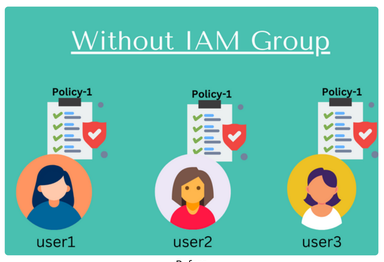
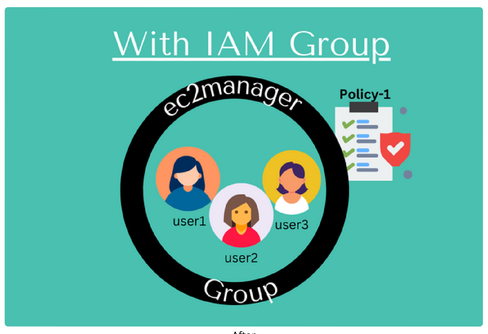
IAM Entities

1. Introduction
   1. IAM entities are the core of access control in AWS
   2. They define who can access resources.
   3. IAM policies define what actions those entities can perform and on which resources
   3. Types of IAM Entities
      1. User
      2. Group
      3. Role
2. IAM Users
   1. Introduction
      1. An individual entity representing a human user who needs access to AWS resources.
      2. Users are used to directly interact with AWS services and resources.
   2. User Properties
      1. Username — Unique identifier for the user, used for signing in
      2. Password — Used for accessing the AWS Management Console
      3. Access Key ID and Secret Access Key
         1. A pair of credentials used for programmatic access to AWS services through the AWS SDK or CLI.
         2. These keys should be treated as highly sensitive and kept secret.
      4. Permissions
         1. Defined by IAM policies attached to the user or groups the user belongs to.
         2. These policies grant specific actions on specific AWS resources.
      5. MFA (Multi-Factor Authentication) — An additional layer of security requiring a second factor (e.g., code from an authenticator app) for login.
   3. Defining Strong User Credentials
      1. An IAM password policy dictates the requirements and restrictions for passwords used by IAM users in your AWS account
      2. It plays a crucial role in safeguarding your resources by enforcing strong password practices.
      3. You can create a custom password policy within the AWS IAM console or use the default policy.
      4. Once defined, the policy applies to all IAM users created after its implementation.
      5. Existing users are not immediately impacted by changes to the password policy, but the new policy will be enforced when they next change their password.
3. IAM Group
   1. Introduction
      1. A collection of IAM users within your AWS account for easier permission management.
      2. Simplify assignment and management of IAM user permissions
      3. Grant users access to specific AWS resources and services based on their group membership
      4. Groups reduces the need for individual user policy management.
   2. Group Use Case
      1. Before



