:doctype: book

include::attributes.txt[]

// Attributes

[.topic] [#cli] = {aws} CDK CLI reference

// Content start

The {aws} Cloud Development Kit ({aws} CDK) Command Line Interface ({aws} CDK CLI), also known as the CDK Toolkit, is the primary tool for interacting with your {aws} CDK app. It executes your app, interrogates the application model you defined, and produces and deploys the {aws} CloudFormation templates generated by the {aws} CDK. It also provides other features useful for creating and working with {aws} CDK projects. This topic contains information about common use cases of the CDK CLI.

The CDK CLI is installed with the Node Package Manager. In most cases, we recommend installing it globally.

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

npm install -g aws-cdk # install latest version npm install -g aws-cdk@X.YY.Z # install specific version —

= [TIP]

If you regularly work with multiple versions of the {aws} CDK, consider installing a matching version of the CDK CLI in individual CDK projects. To do this, omit -g from the npm install command. Then use npx aws-cdk to invoke it. This runs the local version if one exists, falling back to a global version if not.

====

[#cli-commands] == CDK CLI commands

All CDK CLI commands start with cdk, which is followed by a subcommand (list, synthesize, deploy, etc.). Some subcommands have a shorter version (ls, synth, etc.) that is equivalent. Options and arguments follow the subcommand in any order.

For a description of all subcommands, options, and arguments, see xref:ref-cli-cmd[{aws} CDK CLI command reference].

[#cli-options] == Specify options and their values

Command line options begin with two hyphens (--). Some frequently used options have single-letter synonyms that begin with a single hyphen (for example, --app has a synonym -a). The order of options in an CDK CLI command is not important.

All options accept a value, which must follow the option name. The value may be separated from the name by white space or by an equals sign =. The following two options are equivalent.

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

–toolkit-stack-name MyBootstrapStack –toolkit-stack-name=MyBootstrapStack —

Some options are flags (Booleans). You may specify true or false as their value. If you do not provide a value, the value is taken to be true. You may also prefix the option name with no- to imply false.

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

= sets staging flag to true

–staging –staging=true –staging true

= sets staging flag to false

–no-staging –staging=false –staging false —

A few options, namely --context, --parameters, --plugin, --tags, and --trust, may be specified more than once to specify multiple values. These are noted as having [array] type in the CDK CLI help. For example:

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

## cdk bootstrap –tags costCenter=0123 –tags responsibleParty=jdoe

[#cli-help] == Built-in help

The CDK CLI has integrated help. You can see general help about the utility and a list of the provided subcommands by issuing:

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

## cdk –help

To see help for a particular subcommand, for example deploy, specify it before the --help flag.

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

## cdk deploy –help

Issue cdk version to display the version of the CDK CLI. Provide this information when requesting support.

[#version-reporting] == Version reporting

To gain insight into how the {aws} CDK is used, the constructs used by {aws} CDK applications are collected and reported by using a resource identified as +{aws}::CDK::Metadata+. To learn more, see xref:usage-data[Configure {aws} CDK usage data reporting].

[#cli-auth] == Authentication with {aws}

There are different ways in which you can configure programmatic access to {aws} resources, depending on the environment and the {aws} access available to you.

To choose your method of authentication and configure it for the CDK CLI, see xref:configure-access[Configure security credentials for the {aws} CDK CLI].

The recommended approach for new users developing locally, who are not given a method of authentication by their employer, is to set up {aws} IAM Identity Center. This method includes installing the {aws} CLI for ease of configuration and for regularly signing in to the {aws} access portal. If you choose this method, your environment should contain the following elements after you complete the procedure for https://docs.aws.amazon.com/sdkref/latest/guide/access-sso.html[IAM Identity Center authentication] in the *{aws} SDKs and Tools Reference Guide*:

