Gerenciando acesso à AWS com o IAM

"Identity and Access Management"

Definição

O IAM, é um serviço de configuração de usuários, grupos, políticas e funções de acesso a ações e recursos de uma conta na AWS. Ele é o motor de partida para garantir a funcionalidade em todas as atividades executadas em uma conta AWS.

Sua correta configuração, seguindo as boas práticas sugeridas pela AWS, garante um ambiente bem mais seguro para trabalhar.

A lógica é definir usuários ou grupos que terão acesso à conta da AWS, mediante políticas e/ou funções de permissionamento.

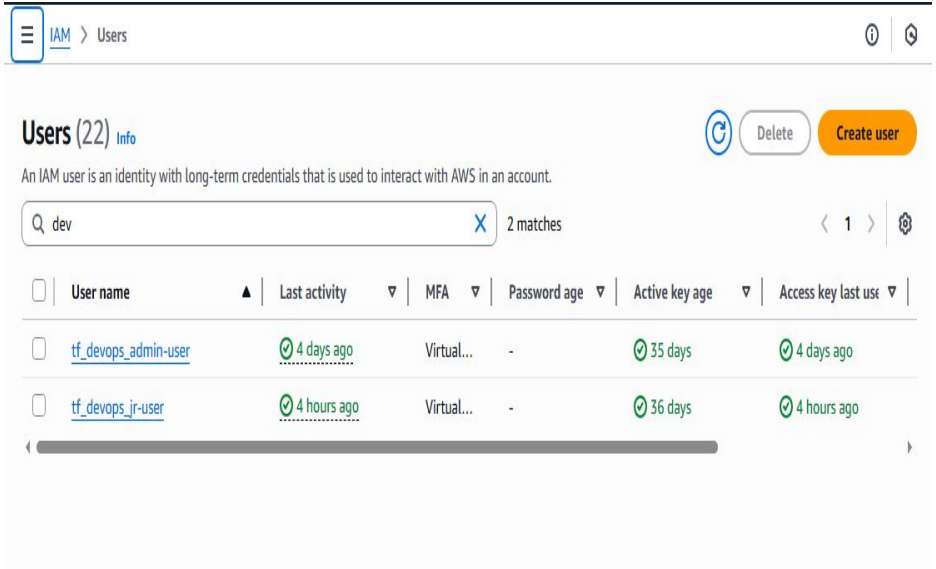
Ou seja, determinar quais ações poderão ser executadas em determinados recursos da AWS.

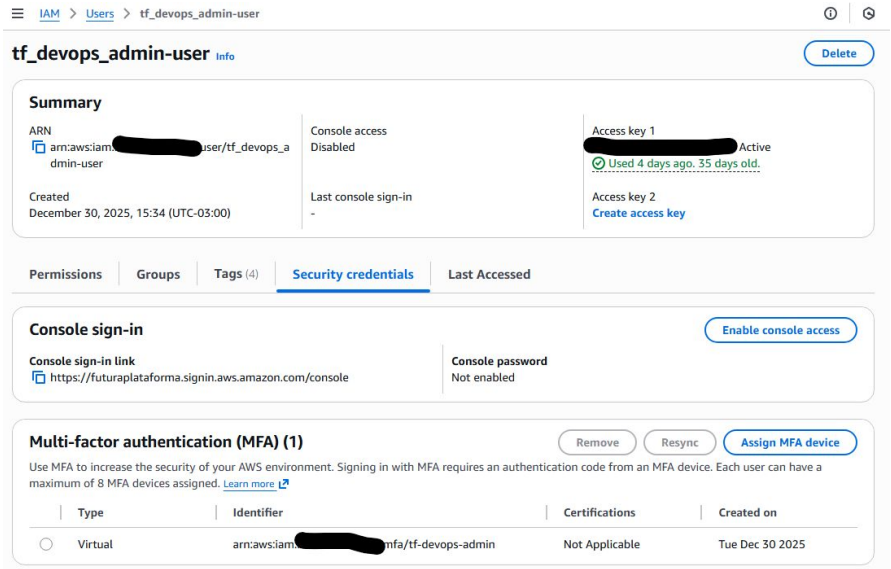
Composição

- Usuários **(Users)**
- Grupos **(Groups)**
- Politicas **(Polices)**
- Funções **(Roles)**

