Bootstrapping - AWS Cloud Development Kit (CDK) v2

Bootstrapping - AWS Cloud Development Kit (CDK) v2

[AWS](http://aws.amazon.com)[Documentation](/index.html)[AWS Cloud Development Kit](/cdk/index.html)[Developer Guide](home.html)

[How to bootstrap](#bootstrapping-howto)[Bootstrapping template](#bootstrapping-template)[Customizing bootstrapping](#bootstrapping-customizing)[Stack synthesizers](#bootstrapping-synthesizers)[Customizing synthesis](#bootstrapping-custom-synth)[The bootstrapping template contract](#bootstrapping-contract)

This is the AWS CDK v2 Developer Guide. If you’re using v1, see the [CDK v1 Developer Guide](../../v1/guide/). CDK v1 enters maintenance on June 1, 2022.

# Bootstrapping

Deploying AWS CDK apps into an AWS [environment](./environments.html) (a combination of an AWS account and region) may require that you provision resources the AWS CDK needs to perform the deployment. These resources include an Amazon S3 bucket for storing files and IAM roles that grant permissions needed to perform deployments. The process of provisioning these initial resources is called *bootstrapping*.

An environment needs to be bootstrapped if any of the following apply.

* An AWS CDK stack being deployed uses [Assets](./assets.html).
* An AWS CloudFormation template generated by the app exceeds 50 kilobytes.

The required resources are defined in a AWS CloudFormation stack, called the *bootstrap stack*, which is usually named CDKToolkit. Like any AWS CloudFormation stack, it appears in the AWS CloudFormation console once it has been deployed.

Note

CDK v2 uses a bootstrap template dubbed the modern template. The legacy template from CDK v1 is not supported in v2.

Environments are independent, so if you want to deploy to multiple environments (different AWS accounts or different regions in the same account), each environment must be bootstrapped separately.

Important

You may incur AWS charges for data stored in the bootstrapped resources.

Note

Older versions of the bootstrap template created a Customer Master Key (CMK) in each bootstrapped environment by default. To avoid charges for the CMK, re-bootstrap these environments using --no-bootstrap-customer-key. The current default is to not use a CMK to avoid these charges.

If you attempt to deploy an AWS CDK application that requires bootstrap resources into an environment that does not have them, you receive an error message telling you that you need to bootstrap the environment.

If you are using CDK Pipelines to deploy into another account’s environment, and you receive a message like the following:

Policy contains a statement with one or more invalid principals

This error message means that the appropriate IAM roles do not exist in the other environment, which is most likely caused by a lack of bootstrapping.

Note

Do not delete and recreate an account’s bootstrap stack if you are using CDK Pipelines to deploy into that account. The pipeline will stop working. To update the bootstrap stack to a new version, instead re-run cdk bootstrap to update the bootstrap stack in place.

## How to bootstrap

Bootstrapping is the deployment of a AWS CloudFormation template to a specific AWS environment (account and region). The bootstrapping template accepts parameters that customize some aspects of the bootstrapped resources (see [Customizing bootstrapping](#bootstrapping-customizing)). Thus, you can bootstrap in one of two ways.

* Use the AWS CDK Toolkit’s **cdk bootstrap** command. This is the simplest method and works well if you have only a few environments to bootstrap.
* Deploy the template provided by the AWS CDK Toolkit using another AWS CloudFormation deployment tool. This lets you use AWS CloudFormation Stack Sets or AWS Control Tower as well as the AWS CloudFormation console or the AWS CLI. You can even make small modifications to the template before deployment. This approach is more flexible and is suitable for large-scale deployments.

It is not an error to bootstrap an environment more than once. If an environment you bootstrap has already been bootstrapped, its bootstrap stack will be upgraded if necessary; otherwise, nothing happens.

### Bootstrapping with the AWS CDK Toolkit

Use the cdk bootstrap command to bootstrap one or more AWS environments. In its basic form, this command bootstraps one or more specified AWS environments (two, in this example).

cdk bootstrap aws://`ACCOUNT-NUMBER-1`/`REGION-1` aws://`ACCOUNT-NUMBER-2`/`REGION-2` ...

