# Identity Access Management (IAM)

Manages **access** of AWS **users** and **resources.**

# IAM – Core Components

IAM allows **management** of access of **users** and **resources**

**IAM Identities**

IAM **Users** — End users who log into the console or interact with AWS resource programmatically

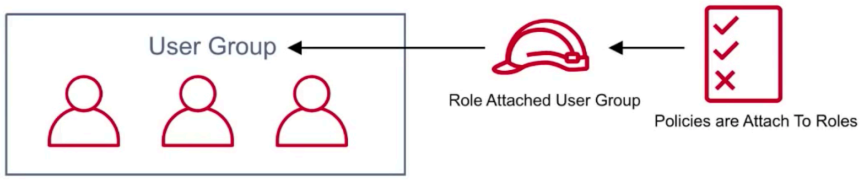
IAM **Groups** — Group up your Users so they all share permission levels of the group
eg. Administrators, Developers, Auditors

IAM **Roles** — Associate permissions to a Role and then assign this to an Users or Groups

IAM **Policies** — JSON documents which grant permissions for a specific user, group, or role to access services. Policies are attached to to **IAM Identities**

# IAM – Core Components

A user can belong to a group.
Roles can be applied to groups to quickly add and remove permissions en-masse to users




A user can have a role directly attached
An policy can be directly attached to a user (called an **Inline Policy**)

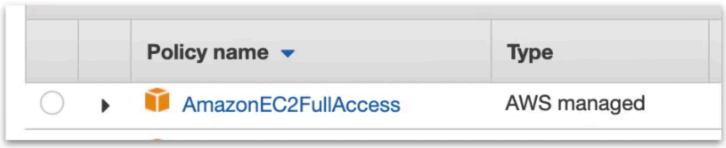

Roles can have many policies attached


Various AWS resources allow you attach roles directly to them.

# IAM – Managed vs Customer vs Inline Policy

### Managed Policies

A policy which is managed by AWS, which you cannot edit. Managed policies are labeled with an **orange box**




### Customer Managed Policies

A policy created by the customer which is editable. Customer policies have no symbol beside them.


### Inline Policies

A policy which is directly attached to the user.

# IAM – Policies

**Version** policy language version. **2012-10-17** is the latest version.

**Statement** container for the policy element you are allowed to have multiples

**Sid** (optional) a way of labeling your statements.

**Effect** Set whether the policy will Allow or Deny

**Principal** account, user, role, or federated user to which you would like to allow or deny access

**Action** list of actions that the policy allows or denies

**Resource** the resource to which the action(s) applies

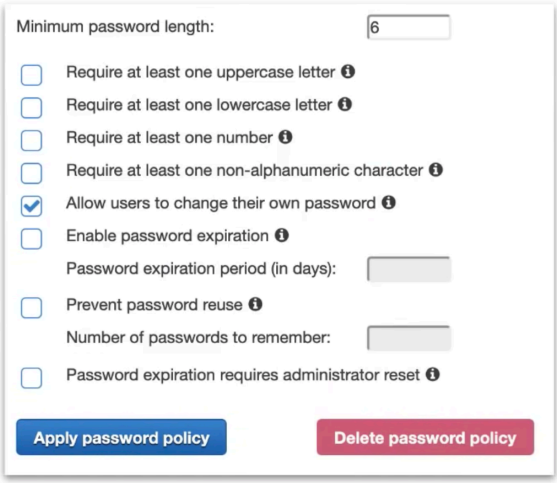
**Condition** (optional) circumstances under which the policy grants permission