* The {aws} CLI, which you use to start an {aws} access portal session before you run your application.
* A https://docs.aws.amazon.com/sdkref/latest/guide/file-format.html[shared {aws} config file] having a [default] profile with a set of configuration values that can be referenced from the {aws} CDK. To find the location of this file, see https://docs.aws.amazon.com/sdkref/latest/guide/file-location.html[Location of the shared files] in the *{aws} SDKs and Tools Reference Guide*.
* The shared config file sets the https://docs.aws.amazon.com/sdkref/latest/guide/feature-region.html[region] setting. This sets the default {aws} Region the {aws} CDK and CDK CLI use for {aws} requests.
* The CDK CLI uses the profile’s link:https://docs.aws.amazon.com/sdkref/latest/guide/feature-sso-credentials.html#feature-sso-credentials-profile[SSO token provider configuration] to acquire credentials before sending requests to {aws}. The sso\_role\_name value, which is an IAM role connected to an IAM Identity Center permission set, should allow access to the {aws} services used in your application.
* The following sample config file shows a default profile set up with SSO token provider configuration. The profile’s sso\_session setting refers to the named link:https://docs.aws.amazon.com/sdkref/latest/guide/file-format.html#section-session[sso-session section]. The sso-session section contains settings to initiate an {aws} access portal session.

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

* [default] sso\_session = ++++++sso\_account\_id = <111122223333> sso\_role\_name = ++++++region = ++++++output = ++++++++++++++++++++++++++++++

[sso-session ++++++] sso\_region = ++++++sso\_start\_url = <https://provided-domain.awsapps.com/start> sso\_registration\_scopes = sso:account:access —-++++++++++++

[#accessportal] === Start an {aws} access portal session

Before accessing {aws} services, you need an active {aws} access portal session for the CDK CLI to use IAM Identity Center authentication to resolve credentials. Depending on your configured session lengths, your access will eventually expire and the CDK CLI will encounter an authentication error. Run the following command in the {aws} CLI to sign in to the {aws} access portal.

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

## aws sso login

If your SSO token provider configuration is using a named profile instead of the default profile, the command is aws sso login --profile <NAME>. Also specify this profile when issuing cdk commands using the --profile option or the AWS\_PROFILE environment variable.

To test if you already have an active session, run the following {aws} CLI command.

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

## aws sts get-caller-identity

The response to this command should report the IAM Identity Center account and permission set configured in the shared config file.

= [NOTE]

If you already have an active {aws} access portal session and run aws sso login, you will not be required to provide credentials.

The sign in process may prompt you to allow the {aws} CLI access to your data. Since the {aws} CLI is built on top of the SDK for Python, permission messages may contain variations of the botocore name.

====

[#cli-environment] == Specify Region and other configuration

The CDK CLI needs to know the {aws} Region that you’re deploying into and how to authenticate with {aws}. This is needed for deployment operations and to retrieve context values during synthesis. Together, your account and Region make up the *environment*.

Region may be specified using environment variables or in configuration files. These are the same variables and files used by other {aws} tools such as the {aws} CLI and the various {aws} SDKs. The CDK CLI looks for this information in the following order.

* The AWS\_DEFAULT\_REGION environment variable.
* A named profile defined in the standard {aws} config file and specified using the --profile option on cdk commands.
* The [default] section of the standard {aws} config file.

Besides specifying {aws} authentication and a Region in the [default] section, you can also add one or more [profile <NAME>] sections, where <NAME> is the name of the profile. For more information about named profiles, see https://docs.aws.amazon.com/sdkref/latest/guide/file-format.html[Shared config and credentials files] in the *{aws} SDKs and Tools Reference Guide*.

The standard {aws} config file is located at ~/.aws/config (macOS/Linux) or %USERPROFILE%\.aws\config (Windows). For details and alternate locations, see https://docs.aws.amazon.com/sdkref/latest/guide/file-location.html[Location of the shared config and credentials files] in the *{aws} SDKs and Tools Reference Guide*

The environment that you specify in your {aws} CDK app by using the stack’s env property is used during synthesis. It’s used to generate an environment-specific {aws} CloudFormation template, and during deployment, it overrides the account or Region specified by one of the preceding methods. For more information, see xref:environments[Environments for the {aws} CDK].

= [NOTE]

The {aws} CDK uses credentials from the same source files as other {aws} tools and SDKs, including the https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html[{aws} Command Line Interface]. However, the {aws} CDK might behave somewhat differently from these tools. It uses the {aws} SDK for JavaScript under the hood. For complete details on setting up credentials for the {aws} SDK for JavaScript, see https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/setting-credentials.html[Setting credentials].