Usuários

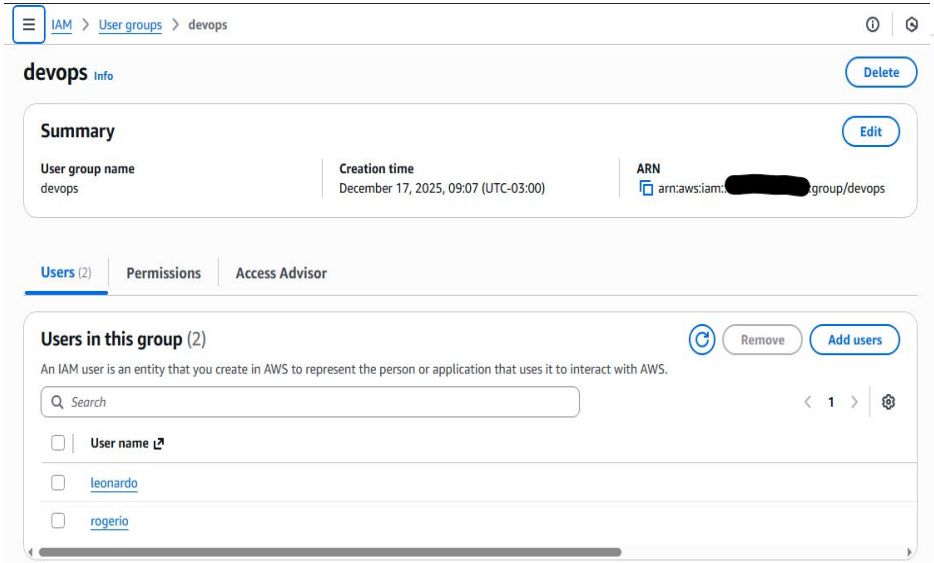
Entidades individuais de acesso à conta AWS.
Ao efetuar o login com o usuário ele assume as políticas de permissionamento que estiverem atribuídas à ele.

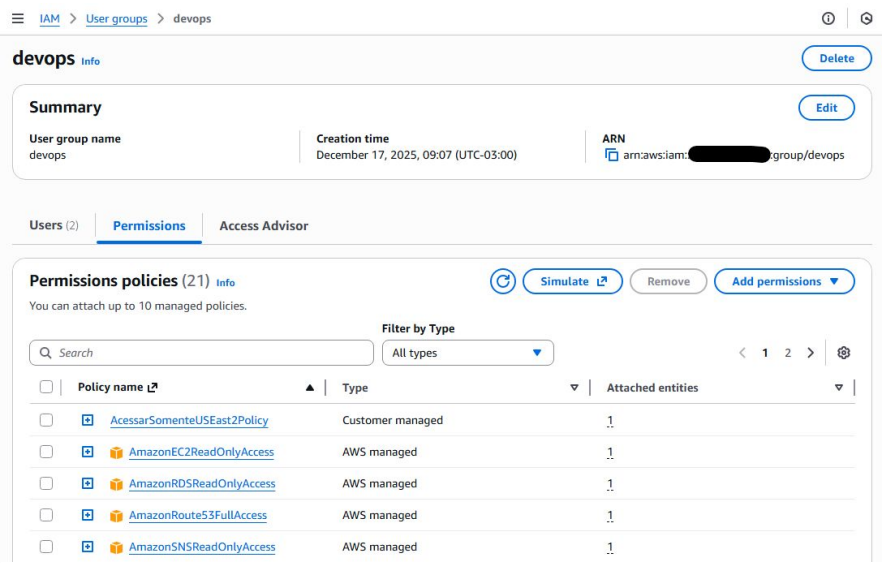




Grupos

Entidades de agrupamento de usuários utilizadas para a configuração de permissões que serão compartilhadas entre todos os membros.
Ou seja: Todos os usuários vinculados à ela possuirão as mesmas permissões.





Políticas

Regras de permissionamento ou recusa de acesso a recursos dentro da conta AWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EC2ReadAndStartStop",
            "Effect": "Allow",
            "Action": [
                "ec2:Describe*",
                "ec2:StartInstances",
                "ec2:StopInstances",
                "ec2:RebootInstances"
            ],
            "Resource": "*"
        }
    ]
}
```

Composição:

Sid → Nome do controle que será efetuado

Effect → Informa se a política é de permissão (Allow) ou negação (DENY)

Action → Quais ações de um determinado recurso serão cobertas pela configuração

Resource → Identificação do recurso que será afetado

Políticas (Exemplos)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyAllIfNoMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:CreateVirtualMFADevice",
                "iam:EnableMFADevice",
                "iam:ListVirtualMFADevices",
                "iam:ListMFADevices",
                "iam:GetUser",
                "iam:ChangePassword",
                "iam:GetAccountPasswordPolicy"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": "false"
                }
            }
        }
    ]
}
```

Nega qualquer ação que não esteja entre as especificadas, **enquanto usuário não estiver logado com a autenticação multi-fator**.

Ele é obrigado a configurar a autenticação para poder ter acesso às demais permissões que possua.

