IAM: Access Control & Policy Management

What is the difference between an allow, implicit and explicit deny in a policy?

Difference Between Allow and Deny in a Policy (e.g., AWS IAM):

- **Allow**: Grants permission to perform an action.
- **Deny**: Explicitly blocks an action, even if another policy allows it.
- **Implicit Deny**: By default, everything is denied unless explicitly allowed.
- **Explicit Deny**: Overrides all allows — it's the strongest rule.

🔐 Key Rules:

- ✅ *Explicit Allow* = Grants access
- ❌ *Explicit Deny* = Blocks access (even if allowed elsewhere)
- ⚠️ *Implicit Deny* = Default for everything not explicitly allowed

---

📜 Example IAM Policy:

```
json

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-bucket/*"
    },
    {
      "Effect": "Deny",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-bucket/private/*"
    }
  ]
```

```
}
```

What this means:

- ✅ Allows reading objects in `my-bucket`.
- ❌ Denies reading objects inside `my-bucket/private/`, **even though it's allowed above**.
- The **deny wins**.

What is the purpose of conditions in IAM policies?

Purpose of Conditions in IAM Policies:

- ✅ **Add fine-grained control** over when a permission is granted.
- ⏱️ **Restrict access** based on time, IP address, MFA, tags, etc.
- 🔒 **Enhance security** by enforcing context-based rules.
- 🛠️ **Customize behavior** for specific users, roles, or environments.

📄 Small Example: Allow S3 access only from a specific IP

```
{
  "Effect": "Allow",
  "Action": "s3:*",
  "Resource": "arn:aws:s3:::my-bucket/*",
  "Condition": {
    "IpAddress": {
      "aws:SourceIp": "203.0.113.0/24"
    }
  }
}
```

1. MFA Required (Multi-Factor Authentication)

```
json

{
  "Effect": "Allow",
  "Action": "ec2:StartInstances",
  "Resource": "*",
  "Condition": {
    "Bool": {
      "aws:MultiFactorAuthPresent": "true"
    }
```

```
  }
}
```

✅ **Purpose**: Allow EC2 start **only if** the user logged in with MFA.

---

2. Allow Based on Resource Tag

```
json
------------
{
  "Effect": "Allow",
  "Action": "ec2:TerminateInstances",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:ResourceTag/Project": "DevTeam"
    }
  }
}
```

✅ **Purpose**: Allow terminating EC2 instances **only if** they have the tag `Project=DevTeam`.

⏰ 3. Time-Based Access

```
json

{
  "Effect": "Allow",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::my-bucket/*",
  "Condition": {
    "DateGreaterThan": {
      "aws:CurrentTime": "2025-01-01T00:00:00Z"
    }
  }
}
```

✅ **Purpose**: Allow S3 upload **only after** 1st Jan 2025.

How do you attach policies to users, groups, or roles?

- Using AWS Console, CLI (aws iam attach-user-policy), or IaC tools like Terraform.

What's the difference between identity-based and resource-based policies?

✅ Identity-Based Policies

- **Attached to**: IAM users, groups, or roles.

- **Purpose**: Define *what actions* a user/role can perform on *which resources*.
- **Most common type**.
- **Default deny** unless explicitly allowed.

📄 **Example**:

```
json

{
  "Effect": "Allow",
  "Action": "s3:ListBucket",
  "Resource": "arn:aws:s3:::my-bucket-xyz"
}
```

📌 Means: The **user/role** can list objects in `my-bucket`.

✅ **Resource-Based Policies**

- **Attached to**: AWS resources (e.g., S3 bucket, SNS topic, Lambda function).
- **Purpose**: Define *who* (principal) can access the resource and *what* they can do.
- Supports **cross-account access**.

📄 **Example** (S3 Bucket Policy):

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::123456789012:user/Alice"
  },
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::my-bucket/*"
}
```

📌 Means: User **Alice** from account `123456789012` can read objects in `my-bucket`.

Summary Table:

Feature	Identity-Based	Resource-Based
Attached To	IAM users, groups, or roles	AWS resources (S3, Lambda, etc.)
Specifies	What user can do	Who can access the resource
Principal field required	❌ No	✅ Yes
Cross-account support	❌ Not directly	✅ Yes

When would you use an S3 bucket policy vs. an IAM policy?

⍰ Use S3 Bucket Policy When:

- ✅ You want to **grant cross-account access** (e.g., allow another AWS account to access your bucket).
- ✅ You need **resource-level permissions** managed **at the bucket** itself.
- ✅ You want to allow **anonymous/public access** (e.g., for static website hosting).
- ✅ You don't manage the **IAM users** (e.g., third-party access).

📄 **Example**:
Allow another AWS account to read objects:

