POLICIES IN AWS

**TOPICS**

1. [What is a Policy](#_Policy)
2. [Identity Based Policies](#_Identity-based_policies)
3. [Resource Based Polices](#_Resource-based_policies)
4. [AWS Managed Policies](#_AWS_managed_policies)
5. [Customer Managed Policies](#_Customer_managed_policies)
6. [Inline Policies](#_Inline_policies)
7. [Control Access Using Policies](#_Controlling_access_to)

## Policy

A policy is an object in AWS that, when associated with an entity or resource, defines their permissions. AWS evaluates these policies when a principal, such as a user, makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, if a policy allows the [GetUser](https://docs.aws.amazon.com/IAM/latest/APIReference/API_GetUser.html) action, then a user with that policy can get user information from the AWS Management Console, the AWS CLI, or the AWS API. When you create an IAM user, you can set up the user to allow console or programmatic access. The IAM user can sign in to the console using a user name and password. Or they can use access keys to work with the CLI or API.

The following policy types, listed in order of frequency, can affect whether a request is authorized. For more details, see [Policy Types](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html#access_policy-types) in the IAM User Guide.

* **Identity-based policies** – You can attach managed and inline policies to IAM identities (users, groups to which users belong, and roles).
* **Resource-based policies** – You can attach inline policies to resources in some AWS services. The most common examples of resource-based policies are Amazon S3 bucket policies and IAM role trust policies. AWS RoboMaker does not support resource-based policies.
* **Organizations SCPs** – You can use an AWS Organizations service control policy (SCP) to apply a permissions boundary to an AWS Organizations organization or organizational unit (OU). Those permissions are applied to all entities within the member accounts.
* **Access control lists (ACLs)** – You can use ACLs to control what principals can access a resource. ACLs are similar to resource-based policies, although they are the only policy type that does not use the JSON policy document structure. AWS RoboMaker does not support ACLs.

These policies types can be categorized as permissions policies or permissions boundaries.

* **Permissions policies** – You can attach permissions policies to a resource in AWS to define the permissions for that object. Within a single account, AWS evaluates all permissions policies together. Permissions policies are the most common policies. You can use the following policy types as permissions policies:
  + **Identity-based policies** – When you attach a managed or inline policy to an IAM user, group, or role, the policy defines the permissions for that entity.
  + **Resource-based policies** – When you attach a JSON policy document to a resource, you define the permissions for that resource. The service must support resource-based policies.
  + **Access control lists (ACLs)** – When you attach an ACL to a resource, you define a list of principals with permission to access that resource. The resource must support ACLs.
* **Permissions boundaries** – You can use policies to define the permissions boundary for an entity (user or role). A permissions boundary controls the maximum permissions that an entity can have. Permissions boundaries are an advanced AWS feature. When more than one permissions boundaries applies to a request, AWS evaluates each permissions boundary separately. You can apply a permissions boundary in the following situations:
  + **Organizations** – You can use an AWS Organizations service control policy (SCP) to apply a permissions boundary to an AWS Organizations organization or organizational unit (OU).
  + **IAM users or roles** – You can use a managed policy for a user or role's permissions boundary. For more information, see [Permissions Boundaries for IAM Entities](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_boundaries) in the IAM User Guide

## Identity-based policies

You can attach policies to IAM identities. For example, you can do the following:

* **Attach a permissions policy to a user or a group in your account** – To grant a user permissions to create an AWS RoboMaker resource, such as a robot applications, you can attach a permissions policy to a user or a group to which the user belongs.
* **Attach a permissions policy to a role (grant cross-account permissions)** – You can attach an identity-based permissions policy to an IAM role to grant cross-account permissions. For example, the administrator in account A can create a role to grant cross-account permissions to another AWS account (for example, account B) or an AWS service as follows:
  1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions on resources in account A.
  2. Account A administrator attaches a trust policy to the role identifying account B as the principal who can assume the role.
  3. Account B administrator can then delegate permissions to assume the role to any users in account B. Doing this allows users in account B to create or access resources in account A. The principal in the trust policy can also be an AWS service principal if you want to grant an AWS service permissions to assume the role.

For more information about using IAM to delegate permissions, see [Access Management](https://docs.aws.amazon.com/IAM/latest/UserGuide/access.html) in the IAM User Guide.

For more information about users, groups, roles, and permissions, see [Identities (Users, Groups, and Roles)](https://docs.aws.amazon.com/IAM/latest/UserGuide/id.html) in the IAM User Guide.

## Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. These policies allow you to specify what actions a specified principal can perform on that resource and under what conditions. The most commonly-known resource-based policy is an Amazon S3 bucket. Resource-based policies are inline policies that exist only on the resource. There are no managed resource-based policies.

Granting permissions to members of other AWS accounts using a resource-based policy has some advantages over an IAM role. For more information, see [How IAM Roles Differ from Resource-based Policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_compare-resource-policies.html) in the IAM User Guide.

AWS RoboMaker does not support resource-based policies.

## AWS managed policies

An AWS managed policy is a standalone policy that is created and administered by AWS. Standalone policy means that the policy has its own Amazon Resource Name (ARN) that includes the policy name. For example, arn:aws:iam::aws:policy/IAMReadOnlyAccess is an AWS managed policy. For more information about ARNs, see [IAM ARNs](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#identifiers-arns).

AWS managed policies are designed to provide permissions for many common use cases. Full access AWS managed policies such as [AmazonDynamoDBFullAccess](https://console.aws.amazon.com/iam/home" \l "policies/arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess" \t "_blank) and [IAMFullAccess](https://console.aws.amazon.com/iam/home" \l "policies/arn:aws:iam::aws:policy/IAMFullAccess" \t "_blank) define permissions for service administrators by granting full access to a service. Power-user AWS managed policies such as [AWSCodeCommitPowerUser](https://console.aws.amazon.com/iam/home#policies/arn:aws:iam::aws:policy/AWSCodeCommitPowerUser) and [AWSKeyManagementServicePowerUser](https://console.aws.amazon.com/iam/home#policies/arn:aws:iam::aws:policy/AWSKeyManagementServicePowerUser) are designed for power users. Partial-access AWS managed policies such as [AmazonMobileAnalyticsWriteOnlyAccess](https://console.aws.amazon.com/iam/home#policies/arn:aws:iam::aws:policy/AmazonMobileAnalyticsWriteOnlyAccess) and [AmazonEC2ReadOnlyAccess](https://console.aws.amazon.com/iam/home#policies/arn:aws:iam::aws:policy/AmazonEC2ReadOnlyAccess) provide specific levels of access to AWS services without allowing [permissions management](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_understand-policy-summary-access-level-summaries.html#access_policies_access-level) access level permissions. AWS managed policies make it easier for you to assign appropriate permissions to users, groups, and roles than if you had to write the policies yourself.

One particularly useful category of AWS managed policies are those designed for job functions. These policies align closely to commonly used job functions in the IT industry. The intent is to make granting permissions for these common job functions easy. One key advantage of using job function policies is that they are maintained and updated by AWS as new services and API operations are introduced. For example, the [AdministratorAccess](https://console.aws.amazon.com/iam/home" \l "policies/arn:aws:iam::aws:policy/AdministratorAccess" \t "_blank) job function provides full access and permissions delegation to every service and resource in AWS. We recommend that this policy is used only for the account administrator. For power users that require full access to every service except limited access to IAM and Organizations, use the [PowerUserAccess](https://console.aws.amazon.com/iam/home" \l "policies/arn:aws:iam::aws:policy/PowerUserAccess" \t "_blank) job function. For a list and descriptions of the job function policies, see [AWS managed policies for job functions](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_job-functions.html).

You cannot change the permissions defined in AWS managed policies. AWS occasionally updates the permissions defined in an AWS managed policy. When AWS does this, the update affects all principal entities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API calls become available for existing services. For example, the AWS managed policy called **ReadOnlyAccess** provides read-only access to all AWS services and resources. When AWS launches a new service, AWS updates the **ReadOnlyAccess** policy to add read-only permissions for the new service. The updated permissions are applied to all principal entities that the policy is attached to.

The following diagram illustrates AWS managed policies. The diagram shows three AWS managed policies: **AdministratorAccess**, **PowerUserAccess**, and **AWSCloudTrailReadOnlyAccess**. Notice that a single AWS managed policy can be attached to principal entities in different AWS accounts, and to different principal entities in a single AWS account.


        Diagram of AWS managed policies
      

## Customer managed policies

You can create standalone policies that you administer in your own AWS account, which we refer to as customer managed policies. You can then attach the policies to multiple principal entities in your AWS account. When you attach a policy to a principal entity, you give the entity the permissions that are defined in the policy.

