Users and Groups

What is IAM?

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. It enables you to manage users, groups, and permissions to allow or deny access to AWS resources.

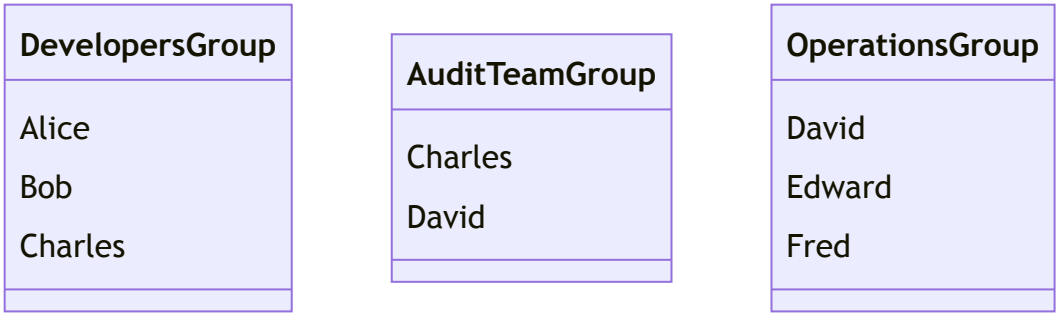
What is a user?

An IAM user is an entity that you create in AWS to represent the person or service that uses it to interact with AWS. A user in AWS consists of a name and credentials. The credentials are used to authenticate the user when interacting with AWS services and resources.

What is a 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. For example, you could have a group called `Admins` and give that group the types of permissions that administrators typically need or a group called `Developers` and give that group the types of permissions that developers typically need.

- Groups only contain users, not other groups.
- users can belong to multiple groups or none at all.



What is a policy?

A policy, in the context of cloud services like AWS, is a document that defines permissions and specifies what actions are allowed or denied for various resources. It is used to manage access to AWS resources and services securely. Policies are written in JSON (JavaScript Object Notation) and include elements such as the version, an identifier, a list of permissions (statements), and effect, action, resource, and condition keys that outline the specifics of the access permissions.

For example, an AWS Identity and Access Management (IAM) policy allows you to specify who (users, groups, roles) has permission to access which AWS resources and what they can do with those resources.

In AWS, you apply the principle of least privilege by granting only the permissions required to perform a task.

> **Note**
>
> The **principle of least privilege** is a security concept that refers to giving a user account or process only those privileges which are essential to perform its intended function. In other words, a user or application should have the minimum levels of access—or permissions—needed to perform their tasks. This helps reduce the attack surface by limiting access rights for users, accounts, and computing processes to only those resources absolutely required to carry out legitimate activities. If a system is compromised, the principle of least privilege can help contain the potential damage by limiting the capabilities that an attacker would have at their disposal.

Policy Structure

A policy consists of the following elements:

- **Version**: The version of the policy language that the policy uses. The current version is 2012-10-17.
- **Statement**: The statement is the main part of the policy. It consists of an array of individual statements, each of which includes the following elements:
  - **Effect**: Whether the statement allows or denies access. The value must be `Allow` or `Deny` and is case sensitive.
  - **Action**: The specific action or actions that the policy allows or denies. For example, `s3:ListBucket` or `s3:*`. Statements must include either an Action or NotAction element.

- **Resource**: The resource to which the statement applies. For example, an Amazon S3 bucket or an IAM user. Statements must include either a Resource or a NotResource element.
- **Condition**: The conditions under which the policy grants permission. For example, to grant permission only if a request is made over SSL.
- **Principal**: The entity that the policy is applied to. For example, an IAM user, an AWS account, or all principals in an account.
- **Sid**: A statement ID that you can use to refer to the statement in other parts of the policy.

Example Policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1234567890",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/ExampleUser"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::example-bucket/*",
      "Condition": {
        "StringEquals": {
          "s3:prefix": "home/"
        },
        "IpAddress": {
          "aws:SourceIp": "192.0.2.0/24"
        }
      }
    }
  ]
}
```