IAM (Identity and Access Management)

O **IAM (Identity and Access Management)** é o serviço da AWS que orquestra o controle de acessos, permitindo gerenciar de forma segura quem pode acessar seus recursos e o que eles podem fazer. Ele oferece um controle refinado sobre permissões, permitindo ou negando acesso a qualquer solicitação com base em políticas definidas.

Exemplo de uso:

Com o IAM, você pode conceder a um usuário a permissão para **listar** os itens de um bucket no S3, mas **não** a permissão para adicionar novos itens. As permissões são altamente personalizáveis, permitindo definir ações específicas que cada identidade (usuário, grupo ou função) pode realizar com os recursos, garantindo segurança e controle granular.

Você pode realizar controle granular de permissões tanto através de Resource-Based Policies (RBP) quanto de Identity-Based Policies (IBP), porém de maneiras diferentes:

Resource-Based Policies (RBP):

Essas políticas são anexadas diretamente a um recurso, como um bucket S3 ou uma fila SQS.

Elas permitem que você controle quem pode acessar o recurso e o que pode fazer com ele, inclusive identidades de outras contas AWS.

O controle é aplicado no nível do recurso em si, permitindo a definição de ações específicas para usuários ou funções.

Exemplo: Você pode permitir que um usuário de outra conta AWS acesse um bucket S3 para listar objetos, mas não para adicioná-los.

Identity-Based Policies (IBP):

Essas políticas são anexadas a identidades (usuários, grupos ou funções) e controlam o que essa identidade pode fazer em todos os recursos da AWS.

O controle é aplicado no nível da identidade, permitindo definir quais ações a identidade pode executar em recursos específicos ou em todos os recursos da conta AWS.

Exemplo: Você pode conceder a um usuário IAM da sua própria conta AWS a permissão para listar instâncias EC2, mas não para iniciar ou parar instâncias.

Controle Granular:

Você pode realizar o controle granular em ambas as abordagens. No caso das RBP, o controle é focado no recurso, enquanto nas IBP, o controle é focado na identidade.

Em muitos casos, as duas políticas são usadas juntas para um controle ainda mais refinado e seguro, aplicando o princípio do mínimo privilégio — ou seja, garantindo que as identidades tenham apenas as permissões necessárias para realizar suas tarefas.

Métodos de Acesso à AWS

- **CLI**: Interface de linha de comando para gerenciar serviços AWS.
- **CloudShell**: Um ambiente shell baseado em navegador, que simula uma instância EC2 (Elastic Compute Cloud) temporária após o login via console. Ele não persiste os dados, oferecendo uma solução mais econômica para executar comandos na AWS.
- **Console**: Interface gráfica da AWS para gerenciar e visualizar serviços.
- **API**: Aplicações podem se conectar e gerenciar serviços AWS diretamente por meio de APIs.

O que podemos criar com IAM?

- **Usuários**: Contas individuais para acessar recursos da AWS.
- **Grupos**: Conjuntos de usuários com permissões similares.
- **Funções (Roles)**: Permissões temporárias atribuídas a usuários ou serviços para acessar recursos AWS.
- **Conexão de Aplicações**: Permitir que aplicações acessem recursos da AWS de forma segura, utilizando funções do IAM.
- **Políticas por Recursos (Resource-Based Policy - RBP)**: Políticas anexadas diretamente aos recursos AWS, controlando quem pode acessá-los.
- **Políticas por Identidade (Identity-Based Policy - IBP)**: Políticas anexadas a identidades (usuários, grupos, funções), controlando o que essas identidades podem fazer.

Tipos de Políticas na AWS

01. Políticas por Recursos (Resource-Based Policy - RBP)

As **Resource-Based Policies** são anexadas diretamente aos recursos da AWS, como buckets no S3, filas no SQS ou tópicos no SNS. Elas especificam quem pode acessar esses recursos e quais ações essas entidades (usuários, funções ou contas) podem realizar.