A great way to create a customer managed policy is to start by copying an existing AWS managed policy. That way you know that the policy is correct at the beginning and all you need to do is customize it to your environment.

The following diagram illustrates customer managed policies. Each policy is an entity in IAM with its own [Amazon Resource Name (ARN)](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#identifiers-arns) that includes the policy name. Notice that the same policy can be attached to multiple principal entities—for example, the same **DynamoDB-books-app** policy is attached to two different IAM roles.


        Diagram of customer managed policies
      

## Inline policies

An inline policy is a policy that's embedded in an IAM identity (a user, group, or role). That is, the policy is an inherent part of the identity. You can create a policy and embed it in an identity, either when you create the identity or later.

The following diagram illustrates inline policies. Each policy is an inherent part of the user, group, or role. Notice that two roles include the same policy (the **DynamoDB-books-app** policy), but they are not sharing a single policy; each role has its own copy of the policy.


        Diagram of inline policies
      

# **Controlling access to AWS resources using policies**

You can use a policy to control access to resources within IAM or all of AWS.

To use a [policy](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html) to control access in AWS, you must understand how AWS grants access. AWS is composed of collections of resources. An IAM user is a resource. An Amazon S3 bucket is a resource. When you use the AWS API, the AWS CLI, or the AWS Management Console to perform an operation (such as creating a user), you send a request for that operation. Your request specifies an action, a resource, a principal entity (user or role), a principal account, and any necessary request information. All of this information provides context.

AWS then checks that you (the principal) are authenticated (signed in) and authorized (have permission) to perform the specified action on the specified resource. During authorization, AWS checks all the policies that apply to the context of your request. Most policies are stored in AWS as [JSON documents](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html#access_policies-json) and specify the permissions for principal entities. For more information about policy types and uses, see [Policies and permissions in IAM](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html).

AWS authorizes the request only if each part of your request is allowed by the policies. To view a diagram of this process, see [Understanding how IAM works](https://docs.aws.amazon.com/IAM/latest/UserGuide/intro-structure.html). For details about how AWS determines whether a request is allowed, see [Policy evaluation logic](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_evaluation-logic.html).

When you create an IAM policy, you can control access to the following:

* [**Principals**](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_controlling.html#access_controlling-principals) – Control what the person making the request (the [principal](https://docs.aws.amazon.com/IAM/latest/UserGuide/intro-structure.html#intro-structure-principal)) is allowed to do.
* [**IAM Identities**](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_controlling.html#access_controlling-identities) – Control which IAM identities (user groups, users, and roles) can be accessed and how.
* [**IAM Policies**](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_controlling.html#access_controlling-policies) – Control who can create, edit, and delete customer managed policies, and who can attach and detach all managed policies.
* [**AWS Resources**](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_controlling.html#access_controlling-resources) – Control who has access to resources using an identity-based policy or a resource-based policy.
* [**AWS Accounts**](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_controlling.html#access_controlling-principal-accounts) – Control whether a request is allowed only for members of a specific account.

Policies let you specify who has access to AWS resources, and what actions they can perform on those resources. Every IAM user starts with no permissions. In other words, by default, users can do nothing, not even view their own access keys. To give a user permission to do something, you can add the permission to the user (that is, attach a policy to the user). Or you can add the user to a user group that has the intended permission.

For example, you might grant a user permission to list his or her own access keys. You might also expand that permission and also let each user create, update, and delete their own keys.

When you give permissions to a user group, all users in that user group get those permissions. For example, you can give the Administrators user group permission to perform any of the IAM actions on any of the AWS account resources. Another example: You can give the Managers user group permission to describe the AWS account's Amazon EC2 instances.

For information about how to delegate basic permissions to your users, user groups, and roles, see [Permissions required to access IAM resources](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_permissions-required.html). For additional examples of policies that illustrate basic permissions, see [Example policies for administering IAM resources](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_delegate-permissions_examples.html).