The following examples illustrate bootstrapping of one and two environments, respectively. (Both use the same AWS account.) As shown in the second example, the aws:// prefix is optional when specifying an environment.

cdk bootstrap aws://123456789012/us-east-1 cdk bootstrap 123456789012/us-east-1 123456789012/us-west-1

The CDK Toolkit always synthesizes the AWS CDK app in the current directory. If you do not specify at least one environment in the cdk bootstrap command, it bootstraps all the environments referenced in the app. If a stack is environment-agnostic (that is, it does not have an env property), the CDK’s environment (for example, the one specified using **--profile**, or the default AWS environment otherwise) is applied to make the stack environment-specific, and that environment is then bootstrapped.

For example, the following command synthesizes the current AWS CDK app using the prod AWS profile, then bootstraps its environments.

cdk bootstrap --profile prod

### Bootstrapping from the AWS CloudFormation template

AWS CDK bootstrapping is performed by an AWS CloudFormation template. To get a copy of this template in the file bootstrap-template.yaml, run the following command.

macOS/Linux

cdk bootstrap --show-template > bootstrap-template.yaml

Windows

On Windows, PowerShell must be used to preserve the encoding of the template.

powershell "cdk bootstrap --show-template | Out-File -encoding utf8 bootstrap-template.yaml"

The template is also available in the [AWS CDK GitHub repository](https://github.com/aws/aws-cdk/blob/master/packages/aws-cdk/lib/api/bootstrap/bootstrap-template.yaml).

Deploy this template using **cdk bootstrap –template TEMPLATE\_FILENAME** or your preferred deployment mechanism for AWS CloudFormation templates. For example, the following command deploys the template using the AWS CLI:

macOS/Linux

aws cloudformation create-stack \ --stack-name CDKToolkit \ --template-body file://bootstrap-template.yaml

Windows

aws cloudformation create-stack ^ --stack-name CDKToolkit ^ --template-body file://bootstrap-template.yaml

## Bootstrapping template

As previously mentioned, AWS CDK v1 supported two bootstrapping templates, legacy and modern. CDK v2 supports only the modern template. For reference, here are the high-level differences between these two templates.

Feature

Legacy (v1 only)

Modern (v1 and v2)

**Cross-account deployments**

Not allowed

Allowed

**AWS CloudFormation Permissions**

Deploys using current user’s permissions (determined by AWS profile, environment variables, etc.)

Deploys using the permissions specified when the bootstrap stack was provisioned (e.g. using --trust)

**Versioning**

Only one version of bootstrap stack is available

Bootstrap stack is versioned; new resources can be added in future versions, and AWS CDK apps can require a minimum version

**Resources**\*

Amazon S3 bucket

Amazon S3 bucket

AWS KMS key

IAM roles

Amazon ECR repository

SSM parameter for versioning

**Resource naming**

Automatically generated

Deterministic

**Bucket encryption**

Default key

Customer-managed key

\* *We will add additional resources to the bootstrap template as needed.*

An environment that has been bootstrapped using the legacy template can (and must) be upgraded to use the modern template for use with CDK v2 by re-bootstrapping. Re-deploy all AWS CDK applications in the environment at least once before deleting the legacy bucket.

## Customizing bootstrapping

There are two ways to customize the bootstrapping resources.

* Use command-line parameters with the cdk bootstrap command. This lets you modify a few aspects of the template.
* Modify the default bootstrap template and deploy it yourself. This gives you unlimited control over the bootstrap resources.

The following command-line options, when used with CDK Toolkit’s **cdk bootstrap**, provide commonly-needed adjustments to the bootstrapping template.

* -**-bootstrap-bucket-name** overrides the name of the Amazon S3 bucket. May require changes to your CDK app (see [Stack synthesizers](#bootstrapping-synthesizers)).
* **--bootstrap-kms-key-id** overrides the AWS KMS key used to encrypt the S3 bucket.
* **--cloudformation-execution-policies** specifies the ARNs of managed policies 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 privileges using the AdministratorAccess policy.
* The policy ARNs must be passed as a single string argument, with the individual ARNs separated by commas. For example:
* --cloudformation-execution-policies "arn:aws:iam::aws:policy/AWSLambda\_FullAccess,arn:aws:iam::aws:policy/AWSCodeDeployFullAccess".
* Important
* To avoid deployment failures, be sure the policies you specify are sufficient for any deployments you will perform in the environment being bootstrapped.
* **--qualifier** a string that is added to the names of all resources in the bootstrap stack. A qualifier lets you avoid resource name clashes when you provision multiple bootstrap stacks in the same environment using **--toolkit-stack-name**. The default is hnb659fds (this value has no significance). Changing the qualifier also requires that your CDK app pass the changed value to the stack synthesizer(see [Stack synthesizers](#bootstrapping-synthesizers)).
* **--tags** adds one or more AWS CloudFormation tags to the bootstrap stack.
* **--trust** lists the AWS accounts that may deploy into the environment being bootstrapped. Use this flag when bootstrapping an environment that a CDK Pipeline in another environment will deploy into. The account doing the bootstrapping is always trusted.
* **--trust-for-lookup** lists the AWS accounts that may look up context information from the environment being bootstrapped. Use this flag to give accounts permission to synthesize stacks that will be deployed into the environment, without actually giving them permission to deploy those stacks directly. Accounts specified under **--trust** are always trusted for context lookup.
* **--termination-protection** prevents the bootstrap stack from being deleted (see [Protecting a stack from being deleted](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-protect-stacks.html) in the AWS CloudFormation User Guide)