Características principais:

- **Anexadas a recursos**: Escritas diretamente nos recursos.
- **Controle de acesso externo**: Permitem acesso a identidades fora da conta AWS (outras contas AWS).
- **Controle granular**: Oferecem controle mais específico sobre o compartilhamento e a segurança do recurso.

Exemplo:

Um bucket S3 pode ter uma Resource-Based Policy permitindo que um usuário de outra conta acesse seu conteúdo.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/John"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::example-bucket/*"
    }
```

```
  ]
}
```

02. Políticas por Identidade (Identity-Based Policy - IBP)

As **Identity-Based Policies** são anexadas a identidades (usuários, grupos ou funções IAM). Elas definem o que essas identidades podem fazer em relação aos recursos da AWS.

Características principais:

- **Anexadas a identidades**: Aplicadas diretamente a usuários, grupos ou funções no IAM.
- **Controle por identidades**: Determinam o que as identidades podem fazer, e em quais recursos.
- **Permissões flexíveis**: Podem permitir ou negar ações.

Exemplo:

Uma política pode definir que um usuário ou função específica tenha permissão para iniciar uma instância EC2.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:StartInstances",
      "Resource":
"arn:aws:ec2:us-west-2:123456789012:instance/i-1234567890abcdef0"
    }
  ]
}
```

Diferenças-chave entre RBP e IBP:

Característica	Resource-Based Policy (RBP)	Identity-Based Policy (IBP)
Onde é anexada	No recurso da AWS (S3, SNS, SQS, etc.)	Na identidade (usuário, grupo, função IAM)
Controle de acesso externo	Permite acesso a identidades de outras contas AWS	Aplica-se apenas a identidades dentro da conta AWS
Permissões explícitas	Define diretamente quem pode acessar o recurso	Define o que a identidade pode fazer
Uso típico	Compartilhamento de recursos entre contas ou controle granular	Gerenciamento de permissões para identidades

03. Criação de Usuário

Ao criar um novo usuário no IAM, ele recebe automaticamente **credenciais de acesso** que permitem o uso de ferramentas como **CLI**, **CloudShell**, **Console da AWS** e **APIs** para interagir com os serviços da AWS. No entanto, **por padrão, o usuário não tem permissões** para realizar qualquer ação nos recursos da AWS.

Para conceder as permissões adequadas, é necessário **anexar políticas de acesso**. Essas políticas podem ser de dois tipos:

- **Políticas por Recursos (RBP - Resource-Based Policies)**: Definem permissões diretamente nos recursos (como um bucket S3 ou uma fila SQS).
- **Políticas por Identidade (IBP - Identity-Based Policies)**: São associadas às identidades (usuários, grupos ou funções) e definem o que essa identidade pode fazer em toda a conta AWS ou em recursos específicos.

Sem essas políticas, o usuário terá acesso ao ambiente, mas não poderá realizar nenhuma ação até que as permissões sejam atribuídas.

04. Criação de Grupos

No IAM, **grupos** são utilizados para **agrupar usuários que necessitam das mesmas permissões** para acessar recursos da AWS. Ao criar um grupo, você pode atribuir a ele políticas de acesso (Identity-Based Policies - IBP), e todos os usuários pertencentes a esse grupo herdarão automaticamente essas permissões.

Essa abordagem facilita a administração de permissões, já que, ao invés de gerenciar políticas individualmente para cada usuário, você pode simplesmente adicionar ou remover usuários do grupo, garantindo que eles tenham o mesmo nível de acesso aos recursos.

05. Roles (Regras)

As Roles no IAM são usadas para permitir que recursos da AWS acessem outros recursos de forma segura, sem a necessidade de credenciais fixas. Ao criar uma Role, você define permissões que podem ser atribuídas a um recurso, como uma instância EC2, permitindo que esse recurso interaja com outros serviços da AWS, como o S3.

Por exemplo, você pode criar uma Role que permite que uma instância EC2 acesse e faça upload de arquivos em um bucket S3. Quando a Role é atribuída à instância EC2, ela herda as permissões necessárias para se comunicar com o S3 sem precisar de chaves de acesso explícitas.

