Overview

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources for your users.

IAM is used to control

- **Identity** – who can use your AWS resources (authentication)
- **Access** – what resources they can use and in what ways (authorization)

IAM also enables access to resources across AWS accounts. It's a Global Service and Service usage is **FREE**

Basic Terminologies

1. Identities:

- An IAM identity represents a user, and can be authenticated and then authorized to perform actions in AWS.

- 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.

2. Resources:

- The user, group, role, policy, and identity provider objects that are stored in IAM.

- As with other AWS services, you can add, edit, and remove resources from IAM.

3. Entities:

- The IAM resource objects that **AWS uses for authentication**.

- These include IAM users, federated users, and assumed IAM roles.

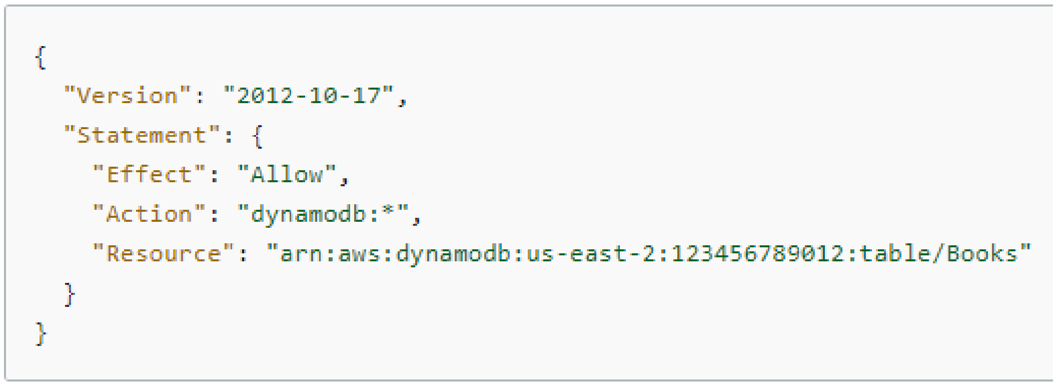
4. Principals:

- A person or application that uses the AWS account root user, an IAM user, or an IAM role to sign in and make requests to AWS.
- Principals include federated users and assumed roles.

IAM Users

1. For greater security and organization, you can give access to your AWS account to specific users—identities that you create with custom permissions
2. When you create an IAM user, they **CAN NOT** access anything in your account until you give them permission.
3. You give permissions to a user by creating an **identity-based policy,** which is a policy that is attached to the user or a group to which the user belongs

   Example:



```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "dynamodb:*",
    "Resource": "arn:aws:dynamodb:us-east-2:123456789012:table/Books"
  }
}
```

Types of users

i. **IAM Root user**
 The user is created when the AWS Account is created.
 User email and password is the credential for the account and he has full administrative rights

ii. **IAM User**
A new user has **implicit deny** for all aws resources when he/she is created.
One or more policies need to add to grant the permission to access AWS resources

iii. **Federated User**
If the users in your organization already have a way to be authenticated, such as by signing in to your corporate network, you don't have to create separate IAM users for them.
Instead, you can **federate** those user identities into AWS.
- Your users already have identities in a corporate directory:
  - Microsoft Active Directory or SAML identity federation
- Your users already have Internet identities:
  - Amazon, Facebook, Google, OpenID connect

iv. **Power Users:**
These are the users who have full access to AWS Services but management of IAM users, groups is **not allowed** to them.

Note: An IAM user doesn't have to represent an actual person; you can create an IAM user in order to generate an access key for an application that runs in your corporate network and needs AWS access.

IAM Groups

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.

IAM Permissions

What is permission and why do we need it?

Permissions let you specify Access to AWS resources.

Permissions are granted to IAM identities (users, groups, and roles) and by default these entities start with **no permissions**.

In other words, IAM identities can do nothing in AWS until you grant them your desired permissions.

How to give permission?

To give entities permissions, you can attach a policy that specifies the type of access, the actions that can be performed, and the resources on which the actions can be performed.

IAM Principals in detail

Use the `Principal` element in a policy to specify the principal that is allowed or denied access to a resource.

You **cannot** use the `Principal` element in an IAM identity-based policy.

You can use it in the **Trust policies for IAM roles and in Resource-based policies**.

You can specify any of the following principals in a policy:

1. AWS account and Root user Principal

When you use an AWS account identifier as the principal in a policy, you delegate authority to the account. An administrator in that account can then grant access to any identity in that account. This includes IAM users and roles in that account.

For example, given an account ID of `123456789012`, you can use either of the following methods to specify that account in the `Principal` element:

```
"Principal": {"AWS": "arn:aws:iam::123456789012:root" }
"Principal": {"AWS": "123456789012" }
"Principal": {"AWS": [ "arn:aws:iam::123456789012:root", "9999999" ]}
```

Some AWS services support additional options for specifying an account principal. For example, Amazon S3 lets you specify a canonical user ID using the following format:

```
"Principal": { "CanonicalUser": "79a59df98218e7cd47ef2be" }
```

2. IAM users Principal

You can specify an individual IAM user (or array of users) as the principal, as in the following examples.

When you specify more than one principal in the element, you grant permissions to each principal. **This is a logical `OR` and not a logical `AND`, because you are authenticated as one principal at a time.**
In a `Principal` element, the user name is case-sensitive.

```
"Principal": {
  "AWS": [
    "arn:aws:iam::AWS-account-ID:user/user-name-1",
    "arn:aws:iam::AWS-account-ID:user/UserName2"
  ]
}
```

When you specify users in a `Principal` element, you **cannot** use a wildcard (`*`) to mean "all users". Principals must always name a specific user or users.

3. Federated users (using web identity or SAML federation) Principal

- Federated web identity user principals

```
"Principal": { "Federated": "cognito-identity.amazonaws.com" }
"Principal": { "Federated": "www.amazon.com" }
"Principal": { "Federated": "graph.facebook.com" }
"Principal": { "Federated": "accounts.google.com" }
```

- Federated SAML user principals

```
"Principal": { "Federated":
"arn:aws:iam::AWS-account-ID:saml-provider/provider-name" }
```

4. IAM role Principal

```
"Principal": { "AWS": "arn:aws:iam::AWS-account-ID:role/role-name" }
```

5. Specific assumed-role session principals

```
"Principal": { "AWS":
"arn:aws:sts::AWS-account-ID:assumed-role/role-name/role-session-name"
}
```

When you specify an assumed-role session in a `Principal` element, you cannot use a wildcard (*) to mean "all sessions". Principals must always name a specific session.

6. AWS services principal

Check the notes from IAM Roles

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "ecs.amazonaws.com",
          "elasticloadbalancing.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

7. Anonymous users (not recommended) principal (Public Access)

For resource-based policies, such as Amazon S3 bucket policies, a wildcard (*) in the principal element specifies all users or public access.

AWS strongly recommend that you do not use a wildcard in the `Principal` element in a role's trust policy unless you otherwise restrict access through a `Condition` element in the policy

The following elements are equivalent:

```
"Principal": "*"
"Principal" : { "AWS" : "*" }
```

AWS EC2 Instance Metadata

It allows AWS EC2 instances to learn about themselves without using an IAM Role for that purpose.

The URL: http://169.254.169.254/latest/meta-data/