Políticas (Exemplos)

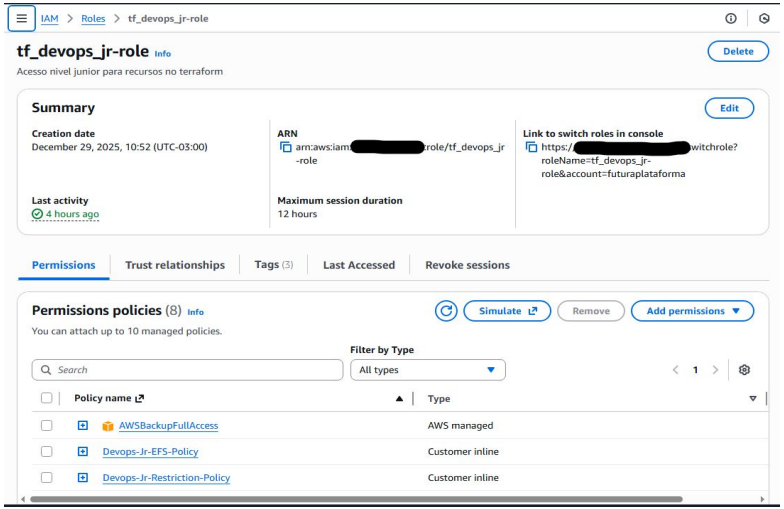
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyCriticalActions",
            "Effect": "Deny",
            "Action": [
                "organizations:*",
                "account:*",
                "billing:*",
                "ec2:Delete*",
                "ecs:Delete*",
                "rds:Delete*"
            ],
            "Resource": "*"
        }
    ]
}
```

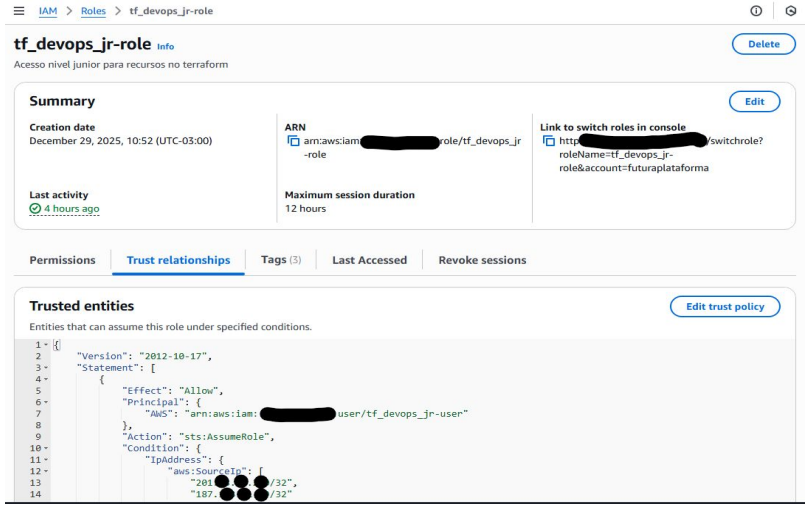
Nega o acesso à recursos como organizations, accounts e billing (recursos administrativos da conta) e não permite ações de exclusão nos recursos EC2, ECS e RDS.

Funções

Configuração de papéis com as políticas de acesso que serão autorizadas para usuários ou serviços que as assumirem.

Podem ser atribuídas diretamente a recursos da AWS ou a usuários (através de uma açã chamada assume role) dando permissão para acessar recursos/serviços que estejam liberados em suas políticas.





Assumindo Funções (Assume Role)

Assume Role, não é nada mais do que assumir uma função previamente configurada com determinadas permissões de acesso.

Antes de mais nada, precisa existir um usuário IAM configurado, sem políticas atribuídas e com credencial de acesso criada para ter acesso a um access key e um secret access key que posteriormente permitirá que obtenhamos as permissões de acesso para nossa máquina. Também é possível tornar o acesso ainda mais seguro habilitando o acesso multi-fator ao seu usuário (MFA).

Para este processo nós Usamos as funções com as políticas de acesso criadas no IAM para que usuários ou serviços possam assumam estes configurações.

Assumindo Funções (Assume Role)

Após efetuadas as configurações de usuário e funções, precisamos executar comandos do AWS CLI (Interface de linhas de comando da AWS) para aplicar essas funções para nosso usuário e assim termos acesso aos recursos liberados por ela.

O primeiro passo é configurar as credenciais do usuário que poderá assumir a função.

Para isso podemos usar alguns recursos como:

- Setar a credencial no arquivo de configurações da aws usando o AWS Configure
- Setar manualmente a variável de ambientes do sistema operacional

Como exemplo usaremos o powershell para a demonstração

Assumindo Funções (Assume Role) - Demonstração

Setaremos manualmente as credenciais e em seguida verificaremos se o usuário está configurado com o comando **aws sts get-caller-identity.**