Exemplo:

Uma Role pode ser configurada para permitir que o EC2 acesse o S3 para armazenar logs ou recuperar arquivos, facilitando a integração entre esses serviços de forma segura.

06. Políticas (Policies)

As **Políticas (Policies)** no IAM são documentos que definem as **permissões** para usuários, grupos ou recursos na AWS. Elas controlam o que cada entidade pode ou não fazer, especificando quais ações podem ser realizadas, em quais recursos e sob quais condições.

Ao atribuir políticas a **grupos**, **usuários**, ou **roles**, você determina **os níveis de acesso** a serviços e recursos da AWS. As políticas podem ser de dois tipos principais:

- **Políticas por Identidade (IBP - Identity-Based Policies)**: Associadas a identidades (usuários, grupos, roles) e definem o que essas identidades podem fazer.
- **Políticas por Recursos (RBP - Resource-Based Policies)**: Associadas diretamente a recursos específicos, como um bucket S3, e determinam quem pode acessá-los e de que forma.

Essas políticas são escritas em formato **JSON**, especificando as permissões de "Allow" (permitir) ou "Deny" (negar), o que garante um controle granular sobre o acesso aos recursos na AWS.

07. Acesso via Console

O **acesso ao Console da AWS** é realizado por meio de **usuário e senha**, fornecendo uma interface gráfica para gerenciar e interagir com os serviços da AWS. Para aumentar a segurança desse acesso, você pode habilitar o **MFA (Autenticação Multifator)**, que adiciona uma camada extra de proteção.

Quando o **MFA** está ativado, além de inserir o usuário e senha, o usuário também precisará fornecer um **código de autenticação** gerado por um dispositivo de MFA, como um aplicativo no celular ou um **token físico** (inserido via USB). Esse código é temporário e único, garantindo que mesmo que alguém tenha acesso às credenciais, não conseguirá acessar o console sem o código MFA.

Em resumo, o **MFA** é uma camada adicional de segurança que protege sua conta AWS contra acessos não autorizados.

8. Acesso via API ou CLI

Para acessar a AWS via **API** ou **CLI (Command Line Interface)**, é necessário obter um par de credenciais, que inclui a **AccessKeyID** e a **SecretAccessKey**. Essas chaves de acesso são **credenciais de longa duração** usadas para autenticar solicitações programáticas à AWS.

Funcionamento:

- **AccessKeyID**: Identifica o usuário ou a entidade que está realizando a solicitação.
- **SecretAccessKey**: Atua como a senha, usada para assinar as solicitações enviadas à AWS.

Ao criar um par de chaves, ambas são **disponibilizadas apenas uma vez** no momento da criação. Portanto, é fundamental **armazená-las com segurança** nesse momento, pois, se perdidas, não poderão ser recuperadas. Nesse caso, você precisará gerar um **novo par de chaves**, o que exigirá a **atualização** em todas as APIs ou sistemas que utilizam as chaves antigas para autenticação.

Boas Práticas:

- Evite o uso de chaves de longa duração sempre que possível, utilizando soluções temporárias como **roles** com **STS (Security Token Service)**.
- Caso as chaves sejam comprometidas, revogue-as imediatamente e crie um novo par.
- Armazene as chaves de maneira segura, utilizando gerenciadores de segredos ou sistemas criptografados.

É possível ter **dois pares de chaves de acesso ativos** por usuário programático no IAM. Essa funcionalidade permite que você **gire chaves de acesso** de forma segura, sem interromper suas operações.

Cenários comuns para o uso de dois pares de chaves:

1. **Rotação de chaves**: Para cumprir boas práticas de segurança, recomenda-se rotacionar periodicamente as chaves de acesso. Tendo dois pares, você pode adicionar um novo par, atualizar suas aplicações para utilizar as novas chaves, e então desativar ou excluir o par antigo.
2. **Migração de aplicações**: Durante a migração de aplicações ou sistemas que dependem de chaves de acesso, ter dois pares ativos permite uma transição suave entre as credenciais.

