interchangeable and serve a similar purpose. One of the mutually exclusive properties are required to configure your resource. For example, the S3Bucket, ImageUri, and ZipFile properties for an +{aws}::Lambda::Function+ resource are mutually exclusive write-only properties. Any one of them can be used to specify your function assets, but you must use one.
* MUTUALLY\_EXCLUSIVE\_TYPES – These are required write-only properties that accept multiple configuration types. For example, the Body property of an +{aws}::ApiGateway::RestApi+ resource accepts an object or string type.
* UNSUPPORTED\_PROPERTIES – These are write-only properties that don’t fall under the other two categories. They are either optional properties or required properties that accept an array of objects.

For more information on write-only properties and how IaC generator manages them when scanning for deployed resources and creating {aws} CloudFormation templates, see https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/generate-IaC-write-only-properties.html[IaC generator and write-only properties] in the *{aws} CloudFormation User Guide*.

After migration, you must specify write-only property values in the new CDK app. The {aws} CDK CLI will append a *Warnings* section to the CDK project’s ReadMe file to document any write-only properties that were identified by IaC generator. The following is an example:

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

= Welcome to your CDK TypeScript project

…

== Warnings

=== Write-only properties

Write-only properties are resource property values that can be written to but can’t be read by {aws} CloudFormation or CDK Migrate. For more information, see https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/generate-IaC-write-only-properties.html[IaC generator and write-only properties].

Write-only properties discovered during migration are organized here by resource ID and categorized by write-only property type. Resolve write-only properties by providing property values in your CDK app. For guidance, see https://docs.aws.amazon.com/cdk/v2/guide/migrate.html#migrate-resources-writeonly[Resolve write-only properties].

=== MyLambdaFunction

* *UNSUPPORTED\_PROPERTIES*: \*\* SnapStart/ApplyOn: Applying SnapStart setting on function resource type.Possible values: [PublishedVersions, None] This property can be replaced with other types \*\* Code/S3ObjectVersion: For versioned objects, the version of the deployment package object to use. This property can be replaced with other exclusive properties
* *MUTUALLY\_EXCLUSIVE\_PROPERTIES*: \*\* Code/S3Bucket: An Amazon S3 bucket in the same {aws} Region as your function. The bucket can be in a different {aws} account. This property can be replaced with other exclusive properties \*\* Code/S3Key: The Amazon S3 key of the deployment package. This property can be replaced with other exclusive properties —

–

* Warnings are organized under headings that identify the resource`’s logical ID that they are associated with.
* {blank}
* == Warnings are categorized by type. These types come directly from IaC generator.

*To resolve write-only properties*:: + . Identify write-only properties to resolve from the *Warnings* section of your CDK project'sReadMefile. Here, you can take note of the resources in your CDK app that may contain write-only properties and identify the write-only property types that were discovered. + .. For ```MUTUALLY\_EXCLUSIVE\_PROPERTIES+, determine which mutually exclusive property to configure in your {aws} CDK app. .. For +MUTUALLY\_EXCLUSIVE\_TYPES, determine which accepted type that you will use to configure the property. .. For UNSUPPORTED\_PROPERTIES```+, determine if the property is optional or required. Then, configure as necessary. . Use guidance from https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/generate-IaC-write-only-properties.html[IaC generator and write-only properties] to reference what the warning types mean. . In your CDK app, write-only property values to resolve will also be specified in the +Props` section of your app. Provide the correct values here. For property descriptions and guidance, you can reference the https://docs.aws.amazon.com/cdk/api/v2/docs/aws-construct-library.html[{aws} CDK API Reference]. +

The following is an example of the Props section within a migrated CDK app with two write-only properties to resolve: + [source,javascript,subs=“verbatim,attributes”] — export interface MyTestAppStackProps extends cdk.StackProps { /\*\*

* The Amazon S3 key of the deployment package. \*/ readonly lambdaFunction00asdfasdfsadf008grk1CodeS3Keym8P82: string; /\*\*
* An Amazon S3 bucket in the same {aws} Region as your function. The bucket can be in a different {aws} account. \*/ readonly lambdaFunction00asdfasdfsadf008grk1CodeS3Bucketzidw8: string; } —

Once you resolve all write-only property values, you’re ready to prepare for deployment.

[#migrate-resources-file] === The migrate.json file

The {aws} CDK CLI creates a migrate.json file in your {aws} CDK project during migration. This file contains reference information on your deployed resources. When you deploy your CDK app for the first time, the {aws} CDK CLI uses this file to reference your deployed resources, associates your resources with the new {aws} CloudFormation stack, and deletes the file.

[#migrate-manage] == Manage and deploy your CDK app

When migrating to {aws} CDK, the new CDK app may not be deployment-ready immediately. This topic describes action items to consider while managing and deploying your new CDK app.

[#migrate-manage-prepare] === Prepare for deployment

Before deploying, you must prepare your CDK app.

*Synthesize your app*:: Use the cdk synth command to synthesize the stack in your CDK app into an {aws} CloudFormation template. + If you migrated from a deployed {aws} CloudFormation stack or template, you can compare the synthesized template to the migrated template to verify resource and property values. + To learn more about cdk synth, see xref:cli-synth[Synthesize stacks].

*Perform a diff*:: If you migrated from a deployed {aws} CloudFormation stack, you can use the cdk diff command to compare with the stack in your new CDK app. + To learn more about cdk diff, see xref:cli-diff[Compare stacks].

*Bootstrap your environment*:: If you are deploying from an {aws} environment for the first time, use cdk bootstrap to prepare your environment. To learn more, see xref:bootstrapping[{aws} CDK bootstrapping].

[#migrate-manage-deploy] === Deploy your CDK app

When you deploy a CDK app, the {aws} CDK CLI utilizes the {aws} CloudFormation service to provision your resources. Resources are bundled into a single stack in the CDK app and are deployed as a single {aws} CloudFormation stack.

Depending on where you migrated from, you can deploy to create a new {aws} CloudFormation stack or update an existing {aws} CloudFormation stack.

*Deploy to create a new {aws} CloudFormation stack*:: If you migrated from deployed resources, the {aws} CDK CLI will automatically create a new {aws} CloudFormation stack at deployment. Your deployed resources will be included in the new {aws} CloudFormation stack. + If you migrated from a local {aws} CloudFormation template that was never deployed, the {aws} CDK CLI will automatically create a new {aws} CloudFormation stack at deployment. + If you migrated from a deployed {aws} CloudFormation stack or local {aws} CloudFormation template that was previously deployed, you can deploy to create a new {aws} CloudFormation stack. To create a new stack, do the following: + –

* Deploy to a new {aws} environment. This consists of using a different {aws} account or deploying to a different {aws} Region.
* {blank}
* == If you want to deploy a new stack to the same {aws} environment of the migrated stack or template, you must modify the stack name in your CDK app to a new value. You must also modify all logical IDs of resources in your CDK app. Then, you can deploy to the same environment to create a new stack and new resources.

*Deploy to update an existing {aws} CloudFormation stack*:: If you migrated from a deployed {aws} CloudFormation stack or local {aws} CloudFormation template that was previously deployed, you can deploy to update the existing {aws} CloudFormation stack. + Verify that the stack name in your CDK app matches the stack name of the deployed {aws} CloudFormation stack and deploy to the same {aws} environment.