Important

The modern bootstrap template effectively grants the permissions implied by the --cloudformation-execution-policies to any AWS account in the --trust list, which by default will extend permissions to read and write to any resource in the bootstrapped account. Make sure to configure the bootstrapping stack with policies and trusted accounts you are comfortable with.

### Customizing the template

When you need more customization than the AWS CDK Toolkit switches can provide, you can modify the bootstrap template to suit your needs. Remember that you can obtain the template by using the **--show-template** flag.

cdk bootstrap --show-template

Any modifications you make must adhere to the [bootstrapping template contract](#bootstrapping-contract).

Deploy your modified template as described in [Bootstrapping from the AWS CloudFormation template](#bootstrapping-howto-cfn), or using **cdk bootstrap –template**.

cdk bootstrap --template bootstrap-template.yaml

## Stack synthesizers

Your AWS CDK app needs to know about the bootstrapping resources available to it in order to successfully synthesize a stack that can be deployed. The *stack synthesizer* is an AWS CDK class that controls how the stack’s template is synthesized, including how it uses bootstrapping resources (for example, how it refers to assets stored in the bootstrap bucket).

The AWS CDK’s built-in stack synthesizers is called DefaultStackSynthesizer. It includes capabilities for cross-account deployments and [CDK Pipelines](./cdk_pipeline.html) deployments.

You can pass a stack synthesizer to a stack when you instantiate it using the synthesizer property.

TypeScript

new MyStack(this, 'MyStack', { // stack properties synthesizer: new DefaultStackSynthesizer({ // synthesizer properties }), });

JavaScript

new MyStack(this, 'MyStack', { // stack properties synthesizer: new DefaultStackSynthesizer({ // synthesizer properties }), });

Python

MyStack(self, "MyStack", # stack properties synthesizer=DefaultStackSynthesizer( # synthesizer properties ))

Java

new MyStack(app, "MyStack", StackProps.builder() // stack properties .synthesizer(DefaultStackSynthesizer.Builder.create() // synthesizer properties .build()) .build();

C#

new MyStack(app, "MyStack", new StackProps // stack properties { Synthesizer = new DefaultStackSynthesizer(new DefaultStackSynthesizerProps { // synthesizer properties }) });

If you don’t provide the synthesizer property, DefaultStackSynthesizer is used.

## Customizing synthesis

Depending on the changes you made to the bootstrap template, you may also need to customize synthesis. The DefaultStackSynthesizer can be customized using the properties described below. If none of these properties provide the customizations you require, you can write your synthesizer as a class that implements IStackSynthesizer (perhaps deriving from DefaultStackSynthesizer).

### Changing the qualifier

The *qualifier* is added to the name of bootstrap resources to distinguish the resources in separate bootstrap stacks. To deploy two different versions of the bootstrap stack in the same environment (AWS account and region), then, the stacks must have different qualifiers. This feature 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 AWS CloudFormation execution role, there are no privilege isolation benefits to having two different bootstrap stacks in a single account, so there is usually no need to change this value.

