Demo-5

What is IAM (Identity and Access Management)? 🔒

AWS IAM (Identity and Access Management) is a service that enables you to **securely manage access** to AWS resources. It allows you to control **who can access AWS services** and **what actions they can perform**.

Key Features of IAM

☑ **User Management** – Create and manage users, groups, and roles.
☑ **Granular Permissions** – Assign specific permissions using policies.
☑ **Multi-Factor Authentication (MFA)** – Adds an extra layer of security.
☑ **Temporary Access** – Use roles for temporary permissions.
☑ **Integration with AWS Services** – Works with EC2, S3, Lambda, RDS, etc.

IAM Components

1. IAM Users 👤

- Represents an individual user with login credentials.
- Can be assigned permissions via **policies**.

2. IAM Groups 🏢

- A collection of users with the same permissions.
- Example: "Developers" group can have access to EC2 but not RDS.

3. IAM Roles 🎭

- Used to grant **temporary access** to AWS services.
- Commonly used by applications, AWS services, or external users.

4. IAM Policies 📜

- JSON documents that define **permissions**.
- Example: Allow a user to read from an S3 bucket but not delete it.

5. IAM MFA (Multi-Factor Authentication) 🔑

- Enhances security by requiring a second factor (OTP, hardware key).

Types of IAM Policies

1. AWS-Managed Policies (Predefined by AWS)

- Example: `AdministratorAccess`, `ReadOnlyAccess`, `AmazonS3FullAccess`

2. Customer-Managed Policies (Custom policies created by users)

- Example: A policy that allows access to a specific S3 bucket.

3. Inline Policies (Directly attached to a user, group, or role)

- Used for specific one-off permissions.

Example of an IAM Policy

- A JSON policy that **grants read-only access to an S3 bucket**:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject"],
      "Resource": ["arn:aws:s3:::example-bucket/*"]
    }
  ]
}
```

Best Practices for IAM

✔ **Follow the Principle of Least Privilege** – Grant only the necessary permissions.
✔ **Enable MFA for All Users** – Protect against unauthorized access.
✔ **Use IAM Roles for Applications** – Avoid storing credentials in code.
✔ **Regularly Audit IAM Users & Permissions** – Use AWS IAM Access Analyzer.
✔ **Rotate Credentials Regularly** – Use temporary credentials when possible.

Use Cases of IAM

- **User Authentication & Authorization** – Secure AWS accounts and resources.
- **Access Control for Applications** – Grant apps controlled access to AWS services.
- **Security & Compliance** – Enforce security policies across an organization.

AWS Organization

What is AWS Organizations? 🏢

AWS Organizations is a service that allows you to **centrally manage and govern multiple AWS accounts** within your organization. It helps with **billing consolidation, security, compliance, and policy enforcement** across multiple accounts.

Key Features of AWS Organizations

☑ **Centralized Account Management** – Manage multiple AWS accounts from one place.

☑ **Consolidated Billing** – Get a single bill for all linked accounts and benefit from cost savings.

AWS Organizations Structure

1. Root Account 🌳

- The primary account that creates and manages the organization.
- Has full control over all accounts in AWS Organizations.

2. Member Accounts 👥

- Sub-accounts that belong to the organization.
- Each account operates independently but follows policies set by the root account.

3. Organizational Units (OUs) 📂

- Logical groups of AWS accounts within AWS Organizations.
- Helps apply policies to specific sets of accounts.

☑ **Fine-Grained Access Control** – Set permissions at different levels using IAM and SCPs.

☑ **Integration with AWS Services** – Works with AWS Control Tower, IAM, and Security Hub.

Example:

Root Account

├── Security OU

│ ├── Logging Account

│ ├── Compliance Account

├── Development OU

│ ├── Dev Account 1

│ ├── Dev Account 2

├── Production OU

├── Prod Account 1

├── Prod Account 2

4. Service Control Policies (SCPs) 🔐

- Organization-wide policies that restrict permissions on AWS accounts.
- Example: **Prevent all member accounts from deleting S3 buckets.**

Benefits of AWS Organizations

✔ **Cost Savings** – Consolidated billing reduces overall AWS costs.
✔ **Improved Security & Compliance** – SCPs enforce security best practices.
✔ **Better Resource Organization** – OUs help structure accounts logically.
✔ **Scalability** – Easily add and manage new AWS accounts.

Example Use Case of AWS Organizations

- **A large company with multiple teams** (Development, Testing, Production) wants to separate accounts but still manage them centrally.
- **AWS Organizations helps** by grouping accounts into OUs and applying policies like security restrictions and budget limits.