You can use [IAM policies](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html) to control who is allowed to create, update, and delete customer managed policies in your AWS account. The following list contains API operations that pertain directly to creating, updating, and deleting policies or policy versions:

* [CreatePolicy](https://docs.aws.amazon.com/IAM/latest/APIReference/API_CreatePolicy.html)
* [CreatePolicyVersion](https://docs.aws.amazon.com/IAM/latest/APIReference/API_CreatePolicyVersion.html)
* [DeletePolicy](https://docs.aws.amazon.com/IAM/latest/APIReference/API_DeletePolicy.html)
* [DeletePolicyVersion](https://docs.aws.amazon.com/IAM/latest/APIReference/API_DeletePolicyVersion.html)
* [SetDefaultPolicyVersion](https://docs.aws.amazon.com/IAM/latest/APIReference/API_SetDefaultPolicyVersion.html)

The API operations in the preceding list correspond to actions that you can allow or deny—that is, permissions that you can grant—using an IAM policy.

Consider the following example policy. It allows a user to create, update (that is, create a new policy version), delete, and set a default version for all customer managed policies in the AWS account. The example policy also allows the user to list policies and get policies. To learn how to create a policy using this example JSON policy document, see [Creating policies on the JSON tab](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_create-console.html#access_policies_create-json-editor).

You can create policies that limit the use of these API operations to affect only the managed policies that you specify. For example, you might want to allow a user to set the default version and delete policy versions, but only for specific customer managed policies. You do this by specifying the policy ARN in the Resource element of the policy that grants these permissions.

The following example shows a policy that allows a user to delete policy versions and set the default version. But these actions are only allowed for the customer managed policies that include the path /TEAM-A/. The customer managed policy ARN is specified in the Resource element of the policy. (In this example the ARN includes a path and a wildcard and thus matches all customer managed policies that include the path /TEAM-A/). To learn how to create a policy using this example JSON policy document, see [Creating policies on the JSON tab](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_create-console.html#access_policies_create-json-editor).

For more information about using paths in the names of customer managed policies, see [Friendly names and paths](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#identifiers-friendly-names).

You can also use IAM policies to allow users to work with only specific managed policies. In effect, you can control which permissions a user is allowed to grant to other principal entities.

The following list shows API operations that pertain directly to attaching and detaching managed policies to and from principal entities:

* [AttachGroupPolicy](https://docs.aws.amazon.com/IAM/latest/APIReference/API_AttachGroupPolicy.html)
* [AttachRolePolicy](https://docs.aws.amazon.com/IAM/latest/APIReference/API_AttachRolePolicy.html)
* [AttachUserPolicy](https://docs.aws.amazon.com/IAM/latest/APIReference/API_AttachUserPolicy.html)
* [DetachGroupPolicy](https://docs.aws.amazon.com/IAM/latest/APIReference/API_DetachGroupPolicy.html)
* [DetachRolePolicy](https://docs.aws.amazon.com/IAM/latest/APIReference/API_DetachRolePolicy.html)
* [DetachUserPolicy](https://docs.aws.amazon.com/IAM/latest/APIReference/API_DetachUserPolicy.html)

You can create policies that limit the use of these API operations to affect only the specific managed policies and/or principal entities that you specify. For example, you might want to allow a user to attach managed policies, but only the managed policies that you specify. Or, you might want to allow a user to attach managed policies, but only to the principal entities that you specify.

The following example policy allows a user to attach managed policies to only the user groups and roles that include the path /TEAM-A/. The user group and role ARNs are specified in the Resource element of the policy. (In this example the ARNs include a path and a wildcard character and thus match all user groups and roles that include the path /TEAM-A/). To learn how to create a policy using this example JSON policy document, see [Creating policies on the JSON tab](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_create-console.html#access_policies_create-json-editor).

You can further limit the actions in the preceding example to affect only specific policies. That is, you can control which permissions a user is allowed to attach to other principal entities—by adding a condition to the policy.

In the following example, the condition ensures that the AttachGroupPolicy and AttachRolePolicy permissions are allowed only when the policy being attached matches one of the specified policies. The condition uses the iam:PolicyARN [condition key](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_condition.html) to determine which policy or policies are allowed to be attached. The following example policy expands on the previous example. It allows a user to attach only the managed policies that include the path /TEAM-A/ to only the user groups and roles that include the path /TEAM-A/.

This policy uses the ArnLike condition operator because the ARN includes a wildcard character. For a specific ARN, use the ArnEquals condition operator. For more information about ArnLike and ArnEquals, see [Amazon Resource Name (ARN) condition operators](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_condition_operators.html#Conditions_ARN) in the *Condition Types* section of the *Policy Element Reference*.

For example, you can limit the use of actions to involve only the managed policies that you specify. You do this by specifying the policy ARN in the Condition element of the policy that grants these permissions.

References:

* https://docs.aws.amazon.com/