====

You may optionally use the --role-arn (or -r) option to specify the ARN of an IAM role that should be used for deployment. This role must be assumable by the {aws} account being used.

[#cli-app-command] == Specify the app command

Many features of the CDK CLI require one or more {aws} CloudFormation templates be synthesized, which in turn requires running your application. The {aws} CDK supports programs written in a variety of languages. Therefore, it uses a configuration option to specify the exact command necessary to run your app. This option can be specified in two ways.

First, and most commonly, it can be specified using the app key inside the file cdk.json. This is in the main directory of your {aws} CDK project. The CDK CLI provides an appropriate command when creating a new project with cdk init. Here is the cdk.json from a fresh TypeScript project, for instance.

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

{ “app”: “npx ts-node bin/hello-cdk.ts” } —

The CDK CLI looks for cdk.json in the current working directory when attempting to run your app. Because of this, you might keep a shell open in your project’s main directory for issuing CDK CLI commands.

The CDK CLI also looks for the app key in ~/.cdk.json (that is, in your home directory) if it can’t find it in ./cdk.json. Adding the app command here can be useful if you usually work with CDK code in the same language.

If you are in some other directory, or to run your app using a command other than the one in cdk.json, use the --app (or -a) option to specify it.

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

## cdk –app “npx ts-node bin/hello-cdk.ts” ls

When deploying, you may also specify a directory containing synthesized cloud assemblies, such as cdk.out, as the value of --app. The specified stacks are deployed from this directory; the app is not synthesized.

[#cli-stacks] == Specify stacks

Many CDK CLI commands (for example, cdk deploy) work on stacks defined in your app. If your app contains only one stack, the CDK CLI assumes you mean that one if you don’t specify a stack explicitly.

Otherwise, you must specify the stack or stacks you want to work with. You can do this by specifying the desired stacks by ID individually on the command line. Recall that the ID is the value specified by the second argument when you instantiate the stack.

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

## cdk synth PipelineStack LambdaStack

You may also use wildcards to specify IDs that match a pattern.

* ? matches any single character
* \\* matches any number of characters (\* alone matches all stacks)
* +\*\*+ matches everything in a hierarchy

You may also use the --all option to specify all stacks.

If your app uses xref:cdk-pipeline[CDK Pipelines], the CDK CLI understands your stacks and stages as a hierarchy. Also, the --all option and the \\* wildcard only match top-level stacks. To match all the stacks, use +\\*\*+. Also use +\*\*+ to indicate all the stacks under a particular hierarchy.

When using wildcards, enclose the pattern in quotes, or escape the wildcards with \. If you don’t, your shell may try to expand the pattern to the names of files in the current directory. At best, this won’t do what you expect; at worst, you could deploy stacks you didn’t intend to. This isn’t strictly necessary on Windows because cmd.exe does not expand wildcards, but is good practice nonetheless.

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

cdk synth “*Stack” # PipelineStack, LambdaStack, etc. cdk synth ‘Stack?’ # StackA, StackB, Stack1, etc. cdk synth*  # All stacks in the app, or all top-level stacks in a CDK Pipelines app cdk synth ‘*’ # All stacks in a CDK Pipelines app cdk synth ’PipelineStack/Prod/*’ # All stacks in Prod stage in a CDK Pipelines app —

= [NOTE]

The order in which you specify the stacks is not necessarily the order in which they will be processed. The CDK CLI accounts for dependencies between stacks when deciding the order in which to process them. For example, let’s say that one stack uses a value produced by another (such as the ARN of a resource defined in the second stack). In this case, the second stack is synthesized before the first one because of this dependency. You can add dependencies between stacks manually using the stack’s +link:https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.Stack.html#addwbrdependencytarget-reason[addDependency()]+ method.

====

[#cli-bootstrap] == Bootstrap your {aws} environment

Deploying stacks with the CDK requires special dedicated {aws} CDK resources to be provisioned. The cdk bootstrap command creates the necessary resources for you. You only need to bootstrap if you are deploying a stack that requires these dedicated resources. See xref:bootstrapping[{aws} CDK bootstrapping] for details.

