All about Policies:

You manage access in AWS by creating policies and attaching them to IAM identities (users, groups of users, or roles) or AWS resources. Policies are **Json documents that define permissions and can be applied to users, groups, and roles.** You manage access in AWS by creating policies and attaching them to IAM identities (users, groups, and roles) or AWS resources.

There are 3 types of policies:

A. **AWS Managed Policies & AWS Managed - Job Functions**
B. **Customer managed policies.**
C. **Inline policies.**

Identity-based policies – Identity-based policies are used to attach to a user or role (to identities) We can attach AWS Managed, Customer Managed or Inline policies to user, group or role can be attached.
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.

A. AWS Managed Policy and AWS Managed- Job Functions:

- Created, managed & maintained by AWS.
- Used for common use cases based on job function.
- You don't have to create policies yourself.
- Can be attached to multiple users, groups, or roles within and across AWS accounts.
- You cannot change the permissions defined in these policies.
- Some AWS managed policies are designed for specific job functions.
    - **Administrator.**
    - **Billing.**
    - **Database Administrator.**
    - **Data Scientist.**
    - **Developer Power User.**
    - **Network Administrator.**
    - **Security Auditor.**
    - **Support User.**
    - **System Administrator.**
    - **View-Only User.**

B. Customer Managed Policy or Custom Policy:

- You can create your own polices with your defined set of permissions and attach to user, group, role or resources.
- Can be attached to multiple users, groups, and roles – **but only within your own account.**
- Can be created by copying an existing AWS managed policy and then customizing it.
- Recommended for use cases where the existing AWS Managed Policies don't meet the needs of your environment.
- AWS recommends that provide the least access to your Identities.

A. Inline Policy:

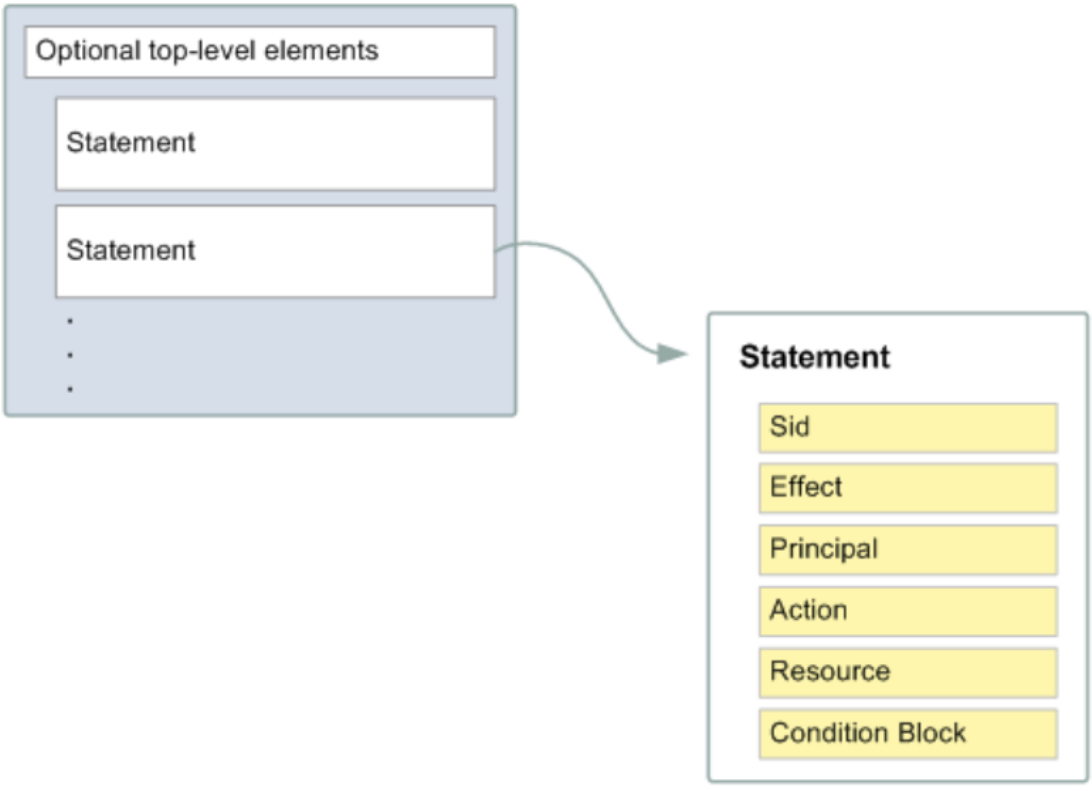
- One kind of custom policies but can be attached to one entity only, cannot be shared.
- Inline policies are embedded within the user, group, or role to which it is applied.
- Strict 1:1 relationship between the entity and the policy.
- When you delete the user, group, or role in which the inline policy is embedded, the policy will also be deleted.
- Inline policies are useful when you want to be sure that the permissions in a policy are not inadvertently assigned to any other user, group, or role.

