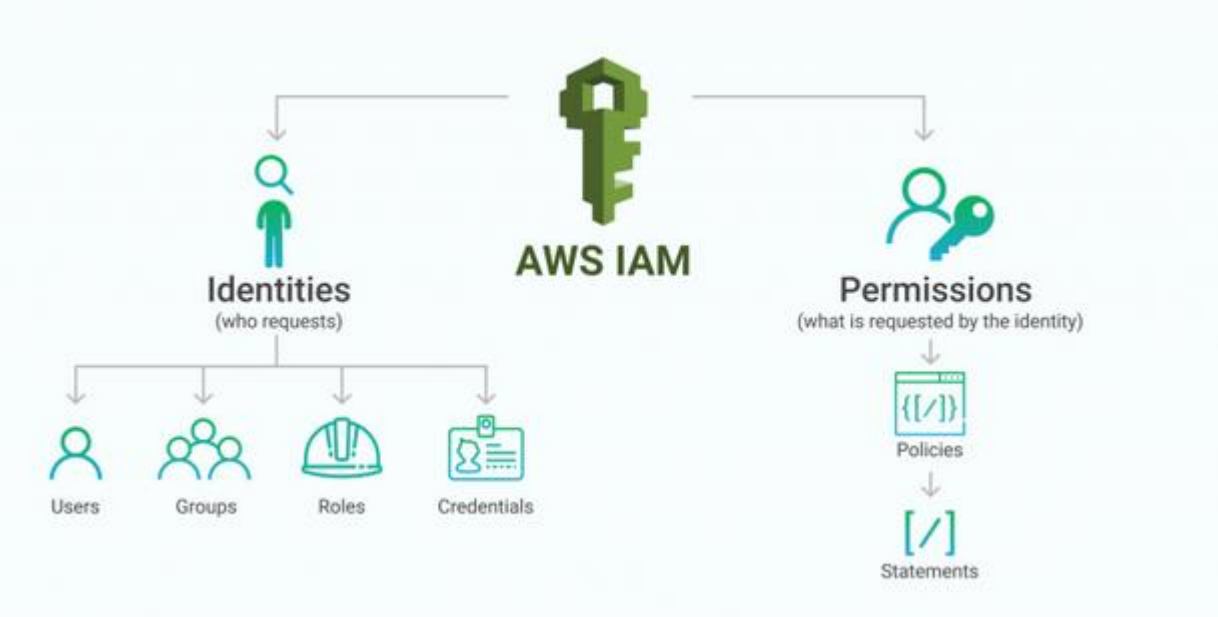
Secure User Management with IAM in AWS:

Introduction to IAM

AWS IAM (Identity and Access Management) is a web service that helps you **securely control access** to AWS resources. Using IAM, you can:

- Create **users** and **groups**
- Assign specific **permissions** using **policies**
- Use **roles** for temporary access
- Enable **Multi-Factor Authentication (MFA)**
- Control which users can access **what**, **how**, and **when**

IAM allows you to **follow the principle of least privilege**—meaning every user gets only the permissions they absolutely need, nothing more.



Key Concepts in IAM

IAM User

- Represents a **single identity** (person or application).
- Can be given long-term **access keys** or **console password**.
- Best practice: No root account usage for daily tasks—create users instead.

IAM Group

- A collection of IAM users.
- Permissions assigned to the group apply to **all users** inside it.

IAM Policy

- A **JSON** document that defines **permissions**.
- Can be attached to a user, group, or role.

IAM Role

- Like a user but **not tied to a single person**.
- Used by applications or services (like EC2) to access other AWS resources securely.

Multi-Factor Authentication (MFA)

- Adds **extra layer of security** by requiring a second factor (e.g., phone or hardware device).

IAM Policies Based on Access Levels

1. S3 Full Access Policy

This policy is designed for users or groups that require complete control over Amazon S3. It allows users to create, delete, upload, download, list, and manage all S3 buckets and objects.

Use Case: Ideal for DevOps engineers or application services managing S3 as a storage backend.

Permissions Granted:

- Create, read, update, delete (CRUD) operations on buckets and objects.
- Manage bucket policies and configurations.

2. S3 Read-Only Access Policy

This policy provides limited, read-only access to all S3 buckets and their contents.

Use Case: Ideal for auditors, developers, or automated services that need to review bucket contents without modifying anything.

Permissions Granted:

- s3:ListBucket
- s3:GetObject

3. EC2 Full Access Policy

This policy allows full administrative control over EC2 services, including managing instances, volumes, security groups, key pairs, and Elastic IPs.

Use Case: Best suited for cloud administrators, DevOps teams, or automation services managing compute infrastructure.

Permissions Granted:

- Start, stop, terminate, and reboot instances.
- Modify security groups, create volumes, and attach or detach them.

4. Administrator Access Policy

This is the most powerful IAM policy in AWS. It provides unrestricted access to all AWS services and resources.

Use Case: Should only be given to highly trusted users such as cloud architects, root account owners, or organization-level admins.

Permissions Granted:

- "Action": "*" and "Resource": "*" — literally every AWS action on every resource.

Security Note: Always attach MFA to administrator users and monitor their activity using AWS CloudTrail.

Best Practices for Assigning IAM Policies

Use least privilege principle: assign only the permissions required.

Apply policies to groups, not individual users, for easier management.

Always enable Multi-Factor Authentication (MFA) for privileged users.

Regularly audit permissions and remove unused access.