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

## cdk bootstrap

If issued with no arguments, as shown here, the cdk bootstrap command synthesizes the current app and bootstraps the environments its stacks will be deployed to. If the app contains environment-agnostic stacks, which don’t explicitly specify an environment, the default account and Region are bootstrapped, or the environment specified using --profile.

Outside of an app, you must explicitly specify the environment to be bootstrapped. You may also do so to bootstrap an environment that’s not specified in your app or local {aws} profile. Credentials must be configured (e.g. in ~/.aws/credentials) for the specified account and Region. You may specify a profile that contains the required credentials.

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

cdk bootstrap ++++++/++++++# e.g. cdk bootstrap 1111111111/us-east-1 cdk bootstrap –profile test 1111111111/us-east-1 —-++++++++++++

= [IMPORTANT]

Each environment (account/region combination) to which you deploy such a stack must be bootstrapped separately.

====

You may incur {aws} charges for what the {aws} CDK stores in the bootstrapped resources. Additionally, if you use --bootstrap-customer-key, an {aws} KMS key will be created, which also incurs charges per environment.

= [NOTE]

Earlier versions of the bootstrap template created a KMS key by default. To avoid charges, re-bootstrap using --no-bootstrap-customer-key.

====

= [NOTE]

CDK CLI v2 does not support the original bootstrap template, dubbed the legacy template, used by default with CDK v1.

====

= [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.

====

[#cli-init] == Create a new app

To create a new app, create a directory for it, then, inside the directory, issue cdk init.