To change the qualifier, configure the DefaultStackSynthesizer either by instantiating the synthesizer with the property:

TypeScript

new MyStack(this, 'MyStack', { synthesizer: new DefaultStackSynthesizer({ qualifier: 'MYQUALIFIER', }), });

JavaScript

new MyStack(this, 'MyStack', { synthesizer: new DefaultStackSynthesizer({ qualifier: 'MYQUALIFIER', }), })

Python

MyStack(self, "MyStack", synthesizer=DefaultStackSynthesizer( qualifier="MYQUALIFIER" ))

Java

new MyStack(app, "MyStack", StackProps.builder() .synthesizer(DefaultStackSynthesizer.Builder.create() .qualifier("MYQUALIFIER") .build()) .build();

C#

new MyStack(app, "MyStack", new StackProps { Synthesizer = new DefaultStackSynthesizer(new DefaultStackSynthesizerProps { Qualifier = "MYQUALIFIER" }) });

Or by configuring the qualifier as a context key in cdk.json.

{ "app": "...", "context": { "@aws-cdk/core:bootstrapQualifier": "MYQUALIFIER" } }

### Changing the resource names

All the other DefaultStackSynthesizer properties relate to the names of the resources in the bootstrapping template. You only need to provide any of these properties if you modified the bootstrap template and changed the resource names or naming scheme.

All properties accept the special placeholders ${Qualifier}, ${AWS::Partition}, ${AWS::AccountId}, and ${AWS::Region}. These placeholders are replaced with the values of the qualifier parameter and with the values of the AWS partition, account ID, and region for the stack’s environment, respectively.

The following example shows the most commonly-used properties for DefaultStackSynthesizer along with their default values, as if you were instantiating the synthesizer. For a complete list, see [DefaultStackSynthesizerProps](https://docs.aws.amazon.com/cdk/api/v2/docs/aws-cdk-lib.DefaultStackSynthesizerProps.html#properties).

TypeScript

new DefaultStackSynthesizer({ // Name of the S3 bucket for file assets fileAssetsBucketName: 'cdk-${Qualifier}-assets-${AWS::AccountId}-${AWS::Region}', bucketPrefix: '', // Name of the ECR repository for Docker image assets imageAssetsRepositoryName: 'cdk-${Qualifier}-container-assets-${AWS::AccountId}-${AWS::Region}', // ARN of the role assumed by the CLI and Pipeline to deploy here deployRoleArn: 'arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-deploy-role-${AWS::AccountId}-${AWS::Region}', deployRoleExternalId: '', // ARN of the role used for file asset publishing (assumed from the deploy role) fileAssetPublishingRoleArn: 'arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-file-publishing-role-${AWS::AccountId}-${AWS::Region}', fileAssetPublishingExternalId: '', // ARN of the role used for Docker asset publishing (assumed from the deploy role) imageAssetPublishingRoleArn: 'arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-image-publishing-role-${AWS::AccountId}-${AWS::Region}', imageAssetPublishingExternalId: '', // ARN of the role passed to CloudFormation to execute the deployments cloudFormationExecutionRole: 'arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-cfn-exec-role-${AWS::AccountId}-${AWS::Region}', // ARN of the role used to look up context information in an environment lookupRoleArn: 'arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-lookup-role-${AWS::AccountId}-${AWS::Region}', lookupRoleExternalId: '', // Name of the SSM parameter which describes the bootstrap stack version number bootstrapStackVersionSsmParameter: '/cdk-bootstrap/${Qualifier}/version', // Add a rule to every template which verifies the required bootstrap stack version generateBootstrapVersionRule: true, })

JavaScript

new DefaultStackSynthesizer({ // Name of the S3 bucket for file assets fileAssetsBucketName: 'cdk-${Qualifier}-assets-${AWS::AccountId}-${AWS::Region}', bucketPrefix: '', // Name of the ECR repository for Docker image assets imageAssetsRepositoryName: 'cdk-${Qualifier}-container-assets-${AWS::AccountId}-${AWS::Region}', // ARN of the role assumed by the CLI and Pipeline to deploy here deployRoleArn: 'arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-deploy-role-${AWS::AccountId}-${AWS::Region}', deployRoleExternalId: '', // ARN of the role used for file asset publishing (assumed from the deploy role) fileAssetPublishingRoleArn: 'arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-file-publishing-role-${AWS::AccountId}-${AWS::Region}', fileAssetPublishingExternalId: '', // ARN of the role used for Docker asset publishing (assumed from the deploy role) imageAssetPublishingRoleArn: 'arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-image-publishing-role-${AWS::AccountId}-${AWS::Region}', imageAssetPublishingExternalId: '', // ARN of the role passed to CloudFormation to execute the deployments cloudFormationExecutionRole: 'arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-cfn-exec-role-${AWS::AccountId}-${AWS::Region}', // ARN of the role used to look up context information in an environment lookupRoleArn: 'arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-lookup-role-${AWS::AccountId}-${AWS::Region}', lookupRoleExternalId: '', // Name of the SSM parameter which describes the bootstrap stack version number bootstrapStackVersionSsmParameter: '/cdk-bootstrap/${Qualifier}/version', // Add a rule to every template which verifies the required bootstrap stack version generateBootstrapVersionRule: true, })

