# AWS-IAM

## Identity and Access Management

# CONTENTS

# What is IAM

- **AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. With IAM, you can centrally manage permissions that control which AWS resources users can access. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.**
- **On a fundamental level, IAM encompasses the following components:**
  - **how individuals are identified in a system (understand the difference between identity management and authentication);**
  - **how roles are identified in a system and how they are assigned to individuals;**
  - **adding, removing and updating individuals and their roles in a system;**
  - **assigning levels of access to individuals or groups of individuals; and**
  - **protecting the sensitive data within the system and securing the system itself.**

# How IAM Works

# IAM Identities (users, user groups, and roles)

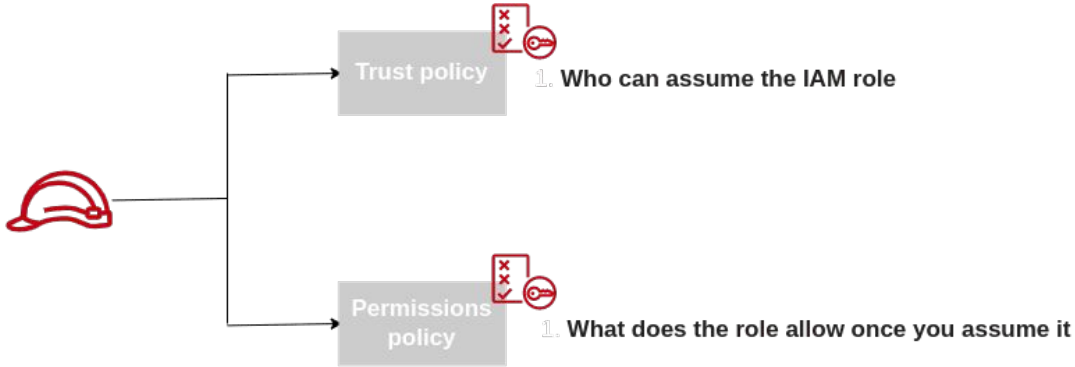
- **The AWS account root user or an administrative user for the account can create *IAM identities*.**
- **An IAM identity provides access to an AWS account. An IAM identity represents a human user or programmatic workload, and can be authenticated and then authorized to perform actions in AWS.**
- **An *IAM user group* is a collection of IAM users managed as a unit.**
- **Each IAM identity can be associated with one or more *policies*. Policies determine what actions a user, role, or member of a user group can perform, on which AWS resources, and under what conditions.**

# IAM Users

- The "identity" aspect of AWS Identity and Access Management (IAM) helps you with the question "Who is that user?", often referred to as *authentication*. IAM users are not separate accounts; they are users within your account. Each user can have its own password for access to the AWS Management Console. You can also create an individual access key for each user so that the user can make programmatic requests to work with resources in your account.
-

# IAM Roles

- **An IAM role is similar to an IAM user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it.**
- **Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session.**
- **You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources. For example, you might want to grant users in your AWS account access to resources they don't usually have, or grant users in one AWS account access to resources in another account**



# Overview of access management: Permissions and policies

The access management portion of AWS Identity and Access Management (IAM) helps you define what a principal entity is allowed to do in an account. A principal entity is a person or application that is authenticated using an IAM entity (user or role). Access management is often referred to as *authorization*. You manage access in AWS by creating policies and attaching them to IAM identities (users, groups of users, or roles) or AWS resources.

A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal uses an IAM entity (user or role) to make a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents.

## Policies and users

- IAM users are identities in the service. When you create an IAM user, they can't access anything in
- your account until you give them permission. You give permissions to a user by creating an identity- based policy, which is a policy that is attached to t user or a group to which the user belongs. The following example shows a JSON policy that allows the user to perform all Amazon DynamoDB actions (dynamodb:*) on the Books table in the 123456789012 account within the us-east-2 Region.

## Policies and groups

- You can organize IAM users into *IAM groups* and attach a policy to a group. In that case, individual users still have their own credentials, but all the users in a group have the permissions that are attached to the group.

# What is ABAC

## What is ABAC for AWS?

**Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM resources, including IAM entities (users or roles) and to AWS resources.**

**You can create a single ABAC policy or small set of policies for your IAM principals. These ABAC policies can be designed to allow operations when the principal's tag matches the resource tag. ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome**

# IAM CLI commands

- Go through the documentation in detail and practice all commands

  https://docs.aws.amazon.com/cli/latest/reference/iam/

# Example

Create an IAM role

- aws iam create-role  --role-name Test-Role --assume-role-policy-document file://Test-Role-Trust-Policy.json

```
{
 "Version": "2012-10-17",
 "Statement": {
        "Effect": "Allow",
        "Principal": {
        "AWS": "arn:aws:iam::851725483404:user/suchintan-desktop"
        },
        "Action": "sts:AssumeRole"
 }
}
```

Test-Role-Trust-Policy.json

# Create a Policy for S3 Read Only access

Create a Permission Policy

- **aws iam create-policy --policy-name Tester-Policy --policy-document Test-access-policy.json**

```
{
        "Version": "2012-10-17",
        "Statement": [
        {
        "Effect": "Allow",
        "Action": [
                "s3:Get*",
                "s3:List*",
                "s3:Describe*",
                "s3-object-lambda:Get*",
                "s3-object-lambda:List*"
        ],
        "Resource": "*"
        }
        ]
}
```

**Test-access-policy.json**

# Attach Permission Policy to a role

- aws iam attach-role-policy --policy-arn "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess" --role-name Test-Role

# Assignment

1. **Write a Bash script to create an IAM user named *Test*. Add Administrator access to the IAM user.**
2. **Write a Bash script to create an IAM role named *Tester-role*. In the same script create a Access policy for Tester-Role named *Tester-Policy*. The policy should provide readonly access to s3. Attach the role to the policy.**
3. **Extend the above bash script to create another user named *Test2*. Add *Test1* and *Test2* to a user group named *testing*(which should also be created through the script). Attach *Tester-Role* to all users in the user group *testing*.**