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

mkdir my-cdk-app cd my-cdk-app cdk init ++++++–language ++++++—-++++++++++++

The supported languages (++++++) are:++++++

[cols=“1,1”, options=“header”] |=== | Code | Language

|=== | typescript | TypeScript |===

|=== | javascript | JavaScript |===

|=== | python | Python |===

|=== | java | Java |===

|=== | csharp | C# |===

|=== | Go | Go |===++++++is an optional template. If the desired template is *app*, the default, you may omit it. The available templates are: [cols=“1,1”, options=“header”] |=== | Template | Description |app (default) |Creates an empty {aws} CDK app. |sample-app |Creates an {aws} CDK app with a stack containing an Amazon SQS queue and an Amazon SNS topic. |=== The templates use the name of the project folder to generate names for files and classes inside your new app. [#cli-list] == List stacks To see a list of the IDs of the stacks in your {aws} CDK application, enter one of the following equivalent commands: [source,none,subs=“verbatim,attributes”] —- cdk list cdk ls —- If your application contains xref:cdk-pipeline[CDK Pipelines] stacks, the CDK CLI displays stack names as paths according to their location in the pipeline hierarchy. (For example, PipelineStack, PipelineStack/Prod, and PipelineStack/Prod/MyService.) If your app contains many stacks, you can specify full or partial stack IDs of the stacks to be listed. For more information, see xref:cli-stacks[Specify stacks]. Add the --long flag to see more information about the stacks, including the stack names and their environments ({aws} account and Region). [#cli-synth] == Synthesize stacks The cdk synthesize command (almost always abbreviated synth) synthesizes a stack defined in your app into a CloudFormation template. [source,none,subs=“verbatim,attributes”] —- cdk synth # if app contains only one stack cdk synth MyStack cdk synth Stack1 Stack2 cdk synth “*” # all stacks in app —- [NOTE] ==== The CDK CLI actually runs your app and synthesizes fresh templates before most operations (such as when deploying or comparing stacks). These templates are stored by default in the cdk.out directory. The cdk synth command simply prints the generated templates for one or more specified stacks. ==== See cdk synth --help for all available options. A few of the most frequently used options are covered in the following section. [#cli-specify-context#] === Specify context values Use the --context or -c option to pass xref:context[runtime context] values to your CDK app. [source,none,subs=”verbatim,attributes”] —- # specify a single context value cdk synth –context key=value MyStack # specify multiple context values (any number) cdk synth –context key1=value1 –context key2=value2 MyStack —- When deploying multiple stacks, the specified context values are normally passed to all of them. If you want, you can specify different values for each stack by prefixing the stack name to the context value. [source,none,subs=”verbatim,attributes”] —- # different context values for each stack cdk synth –context Stack1:key=value Stack2:key=value Stack1 Stack2 —- [#cli-specify-format] === Specify display format By default, the synthesized template is displayed in YAML format. Add the --json flag to display it in JSON format instead. [source,none,subs=”verbatim,attributes”] —- cdk synth –json MyStack —- [#cli-specify-output] === Specify the output directory Add the --output (-o) option to write the synthesized templates to a directory other than cdk.out. [source,none,subs=”verbatim,attributes”] —- cdk synth –output=~/templates —- [#cli-deploy] == Deploy stacks The cdk deploy subcommand deploys one or more specified stacks to your {aws} account. [source,none,subs=”verbatim,attributes”] —- cdk deploy # if app contains only one stack cdk deploy MyStack cdk deploy Stack1 Stack2 cdk deploy ”*” # all stacks in app —- [NOTE] ==== The CDK CLI runs your app and synthesizes fresh {aws} CloudFormation templates before deploying anything. Therefore, most command line options you can use with cdk synth (for example, --context) can also be used with cdk deploy. ==== See cdk deploy --help for all available options. A few of the most useful options are covered in the following section. [#cli-deploy-nosynth] === Skip synthesis The cdk deploy command normally synthesizes your app’s stacks before deploying to make sure that the deployment reflects the latest version of your app. If you know that you haven’t changed your code since your last cdk synth, you can suppress the redundant synthesis step when deploying. To do so, specify your project’s cdk.out directory in the --app option. [source,none,subs=“verbatim,attributes”] —- cdk deploy –app cdk.out StackOne StackTwo —- [#cli-deploy-norollback] === Disable rollback {aws} CloudFormation has the ability to roll back changes so that deployments are atomic. This means that they either succeed or fail as a whole. The {aws} CDK inherits this capability because it synthesizes and deploys {aws} CloudFormation templates. Rollback makes sure that your resources are in a consistent state at all times, which is