What is IAM?

* AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources.
* IAM enables you to create and manage users, groups, roles, and permissions, making it easier to follow the principle of least privilege—granting users only the permissions they need to perform their tasks and nothing more

**Key components of IAM include:**

* **Users:** IAM users are the entities representing individuals who need access to your AWS account. Each user is assigned a unique set of security credentials (access key and secret access key) for programmatic access to AWS services.
* **Groups:** IAM groups allow you to group users together based on their roles or functions. Permissions can be assigned to groups instead of individual users, simplifying permission management.
* **Roles:** IAM roles are sets of permissions that can be assumed by AWS services or users from another AWS account. Roles are commonly used to grant temporary access for specific tasks, like allowing an EC2 instance to access an S3 bucket.
* **Policies:** IAM policies are JSON documents that define permissions. You can attach policies to users, groups, or roles, specifying what actions are allowed or denied on specific AWS resources.

**Q: What are the key features of AWS IAM?**

Answer: The key features of AWS IAM are:

* Identity Management: Manage IAM users, groups, and roles to control access to AWS resources.
* Fine-Grained Access Control: Assign precise permissions using IAM policies for granular control over resource access.
* Multi-Factor Authentication (MFA): Enable an extra layer of security by requiring users to provide additional authentication factors.
* Access Key Management: Generate and manage access keys for programmatic access to AWS services.
* Integration with AWS Services: Seamlessly integrate IAM with various AWS services for secure access management.
* Identity Federation: Enable single sign-on (SSO) and federated access using external identity providers (e.g., Active Directory).
* Security Token Service (STS): Obtain temporary security credentials for cross-account access and identity delegation.
* Audit and Compliance: Monitor and log IAM events for auditing, compliance, and security analysis.
* Cross-Account Access: Grant permissions to users in other AWS accounts through IAM roles.
* AWS Organizations Integration: Centrally manage IAM policies and accounts across multiple AWS accounts using AWS Organizations, etc.

**What is Roles in IAM?**

AWS Identity and Access Management (IAM), a role is a way to grant permissions to entities so they can perform actions on your AWS resources. Roles are used primarily for secure cross-account access and temporary access needs. They provide a way to delegate access without sharing long-term credentials like access keys.

Roles can be assumed by AWS services, such as Amazon EC2 instances or Lambda functions, or by external identities like users or applications from other AWS accounts.

**Q. What happens if you have one IAM statement that allows a principal to perform an operation on a resource and another statement that denies that same operation on the same resource?**

The **Deny** effect always wins when multiple statements apply.

**What are the basic differences between IAM roles and users**

* **IAM Users:** Users are individuals within your AWS account who need to interact with AWS resources. They can be assigned permissions directly through policies and can authenticate using their own credentials (username and password or access keys).
* **IAM Roles:** Roles are used to delegate permissions to entities that are not within your AWS account, such as AWS services, applications, or external identities. Roles are assumed to obtain temporary security credentials for performing actions on AWS resources.

**Policy Types**

A IAM policy is an object that defines the permissions attached to an identity or resource. The following policy types supported AWS IAM:

**1.Identity-based Policies**

Identity-based policies are policies that define what actions an IAM identity (user, group, or role) are allowed to take against specific resources. A common example of when you would create an identity-based policy is when you want to grant permissions to an IAM user access to perforn certain actions such as ec2:DescribeInstances, s3:ListBuckets, s3:PutObject, cloudformation:CreateStack, etc.2. 2.**Resource-based Policies**

Resource-based policies are policies that are associated or attached to a specific resource. They are used to define who (principals) has access to a resource regardless of whether the resource exists in the same account or another AWS account.

Ex.Some popular resource-based policies are S3 bucket policies and IAM role trust policies (aka trust relationships)

**3. permission boundaries**

In AWS Identity and Access Management (IAM), permission boundaries are a feature that helps you control the maximum permissions that an identity-based policy can grant. Permission boundaries set the scope of permissions that a policy can grant, even if the identity attaching the policy has higher permissions.

**4. Service Control Policies**

AWS Organizations not only allows you to divide your organization into logical business units and manage AWS billing across many AWS accounts, it also enables you to set permissions across your AWS accounts using service control policies (SCPs).

**5. session policy**

In AWS Identity and Access Management (IAM), a session policy is a policy that defines permissions for a temporary session created when assuming an IAM role. When an entity, such as an AWS service or an IAM user, assumes a role, it receives temporary security credentials including an access key, secret key, and session token. These credentials allow the entity to perform actions according to the permissions granted by the role's session policy.

**Q: Explain the IAM Policy Structure?**

Answer: A JSON policy document contains following elements:

**Version** – The version of the policy language (2012-10-17). As a best practice, use the latest version.

**Statement** – Main policy element as a container. You can add multiple statements in a policy.

**Sid**– It is an optional identifier for the policy to differentiate between your statements.

**Effect** – Grant permission (allow or deny)

**Principal**– Who can access it. In case of resource-based policy, you need to add a principle (account, user, role, or federated) to define allow or deny access.

**Action –** List of actions that the policy allows or denies.

**Resource –** If you create an IAM permissions policy, you need to specify the list of resources to which the actions apply. In case of resource-based policy, this element is optional.