Interview Question: what is difference between Custom Policy and inline policy?

JSON Document and Policy structure:

It is not necessary for you to understand the JSON syntax. You can use the visual editor in the AWS Management Console to create and edit customer managed policies without ever using JSON.

Structure:



- **Version** – Specify the version of the policy language that you want to use. We recommend that you use the latest 2012-10-17 version.

- **Statement** – Acts as a container for the elements. You can include more than one statement in a policy.
- **Sid** (Optional) – Include an optional **statement ID** to differentiate between your statements.
- **Effect** – it can be either **Allow** or **Deny** - indicate whether the policy allows or denies access.
- **Principal** (Required in only some circumstances) – If you create a resource-based policy, you must indicate the account, user, role, or federated user to which you would like to allow or deny access. If you are creating an IAM permissions policy to attach to a user or role, you cannot include this element. The principal is implied as that user or role.
- **Action** – Include a **list of actions that the policy allows or denies.**
- **Resource** (Required in only some circumstances) – If you create an IAM permissions policy, you must specify a list of resources to which the actions apply. If you create a resource-based policy, this element is optional. If you do not include this element, then the resource to which the action applies is the resource to which the policy is attached.
- **Condition** (Optional) – Specify the circumstances under which the policy grants permission.

Example:

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
      "Sid": "FirstStatement",
      "Effect": "Allow",
      "Action": ["iam:ChangePassword"],
      "Resource": "*"
      },
   ]
}
```

The reason why we don't mention principal in Identity based policy is because we are attaching policy to principal itself to Allow or Deny actions on resources. So Root user, IAM User or Application are the principals and if we are attaching policy to them to allow/deny access of resources we don't have to mention principal in policy separately.

Lets say - if a policy allows the GetUser action, then a user with that policy can get user information from the AWS Management Console, the AWS CLI, or the AWS API.

If a policy allows all access but not the CreateUser access, then user with that policy can list, read or edit resources in IAM but wont be able to create a new user.

AWS has distributed level of access into small actions for every service:

For example: IAM has below set of actions, we can allow or deny below actions with the help of policies.

AddClientIDToOpenIDConnectProvider	AttachGroupPolicy	CreateAccessKey
AddRoleToInstanceProfile	AttachRolePolicy	CreateAccountAlias
AddUserToGroup	AttachUserPolicy	CreateGroup
	ChangePassword	CreateInstanceProfile

CreateLoginProfile
CreateOpenIDConnectProvider
CreatePolicy
CreatePolicyVersion
CreateRole
CreateSAMLProvider
CreateServiceLinkedRole
CreateServiceSpecificCredential
CreateUser
CreateVirtualMFADevice
DeactivateMFADevice
DeleteAccessKey
DeleteAccountAlias
DeleteAccountPasswordPolicy
DeleteGroup
DeleteGroupPolicy
DeleteInstanceProfile
DeleteLoginProfile
DeleteOpenIDConnectProvider
DeletePolicy
DeletePolicyVersion
DeleteRole
DeleteRolePermissionsBoundary
DeleteRolePolicy
DeleteSAMLProvider
DeleteServerCertificate