vital for production stacks. However, while you’re still developing your infrastructure, some failures are inevitable, and rolling back failed deployments can slow you down. For this reason, the CDK CLI lets you disable rollback by adding --no-rollback to your cdk deploy command. With this flag, failed deployments are not rolled back. Instead, resources deployed before the failed resource remain in place, and the next deployment starts with the failed resource. You’ll spend a lot less time waiting for deployments and a lot more time developing your infrastructure. [#cli-deploy-hotswap] === Hot swapping Use the --hotswap flag with cdk deploy to attempt to update your {aws} resources directly instead of generating an {aws} CloudFormation change set and deploying it. Deployment falls back to {aws} CloudFormation deployment if hot swapping is not possible. Currently hot swapping supports Lambda functions, Step Functions state machines, and Amazon ECS container images. The --hotswap flag also disables rollback (i.e., implies --no-rollback). [IMPORTANT] ==== Hot-swapping is not recommended for production deployments. ==== [#cli-deploy-watch] === Watch mode The CDK CLI’s watch mode (cdk deploy --watch, or cdk watch for short) continuously monitors your CDK app’s source files and assets for changes. It immediately performs a deployment of the specified stacks when a change is detected. By default, these deployments use the --hotswap flag, which fast-tracks deployment of changes to Lambda functions. It also falls back to deploying through {aws} CloudFormation if you have changed infrastructure configuration. To have cdk watch always perform full {aws} CloudFormation deployments, add the --no-hotswap flag to cdk watch. Any changes made while cdk watch is already performing a deployment are combined into a single deployment, which begins as soon as the in-progress deployment is complete. Watch mode uses the "watch" key in the project’s cdk.json to determine which files to monitor. By default, these files are your application files and assets, but this can be changed by modifying the "include" and "exclude" entries in the "watch" key. The following cdk.json file shows an example of these entries. [source,json,subs=“verbatim,attributes”] —- { “app”: “mvn -e -q compile exec:java”, “watch”: { “include”: “src/main/\*\*“,”exclude”: “target/*” } } —- cdk watch executes the "build" command from cdk.json to build your app before synthesis. If your deployment requires any commands to build or package your Lambda code (or anything else that’s not in your CDK app), add it here. Git-style wildcards, both \\* and \\*\*, can be used in the "watch" and "build" keys. Each path is interpreted relative to the parent directory of cdk.json. The default value of include is \\*\*/\*, meaning all files and directories in the project root directory. exclude is optional. [IMPORTANT] ==== Watch mode is not recommended for production deployments. ==== [#cli-specify-parameters] === Specify {aws} CloudFormation parameters The CDK CLI supports specifying {aws} CloudFormation xref:parameters[parameters] at deployment. You may provide these on the command line following the --parameters flag. [source,none,subs=”verbatim,attributes”] —- cdk deploy MyStack –parameters uploadBucketName=UploadBucket —- To define multiple parameters, use multiple --parameters flags. [source,none,subs=”verbatim,attributes”] —- cdk deploy MyStack –parameters uploadBucketName=UpBucket –parameters downloadBucketName=DownBucket —- If you are deploying multiple stacks, you can specify a different value of each parameter for each stack. To do so, prefix the name of the parameter with the stack name and a colon. Otherwise, the same value is passed to all stacks. [source,none,subs=”verbatim,attributes”] —- cdk deploy MyStack YourStack –parameters MyStack:uploadBucketName=UploadBucket –parameters YourStack:uploadBucketName=UpBucket —- By default, the {aws} CDK retains values of parameters from previous deployments and uses them in later deployments if they are not specified explicitly. Use the --no-previous-parameters flag to require all parameters to be specified. [#cli-specify-outputs-file] === Specify outputs file If your stack declares {aws} CloudFormation outputs, these are normally displayed on the screen at the conclusion of deployment. To write them to a file in JSON format, use the --outputs-file flag. [source,none,subs=”verbatim,attributes”] —- cdk deploy –outputs-file outputs.json MyStack —- [#cli-security] === Approve security-related changes To protect you against unintended changes that affect your security posture, the CDK CLI prompts you to approve security-related changes before deploying them. You can specify the level of change that requires approval: [source,none,subs=”verbatim,attributes”] —- cdk deploy –require-approval ++++++—- +++<LEVEL>+++ can be one of the following: [cols=”1,1”, options=”header”] |=== | Term | Meaning |never |Approval is never required |any-change |Requires approval on any IAM or security-group-related change |broadening (default) |Requires approval when IAM statements