Importante:

- Cada usuário programático pode ter **no máximo dois pares de chaves de acesso** ativos ao mesmo tempo.
- Ao criar novas chaves de acesso, lembre-se de seguir as boas práticas de segurança, como **armazenar as chaves de forma segura** e revogar as chaves antigas quando não forem mais necessárias.

Isso garante maior flexibilidade na gestão de credenciais sem comprometer a segurança.

Na **API** e **CLI**, ao invés de usar diretamente **chaves de acesso permanentes**, é possível configurar o uso de **roles** (regras) para permitir que os usuários programáticos acessem recursos da AWS de maneira mais segura. As **roles** são especialmente úteis em cenários onde é preferível utilizar **credenciais temporárias** e mais seguras.

Funcionamento:

1. **Roles com STS (Security Token Service)**: Ao utilizar uma role, você pode gerar credenciais temporárias com o **STS**. Essas credenciais temporárias são utilizadas para acessar os recursos, e possuem um tempo de expiração, o que as torna mais seguras que chaves de acesso de longa duração.
2. **Atribuição de roles a instâncias EC2**: Outra aplicação comum é atribuir uma role a instâncias **EC2** ou a serviços da AWS, permitindo que esses serviços acessem recursos sem a necessidade de gerenciar chaves de acesso diretamente.

Vantagens:

- **Maior segurança**: As roles evitam o uso de credenciais permanentes, utilizando credenciais temporárias que expiram automaticamente.
- **Controle refinado**: As roles podem ter permissões específicas, limitando o que o usuário ou serviço pode fazer com os recursos.
- **Gerenciamento centralizado**: As roles são gerenciadas pelo IAM, permitindo controlar e auditar facilmente o acesso aos recursos.

Portanto, sim, a API e a CLI podem utilizar **roles** atribuídas a usuários programáticos para acessar recursos da AWS, proporcionando um controle mais seguro e eficiente do acesso.

O **AWS STS (Security Token Service)** pode utilizar **AccessKeyID** e **SecretAccessKey** para gerar **credenciais temporárias**, mas isso depende do contexto e da maneira como você solicita essas credenciais.

Aqui está como funciona em diferentes cenários:

1. Credenciais temporárias para usuários IAM:

Quando você solicita credenciais temporárias usando o STS com um **usuário programático IAM** (que possui uma **AccessKeyID** e **SecretAccessKey**), o STS usa essas chaves para autenticar a solicitação e então retorna um conjunto de **credenciais temporárias**. Essas credenciais temporárias incluem:

- Um **AccessKeyID** temporário.
- Uma **SecretAccessKey** temporária.
- Um **token de segurança** adicional (Session Token) que expira após um período definido.

Essas credenciais temporárias são usadas para acessar recursos da AWS durante o tempo de validade.

2. Assumindo uma role:

Ao usar o STS para **assumir uma role** (como quando você atribui uma role a uma instância EC2 ou a um serviço), não são necessárias **AccessKeyID** e **SecretAccessKey** permanentes. Em vez disso, você assume a role e o STS gera automaticamente as credenciais temporárias para o serviço ou instância, sem precisar de credenciais de longa duração.

Nesse cenário, o serviço que assume a role obtém as credenciais temporárias diretamente e não precisa de chaves permanentes, tornando o processo mais seguro.

Resumo:

- Quando você solicita credenciais temporárias como um **usuário IAM**, o STS usa suas **AccessKeyID** e **SecretAccessKey** permanentes para gerar as credenciais temporárias.
- Quando você **assume uma role**, como em instâncias EC2 ou serviços da AWS, o STS gera diretamente as credenciais temporárias sem precisar de **AccessKeyID** e **SecretAccessKey** permanentes.

Esses mecanismos ajudam a melhorar a segurança ao limitar o uso de chaves permanentes e incentivar o uso de credenciais temporárias que expiram após um tempo determinado.