**Condition –** It is an optional element. You can specify the condition under which the policy grants permission.

**What are IAM users’ access and secret keys?**

IAM users’ access and secret keys are credentials used for programmatic access to AWS services and resources.

**Access Key:** An access key is a unique identifier associated with an IAM user or an IAM role. It is a long string of characters (e.g., AKIAIOSFODNN7EXAMPLE) that is used to authenticate requests made to AWS services.

**Secret Key:** A secret key is a cryptographic key associated with an access key. It is a separate, confidential value (e.g., wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY) that must be kept secure.

**What is CloudTrail in AWS?**

It’s a service, which records the logs of each IAM entity. So that you can use these logs for auditing and compliance purposes.

In these logs you will get answers for Who, When, Where, What, Which

Who made the request.

When was the request made.

What was the request about.

Which resources were acted upon in response to the reques.

**WHAT IS MFA?**

1.MFA stands for "Multi-Factor Authentication." It is a security mechanism used to enhance the protection of user accounts and systems by requiring multiple forms of verification before granting access. In the context of AWS (Amazon Web Services), MFA adds an extra layer of security to user accounts when accessing AWS resources and services.

2. When MFA is enabled, users are required to provide not only their password but also a temporary code generated by their MFA device or app. This makes it much more difficult for unauthorized individuals to gain access to AWS resources, even if they have the user's password.

**SERVICES SUPPORTED BY RESOURCE-BASED POLICIES.**

As of my last update in September 2021, a wide range of Amazon Web Services (AWS) services support resource-based policies. These policies allow you to define permissions directly on a resource (like an S3 bucket, Lambda function, or SQS queue) rather than defining them in an Identity and Access Management (IAM) policy. This provides more fine-grained control over who can access the resource and what actions they can perform.

Here are some AWS services that support resource-based policies:

1. \*\*Amazon S3 (Simple Storage Service)\*\*: S3 buckets and objects can have resource-based policies that control access to them.

2. \*\*AWS Lambda\*\*: Lambda functions can have resource-based permissions that define who can invoke them directly.

3. \*\*Amazon SQS (Simple Queue Service)\*\*: SQS queues can have resource-based policies that grant permissions to specific AWS accounts or IAM users/roles to send or receive messages from the queue.

4. \*\*Amazon API Gateway\*\*: API Gateway APIs can have resource-based policies to control access to your APIs, allowing or denying access to specific AWS accounts or IAM users/roles.

5. \*\*Amazon DynamoDB\*\*: DynamoDB tables can have fine-grained access control through resource-based policies, allowing you to specify who can perform specific actions on the table.

6. \*\*AWS Key Management Service (KMS)\*\*: KMS keys can have resource-based policies that define who can use the key for encryption and decryption.

**What is AWS STS**

* You can use the AWS Security Token Service (AWS STS) to create and provide trusted users with temporary security credentials that can control access to your AWS resources.
* Temporary security credentials are short-term, as the name implies. They can be configured to last for anywhere from a few minutes to several hours. After the credentials expire, AWS no longer recognizes them or allows any kind of access from API requests made with them.

**Amazon Resource Names (ARNs)**

Amazon Resource Names (ARNs) uniquely identify AWS resources. We require an ARN when you need to specify a resource unambiguously across all of AWS, such as in IAM policies, Amazon Relational Database Service (Amazon RDS) tags, and API calls.

* **ARN format**

The following are the general formats for ARNs. The specific formats depend on the resource. To use an ARN, replace the italicized text with the resource-specific information. Be aware that the ARNs for some resources omit the Region, the account ID, or both the Region and the account ID.

arn:partition:service:region:account-id:resource-id

arn:partition:service:region:account-id:resource-type/resource-id

arn:partition:service:region:account-id:resource-type:resource-id

**Partition --** The partition in which the resource is located. A *partition* is a group of AWS Regions. Each AWS account is scoped to one partition.

The following are the supported partitions:

* aws - AWS Regions
* aws-cn - China Regions
* aws-us-gov - AWS GovCloud (US) Regions

**service**

The service namespace that identifies the AWS product.

**region**

The Region code. For example, us-east-2 for US East (Ohio). For the list of Region codes, see [Regional endpoints](https://docs.aws.amazon.com/general/latest/gr/rande.html#regional-endpoints) in the *AWS General Reference*.

**account-id**

The ID of the AWS account that owns the resource, without the hyphens. For example, 123456789012.

**resource-type**

The resource type. For example, vpc for a virtual private cloud (VPC).

**resource-id**

The resource identifier. This is the name of the resource, the ID of the resource, or a [resource path](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference-arns.html#arns-paths). Some resource identifiers include a parent resource (sub-resource-type/parent-resource/sub-resource) or a qualifier such as a version (resource-type:resource-name:qualifier).

###### Examples

* IAM user

arn:aws:iam::*123456789012*:user/*johndoe*

* SNS topic

arn:aws:sns:*us-east-1*:*123456789012*:*example-sns-topic-name*

* VPC

arn:aws:ec2:*us-east-1*:*123456789012*:vpc/*vpc-0e9801d129EXAMPLE*