```
json

{
  "Effect": "Allow",
  "Principal": { "AWS": "arn:aws:iam::123456789012:user/OtherUser" },
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::my-bucket-xyz/*"
}
```

ARN: amazon resource name

🕵 Use IAM Policy When:

- ✅ You want to control access for **users, groups, or roles** within **your own AWS account**.
- ✅ You want **centralized access management** via IAM.
- ✅ You need to apply permissions to **multiple services**, not just S3.
- ✅ You want to enforce permissions using **MFA**, tags, or conditions tied to the user/role.

📄 **Example**:
Allow a user to upload to a specific S3 bucket:

```
json

{
  "Effect": "Allow",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::my-bucket/uploads/*"
}
```

Summary Table

Feature	S3 Bucket Policy	IAM Policy
Attached To	S3 bucket (resource-based)	IAM user/role/group (identity-based)

Feature	S3 Bucket Policy	IAM Policy
Cross-account Access	✅ Yes	❌ Not directly
Public/Anonymous Access	✅ Yes	❌ No
Centralized User Permissions	❌ No	✅ Yes
Works across multiple services	❌ No (S3 only)	✅ Yes

If IAM policy says user ABCD has full access to S3.

But bucket policy of my-xyz-bucket says

```
{
 "Effect": "Allow",
 "Principal": { "AWS": "arn:aws:iam::123456789012:user/OtherUser" },
 "Action": "s3:GetObject",
 "Resource": "arn:aws:s3:::my-bucket-xyz/*"
}
```

Can ABCD user access my-bucket-xyz/ bucket ?

Can user/ABCD access my-xyz-bucket?

✅ **YES**, `user/ABCD` **can access** the bucket **because their IAM policy allows it**.

👉 The bucket policy does **not deny** ABCD — it is simply silent about ABCD.

In AWS:

- IAM user permissions are evaluated **independently**.
- Bucket policy is like a **resource-based policy** that can **add more access**, but it doesn't restrict users unless it has an explicit **Deny**.

Key Points:

- **Explicit Deny** in either IAM or bucket policy overrides any Allow.
- If **IAM allows** and **bucket policy doesn't deny**, access is granted.
- Bucket policy is often used to give **cross-account access** or **public access**.

How do S3 bucket policies, KMS key policies, and IAM policies interact?

All three must allow access

To successfully access a KMS-encrypted S3 object, the **request must be allowed by**:

1. ✅ The **IAM policy** (who is making the request)
2. ✅ The **S3 bucket policy** (on the resource)
3. ✅ The **KMS key policy** (encryption key access)

If **any one denies**, the request fails.

Roles Each One Plays

1. 🕵️ IAM Policy

- Attached to: User, group, or role.
- Says: *"This person is allowed to call S3:GetObject or kms:Decrypt."*

2. □ S3 Bucket Policy

- Attached to: The bucket.
- Says: *"This person/service can read/write objects in this bucket."*

3. 🔐 KMS Key Policy

- Attached to: The KMS key.
- Says: *"This person/service can use this key for encryption or decryption."*

📄 Example: Alice wants to download an encrypted file

Alice's request will succeed only if:

- ✅ Her **IAM policy** allows `s3:GetObject` **and** `kms:Decrypt`.
- ✅ The **S3 bucket policy** allows her to `s3:GetObject`.
- ✅ The **KMS key policy** allows her to `kms:Decrypt`.

If **any one of these denies** access, the download will fail.

:

---

1. What is the difference between an allow and a deny in a policy?

Q1: In AWS IAM, what happens if a user is granted "Allow" in one policy but "Deny" in another for the same action?

A) The user is allowed the action
B) The policies cancel each other
C) The explicit **Deny overrides the Allow**
D) The request is ignored

✅ **Correct Answer:** C) The explicit **Deny overrides the Allow**

---

2. What is the purpose of conditions in IAM policies?

Q2: Why are **conditions** used in IAM policies?

A) To define the maximum session duration
B) To add **context-based restrictions** like IP, time, or MFA
C) To enforce two-factor authentication by default
D) To avoid using bucket policies

✅ **Correct Answer:** B) To add **context-based restrictions** like IP, time, or MFA

---

3. What's the difference between identity-based and resource-based policies?

Q3: Which of the following is **true** about **resource-based policies**?

A) They can be attached to IAM users and groups
B) They define what actions users in your account can take
C) They support cross-account access using the `Principal` field
D) They are more secure than identity-based policies by default

✅ **Correct Answer:** C) They support **cross-account access** using the `Principal` field