include::attributes.txt[]

// Attributes

[.topic] [#bootstrapping-customizing] = Customize {aws} CDK bootstrapping :info\_titleabbrev: Customize bootstrapping :keywords: {aws} CDK, {aws} Cloud Development Kit ({aws} CDK), {aws} account, {aws} Region, Bootstrapping, Bootstrap, Environment, Customize

== [abstract]

## You can customize {aws} Cloud Development Kit ({aws} CDK) bootstrapping by using the {aws} CDK Command Line Interface ({aws} CDK CLI) or by modifying and deploying the {aws} CloudFormation bootstrap template.

// Content start

You can customize {aws} Cloud Development Kit ({aws} CDK) bootstrapping by using the {aws} CDK Command Line Interface ({aws} CDK CLI) or by modifying and deploying the {aws} CloudFormation bootstrap template.

For an introduction to bootstrapping, see xref:bootstrapping[{aws} CDK bootstrapping].

[#bootstrapping-customizing-cli] == Use the CDK CLI to customize bootstrapping

The following are a few examples of how you can customize bootstrapping by using the CDK CLI. For a list of all cdk bootstrap options, see xref:ref-cli-cmd-bootstrap[cdk bootstrap].

[#bootstrapping-customizing-cli-s3-name] *Override the name of the Amazon S3 bucket*:: Use the --bootstrap-bucket-name option to override the default Amazon S3 bucket name. This may require that you modify template synthesis. For more information, see xref:bootstrapping-custom-synth[Customize CDK stack synthesis].

[#bootstrapping-customizing-keys] *Modify server-side encryption keys for the Amazon S3 bucket*:: By default, the Amazon S3 bucket in the bootstrap stack is configure to use {aws} managed keys for server-side encryption. To use an existing customer managed key, use the --bootstrap-kms-key-id option and provide a value for the {aws} Key Management Service ({aws} KMS) key to use. If you want more control over the encryption key, provide --bootstrap-customer-key to use a customer managed key.

[#bootstrapping-customizing-cli-deploy-role] *Attach managed policies to the deployment role assumed by {aws} CloudFormation*:: By default, stacks are deployed with full administrator permissions using the AdministratorAccess policy. To use your own managed policies, use the --cloudformation-execution-policies option and provide the ARNs of the managed policies to attach to the deployment role. + To provide multiple policies, pass them a single string, separated by commas: + [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. + [#bootstrapping-customizing-cli-qualifier] *Change the qualifier that is added to the names of resources in your bootstrap stack*:: By default, the hnb659fds qualifier is added to the physical ID of resources in your bootstrap stack. To change this value, use the --qualifier option. + This modification is useful when provisioning multiple bootstrap stacks in the same environment to avoid name clashes. + Changing the qualifier is intended for name isolation between automated tests of the CDK itself. Unless you can very precisely scope down the IAM permissions given to the CloudFormation execution role, there are no permission isolation benefits to having two different bootstrap stacks in a single account. Therefore, there’s usually no need to change this value. + 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]. + [#bootstrapping-customizing-cli-tags] *Add tags to the bootstrap stack*:: Use the --tags option in the format of KEY=VALUE to add CloudFormation tags to your bootstrap stack. + [#bootstrapping-customizing-cli-accounts-deploy] *Specify additional {aws} accounts that can deploy into the environment being bootstrapped*:: Use the --trust option to provide additional {aws} accounts that are allowed to deploy into the environment being bootstrapped. By default, the account performing the bootstrapping will always be trusted. + This option is useful when you are bootstrapping an environment that a CDK [.noloc]Pipeline from another environment will deploy into. + When you use this option, you must also provide --cloudformation-execution-policies. + 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. —

[#bootstrapping-customizing-cli-accounts-lookup] *Specify additional {aws} accounts that can look up information in the environment being bootstrapped*:: + Use the --trust-for-lookup option to specify {aws} accounts that are allowed to look up context information from the environment being bootstrapped. This option is useful to give accounts permission to synthesize stacks that will be deployed into the environment, without actually giving them permission to deploy those stacks directly.

[#bootstrapping-customizing-cli-protection] *Enable termination protection for the bootstrap stack*:: If a bootstrap stack is deleted, the {aws} resources that were originally provisioned in the environment will also be deleted. After your environment is bootstrapped, we recommend that you don’t delete and recreate the environment’s bootstrap stack, unless you are intentionally doing so. Instead, try to update the bootstrap stack to a new version by running the cdk bootstrap command again. + Use the --termination-protection option to manage termination protection settings for the bootstrap stack. By enabling termination protection, you prevent the bootstrap stack and its resources from being accidentally deleted. This is especially important if you use CDK [.noloc]Pipelines since there is no general recovery option if you accidentally delete the bootstrap stack. + After enabling termination protection, you can use the {aws} CLI or {aws} CloudFormation console to verify. + *To enable termination protection*::: + . Run the following command to enable termination protection on a new or existing bootstrap stack: + [source,none,subs=“verbatim,attributes”] — $ cdk bootstrap –termination-protection — + . Use the {aws} CLI or CloudFormation console to verify. The following is an example, using the {aws} CLI. If you modified your bootstrap stack name, replace CDKToolkit with your stack name: + [source,none,subs=“verbatim,attributes”] — $ aws cloudformation describe-stacks –stack-name ++++++–query “Stacks[0].EnableTerminationProtection” true —-++++++

[#bootstrapping-customizing-template] == Modify the default bootstrap template

When you need more customization than the CDK CLI can provide, you can modify the bootstrap template as needed. Then, deploy the template to bootstrap your environment.