Python

DefaultStackSynthesizer( # Name of the S3 bucket for file assets file\_assets\_bucket\_name="cdk-${Qualifier}-assets-${AWS::AccountId}-${AWS::Region}", bucket\_prefix="", # Name of the ECR repository for Docker image assets image\_assets\_repository\_name="cdk-${Qualifier}-container-assets-${AWS::AccountId}-${AWS::Region}", # ARN of the role assumed by the CLI and Pipeline to deploy here deploy\_role\_arn="arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-deploy-role-${AWS::AccountId}-${AWS::Region}", deploy\_role\_external\_id="", # ARN of the role used for file asset publishing (assumed from the deploy role) file\_asset\_publishing\_role\_arn="arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-file-publishing-role-${AWS::AccountId}-${AWS::Region}", file\_asset\_publishing\_external\_id="", # ARN of the role used for Docker asset publishing (assumed from the deploy role) image\_asset\_publishing\_role\_arn="arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-image-publishing-role-${AWS::AccountId}-${AWS::Region}", image\_asset\_publishing\_external\_id="", # ARN of the role passed to CloudFormation to execute the deployments cloud\_formation\_execution\_role="arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-cfn-exec-role-${AWS::AccountId}-${AWS::Region}", # ARN of the role used to look up context information in an environment lookup\_role\_arn="arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-lookup-role-${AWS::AccountId}-${AWS::Region}", lookup\_role\_external\_id="", # Name of the SSM parameter which describes the bootstrap stack version number bootstrap\_stack\_version\_ssm\_parameter="/cdk-bootstrap/${Qualifier}/version", # Add a rule to every template which verifies the required bootstrap stack version generate\_bootstrap\_version\_rule=True, )

Java

DefaultStackSynthesizer.Builder.create() // Name of the S3 bucket for file assets .fileAssetsBucketName("cdk-${Qualifier}-assets-${AWS::AccountId}-${AWS::Region}") .bucketPrefix('') // Name of the ECR repository for Docker image assets .imageAssetsRepositoryName("cdk-${Qualifier}-container-assets-${AWS::AccountId}-${AWS::Region}") // ARN of the role assumed by the CLI and Pipeline to deploy here .deployRoleArn("arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-deploy-role-${AWS::AccountId}-${AWS::Region}") .deployRoleExternalId("") // ARN of the role used for file asset publishing (assumed from the deploy role) .fileAssetPublishingRoleArn("arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-file-publishing-role-${AWS::AccountId}-${AWS::Region}") .fileAssetPublishingExternalId("") // ARN of the role used for Docker asset publishing (assumed from the deploy role) .imageAssetPublishingRoleArn("arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-image-publishing-role-${AWS::AccountId}-${AWS::Region}") .imageAssetPublishingExternalId("") // ARN of the role passed to CloudFormation to execute the deployments .cloudFormationExecutionRole("arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-cfn-exec-role-${AWS::AccountId}-${AWS::Region}") .lookupRoleArn("arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-lookup-role-${AWS::AccountId}-${AWS::Region}") .lookupRoleExternalId("") // Name of the SSM parameter which describes the bootstrap stack version number .bootstrapStackVersionSsmParameter("/cdk-bootstrap/${Qualifier}/version") // Add a rule to every template which verifies the required bootstrap stack version .generateBootstrapVersionRule(true) .build()

