MODULE 7

Praneesh Sharma

21052264

**AWS Identity and Access Management (IAM):** Involves the application of controls to users who need access to computing resources.

**Role:** An IAM identity that you can create in your account that has specific permissions.

**User:** An entity that you create in Amazon Web Services (AWS) to represent the person or application that uses it to interact with AWS. A user in AWS consists of a name and credentials.

**Security Group:** A security group acts as a virtual firewall for your instance to control inbound and outbound traffic.

**Policy:** An object in AWS that, when associated with an identity or a resource, defines its permissions. AWS evaluates these policies when a principal entity (user or role) makes a request.

**Amazon Inspector:** Helps customers identify security vulnerabilities and deviations from security best practices in applications, before they are deployed and while they are running in a production environment.

**Group:** An IAM group is a collection of IAM users. Groups let you specify permissions for multiple users, which can make it easier to manage the permissions for those users.

**Root user:** When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account.

**Credential:** AWS security credentials verify who you are and whether you have permission to access the resources that you are requesting.

**Enable multi-factor authentication:** This approach to authentication requires two or more independent pieces of information to be authenticated.

**JavaScript Object Notation (JSON):** A syntax for storing and exchanging data.

**Multi-Factor Authentication (MFA):** A security system that requires more than one method of authentication from independent categories of credentials to verify the user's identity for a login or other transaction.

Security is essential when using cloud resources for processing and storing data. Because databases, websites, and apps in the cloud can process sensitive information like banking and medical records, these cloud resources need to be restrictive of who can access them and what privileges they have while doing so. IAM involves the application of controls to users who need access to computing resources.

**AWS Identitites:**

At the highest level is the *root user*. This is the identity that created the AWS account. The root user has access to every aspect of AWS and acts as a universal administrator. The root user credentials should never be given out, and it is not even recommended for the account creator to do everyday tasks as the root user. Instead, the root user account should be used to make an administrator account.

An *IAM user* is an entity created in AWS. It represents the person using the AWS services and gives people the ability to sign in to AWS. A user will be assigned a name and password to access the AWS console. When creating a user, it is considered a best practice to assign them to a *group*that has the appropriate permissions *policy*.

A *group*is a collection of *IAM users*. You can use groups to specify permissions for a collection of users, which can make those permissions easier to manage for those users.Any user in that group automatically has the permissions that are assigned to the group. If a new user joins your organization and needs administrator privileges, you can assign the appropriate permissions by adding the user to that group.

*IAM roles*are similar to *users*in that they are identities with permission policies that determine what the identity can and cannot do in AWS. However, a role does not have any credentials (password or access keys) associated with it. Instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. An IAM user can assume a role to temporarily take on different permissions for a specific task. Roles are useful in instances where a mobile app is accessing your AWS data. By assigning a role when the user logs in, they are granted temporary access with some permissions, but not permanent credentials.

A policy, when attached to a *user*,*role*, or *group*, defines their permissions. Policies are stored in AWS as JSON documents. It is best practice to assign *policies*to *groups*and then assign each *user*and *role*to a group when created.

**Lab 7**

1. Services > Security, Identity and Compliance > Identity Access and Management > User Groups
2. S3-Support Group > Users> Add Users >user-1 > Add Users
3. Added user-2 to EC2-Support group and user-3 to EC2-Admin group
4. Access the console sign-in url from the dashboard
5. Use this link to sign into the user1 account. Access the EC2 Instances tab. No instances will be displayed.
6. Use the sign-in url to sign into the user2 account. Access the EC2 instances tab.
7. Stop the instance from the Instance state menu, An error message will be displayed
8. Use the sign-in url to sign into the user3 account. Access the EC2 instances tab.
9. Stop the instance from the Instance state menu, The instance will successfully stop.