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’