include::attributes.txt[]

// Attributes

[.topic] [[best-practices-security,best-practices-security.title]] = {aws} CDK security best practices :info\_titleabbrev: Security :keywords: {aws} CDK, IAM, security, permissions, infrastructure, {aws} CloudFormation, {aws} CDK deployments

== [abstract]

## The {aws} Cloud Development Kit ({aws} CDK) is a powerful tool that developers can use to configure {aws} services and provision infrastructure on {aws}. With any tool that provides such control and capabilities, organizations will need to establish policies and practices to ensure that the tool is being used in safe and secure ways. For example, organizations may want to restrict developer access to specific services to ensure that they can’t tamper with compliance or cost control measures that are configured in the account.

// Content start

The {aws} Cloud Development Kit ({aws} CDK) is a powerful tool that developers can use to configure {aws} services and provision infrastructure on {aws}. With any tool that provides such control and capabilities, organizations will need to establish policies and practices to ensure that the tool is being used in safe and secure ways. For example, organizations may want to restrict developer access to specific services to ensure that they can’t tamper with compliance or cost control measures that are configured in the account.

Often, there can be a tension between security and productivity, and each organization needs to establish the proper balance for themselves. This topic provides security best practices for the {aws} CDK that you can consider as you create and implement your own security policies. The following best practices are general guidelines and don`’t represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful considerations rather than prescriptions.

[#best-practices-security-iam] == Follow IAM security best practices

{aws} Identity and Access Management (IAM) is a web service that helps you securely control access to {aws} resources. Organizations, individuals, and the {aws} CDK use IAM to manage permissions that determine the actions that can be performed on {aws} resources. When using IAM, follow the IAM security best practices. For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/IAMBestPracticesAndUseCases.html[Security best practices and use cases in {aws} Identity and Access Management] in the *IAM User Guide*.

[#best-practices-security-permissions] == Manage permissions for the {aws} CDK

When you use the {aws} CDK across your organization to develop and manage your infrastructure, you’ll want to consider the following scenarios where managing permissions will be important:

* *Permissions for {aws} CDK deployments* – These permissions determine who can make changes to your {aws} resources and what changes they can make.
* *Permissions between resources* – These are the permissions that allow interactions between the {aws} resources that you create and manage with the {aws} CDK.

[#best-practices-security-permissions-deployments] === Manage permissions for {aws} CDK deployments

Developers use the {aws} CDK to define infrastructure locally on their development machines. This infrastructure is implemented in {aws} environments through deployments that typically involve using the {aws} CDK Command Line Interface ({aws} CDK CLI). With deployments, you may want to control what changes developers can make in your environments. For example, you might have an Amazon Virtual Private Cloud (Amazon VPC) resource that you don’t want developers to modify.

By default, the CDK CLI uses a combination of the actor’s security credentials and IAM roles that are created during bootstrapping to receive permissions for deployments. The actor’s security credentials are first used for authentication and IAM roles are then assumed to perform various actions during deployment, such as using the {aws} CloudFormation service to create resources. For more information on how CDK deployments work, including the IAM roles that are used, see xref:deploy[Deploy {aws} CDK applications].

To restrict who can perform deployments and the actions that can be performed during deployment, consider the following:

* The actor’s security credentials are the first set of credentials used to authenticate to {aws}. From here, the permissions used to perform actions during deployment are granted to the IAM roles that are assumed during the deployment workflow. You can restrict who can perform deployments by limiting who can assume these roles. You can also restrict the actions that can be performed during deployment by replacing these IAM roles with your own.
* Permissions for performing deployments are given to the DeploymentActionRole. You can control permissions for who can perform deployments by limiting who can assume this role. By using a role for deployments, you can perform cross-account deployments since the role can be assumed by {aws} identities in a different account. By default, all identities in the same {aws} account with the appropriate AssumeRole policy statement can assume this role.
* Permissions for creating and modifying resources through {aws} CloudFormation are given to the CloudFormationExecutionRole. This role also requires permission to read from the bootstrap resources. You control the permissions that CDK deployments have by using a managed policy for the CloudFormationExecutionRole and optionally by configuring a permissions boundary. By default, this role has AdministratorAccess permissions with no permission boundary.
* Permissions for interacting with bootstrap resources are given to the FilePublishingRole and ImagePublishingRole. The actor performing deployments must have permission to assume these roles. By default, all identities in the same {aws} account with the appropriate AssumeRole policy statement can assume this role.
* Permissions for accessing bootstrap resources to perform lookups are given to the LookupRole. The actor performing deployments must have permission to assume this role. By default, this role has readOnly access to the bootstrap resources. By default, all identities in the same {aws} account with the appropriate AssumeRole policy statement can assume this role.

To configure the IAM identities in your {aws} account with permission to assume these roles, add a policy with the following policy statement to the identities:

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

{ “Version”: “2012-10-17”, “Statement”: [{ “Sid”: “AssumeCDKRoles”, “Effect”: “Allow”, “Action”: “sts:AssumeRole”, “Resource”: “\*“,”Condition”: { “StringEquals”: { “iam:ResourceTag/aws-cdk:bootstrap-role”: [ “image-publishing”, “file-publishing”, “deploy”, “lookup” ] } } }] } —

[#best-practices-security-permissions-deployments-roles] ==== Modify the permissions for the roles assumed during deployment

By modifying permissions for the roles assumed during deployment, you can manage the actions that can be performed during deployment. To modify permissions, you create your own IAM roles and specify them when bootstrapping your environment. When you customize bootstrapping, you will have to customize synthesis. For general instructions, see xref:bootstrapping-customizing[Customize {aws} CDK bootstrapping].