or traffic rules are added; removals don’t require approval |=== The setting can also be configured in the cdk.json file. [source,json,subs=”verbatim,attributes”] —- { ”app”: ”...”, ”requireApproval”: ”never” } —- [#cli-diff] == Compare stacks The cdk diff command compares the current version of a stack (and its dependencies) defined in your app with the already-deployed versions, or with a saved {aws} CloudFormation template, and displays a list of changes. —- Stack HelloCdkStack IAM Statement Changes ┌───┬──────────────────────────────┬────────┬──────────────────────────────┬──────────────────────────────┬───────────┐ │ │ Resource │ Effect │ Action │ Principal │ Condition │ ├───┼──────────────────────────────┼────────┼──────────────────────────────┼──────────────────────────────┼───────────┤ │ + │ ${Custom::S3AutoDeleteObject │ Allow │ sts:AssumeRole │ Service:lambda.amazonaws.com │ │ │ │ sCustomResourceProvider/Role │ │ │ │ │ │ │ .Arn} │ │ │ │ │ ├───┼──────────────────────────────┼────────┼──────────────────────────────┼──────────────────────────────┼───────────┤ │ + │ ${MyFirstBucket.Arn} │ Allow │ s3:DeleteObject\* │ \{aws}:${Custom::S3AutoDeleteOb │ │ │ │ ${MyFirstBucket.Arn}/* │ │ s3:GetBucket\* │ jectsCustomResourceProvider/ │ │ │ │ │ │ s3:GetObject\* │ Role.Arn} │ │ │ │ │ │ s3:List\* │ │ │ └───┴──────────────────────────────┴────────┴──────────────────────────────┴──────────────────────────────┴───────────┘ IAM Policy Changes ┌───┬────────────────────────────────────────────────────────┬────────────────────────────────────────────────────────┐ │ │ Resource │ Managed Policy ARN │ ├───┼────────────────────────────────────────────────────────┼────────────────────────────────────────────────────────┤ │ + │ ${Custom::S3AutoDeleteObjectsCustomResourceProvider/Ro │ {"Fn::Sub":"arn:${{aws}::Partition}:iam::aws:policy/serv │ │ │ le} │ ice-role/AWSLambdaBasicExecutionRole”} │ └───┴────────────────────────────────────────────────────────┴────────────────────────────────────────────────────────┘ (NOTE: There may be security-related changes not in this list. See https://github.com/aws/aws-cdk/issues/1299) Parameters [+] Parameter AssetParameters/4cd61014b71160e8c66fe167e43710d5ba068b80b134e9bd84508cf9238b2392/S3Bucket AssetParameters4cd61014b71160e8c66fe167e43710d5ba068b80b134e9bd84508cf9238b2392S3BucketBF7A7F3F: {“Type”:“String”,“Description”:“S3 bucket for asset "4cd61014b71160e8c66fe167e43710d5ba068b80b134e9bd84508cf9238b2392"”} [+] Parameter AssetParameters/4cd61014b71160e8c66fe167e43710d5ba068b80b134e9bd84508cf9238b2392/S3VersionKey AssetParameters4cd61014b71160e8c66fe167e43710d5ba068b80b134e9bd84508cf9238b2392S3VersionKeyFAF93626: {“Type”:“String”,“Description”:“S3 key for asset version "4cd61014b71160e8c66fe167e43710d5ba068b80b134e9bd84508cf9238b2392"”} [+] Parameter AssetParameters/4cd61014b71160e8c66fe167e43710d5ba068b80b134e9bd84508cf9238b2392/ArtifactHash AssetParameters4cd61014b71160e8c66fe167e43710d5ba068b80b134e9bd84508cf9238b2392ArtifactHashE56CD69A: {“Type”:“String”,“Description”:“Artifact hash for asset "4cd61014b71160e8c66fe167e43710d5ba068b80b134e9bd84508cf9238b2392"”} Resources [+] {aws}::S3::BucketPolicy MyFirstBucket/Policy MyFirstBucketPolicy3243DEFD [+] Custom::S3AutoDeleteObjects MyFirstBucket/AutoDeleteObjectsCustomResource MyFirstBucketAutoDeleteObjectsCustomResourceC52FCF6E [+] {aws}::IAM::Role Custom::S3AutoDeleteObjectsCustomResourceProvider/Role CustomS3AutoDeleteObjectsCustomResourceProviderRole3B1BD092 [+] {aws}::Lambda::Function Custom::S3AutoDeleteObjectsCustomResourceProvider/Handler CustomS3AutoDeleteObjectsCustomResourceProviderHandler9D90184F [~] {aws}::S3::Bucket MyFirstBucket MyFirstBucketB8884501 ├─ [~] DeletionPolicy │ ├─ [-] Retain │ └─ [+] Delete └─ [~] UpdateReplacePolicy ├─ [-] Retain └─ [+] Delete —- To compare your app’s stacks with the existing deployment: [source,none,subs=“verbatim,attributes”] —- cdk diff MyStack —- To compare your app’s stacks with a saved CloudFormation template: [source,none,subs=“verbatim,attributes”] —- cdk diff –template ~/stacks/MyStack.old MyStack —- [#cli-import] == Import existing resources into a stack You can use the cdk import command to bring resources under the management of CloudFormation for a particular {aws} CDK stack. This is useful if you are migrating to {aws} CDK, or are moving resources between stacks or changing their logical id. cdk import uses https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/resource-import.html[CloudFormation resource imports]. See the https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/resource-import-supported-resources.html[list of resources that can be imported here]. To import an existing resource into a {aws} CDK stack, follow the following steps: \* Make sure the resource is not currently being managed by any other CloudFormation stack. If it is, first set the removal policy to RemovalPolicy.RETAIN in the stack the resource is currently in and perform a deployment. Then, remove the resource from the stack and perform another deployment. This process will make sure that the resource is no longer managed by