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "Deny-Barclay-S3-Access",
    "Effect": "Deny",
    "Action": "s3:*",
    "Principal": {"AWS": ["arn:aws:iam::123456789012:barclay"]},
    "Resource": "arn:aws:s3:::my-bucket"
  },
  {
    "Effect": "Allow",
    "Action": "iam:CreateServiceLinkedRole",
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "iam:AWSServiceName": [
          "rds.amazonaws.com",
          "rds.application-autoscaling.amazonaws.com"
        ]
      }
    }
  }]
}
```

# IAM – Password Policy

In IAM you can set a **Password Policy**

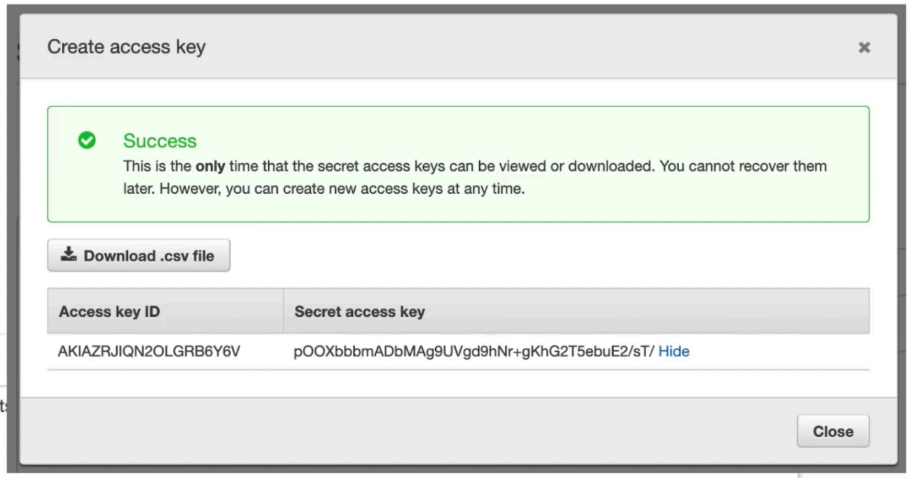
To set the minimum requirements of a password and **rotate** passwords so users have to update their passwords after X days

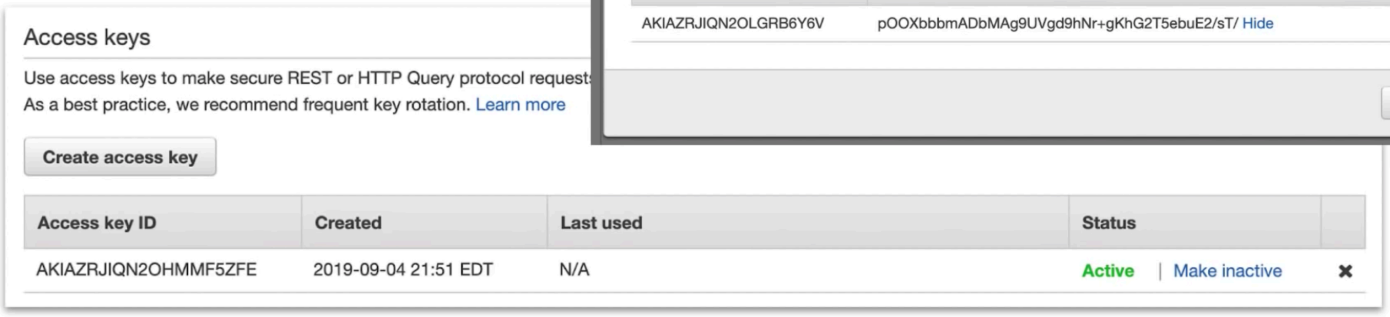


# IAM – Access Keys

Access Keys allow users to interact with AWS service **programmatically** via the AWS CLI or AWS SDK

You're allowed two Access keys per user.



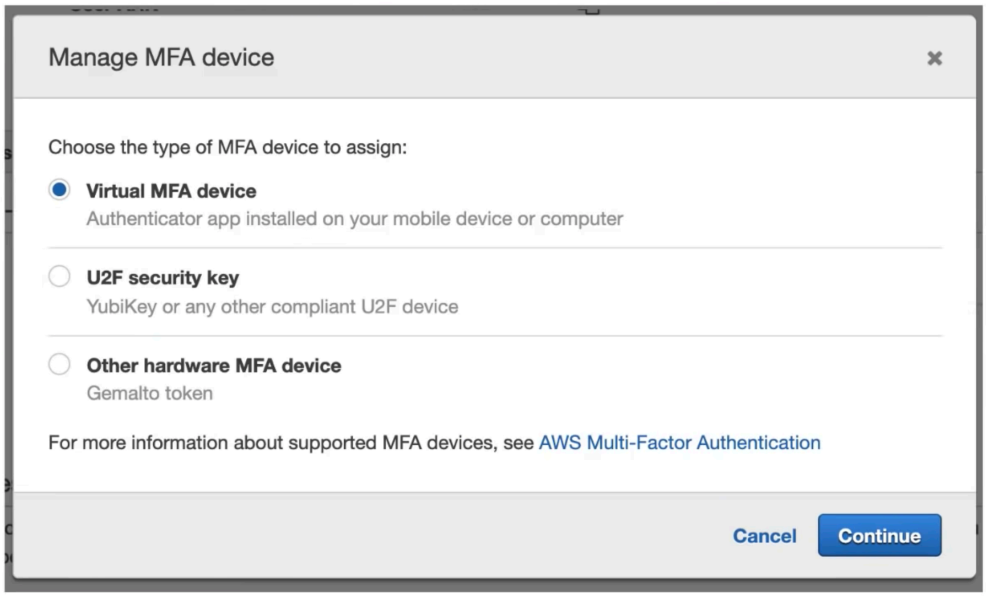



# IAM – MFA

Multi-factor authentication (MFA ) can be turned on per user.

The user has to turn on MFA themselves, Administrator cannot directly enforce users to have MFA.

They Administrator account could create a policy requiring MFA to access certain resources.



# IAM CheatSheet

- **Identity Access Management** is used to manage **access** to users and resources
- IAM is a universal system. (applied to all regions at the same time). IAM is a free service
- A root account is the account initially created when AWS is set up (full administrator)
- New IAM accounts have no permissions by default until granted
- New users get assigned an Access Key Id and Secret when first created when you give them programmatic access
- Access Keys are only used for CLI and SDK (cannot access console)
- Access keys are only shown once when created. If lost they must be deleted/recreated again.
- Always setup MFA for Root Accounts
- Users must enable MFA on their own, Administrator cannot turn it on for each user
- IAM allows your set password policies to set minimum password requirements or rotate passwords
- **IAM Identities** as Users, Groups, and Roles
- **IAM Users** End users who log into the console or interact with AWS resources programmatically
- **IAM Groups** Group up your Users so they all share permission levels of the group
- eg. Administrators, Developers, Auditors
- **IAM Roles** Associate permissions to a Role and then assign this to an Users or Groups
- **IAM Policies** JSON documents which grant permissions for a specific user, group, or role to access services. Policies are attached to to IAM Identities
- **Managed Policies** are policies provided by AWS and cannot be edited
- **Customer Managed Policies** are policies created by use the customer, which you can edit
- **Inline Policies** are policies which are directly attached to a user