[#best-practices-security-permissions-deployments-creds] ==== Modify the security credentials and roles used during deployment

The roles and bootstrap resources that are used during deployments are determined by the CDK stack synthesizer that you use. To modify this behavior, you can customize synthesis. For more information, see xref:configure-synth[Configure and perform CDK stack synthesis].

[#best-practices-security-permissions-deployments-least] ==== Considerations for granting least privilege access

Granting least privilege access is a security best practice that we recommend that you consider as you develop your security strategy. For more information, see link:https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/sec\_permissions\_least\_privileges.html[SEC03-BP02 Grant least privilege access] in the *{aws} Well-Architected Framework Guide*.

Often, granting least privilege access involves restricting IAM policies to the minimum access necessary to perform a given task. Attempting to grant least privilege access through fine-grained permissions with the CDK using this approach can impact CDK deployments and cause you to have to create wider-scoped permissions than you`’d like. The following are a few things to consider when using this approach:

* Determining an exhaustive list of permissions that allow developers to use the {aws} CDK to provision infrastructure through CloudFormation is difficult and complex.
* If you want to be fine-grained, permissions may become too long to fit within the maximum length of IAM policy documents.
* Providing an incomplete set of permissions can severely impact developer productivity and deployments.

With the CDK, deployments are performed using CloudFormation. CloudFormation initiates a set of {aws} API calls in order using the permissions that are provided. The permissions necessary at any point in time depends on many factors:

* The {aws} services that are being modified. Specifically, the resources and properties that are being used and changed.
* The current state of the CloudFormation stack.
* Issues that may occur during deployments and if rollbacks are needed, which will require Delete permissions in addition to Create.

When the provided permissions are incomplete, manual intervention will be required. The following are a few examples:

* If you discover incomplete permissions during roll forward, you’ll need to pause deployment, and take time to discuss and provision new permissions before continuing.
* If deployment rolls back and the permissions to apply the roll back are missing, it may leave your CloudFormation stack in a state that will require a lot of manual work to recover from.

Since this approach can result in complications and severely limit developer productivity, we don’t recommend it. Instead, we recommend implementing guardrails and preventing bypass.

[#best-practices-security-permissions-deployments-guardrails] ==== Implementing guardrails and preventing bypass

You can implement guardrails, compliance rules, auditing, and monitoring by using services such as {aws} Control Tower, {aws} Config, {aws} CloudTrail, {aws} Security Hub, and others. With this approach, you grant developers permission to do everything, except tampering with the existing validation mechanisms. Developers have the freedom to implement changes quickly, as long as they stay within policy. This is the approach we recommend when using the {aws} CDK. For more information on guardrails, see https://docs.aws.amazon.com/wellarchitected/latest/management-and-governance-guide/controls.html[Controls] in the *Management and Governance Cloud Environment Guide*.

We also recommend using *permissions boundaries* or *service control policies (SCPs)* as a way of implementing guardrails. For more information on implementing permissions boundaries with the {aws} CDK, see xref:customize-permissions-boundaries[Create and apply permissions boundaries for the {aws} CDK].

If you are using any compliance control mechanisms, set them up during the bootstrapping phase. Make sure that the CloudFormationExecutionRole or developer-accessible identities have policies or permissions boundaries attached that prevent bypass of the mechanisms that you put in place. The appropriate policies depends on the specific mechanisms that you use.

[#best-practices-security-permissions-resources] === Manage permissions between resources provisioned by the {aws} CDK

How you manage permissions between resources that are provisioned by the {aws} CDK depends on whether you allow the CDK to create roles and policies.

When you use L2 constructs from the {aws} Construct Library to define your infrastructure, you can use the provided grant methods to provision permissions between resources. With grant methods, you specify the type of access you want between resources and the {aws} CDK provisions least privilege IAM roles to accomplish your intent. This approach meets security requirements for most organizations while being efficient for developers. For more information, see xref:define-iam-l2[Define permissions for L2 constructs with the {aws} CDK].

If you want to work around this feature by replacing the automatically generated roles with manually created ones, consider the following:

* Your IAM roles will need to be manually created, slowing down application development.
* When IAM roles need to be manually created and managed, people will often combine multiple logical roles into a single role to make them easier to manage. This runs counter to the least privilege principle.
* Since these roles will need to be created before deployment, the resources that need to be referenced will not yet exist. Therefore, you`’ll need to use wildcards, which runs counter to the least privilege principle.
* A common workaround to using wildcards is to mandate that all resources be given a predictable name. However, this interferes with CloudFormation`’s ability to replace resources when necessary and may slow down or block development. Because of this, we recommend that you allow CloudFormation to create unique resource names for you.
* It will be impossible to perform continuous delivery since manual actions must be performed prior to every deployment.

When organizations want to prevent the CDK from creating roles, it is usually to prevent developers from being able to create IAM roles. The concern is that by giving developers permission to create IAM roles using the {aws} CDK, they could possibly elevate their own privileges. To mitigate against this, we recommend using *permission boundaries* or *service control policies (SCPs)*. With permission boundaries, you can set limits for what developers and the CDK are allowed to do. For more information on using permission boundaries with the CDK, see xref:customize-permissions-boundaries[Create and apply permissions boundaries for the {aws} CDK].