CloudFormation but does not delete it. \* Run a cdk diff to make sure there are no pending changes to the {aws} CDK stack you want to import resources into. The only changes allowed in an “import” operation are the addition of new resources which you want to import. \* Add constructs for the resources you want to import to your stack. For example, if you want to import an Amazon S3 bucket, add something like new s3.Bucket(this, 'ImportedS3Bucket', {});. Do not make any modifications to any other resource. + You must also make sure to exactly model the state that the resource currently has into the definition. For the example of the bucket, be sure to include {aws} KMS keys, life cycle policies, and anything else that’s relevant about the bucket. If you do not, subsequent update operations may not do what you expect. + You can choose whether or not to include the physical bucket name. We usually recommend to not include resource names into your {aws} CDK resource definitions so that it becomes easier to deploy your resources multiple times. + \* Run cdk import +++<STACKNAME>+++. \* If the resource names are not in your model, the CLI will prompt you to pass in the actual names of the resources you are importing. After this, the import starts. \* When cdk import reports success, the resource is now managed by {aws} CDK and CloudFormation. Any subsequent changes you make to the resource properties in your {aws} CDK app the construct configuration will be applied on the next deployment. \* To confirm that the resource definition in your {aws} CDK app matches the current state of the resource, you can start an https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-stack-drift.html[CloudFormation drift detection operation]. This feature currently does not support importing resources into nested stacks. [#cli-config] == Configuration (cdk.json) Default values for many CDK CLI command line flags can be stored in a project’s cdk.json file or in the .cdk.json file in your user directory. Following is an alphabetical reference to the supported configuration settings. [cols=“1,1,1”, options=“header”] |=== | Key | Notes | CDK CLI option |app |The command that executes the CDK application. |--app |assetMetadata |If false, CDK does not add metadata to resources that use assets. |--no-asset-metadata |bootstrapKmsKeyId |Overrides the ID of the {aws} KMS key used to encrypt the Amazon S3 deployment bucket. |--bootstrap-kms-key-id |build |The command that compiles or builds the CDK application before synthesis. Not permitted in ~/.cdk.json. |--build |browser |The command for launching a Web browser for the cdk docs subcommand. |--browser |context |See xref:context[Context values and the {aws} CDK]. Context values in a configuration file will not be erased by cdk context --clear. (The CDK CLI places cached context values in cdk.context.json.) |--context |debug |If true, CDK CLI emits more detailed information useful for debugging. |--debug |language |The language to be used for initializing new projects. |--language |lookups |If false, no context lookups are permitted. Synthesis will fail if any context lookups need to be performed. |--no-lookups |notices |If false, suppresses the display of messages about security vulnerabilities, regressions, and unsupported versions. |--no-notices |output |The name of the directory into which the synthesized cloud assembly will be emitted (default "cdk.out"). |--output |outputsFile |The file to which {aws} CloudFormation outputs from deployed stacks will be written (in JSON format). |--outputs-file |pathMetadata |If false, CDK path metadata is not added to synthesized templates. |--no-path-metadata |plugin |JSON array specifying the package names or local paths of packages that extend the CDK |--plugin |profile |Name of the default {aws} profile used for specifying Region and account credentials. |--profile |progress |If set to "events", the CDK CLI displays all {aws} CloudFormation events during deployment, rather than a progress bar. |--progress |requireApproval |Default approval level for security changes. See xref:cli-security[Approve security-related changes] |--require-approval |rollback |If false, failed deployments are not rolled back. |--no-rollback |staging |If false, assets are not copied to the output directory (use for local debugging of the source files with {aws} SAM). |--no-staging |tags |JSON object containing tags (key-value pairs) for the stack. |--tags |toolkitBucketName |The name of the Amazon S3 bucket used for deploying assets such as Lambda functions and container images (see xref:cli-bootstrap[Bootstrap your {aws} environment]). |--toolkit-bucket-name |toolkitStackName |The name of the bootstrap stack (see xref:cli-bootstrap[Bootstrap your {aws} environment]). |--toolkit-stack-name |versionReporting |If false, opts out of version reporting. |--no-version-reporting |watch |JSON object containing "include" and "exclude" keys that indicate which files should (or should not) trigger a rebuild of the project when changed. See xref:cli-deploy-watch[Watch mode]. |--watch |===++++++++++++++++++++++++