Overview

A policy is an object in AWS that, when associated with an **identity** or **resource**, defines its permissions.

Policy contains permissions, which determines whether the request is allowed or denied

IAM Policies specify what you are allowed to do with any AWS resources.

They are **Global** and apply to all areas of AWS.

Policies are attached to either **Identities** (Users, Group, or Roles) or **Resources** (ex. S3)

Policy Types

1. Identity-based policies

Attach **managed and inline policies** to IAM **identities** (users, groups to which users belong, or roles).
Identity-based policies grant permissions to an identity.

2. Resource-based policies

Attach **inline policies** to **resources**.
The most common examples of resource-based policies are Amazon S3 bucket policies and IAM role trust policies.
Resource-based policies grant permissions to the principal that is specified in the policy.
Principals can be in the same account as the resource or in other accounts.

3. Permissions boundaries

Use a managed policy as the permissions boundary for an IAM entity (user or role).
That policy defines the maximum permissions that the **identity-based policies** can grant to an entity, but **does not grant permissions**.
Permissions boundaries **do not define the maximum permissions that a resource-based policy** can grant to an entity.

4. Organizations SCPs

Use an AWS Organizations service control policy (SCP) to define the maximum permissions for account members of an organization or organizational unit (OU).
SCPs limit permissions that identity-based policies or resource-based policies grant to entities (users or roles) within the account, **but do not grant permissions**.

5. Access control lists (ACLs)

Use ACLs to control which principals in other accounts can access the resource to which the ACL is attached.
ACLs are similar to resource-based policies, although they are the only policy type that does not use the **JSON policy document structure**.
ACLs are cross-account permissions policies that grant permissions to the specified principal.
ACLs **cannot grant permissions to entities within the same account.**

6. Session policies

Pass advanced session policies when you use the AWS CLI or AWS API to assume a role or a federated user.
Session policies **limit the permissions that the role or user's identity-based policies grant to the session**.
Session policies **limit permissions for a created session, but do not grant permissions.**

1. Identity based policies

These are the policies that you **can attach to an Identity.** These can define what Identity can do.Ex: Users, Groups, Roles

***Identity based policies control* what actions the identity can perform, on which resources, and under what conditions**

These can be further categorized into the following 2 types

A. Managed Policies
B. Inline Policies

A.Managed Policies:

These are standalone identity based policies that you can attach to:

- **Multiple** Users
- Groups
- Roles

These can be divided in two categories as below:

I. AWS Managed Policies:

These are created and managed by AWS. These are not modifiable.
These are **READ ONLY**

II. Customer Managed Policies:

These are created and managed by customers

B.Inline Policies:

These are created and managed that are embedded directly to: A **Single** **User, Group or a Role**.
One to one relation between user/group/role and policy.

In most cases, AWS doesn't recommend using inline policies.
Example for Identity based policy:

```
{
    "Version": "2012-10-17",
    "Statement": {
            "Effect": "Allow",
            "Action": "dynamodb:*",
            "Resource": "arn:aws:dynamodb:us-east-
2:123456789012:table/Books"
    }
}
```

EAR is Identity based policy document attributes**. E - Effect A - Action R - Resource**

Identity-based policies with boundaries

Identity-based policies grant permission to the entity, and permissions boundaries limit those permissions.

2. Resource based policy

These are the policies that **you can attach to a Resource**, such as S3 bucket

Resource-based policies are **inline** policies. There are **no managed resource-based** policies.

Resource-based policies **control** **what actions a specified principal can perform on that resource and under what conditions.**