*To modify and deploy the default bootstrap template*:: + . Obtain the default bootstrap template using the --show-template option. By default, the CDK CLI will output the template in your terminal window. You can modify the CDK CLI command to save the template to your local machine. The following is an example: + [source,none,subs=“verbatim,attributes”] — $ cdk bootstrap –show-template > +++<my-bootstrap-template.yaml>+++—- . Modify the bootstrap template as needed. Any changes that you make should adhere to the bootstrapping template contract. For more information on the bootstrapping template contract, see xref:bootstrapping-contract[Follow the bootstrap contract]. + To ensure that your customizations are not accidentally overwritten later by someone running cdk bootstrap using the default template, change the default value of the BootstrapVariant template parameter. The CDK CLI will only allow overwriting the bootstrap stack with templates that have the same BootstrapVariant and an equal or higher version than the template that is currently deployed. + . Deploy your modified template using your preferred {aws} CloudFormation deployment method. The following is an example that uses the CDK CLI: + [source,none,subs=“verbatim,attributes”] —- $ cdk bootstrap –template +++<my-bootstrap-template.yaml>+++—-+++</my-bootstrap-template.yaml>++++++</my-bootstrap-template.yaml>+++

[#bootstrapping-contract] == Follow the bootstrap contract

For your CDK apps to properly deploy, the CloudFormation templates produced during synthesis must correctly specify the resources created during bootstrapping. These resources are commonly referred to as *bootstrap resources*. Bootstrapping creates resources in your {aws} environment that are used by the {aws} CDK to perform deployments and manage application assets. Synthesis produces CloudFormation templates from each CDK stack in your application. These templates don’t just define the {aws} resources that will be provisioned from your application. They also specify the bootstrap resources to use during deployment.

During synthesis, the CDK CLI doesn’t know specifically how your {aws} environment has been bootstrapped. Instead, the CDK CLI produces CloudFormation templates based on the synthesizer that you configure. Therefore, when you customize bootstrapping, you may need to customize synthesis. For instructions on customizing synthesis, see xref:bootstrapping-custom-synth[Customize CDK stack synthesis]. The purpose is to ensure that your synthesized CloudFormation templates are compatible with your bootstrapped environment. This compatibility is referred to as the *bootstrap contract*.

The simplest method to customize stack synthesis is by modifying the DefaultStackSynthesizer class in your Stack instance. If you require customization beyond what this class can offer, you can write your own synthesizer as a class that implements +link:https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.IStackSynthesizer.html[IStackSynthesizer]+ (perhaps deriving from DefaultStackSynthesizer).

When you customize bootstrapping, follow the bootstrap template contract to remain compatible with DefaultStackSynthesizer. If you modify bootstrapping beyond the bootstrap template contract, you will need to write your own synthesizer.

[#bootstrapping-contract-versioning] === Versioning

The bootstrap template should contain a resource to create an Amazon EC2 Systems Manager (SSM) parameter with a well-known name and an output to reflect the template’s version:

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

Resources: CdkBootstrapVersion: Type: {aws}::SSM::Parameter Properties: Type: String Name: Fn::Sub: ‘/cdk-bootstrap/${Qualifier}/version’ Value: 4 Outputs: BootstrapVersion: Value: Fn::GetAtt: [CdkBootstrapVersion, Value] —

[#bootstrapping-contract-roles] === Roles

The DefaultStackSynthesizer requires five IAM roles for five different purposes. If you are not using the default roles, you must specify your IAM role ARNs within your DefaultStackSynthesizer object. The roles are as follows:

* The *deployment role* is assumed by the CDK CLI and by {aws} CodePipeline to deploy into an environment. Its AssumeRolePolicy controls who can deploy into the environment. In the template, you can see the permissions that this role needs.
* The *lookup role* is assumed by the CDK CLI to perform context lookups in an environment. Its AssumeRolePolicy controls who can deploy into the environment. The permissions this role needs can be seen in the template.
* The *file publishing role* and the *image publishing role* are assumed by the CDK CLI and by {aws} CodeBuild projects to publish assets into an environment. They’re used to write to the Amazon S3 bucket and the Amazon ECR repository, respectively. These roles require write access to these resources.
* *The {aws} CloudFormation execution role* is passed to {aws} CloudFormation to perform the actual deployment. Its permissions are the permissions that the deployment will execute under. The permissions are passed to the stack as a parameter that lists managed policy ARNs.

[#bootstrapping-contract-outputs] === Outputs

The CDK CLI requires that the following CloudFormation outputs exist on the bootstrap stack:

* BucketName – The name of the file asset bucket.
* BucketDomainName – The file asset bucket in domain name format.
* BootstrapVersion – The current version of the bootstrap stack.

[#bootstrapping-contract-history] === Template history

The bootstrap template is versioned and evolves over time with the {aws} CDK itself. If you provide your own bootstrap template, keep it up to date with the canonical default template. You want to make sure that your template continues to work with all CDK features. For more information, see xref:bootstrap-template-history[Bootstrap template version history].