Project: Create IAM Users, Groups, and Permissions for a Company

Scenario:

You are the DevOps engineer at a startup. You need to:

- Create 3 IAM users for different team members.
- Create 2 groups: Developers and Admins.
- Assign proper permissions using managed policies.
- Enable MFA for Admins.
- Ensure secure, least-privileged access.

Steps to Implement (Using AWS Console)

Step 1: Login to AWS Console

- Go to https://console.aws.amazon.com/iam
- Use your admin credentials.

Step 2: Create IAM Groups

- Navigate to **IAM > Groups** → Click **Create group**
- Group Name: Developers
    - Attach Policy: AmazonEC2ReadOnlyAccess
- Group Name: Admins
    - Attach Policy: AdministratorAccess

Step 3: Create IAM Users

- IAM > Users → Click **Add user**
- User 1: alice-dev
- User 2: bob-dev
- User 3: charlie-admin

- Select: **Programmatic access** + **AWS Management Console access**
- Set temporary password
- Assign groups accordingly:
    - alice-dev and bob-dev → Developers
    - charlie-admin → Admins

Step 4: Enforce MFA

- IAM > Users > charlie-admin → **Security Credentials**
- Activate **MFA device**
    - Use Virtual MFA (Google Authenticator or Authy)
    - Scan QR code
    - Enter two consecutive MFA codes

Step 5: Create a Custom Policy (Optional)

- IAM > Policies → Create Policy
- Choose **JSON**, paste:

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

- Name it: ListS3ExampleBucket
- Attach this policy to alice-dev to allow only listing one bucket.

Step 6: Test IAM Users

- Log out and log in as each user
- Try accessing services:
    - Developers should only see EC2 in read-only mode
    - Admin should have full access
    - MFA required for Admin
- **AM Users, Groups, Permissions, and MFA – Expected Setup**
- In this project, three IAM users are created and assigned to appropriate IAM groups with defined permissions and Multi-Factor Authentication (MFA) configurations.
- **Alice-dev** is a developer and has been added to the **Developers** group. She has read-only access to EC2 services along with permissions to list S3 buckets. However, Multi-Factor Authentication (MFA) has **not** been enabled for her account.
- **Bob-dev**, another developer, also belongs to the **Developers** group. He has been granted **EC2 ReadOnly** permissions, but similar to Alice, he does **not** have MFA enabled for his account.
- **Charlie-admin** is an administrator and is part of the **Admins** group. He has been provided **Full Admin** privileges, which means he can manage all AWS services and resources. For enhanced security, **MFA is enabled** for Charlie's account.
- This setup ensures that users have access based on their roles and responsibilities while following best practices, like enabling MFA for privileged accounts. It also demonstrates the importance of grouping users and assigning policies at the group level for better manageability and security in AWS Identity and Access Management (IAM).
-

Conclusion:

This project introduces the power of IAM for **secure access control**. Instead of sharing the root account or giving everyone admin rights, IAM helps define roles and responsibilities clearly. It also lays the foundation for more advanced setups like **cross-account roles**, **federated access**, or **IAM policies with conditions**.

Creating IAM Group svglobal and Assigning Administrator Access to Students

Project Objective:

To create a user group in AWS Identity and Access Management (IAM) called svglobal and manually assign **AdministratorAccess** to all the students listed. This group helps manage multiple users easily by assigning policies at the group level.

Key IAM Concepts:

- **IAM (Identity and Access Management):** A secure AWS service that controls users' access to AWS resources.
- **IAM Users:** Individual identities with specific credentials and permissions.
- **IAM Groups:** A collection of IAM users. Policies assigned to a group apply to all users in that group.
- **Policies:** Permissions that define what actions users can perform on AWS resources.

Steps Followed (Manually through AWS Console):

1. **Sign in to AWS Management Console**
   Logged into the AWS account as a user with administrative privileges.
2. **Navigate to IAM Dashboard**
   - Go to Services > IAM.
   - Open the **Groups** section.
3. **Create a New Group svglobal**
   - Click on **Create group**.
   - Enter **Group Name**: svglobal
   - Proceed to the next step to attach policies.
4. **Attach Administrator Access Policy**
   - In the list of policies, search for AdministratorAccess.
   - Select the checkbox next to **AdministratorAccess**.
   - Click **Next** and then **Create Group**.

5. **Create IAM Users (if not already created)**
    - Go to **Users** in IAM.
    - Click **Add users**.
    - Enter usernames and create login credentials (password).
    - Example usernames:
        - Adithya
        - Aishwarya
        - bsravya
        - Gayathri
        - goutham
        - gsravya
        - himabindu
        - Jayathi
        - maneesha
        - Meghana
        - Rachana
        - Sindhu
        - Sowmya
        - Spandana
        - sravya
        - Swetha
6. **Add Users to the Group**
    - For each user:
        - Go to the **Users** tab.
        - Click on the user's name.
        - In the **Groups** section, click **Add user to groups**.
        - Select the svglobal group.

- Click **Add to Group**.

Results:

- The IAM group svglobal was successfully created.
- The AdministratorAccess policy was assigned to the group.
- All listed students were added to this group.
- As a result, each student now has full administrative permissions across the AWS account.

Conclusion:

This project demonstrates a practical example of using AWS IAM to manage multiple users efficiently. Instead of assigning permissions to each user individually, grouping them under svglobal simplifies administration and ensures consistent access levels.