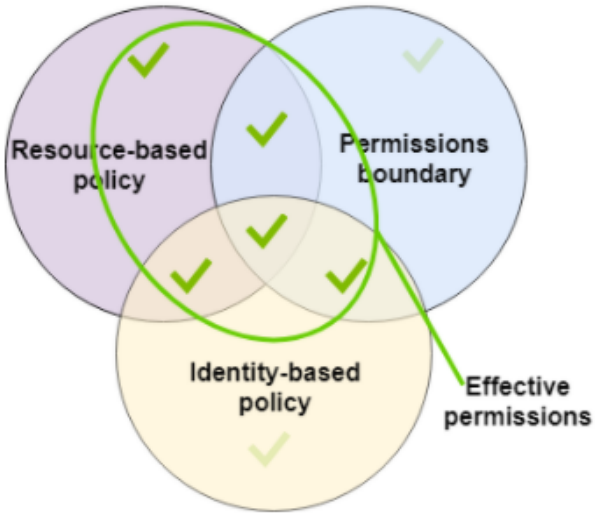
To enable **cross-account access**, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::777788889999:user/bob"},
        "Action": [
            "s3:PutObject",
            "s3:PutObjectAcl"
        ]
    }
```

EAP is Resource based policy document attributes**. E - Effect A - Action P - Principal**

Resource based policy and Identity-based policies with boundaries

In this case, the effective permissions are everything that is allowed by the resource-based policy *and* the intersection of the permissions boundary and the identity-based policy. An explicit deny in any of these policies overrides the allow.



3. Permissions Boundaries

A permissions boundary is an advanced feature in which you set the **maximum permissions** that an identity-based policy can grant to an IAM entity.
When you set a permissions boundary for an entity, the entity can perform only the actions that are allowed **by both its identity-based policies and its permissions boundaries**.
Resource-based policies that specify the user or role as the principal **are not limited by the permissions boundary**.
An explicit deny in any of these policies overrides the allow

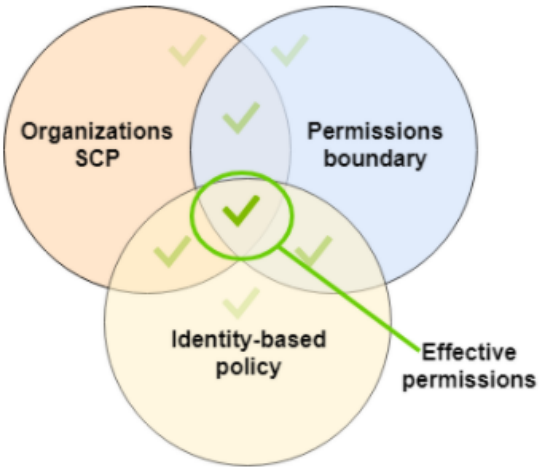
https://www.youtube.com/watch?v=_4Gw3T1otS4&t=826s&ab_channel=KnowledgeIndiaAWSAzureTutorials

4. Service control policies (SCP)

AWS Organizations is a service for grouping and centrally managing the AWS accounts that your business owns.
If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts.
SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU).
The **SCP limits permissions for entities in member accounts**, including each AWS account root user.
An explicit deny in any of these policies overrides the allow.

Service Control policy and Identity-based policies with boundaries

In this case, the request is allowed only if all three policy types allow it. The effective permissions are the intersection of all three policy types. An explicit deny in any of these policies overrides the allow.



5. Access control list (ACL)

Access control lists (ACLs) are service policies that allow you to control which principals in another account can access a resource

ACLs **cannot** be used to control access for a principal within the **same account**.
ACLs are similar to resource-based policies, although they are the **only policy type that does not use the JSON policy document format.**

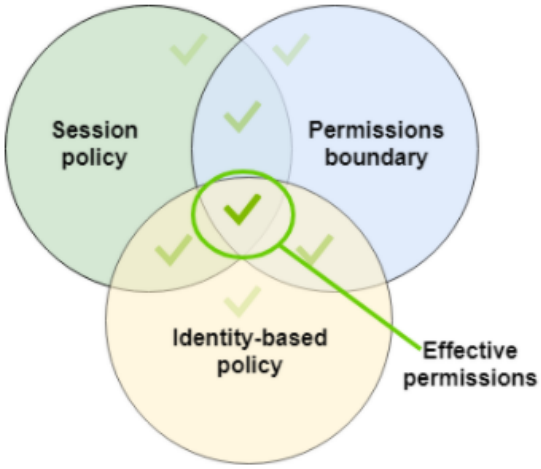
Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs.

6. Session policies

Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user.
The permissions for a session are the intersection of the identity-based policies for the IAM entity (user or role) used to create the session and the session policies.
Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow.

Session policy and Identity-based policies with boundaries

The effective permissions for this set of policy types are the intersection of all three policy types. An explicit deny in any of these policies overrides the allow. For more information about session policies



IAM Policy Simulator

Online tool to test your policy. It requires calling API operations to do two things:

1. Evaluate the policies and return the list of context keys that they reference.
2. Simulate the policies, providing a list of actions, resources, and context keys.

Use the aws iam simulate-custom-policy command

The IAM console includes ***policy summary tables*** that describe the access level, resources, and conditions that are allowed or denied for each service in a policy.

IAM Policy Simulator just reports the result Allowed or Denied

AWS CLI Dry Run

We can include a dry-run command in the CLI to ensure that policy has rights to perform operations before it actually does that operation.

Ex: Creating an EC2 instance using CLI. If we include --dry-run in the cli command we can ensure that the EC2 instance has a valid policy to create instances without actually creating it.

IAM Policy Variables

Policy variables provide a feature to specify placeholders in a policy.

When the policy is evaluated, the policy variables are replaced with values that come from the request itself

Policy variables allow a single policy to be applied to a group of users to control access for e.g. all user having access to S3 bucket folder with their name only

Policy variable is marked using a $ prefix followed by a pair of curly braces ({ }). Inside the ${ } characters, with the name of the value from the request that you want to use in the policy

Policy variables work only with policies defined with Version **2012-10-17**

Policy variables can only be used in the Resource element and in string comparisons in the Condition element

Policy variables are case sensitive and include variables like aws:username, aws:userid, aws:SourceIp, aws:CurrentTime etc.