DeleteServiceLinkedRole
DeleteServiceSpecificCredential
DeleteSigningCertificate
DeleteSSHPublicKey
DeleteUser
DeleteUserPermissionsBoundary
DeleteUserPolicy
DeleteVirtualMFADevice
DetachGroupPolicy
DetachRolePolicy
DetachUserPolicy
EnableMFADevice
GenerateCredentialReport
GenerateOrganizationsAccessReport
GenerateServiceLastAccessedDetails
GetAccessKeyLastUsed
GetAccountAuthorizationDetails
GetAccountPasswordPolicy
GetAccountSummary
GetContextKeysForCustomPolicy
GetContextKeysForPrincipalPolicy
GetCredentialReport
GetGroup
GetGroupPolicy
GetInstanceProfile
GetLoginProfile
GetMFADevice
GetOpenIDConnectProvider
GetOrganizationsAccessReport
GetPolicy
GetPolicyVersion
GetRole
GetRolePolicy
GetSAMLProvider
GetServerCertificate
GetServiceLastAccessedDetails
GetServiceLastAccessedDetailsWithEntities
GetServiceLinkedRoleDeletionStatus
GetSSHPublicKey
GetUser
GetUserPolicy
ListAccessKeys
ListAccountAliases
ListAttachedGroupPolicies
ListAttachedRolePolicies
ListAttachedUserPolicies
ListEntitiesForPolicy
ListGroupPolicies
ListGroups
ListGroupsForUser
ListInstanceProfiles
ListInstanceProfilesForRole
ListInstanceProfileTags
ListMFADevices
ListMFADeviceTags
ListOpenIDConnectProviders
ListOpenIDConnectProviderTags
ListPolicies
ListPoliciesGrantingServiceAccess
ListPolicyTags
ListPolicyVersions
ListRolePolicies
ListRoles
ListRoleTags
ListSAMLProviders
ListSAMLProviderTags
ListServerCertificates
ListServerCertificateTags
ListServiceSpecificCredentials
ListSigningCertificates
ListSSHPublicKeys
ListUserPolicies
ListUsers
ListUserTags
ListVirtualMFADevices
PutGroupPolicy
PutRolePermissionsBoundary
PutRolePolicy
PutUserPermissionsBoundary
PutUserPolicy
RemoveClientIDFromOpenIDConnectProvider
RemoveRoleFromInstanceProfile
RemoveUserFromGroup
ResetServiceSpecificCredential
ResyncMFADevice
SetDefaultPolicyVersion
SetSecurityTokenServicePreferences
SimulateCustomPolicy
SimulatePrincipalPolicy
TagInstanceProfile
TagMFADevice
TagOpenIDConnectProvider

TagPolicy
TagRole
TagSAMLProvider
TagServerCertificate
TagUser
UntagInstanceProfile
UntagMFADevice
UntagOpenIDConnectProvider
UntagPolicy
UntagRole
UntagSAMLProvider
UntagServerCertificate
UntagUser
UpdateAccessKey
UpdateAccountPasswordPolicy
UpdateAssumeRolePolicy
UpdateGroup
UpdateLoginProfile
UpdateOpenIDConnectProviderThumbprint
UpdateRole
UpdateRoleDescription
UpdateSAMLProvider
UpdateServerCertificate
UpdateServiceSpecificCredential
UpdateSigningCertificate
UpdateSSHPublicKey
UpdateUser
UploadServerCertificate
UploadSigningCertificate
UploadSSHPublicKey

IAM Policy Evaluation Logic

- By default, all requests are implicitly denied. (Alternatively, by default, the AWS account root user has full access).
- An explicit allow in an identity-based or resource-based policy overrides this default.
- If a permissions boundary, Organizations SCP, or session policy is present, it might override the allow with an implicit deny.
- An explicit deny in any policy overrides any allows.

IAM permissions boundaries – Permissions boundaries are an advanced feature that sets the maximum permissions that an identity-based policy can grant to an IAM entity (user or role).
AWS Organizations service control policies (SCPs) – Organizations SCPs specify the maximum permissions for an organization or organizational unit (OU). Session policies – Session policies are advanced policies that you pass as parameters when you programmatically create a temporary session for a role or federated user.