C#

new DefaultStackSynthesizer(new DefaultStackSynthesizerProps { // Name of the S3 bucket for file assets FileAssetsBucketName = "cdk-${Qualifier}-assets-${AWS::AccountId}-${AWS::Region}", BucketPrefix = "", // Name of the ECR repository for Docker image assets ImageAssetsRepositoryName = "cdk-${Qualifier}-container-assets-${AWS::AccountId}-${AWS::Region}", // ARN of the role assumed by the CLI and Pipeline to deploy here DeployRoleArn = "arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-deploy-role-${AWS::AccountId}-${AWS::Region}", DeployRoleExternalId = "", // ARN of the role used for file asset publishing (assumed from the deploy role) FileAssetPublishingRoleArn = "arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-file-publishing-role-${AWS::AccountId}-${AWS::Region}", FileAssetPublishingExternalId = "", // ARN of the role used for Docker asset publishing (assumed from the deploy role) ImageAssetPublishingRoleArn = "arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-image-publishing-role-${AWS::AccountId}-${AWS::Region}", ImageAssetPublishingExternalId = "", // ARN of the role passed to CloudFormation to execute the deployments CloudFormationExecutionRole = "arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-cfn-exec-role-${AWS::AccountId}-${AWS::Region}", LookupRoleArn = "arn:${AWS::Partition}:iam::${AWS::AccountId}:role/cdk-${Qualifier}-lookup-role-${AWS::AccountId}-${AWS::Region}", LookupRoleExternalId = "", // Name of the SSM parameter which describes the bootstrap stack version number BootstrapStackVersionSsmParameter = "/cdk-bootstrap/${Qualifier}/version", // Add a rule to every template which verifies the required bootstrap stack version GenerateBootstrapVersionRule = true, })

## The bootstrapping template contract

The requirements of the bootstrapping stack depend on the stack synthesizer in use. If you write your own stack synthesizer, you have complete control of the bootstrap resources that your synthesizer requires and how the synthesizer finds them. This section describes the expectations that the DefaultStackSynthesizer has of the bootstrapping template.

### Versioning

The template should contain a resource to create an SSM parameter with a well-known name and an output to reflect the template’s version.

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]

### Roles

The DefaultStackSynthesizer requires five IAM roles for five different purposes. If you are not using the default roles, the synthesizer needs to be told the ARNs for the roles you want to use. The roles are:

* The *deployment role* is assumed by the AWS CDK Toolkit and by AWS CodePipeline to deploy into an environment. Its AssumeRolePolicy controls who can deploy into the environment. The permissions this role needs can be seen in the template.
* The *lookup role* is assumed by the AWS CDK Toolkit 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 AWS CDK Toolkit and by AWS CodeBuild projects to publish assets into an environment: that is, to write to the S3 bucket and the 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.

### Outputs

The AWS CDK Toolkit 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

### 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 to ensure that yours continues to work with all CDK features. This section contains a list of the changes made in each version.

Template version

AWS CDK version

Changes

**1**

1.40.0

Initial version of template with Bucket, Key, Repository and Roles.

**2**

1.45.0

Split asset publishing role into separate file and image publishing roles.

**3**

1.46.0

Add FileAssetKeyArn export to be able to add decrypt permissions to asset consumers.

**4**

1.61.0

KMS permissions are now implicit via S3 and no longer require FileAsetKeyArn, Add CdkBootstrapVersion SSM parameter so the bootstrap stack version can be verified without knowing the stack name.

**5**

1.87.0

Deployment role can read SSM parameter.

**6**

1.108.0

Add lookup role separate from deployment role.

**6**

1.109.0

Attach aws-cdk:bootstrap-role tag to deployment, file publishing, and image publishing roles.

**7**

1.110.0

Deployment role can no longer read Buckets in the target account directly (however, this role is effectively an administrator, and could always use its AWS CloudFormation permissions to make the bucket readable anyway).

**8**

1.114.0

The lookup role has full read-only permissions to the target environment, and has a aws-cdk:bootstrap-role tag as well.

**9**

2.1.0

Fixes S3 asset uploads from being rejected by commonly referenced encryption SCP.

**10**

2.4.0

ECR ScanOnPush is now enabled by default.

**11**

2.18.0

Adds policy allowing Lambda to pull from Amazon ECR repos so it survives rebootstrapping.

**12**

2.20.0

Adds support for experimental **cdk import**.

Warning **Javascript is disabled or is unavailable in your browser.**

To use the Amazon Web Services Documentation, Javascript must be enabled. Please refer to your browser’s Help pages for instructions.

[Document Conventions](/general/latest/gr/docconventions.html)

Escape hatches

Best practices

Did this page help you? - Yes

Thanks for letting us know we’re doing a good job!

If you’ve got a moment, please tell us what we did right so we can do more of it.

Did this page help you? - No

Thanks for letting us know this page needs work. We’re sorry we let you down.

If you’ve got a moment, please tell us how we can make the documentation better.