      2. After



   3. Key Points
      1. Groups do not have individual login credentials
      2. Users can belong to multiple groups, inheriting permissions from each
      3. Regularly review and update group memberships and permissions to maintain security and compliance.
4. Role
   1. Introduction
      1. An IAM role is an IAM identity that you can create in your account that has specific permissions.
      2. A role is for anyone who needs it, not just for one person, unlike an IAM user
      3. A role gives you temporary credentials when assumed, unlike IAM users, for whom we create long-term passwords or access keys
      4. You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources.
      5. Roles are designed for various use cases, each enabling specific scenarios of access control and delegation
   2. Role Use Case
      1. AWS service
         1. AWS offers over 200+ services for computing, storage, database management, analytics, machine learning, and more.
         2. By default, services under a single AWS account operate in isolation and cannot interact with each other.
         3. Direct access or interaction between services requires explicit authorization to ensure security.
         4. This security principle is applied to limit access to only the resources needed for a service to perform its tasks, enhancing overall security.
         5. Example
            1. Imagine you have a web application running on an EC2 instance (virtual server) in AWS.
            2. This application needs to upload files to an S3 bucket (storage) for user uploads.
            3. Here's the traditional, insecure approach — Store Access Keys on EC2
               1. You need store the access key and secret key for an IAM user with S3 upload permissions directly on the EC2 instance.
               2. This is risky because anyone who accesses the server can steal these credentials.
            4. Here's how a service role improves security
               1. Create an IAM Role — You create an IAM role with a policy that allows uploading files to S3.
               2. Attach Service Role to EC2 Instance — When launching the EC2 instance, you associate it with this IAM role.
               3. Automatic Temporary Credentials — When the EC2 instance starts, it automatically assumes the role and receives temporary security credentials with limited permissions (only S3 upload).
               4. Application Uses Temporary Credentials — Your web application can then use these temporary credentials to upload files to S3 securely, without needing the actual long-term access keys.
      2. AWS Account | Assume Role
         1. Introduction
            1. Assume Role is an AWS IAM (Identity and Access Management) action that allows an IAM entity (such as a user, AWS service, or application) to temporarily adopt the permissions of an IAM role.
            2. When an IAM entity assumes a role, it receives a set of temporary security credentials that grant it permissions based on the policies attached to the role.
            3. AWS Security Token Service (STS) grants temporary credentials, with customizable lifespans ranging from 15 minutes to 12 hours and Default Value is 1 hour, to access AWS resources securely using role
            4. When working with IAM roles in AWS, there are primarily two options for how roles can be assumed:
               1. Same Account Access
               2. Cross-Account Access
         2. Same Account Access
            1. Introduction
               1. IAM roles are created within the same AWS account where the resources are located.
               2. It's like giving certain individuals or applications specific responsibilities or access rights within your organization's AWS environment.
            2. Examples
               1. Scenario — Amit, a developer, needs daily access to EC2 instances but only occasional access to an S3 bucket
               2. Solution
                  1. IAM User (Amit) — Create an IAM user named "Amit" for his daily tasks.
                  2. Directly Attached Policy (EC2-FullAccess)
                     1. Attach a policy granting "Amit" full access (ec2full) to manage EC2 instances
                     2. This provides him with the necessary permissions for his daily development work.
                  3. IAM Role (S3-TempAccess) — Create an IAM role named "S3-TempAccess."
                  4. Role Policy (AmazonS3FullAccess) — Attach a policy granting the "S3-TempAccess" role full access access on S3.
                  5. Assign (S3-TempAccess) Role To User (Amit) — Go To Role And Add Amit ARN
               3. Advantages
                  1. Security
                     1. Temporary credentials from the assumed role add another layer of security.
                     2. Even if Amit's user credentials are compromised, the attacker wouldn't have access to S3 after the credentials expire.
                  2. Centralized Management — You can update the role's policy or the "Dev-Assume-Role" policy to grant or revoke access for other developers without modifying individual user permissions.
                  3. Auditing — AWS logs the assumption of the role, providing an audit trail of who accessed S3 Bucket and when
                  4. Additional Considerations — For added security, consider using MFA (Multi-Factor Authentication) when assuming the role.
         3. Cross-Account Access
            1. Introduction
               1. In IAM, assume role functionality can also be used for cross-account access, allowing users or resources in one AWS account to access resources in another account.
            2. Example
               1. Imagine two companies
                  1. CloudStore Pvt. Ltd. (Company A) — Provides cloud storage solutions, using Amazon S3 to store a vast collection of high-resolution images and editing assets.
                  2. PhotoMagic Pvt. Ltd. (Company B) — Specializes in online photo editing services and wants to access CloudStore's S3 bucket containing the dataset for enhancing their photo editing application.
               2. Requirement — PhotoMagic requires access to CloudStore's datasets stored in an Amazon S3 bucket to download high-resolution images for their editing platform.
               3. Option 1
                  1. Creating an IAM User for PhotoMagic in CloudStore's Account — CloudStore creates a dedicated IAM user for PhotoMagic, granting it permissions to access the specified S3 bucket.
                  2. Disadvantages of Option 1
                     1. Security Risks
                        1. Sharing IAM user credentials poses a significant security risk.
                        2. If these credentials are compromised, unauthorized access could occur, leading to potential data breaches.
                     2. Credential Management — CloudStore is responsible for securely sharing and managing the lifecycle of these credentials
                     3. Limited Audit Trail — It's harder to track who accessed what and when, as actions taken by PhotoMagic using the IAM user credentials will not be easily distinguishable from actions taken by CloudStore's own users.
               4. Option 2
                  1. Creating a Role in CloudStore for PhotoMagic to Assume
                     1. CloudStore sets up an IAM role with the necessary permissions for the S3 bucket.
                     2. This role's trust policy allows entities from PhotoMagic's AWS account to assume the role, providing temporary access to the bucket.
                     3. In This Example CloudStore Pvt. Ltd. (Company A) Is Trusting Account & PhotoMagic Pvt. Ltd. (Company B) is Trusted Account
                     4. Requirements
                        1. Establish a trust relationship by specifying PhotoMagic's AWS account ID in the role's trust policy.
                        2. PhotoMagic assumes the role using AWS STS, obtaining temporary credentials for secure access to the S3 bucket.
                  2. Advantages of Option 2
                     1. Enhanced Security — Temporary credentials reduce the risk of long-term credential compromise.
                     2. Ease of Management — No need for CloudStore to manage separate IAM user credentials for PhotoMagic, simplifying administrative tasks.
                     3. Scalability — The solution scales effortlessly as PhotoMagic's needs grow or if additional partners require access.
                     4. Improved Audit Trail — Access and actions are logged with the assumption of the role, providing a clear audit trail of who accessed the S3 bucket and what actions were performed.
5. 3 & 4 (70)
6. 5 (34)