```
$env:AWS_ACCESS_KEY_ID     = 'AKIAHFH6FGH213GFFG'
$env:AWS_SECRET_ACCESS_KEY = 'FGFGDF564FDG52FD54FGG'
$env:AWS_DEFAULT_REGION    = 'us-east-1'
```

```
C:\> aws sts get-caller-identity

{
    "UserId": "AKIAHFH6FGH213GFFG",
    "Account": "123456789012",
    "Arn": "arn:aws:iam::123456789012:user/seu-usuario"
}
```

Assumindo Funções (Assume Role) - Demonstração

Feito isso, agora vamos executar o comando que vai assumir para nosso usuário as permissões configuradas para a nossa função.

```
c:\> aws sts assume-role --role-arn
arn:aws:iam::123456789012:role/s3-role --role-session-name
sessao-s3 --duration-seconds 3600 --output json

{
  "Credentials": {
    "AccessKeyId": "ASIA5FEXAMPLE123456",
    "SecretAccessKey":
"wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken":
"IQoJb3JpZ2luX2VjEJr//////////wEaCXVzLWVhc3QtMSJHMEUCIQDzExample
SessionTokenPart1ExampleSessionTokenPart2ExampleSessionTokenPart
3",
    "Expiration": "2026-01-05T15:42:31Z"
  },
  "AssumedRoleUser": {
    "AssumedRoleId": "AROAEXAMPLEID:sessao-s3",
    "Arn": "arn:aws:sts::123456789012:assumed-role/s3-
role/sessao-s3"
  }
}
```

- ARN da nossa Função AWS;
- Nome para a sessão;
- Tempo em segundos que permanecerá ativa (neste caso 3600s = 1h).

Assumindo Funções (Assume Role) - Demonstração

Para assumir a função limpamos as credenciais da nossa sessão e informamos as novas credenciais retornadas pelo comando **aws sts assume-role**.

```
Remove-Item Env:AWS_ACCESS_KEY_ID -ErrorAction SilentlyContinue
Remove-Item Env:AWS_SECRET_ACCESS_KEY -ErrorAction
SilentlyContinue
Remove-Item Env:AWS_SESSION_TOKEN -ErrorAction SilentlyContinue
```

```
$env:AWS_ACCESS_KEY_ID     = 'ASIA5FEXAMPLE123456'
$env:AWS_SECRET_ACCESS_KEY =
'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'
$env:AWS_SESSION_TOKEN     =
'IQoJb3JpZ2luX2VjEJr/////////wEaCXVzLWVhc3QtMSJHMEUCIQDzExample
SessionTokenPart1ExampleSessionTokenPart2ExampleSessionTokenPart
3'
$env:AWS_CREDENTIAL_EXPIRATION = '2026-01-05T15:42:31Z'
```

Assumindo Funções (Assume Role) - Demonstração

Após isso, executando novamente comando **aws sts get-caller-identity**, não serão mais retornadas informações sobre o usuário, mas sim sobre a função assumida.

```
C:\> aws sts get-caller-identity

{
    "UserId": "AROA7BFF9FGHFG2GX1XD5D:sessao-s3",
    "Account": "123456789012",
    "Arn": "arn:aws:sts::123456789012:assumed-role/s3-
role/sessao-s3"
}
```

Assumindo Funções (Assume Role) - Demonstração

A partir deste momento é possível acessar todos os recursos que estiverem configurados para a função .

```
c:\> aws s3 ls
2023-04-11 11:24:11 funcionarios
2023-04-11 14:01:23 fotos
2024-12-30 17:31:37 pagamentos
```

Assumindo Funções (Assume Role) - Observações

Após vencer o período de expiração ou caso a sessão do Powershell seja fechada e aberta uma nova, não será mais possível acessar os recursos AWS, a menos que refaça todo o processo de autenticação.

Isso parece ruim, mas na verdade é MUITO BOM, pois garante segurança ao acesso na conta AWS. Pois você abre uma sessão, se autentica, faz o que precisa ser feito e ao final o acesso é extinto, prevenindo que terceiros tenham acesso indevido.

Neste exemplo, de forma didática executamos os comandos e informamos valores manualmente. Mas profissionalmente usaríamos scripts para automatizar este processo.

Importante

Qualquer ação ou recurso que não esteja discriminado nas políticas de um usuário, grupo ou função é automaticamente negado pela AWS.

Fica a cargo do administrador da conta definir os acessos e ações para cada recurso.


DÚVIDAS